Comment on "SRAM-PUF Based Entities Authentication Scheme for Resource-constrained IoT Devices"

Michael Amar[ℵ] , Amit Kama[ℵ], Kang Wang[ℶ], Yossi Oren[ℵ]
{amarmic@post. | kamaa@post. | yos@ } bgu.ac.il

[ℵ] Dept. of Software and Information Systems Engineering, Ben Gurion University of the Negev, Israel.
[ℶ]Alibaba Group

Abstract. The cloud-based Internet of Things (IoT) creates opportunities for more direct integration of the physical world and computer-based systems, allowing advanced applications based on sensing, analyzing and controlling the physical world. IoT deployments, however, are at a particular risk of counterfeiting, through which an adversary can corrupt the entire ecosystem. Therefore, entity authentication of edge devices is considered an essential part of the security of IoT systems. A recent paper of Farha *et al.* suggested an entity authentication scheme suitable for low-resource IoT edge devices, which relies on SRAM-based physically unclonable functions (PUFs). In this paper we analyze this scheme. We show that, while it claims to offer strong PUF functionality, the scheme creates only a weak PUF: an active attacker can completely read out the secret PUF response of the edge device after a very small amount of queries, converting the scheme into a weak PUF scheme which can then be counterfeited easily. After analyzing the scheme, we propose an alternative construction for an authentication method based on SRAM-PUF which better protects the secret SRAM startup state.

Keywords: Entity authentication, SRAM-PUF, Physical security, Security analysis.

1 Introduction

[1] The Internet of Things is an important new application domain for connected computing. It facilitates the connection of physical objects, such as sensors and actuators, to the digital domain. This synergy allows us to monitor and control far-off devices with improved accessibility, and to automate management processes. For example, a smart traffic control system can optimize vehicle routing with the aid of traffic sensors. A typical IoT deployment consists of a large number of edge devices connected to a powerful central cloud server. It is important

[1] M. Amar and A. Kama contributed equally to this article.

to keep in mind that many IoT edge devices, especially the low budget ones, have restricted computational power and limited energy budgets.

Because IoT deployments rely on many low-powered edge devices, they are at a particular risk of counterfeiting. An adversary who counterfeits an edge device can corrupt the entire ecosystem, by sending false data to the cloud server. Such an adversary can also gain financial benefit by reverse-engineering or stealing original designs, and then making cheap clones of the edge devices that abuse the resources provided by the cloud server. A particular risk comes from the overproduction scenario, in which a subcontractor hired to produce a limited amount of IoT edge devices produces a much larger amount, and sells the surplus by itself. To protect against this risk, each IoT edge device needs to be uniquely identified and authenticated before connecting to the cloud server. Traditional ID mechanisms, however, are not feasible in IoT, due to the constrained runtime environment of the IoT edge device. Other mechanisms, which rely on a standalone secure element chips or e-fuses, introduce additional costs to the edge device and make them more difficult to deploy.

A recent paper of Farha *et al.* [4] suggested an entity authentication scheme suitable for low-resource IoT edge devices, based on physically unclonable functions, or PUFs [6,15]. A PUF is a function that maps between challenges and responses, and is embodied by a certain physical device. A good PUF is easy to evaluate on one particular device, but hard to characterize and replicate on a different device. A PUF is considered *weak* if the number of challenge-response pairs (CRPs) is limited, and *strong* if there is a significant number of CRPs [7]. As more formally stated by Herder *et al.* [8], one of the requirements for a strong PUF is: “an adversary given a polynomial-sized sample of adaptively chosen CRPs cannot predict the response to a new, randomly chosen challenge.”

To use PUFs for authentication, the verifier, or authenticator, first issues a series of challenges to the prover in a safe setting, and records the responses in a database. Then, to authenticate, the verifier issues one of the stored challenges to the prover, collects the response, and compares it to the pre-recorded one. Obviously a strong PUF is better for the purpose of authentication than a weak PUF, since the limited number of available CRPs can severely limit the usable runtime of the authentication system.

