SHAIP: Secure Hamming Distance for Authentication of Intrinsic PUFs

SIAM UMAR HUSSAIN, M. SADEGH RIAZI, and FARINAZ KOUSHANFAR,
University of California San Diego

In this article, we present SHAIP, a secure Hamming distance–based mutual authentication protocol. It allows an unlimited number of authentications by employing an intrinsic Physical Unclonable Function (PUF). PUFs are being increasingly employed for remote authentication of devices. Most of these devices have limited resources. Therefore, the intrinsic PUFs are most suitable for this task as they can be built with little or no modification to the underlying hardware platform. One major drawback of the current authentication schemes is that they expose the PUF response. This makes the intrinsic PUFs, which have a limited number of challenge-response pairs, unusable after a certain number of authentication sessions. Moreover, these schemes are one way in the sense that they only allow one party, the prover, to authenticate herself to the verifier. We propose a symmetric mutual authentication scheme based on secure (privacy-preserving) computation of the Hamming distance between the PUF response from the remote device and reference response stored at the verifier end. This allows both parties to authenticate each other without revealing their respective sets of inputs. We show that our scheme is effective with all state-of-the-art intrinsic PUFs. The proposed scheme is lightweight and does not require any modification to the underlying hardware.

CCS Concepts: • **Security and privacy → Authentication**; **Embedded systems security**;

Additional Key Words and Phrases: Physical unclonable function, intrinsic PUF, remote authentication, biometric authentication, secure hamming distance

ACM Reference format:
Siam Umar Hussain, M. Sadegh Riazi, and Farinaz Koushanfar. 2018. SHAIP: Secure Hamming Distance for Authentication of Intrinsic PUFs. *ACM Trans. Des. Autom. Electron. Syst.* 23, 6, Article 75 (December 2018), 20 pages.
https://doi.org/10.1145/3274669

1 INTRODUCTION

With the recent rapid surge of the Internet of Things (IoT) paradigm, remote authentication of devices holding sensitive information has become one of the primary security concerns. Traditional methods of authentication using secret keys stored in a non-volatile memory have been

This work is supported in parts by an Office of Naval Research grant (N00014-17-1-2500), National Science Foundation grants (CSN-1649423), Semiconductor Research Corporation grant (2016-TS-2690) and Multidisciplinary Research Program of the University Research Initiative grant (FA9550-14-1-0351). We would also like to sincerely thank Xavier Carpent for his suggestions to improve the work.
Authors' addresses: S. U. Hussain, M. S. Riazi, and F. Koushanfar, Department of Electrical and Computer Engineering, Jacobs Hall, EBU1, 2nd Floor, Jacobs School of Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093; emails: {siamumar, mriazi, farinaz}@ucsd.edu.

1084-4309/2018/12-ART75 $15.00
https://doi.org/10.1145/3274669

shown to be vulnerable to physical attacks [1, 2]. Therefore, Physical Unclonable Functions (PUF) has become a key element in remote authentication of resource-constrained devices. PUFs offer a secure and lightweight alternative where the secret is not stored rather extracted from the unique physical properties of the authenticating device [3].

A number of PUF-based authentication schemes have been presented in literature [4–9]. However, all these schemes involve key updates and therefore require *strong* PUFs, which have an exponential number of challenge-response pairs (CRP) as opposed to *weak* PUFs, which have a limited number of CRPs. Strong PUFs, in general, require dedicated hardware and may not be suitable for resource-constrained devices that constitute the majority of the connected devices in IoT. Moreover, in IoT, the task of authentication is being distributed among nodes at different levels of the network to scalably manage the massive web of various entities. In such scenarios, holding a large CRP database at the verifier end may become impractical.