The scheme of Farha *et al.* [4] relies on static random access memory (SRAM) based PUFs. SRAM is a memory technology widely used in low-cost processors. Each SRAM cell is composed of two cross-coupled inverters designed to be symmetric. Process variations, however, disrupt this symmetry. As a result, each individual SRAM cell has its own bias toward a preferable start-up value (either 1, 0, or random). The start-up values are random between devices but are largely static per device. Thus, using an array of SRAM cells generates a unique fingerprint, as originally proposed by Holcomb *et al.* [9]. SRAM PUF designs must compensate for bits in the SRAM with random or unreliable startup states. To do so, the algorithm which extracts the fingerprint from SRAM cells typically makes use of helper data, which identifies the proper subset of bits to use, as well as error-correcting codes, which compensate for the remaining small amount of startup instability.

SRAM PUFs are intrinsically weak PUFs, since the device only has a single response, namely its SRAM startup value. This limits their utility as a mechanism for authentication. Farha *et al.* designed a scheme that aims to convert the SRAM-based weak PUF into a strong PUF. Their proposed scheme uses CRPs represented by re-ordered memory addresses as challenges, and the corresponding SRAM cells' startup values as responses. From a security perspective, the proposed scheme was designed to be robust against attempts to authenticate a fake end device to the system. This includes man-in-the-middle attacks, side-channel attacks, extraction of hashing tables, and brute force attacks.

In this article we analyze the scheme of Farha *et al.* and show that, while it claims to offer strong PUF functionality, the scheme creates only a weak PUF. In particular, as we show below, an active attacker can completely read out the secret PUF response of the edge device after a very small amount of queries, converting the scheme into a weak PUF scheme, which can then be counterfeited easily. After studying the security of Farha *et al.*'s scheme, we next propose an alternative construction for SRAM-PUF based authentication, which better protects the secret SRAM startup state.

2 The Scheme of Farha *et al.*

The proposed scheme is meant to allow end devices (provers) to prove their identity to a smart gateway (verifier). The scheme consists of two phases: the enrollment phase, in which the PUF response is calculated and stored, and the authentication phase, in which the edge device interacts with the verifier to prove that it knows the value of the secret PUF response. The attack we describe in this paper targets the authentication phase.

2.1 Enrollment phase

In the enrollment phase of the proposed scheme, the edge device carries out the following steps:

- Read out 1KB of SRAM data for 50 times.
- Create a stability map of those SRAM values.
- Run a simple algorithm to select the stable cells, while keeping the final Hamming weight to $\sim$50%.
- Generate helper data with an ECC algorithm.
- Store the stability map and helper data in the non-volatile storage of the edge device.
- Transmit the generated fingerprint (100 bits) to the authenticator device (i.e., smart gateway) via an eavesdropping-resistant channel.

2.2 Authentication phase

In the authentication phase, the authenticator (i.e., smart gateway) sends a challenge to the prover (i.e., edge device), retrieves a response, and finally checks the response for correctness. This operation is summarized in Figure 1.

- The authenticator chooses a random re-ordering of the set of stable SRAM cell addresses. While the original paper simply calls this value "challenge", we note that it is a permutation, and mark it here as π.
- The authenticator calculates a random value *nonce*, and sends it to the prover together with π as a challange.
- The prover reads out the raw SRAM data, and processes it using the stored helper data, to create the secret PUF response, which we denote as PUF.
- The prover applies the permutation π to PUF, reordering its bits, resulting in a 100-bit value $RPUF = \pi(PUF)$.
- Finally, the prover uses a XOR operation to combine the nonce, the original PUF response, and the reordered PUF response, and sends back a hash of the response - $H(nonce \oplus PUF \oplus \pi(PUF))$.
- To verify, the authenticator similarly calculates $H(nonce \oplus PUF \oplus \pi(PUF))$ using its stored value for the PUF response. If the values match, authentication succeeds.

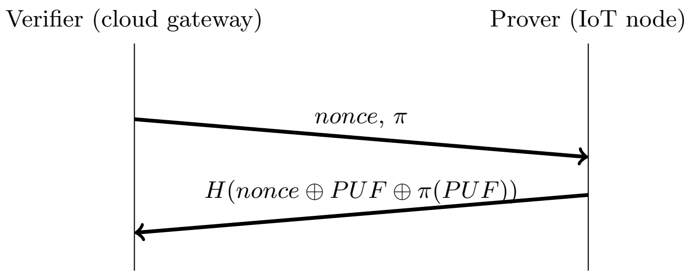

Fig. 1. Description of the scheme of Farha *et al.*, based on [4].

The authors of [4] argue that their scheme implements a strong PUF, since there are $100! \approx 9.3 * 10^{157}$ possible challenge permutations, each resulting in a different response from the prover.

3 Our Attack

We now describe an attack on the authentication phase of the scheme, which transforms it into a weak PUF. Let PUF be the secret PUF response of the prover. The objective of the attack is to recover this secret value using a small number of authentications. Now let PUF_i be the i'th bit of this response, counting from zero, and let $swap_{i,j}()$ be a permutation which swaps the values of bits i and j in its input, where the least-significant bit index is zero. For example: $swap_{1,2}(0100_b) = 0010_b$. We note that if $PUF_i = PUF_j$ then immediately $swap_{i,j}(PUF) = PUF$.

This property of the permutation gives an active attacker, for instance in a man-in-the-middle setting, the ability to determine whether two bits in the PUF response are identical. Thus we can mount the following attack: The attacker first fixes a random value n_0 for *nonce* and selects a random initial permutation π_0. Next, the attacker progressively swaps bits in π_0. After each bit swap, the attacker requests another authentication response from the prover, obtaining a series of responses, as follows:

$$\begin{aligned} R_0 &= H(n_0 \oplus PUF \oplus \pi_0(PUF)), \\ R_1 &= H(n_0 \oplus PUF \oplus \pi_0(swap_{0,1}(PUF))), \\ R_2 &= H(n_0 \oplus PUF \oplus \pi_0(swap_{0,2}(PUF))), \\ &\dots \\ R_{99} &= H(n_0 \oplus PUF \oplus \pi_0(swap_{0,99}(PUF))) \end{aligned}$$

To recover the secret value of PUF, we note that for all i, if $PUF_i = PUF_0$ then $R_i = R_0$[2]. Therefore, after collecting 100 responses, the attacker knows which bits of the secret PUF response are equal to PUF_0, and which are different. All that is left for the attacker is to try two possible assignments for PUF – one in which $PUF_0 = 0$ and one in which $PUF_0 = 1$ – and verify which of the two assignments can be used to produce the correct value of R_0 originally sent by the prover. Once the attack concludes, the attacker knows the value of PUF, and can now produce a counterfeit edge device which can answer any authentication challenge.

Note that the attack can be optimized and made harder to detect. We first note that the attacker does not have to recover all 100 bits of the PUF, but only enough bits to make brute-forcing over the remaining bits practical. Considering that modern computing hardware can perform more than 2^{30} hash operations per second [21], an attacker only needs to recover 50 bits of the PUF response, then brute-force the remaining 50 bits in less than a week of post-processing. The attacker can also evade detection through the use of fresh nonces and permutations for each bit, that is:

[2] If $PUF_i \neq PUF_0$ then with very high probability $R_i \neq R_0$, depending on the properties of the hash function H

$$\begin{aligned}
R_{0,1} &= H(n_1 \oplus PUF \oplus \pi_1(PUF)),\\
R_1 &= H(n_1 \oplus PUF \oplus \pi_1(swap_{0,1}(PUF))),\\
R_{0,2} &= H(n_2 \oplus PUF \oplus \pi_2(PUF)),\\
R_2 &= H(n_2 \oplus PUF \oplus \pi_2(swap_{0,2}(PUF))),\\
&\dots\\
R_{0,99} &= H(n_{99} \oplus PUF \oplus \pi_{99}(PUF)),\\
R_{99} &= H(n_{99} \oplus PUF \oplus \pi_{99}(swap_{0,99}(PUF)))
\end{aligned}$$

This modification doubles the amount of required queries, but makes the attack much harder to detect.