Intrinsic PUFs [10] are more suitable of IoT since they can be instantiated in off-the-shelf devices with little or no modification to the underlying hardware. All variants of intrinsic PUFs are weak as they offer a limited number of CRPs. The current schemes involving weak PUFs allow a limited number of authentications [11, 12]. To the best of our knowledge, there have been only two weak PUF authentication schemes [13, 14] that allow unlimited mutual authentication sessions. Both schemes employ Fuzzy Extractor to limit the exposure of the PUF response. The scheme in Reference [13] still leaks some information and thus, to ensure security, requires a minimum number of bits (1785 bits to ensure 128-bit security) that may not be available from many intrinsic PUFs. A common limitation of both of these methods is that their proof of correctness must assume certain properties of the distribution of the PUF response, and requires empirical verification. The work in Reference [14] verifies the assumptions by experimental evaluation of the Ring Oscillator (RO)-based PUF. However, there is no guarantee that these assumptions will hold for all different PUF designs (or even for RO-based PUFs on a different hardware platform). Another drawback of these schemes as well as most other PUF-based authentication schemes is that they assume the response to be in binary form. A number of recent intrinsic PUF designs, e.g., the DRAM PUF proposed in Reference [12], generates the PUF response as a set of integers. Authentication of these PUFs requires secure set operations and thus is not compatible with the schemes presented in Reference [13, 14].

In this article, we propose an authentication scheme where the prover and the verifier do not reveal a single bit of the PUF response to each other. Thus this scheme supports unlimited mutual authentication sessions even with weak PUFs. Unlike in Reference [13] it requires only 237 bits to ensure 128-bit security in an equivalent setting. More importantly, our scheme does not require any assumption on the distribution of the PUF response. Moreover, it supports PUF response both in binary form and as a set of integers. Another feature of this protocol is that it allows successful authentication even with the presence of noise in responses at different interrogations of the PUF. Therefore, it does not require any error correction scheme as most of the PUF-based protocols [8, 9, 13, 14]. Since one of our primary goals is authentication with intrinsic PUFs we design the scheme such that it can be implemented without any additional hardware.

Our scheme is based on the secure (privacy-preserving) computation of the Hamming distance between the PUF and reference responses from the prover and the verifier, respectively. Interestingly, secure Hamming distance has been employed in the field of biometric authentication. This field faces similar challenges since the biometric keys tend to be noisy and limited in supply, just like the responses of weak PUFs. A number of these works [15, 16] presents protocols for secure computation of Hamming distance in the *malicious* security model, where any party can deviate from the accepted behavior to learn more information about the other party's data or to produce incorrect results. These protocols are proven to be secure in the sense that they do not *leak* any

information to any party other than the protocol output. However, we show that the security of the biometric keys can still be breached just from the information held by an individual party.

In our scheme, we first construct an authentication function such that its secure computation does not allow any malicious party to produce a false result or deduce the inputs of the honest party in polynomial (in terms of the number of bits in the response) number of attempts. We then utilize the well known Secure Function Evaluation (SFE) protocol named Yao's Garbled Circuit (GC) [17] to securely compute the authentication function. The original Yao's protocol was secure in *honest-but-curious* security model, which assumes that both parties follow the protocol honestly yet may try to learn additional information from the information at hand. However, subsequent enhancements [18–21] have made it secure in the malicious security model. To add support for the PUFs with integer responses we employ Locality Sensitive Hashing (LSH) [22, 23]. It translates the authentication involving set operations to our Hamming distance–based authentication function. In brief, our contributions are as follows:

- We present a mutual authentication scheme for weak intrinsic PUFs based on the secure computation of Hamming distance between the PUF response from the prover and the reference response held at the verifier.
- We prove that our authentication scheme does not allow any malicious party to produce a false result or deduce the inputs of the honest party in polynomial (in terms of the number of bits in the response) number of attempts.
- Our authentication scheme supports PUF response in the classic binary form as well as a set of integers as produced by a number recent intrinsic PUF implementations.
- The authentication scheme supports mutual authentication even with the presence of noise in the PUF response and thus eliminates the need for error correction methods.
- Implementation of the scheme demonstrates the practicality of the design. With maximum allowed fraction of mismatched bits between an authentic pair of responses from the same PUF set to 10%, the protocol execution takes 81 ms on a desktop processor and 487 ms on an embedded processor.