4 An Alternative Construction

The motivation behind the novel authentication scheme proposed by [4] was the claimed inability of IoT edge devices to support advanced asymmetric cryptography. As the authors state, such devices are "[...] unable to run the security methods that require intensive computational operations". This led the authors to seek attempt a custom construction which, as we showed, is insecure. In the remainder of the paper, we briefly propose an alternative construction based on asymmetric cryptography. We begin with a formal definition, follow with some security proofs and conclude with a survey of related art.

4.1 Formal Definition

The ability of low-resource devices to run public-key cryptography was demonstrated in a series of works centered on cryptography for low-powered RFID tags. In particular, in 2008 Oren and Feldhofer presented WIPR, a full-strength public-key encryption which can be implemented on a low-resource device [14]. A full microcontroller implementation of the scheme was provided in [1], showing that it has very modest storage and runtime costs appropriate for low-resource deployments. A variant of the scheme proposed in [22], called WIPR-SAEP, provides a stronger security guarantee at a slightly higher implementation cost, assuming the prover device has an existing implementation of a hash function.

The scheme is a straightforward implementation of randomized public key-based authentication, where the prover has the public key and a secret ID, and the verifier has the private key. The scheme begins when the verifier sends the prover a random nonce. Next, the prover encrypts the nonce, together with its own secret ID and some randomness, using the public key, and sends it to the verifier. Finally, the verifier decrypts the message using the private key, verifies the correctness of the nonce and learns the secret ID. The randomness is used to prevent replay attacks and to provide forward and backward secrecy.

To apply this scheme to a low-resource setting, the WIPR scheme of Arbit *et al.* applies several resource-saving steps which were proven not to affect security. First, the chosen public key scheme is the Randomized Rabin encryption scheme, a variant of Rabin's encryption scheme of 1970 [16] proposed simultaneously by Shamir and Naccache [18,13]. In Randomized Rabin, the expensive modular reduction step is replaced with the addition of a large random multiple of the public key n. As Shamir showed, if the random factor is chosen properly, this scheme is as secure as the standard Rabin scheme, which is in turn at least as hard as integer factorization. The encryption step, which must be performed on the low-resource prover, is thus transformed into two multiply-accumulate operations which do not require modular reductions, resulting in a very low ROM and RAM footprint. Second, the public decryption key is stored in a memory-efficient way, using Johnston's method [11]. Third, the random strings required by the scheme are stored in a space-efficient way based on reversible ciphers.

A brief description of the scheme is provided in the Figure below. r_1 and r_2 are random bit sequences generated by the prover, and MIX is a byte-interleaving function designed to prevent large blocks of sender- or prover-generated blocks from appearing in the data. In the example below, PUF is the plain PUF response of the IoT node, which is sent in its entirety to the server. It is certainly possible to integrate this scheme with other schemes such as ID-AKE [12] or TCALAS [20] by replacing PUF with a protocol-specific payload.

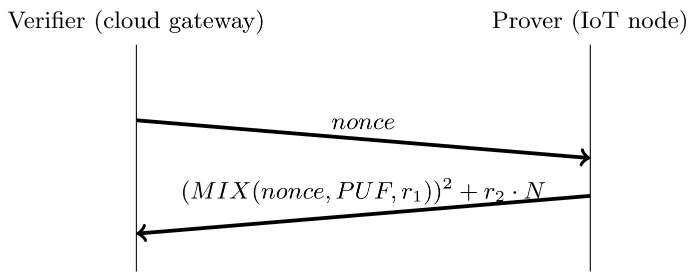

Fig. 2. A modified scheme based on WIPR [1].

4.2 Security Analysis

As stated by Oren and Feldhofer [14], the following claims hold regarding the security of the scheme, even if we assume the adversary has complete knowledge of the entire system other than the private key. Most of these claims hold for any randomized encryption scheme based on public keys. We also describe the security of the scheme in an ephemeral secret leakage (ESL) setting [3], in which the randomness is controlled by the adversary.

– *Secrecy*: An adversary observing a protocol exchange cannot learn anything about the value of the secret PUF response, even if the adversary knows the value of the public key n. This follows naturally from the use of public-key encryption.
– *Full backward and forward privacy*: An adversary cannot determine whether a node it currently holds was a part of any past or future protocol exchange it has recorded, even if the adversary knows the value of the node's PUF response.
– *Metadata privacy*: The adversary cannot determine whether a certain public key n was used in the protocol exchange.
– *Implicit verifier authentication*: Since only a verifier who holds a private key can decipher data coming from the node, a succesful protocol execution serves as an implicit way of authenticating the verifier. This lets the two parties create a secure data channel.
– *Security under ESL*: An adversary who can control the verifier-side randomness gains no advantage. An adversary who can control the prover-side randomness can decrypt ciphertexts without knowing the private key, allowing the recovery of the PUF's response. However, even such an adversary cannot break the forward and backward privacy property.

We note that this scheme does not claim to convert the SRAM PUF into a strong PUF; specifically, a server-side compromise, which recovers either the secret PUF response or the secret decryption key, will allow devices to be counterfeited. However, unless the server is compromised, this scheme can still provide authentication over an extremely large number of challenges. The scheme therefore achieves the implementation objectives and security claims of Farha *et al.*.

4.3 Related Work

As stated in [17], in 2002 Pappu [15] introduced the concept of PUFs as Physical One-Way Functions. The indicated technology was based on the response obtained when shining a laser on a bubble-filled transparent epoxy wafer. Gassend et al. [5] introduced Silicon Physical Random Functions in 2002, which use manufacturing process variations in ICs to uniquely characterize each IC. The statistical delay variations of transistors and wires in the IC were used to create a parameterized self-oscillating circuit to measure frequency for characterization. This circuit is nowadays known as a Ring Oscillator PUF. Besides hardware intrinsic PUFs based on delay measurements, a second type, based on the measurement of start-up values of memory cells, is known. This type includes the use of the initial start-up values of SRAM as digital fingerprint introduced by Guajardo et al. in 2007 [7], and also by Holcomb et al. [10] that year.

There is a large body of work on low-resource authentication schemes, and in particular on authentication schemes using PUFs. Many of these works focus on authenticated key agreement (AKA) and authenticated key exchange (AKE). In

2002 Canetti and Krawczyk introduced the notion of ESL security, which analyzes the security of a system even if the random nonces are controlled or known by the adversary [3]. This setting is particularly interesting in the smart home environment, when randomness generation and extraction can be offloaded to a central server. For example, Liu et al. presented an authenticated key agreement protocol secure against ESL attacks [12]. This protocol is designed for three parties – a client, an application server and an authentication server. Shuai et al. present an efficient authentication scheme for smart home environments based on elliptic-curve cryptography [19]. This scheme was later analyzed by Banerjee et al., who present a version with improved security [2]. The scheme we present in this paper can be used as a building-block for one of these more advanced protocols.

5 Author Response

We sent a draft of this report to Farha *et al.*, the authors of the original paper, during February 2022. We plan to update this section of the report with their response, as it arrives.

6 Conclusion

In this comment, we discussed the recent scheme of Farha *et al.*. We showed how an active attacker can recover the secret PUF response from this scheme after exchanging a very small number of messages with the prover. Thus, we conclude that the proposed scheme does not provide strong PUF ability. We next suggest an alternative scheme which achieves the security aims of Farha *et al.* using a low-resource version of a standard public-key asymmetric cipher.

IoT is a novel technological field with unique technical constraints. Even so, when introducing new security constructions such as authentication schemes, they should be carefully analyzed to make sure they do not introduce unexpected vulnerabilities. The designers of future IoT security schemes are encouraged to take inspiration from prior work on low-resource security constructions smart cards, RFID tags and similar low-resource devices.

References

1. Alex Arbit, Yossef Oren, and Avishai Wool. A secure supply-chain RFID system that respects your privacy. *IEEE Pervasive Comput.*, 13(2):52–60, 2014.
2. Soumya Banerjee, Vanga Odelu, Ashok Kumar Das, Samiran Chattopadhyay, and Youngho Park. An efficient, anonymous and robust authentication scheme for smart home environments. *Sensors*, 20(4):1215, 2020.
3. Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2002.

4. Fadi Farha, Huansheng Ning, Karim Ali, Liming Chen, and Christopher Nugent. SRAM-PUF-based entities authentication scheme for resource-constrained IoT devices. *IEEE Internet Things J.*, 8(7):5904–5913, 2021.
5. Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 148–160, New York, NY, USA, 2002. Association for Computing Machinery.
6. Blaise Gassend, Dwaine E. Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *CCS*, pages 148–160. ACM, 2002.
7. Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA intrinsic pufs and their use for IP protection. In *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.
8. Charles Herder, Meng-Day (Mandel) Yu, Farinaz Koushanfar, and Srinivas Devadas. Physical unclonable functions and applications: A tutorial. *Proc. IEEE*, 102(8):1126–1141, 2014.
9. Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Computers*, 58(9):1198–1210, 2009.
10. Daniel E Holcomb, Wayne P Burleson, Kevin Fu, et al. Initial sram state as a fingerprint and source of true random numbers for rfid tags. In *Proceedings of the Conference on RFID Security*, volume 7, page 01, 2007.
11. Anna M. Johnston. Digitally watermarking RSA moduli. *IACR Cryptol. ePrint Arch.*, page 13, 2001.
12. Chao-Liang Liu, Wang-Jui Tsai, Ting-Yi Chang, and Ta-Ming Liu. Ephemeral-secret-leakage secure id-based three-party authenticated key agreement protocol for mobile distributed computing environments. *Symmetry*, 10(4):84, 2018.
13. David Naccache. Method, sender apparatus and receiver apparatus for modulo operation. European patent application no. 91402958.2, Filed 10/27/1992.
14. Yossef Oren and Martin Feldhofer. WIPR – public-key identification on two grains of sand. In Sandra Dominikus, editor, *Workshop on RFID Security 2008*, pages 15 – 27, July 2008. `http://iss.oy.ne.ro/WIPR`.
15. Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002.
16. Michael Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, MIT, Cambridge, MA, USA, 1979.
17. Geert Jan Schrijen and Vincent van der Leest. Comparative analysis of SRAM memories used as PUF primitives. In Wolfgang Rosenstiel and Lothar Thiele, editors, *2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, Dresden, Germany, March 12-16, 2012*, pages 1319–1324. IEEE, 2012.
18. Adi Shamir. Memory efficient variants of public-key schemes for smart card applications. In *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 445–449. Springer, 1994.
19. Mengxia Shuai, Nenghai Yu, Hongxia Wang, and Ling Xiong. Anonymous authentication scheme for smart home environment with provable security. *Computers & Security*, 86:132–146, 2019.
20. Jangirala Srinivas, Ashok Kumar Das, Neeraj Kumar, and Joel J. P. C. Rodrigues. TCALAS: temporal credential-based anonymous lightweight authentication scheme for internet of drones environment. *IEEE Trans. Veh. Technol.*, 68(7):6903–6916, 2019.
21. Michael Bedford Taylor. The evolution of bitcoin hardware. *Computer*, 50(9):58–66, 2017.

22. Jiang Wu and Douglas R. Stinson. How to improve security and reduce hardware demands of the wipr rfid protocol. In *2009 IEEE International Conference on RFID*, pages 192–199, 2009.