The rest of the article is organized as follows. In the next section, we provide a brief background on PUFs (with the focus on intrinsic PUFs) and the cryptographic protocols employed in this article. Sections 3 and 4 outline the authentication scheme and analyze the security, respectively. The details of the implementation is described in Section 5 and its performance is evaluated in Section 6. A survey of the related literature is provided in Section 7. The article concludes in Section 8.

2 BACKGROUND

2.1 Physical Unclonable Function

A PUF is a function whose output (response) depends on both the applied input (challenge) and the unique physical properties of the hardware where it resides. It utilizes the inherent natural physical disorder, e.g., silicon manufacturing variations, of the device to create a unique signature (fingerprint), of that device. The challenge and response together form one CRP. In the exact sense, for a PUF to be used in authentication it must always generate the same response when interrogated multiple times by the same challenge. However, in practice, this criteria is difficult to meet precisely since the response bits are not perfectly reproducible. Transistor noise, as well as various environmental variations (supply voltage, temperature, etc.), introduces noise in the generated response. To ensure usability, the major portion of the response should be stable over multiple interrogations. Some authentication schemes employ error correction methods that can regenerate the reference PUF response given that the number of noisy bits is within a certain limit [24]. In

this work, for binary PUF response, we compute the Hamming distance between generated and reference PUF response and compare it against a certain threshold for authentication decision.

There are two broad variants of PUFs: strong PUF and weak PUF [25]. Strong PUF supports a large number (usually exponential in term of a number of challenge bit) of CRPs. This ensures that even if an adversary gains access to a large subset of CRPs, it cannot predict the ones not already known. However, weak PUFs provide only a limited space of CRPs. In applications involving weak PUFs, it is assumed that its responses are not accessible by adversaries.

In this article, we are particularly interested in intrinsic PUFs that can be instantiated in off-the-shelf devices without any modification to the hardware. They have limited CRP space and thus are considered weak PUFs. The most widely examined intrinsic PUFs are the one based on Static Random-Access Memory (SRAM) [26–28]. In SRAM PUFs, the start-up values of the bi-stable SRAM cell constitutes the PUF response that is a binary string. Even though SRAM PUFs have been shown to possess good PUF characteristics, their one major drawback is that since the response is based on start-up values, the authentication can only be performed at boot time or stored in a memory, which subverts the main motivation behind using a PUF.

Recently Dynamic Random Access Memory (DRAM) has been introduced as a PUF construct [12, 29, 30]. The primary advantage of a DRAM PUF is that it can be interrogated at runtime of the operating system. If the periodic refresh of DRAM cells is turned off or delayed, then the charges of the cells decay in a manner unique to each cell resulting in the flipping of the stored value. At a certain delay, the indices of the flipped bits constitute the PUF response. Thus the response of a DRAM PUF is a set of integers rather than a bit stream in generic PUFs. If the DRAM PUF is used for authentication, then the similarity of the PUF response with the reference response stored at the verifier is measured using the *Jaccard similarity*. If the response generated by the PUF is R_{PUF} and the reference response saved at the verifier is R_{ref} the Jaccard similarity, then the $\mathcal{J}$ between these two sets of integers is given by

$$\mathcal{J} = \frac{|R_{PUF} \cap R_{ref}|}{|R_{PUF} \cup R_{ref}|}. \tag{1}$$

In Section 3.5, we will show how the problem of computation of $\mathcal{J}$ can be translated to the computation of the Hamming distance of two binary strings.

2.2 Cryptographic Protocols

We now provide a brief background on the cryptographic protocols employed in this work.

Oblivious Transfer. Oblivious Transfer (OT)[31] is a cryptographic protocol between a receiver R and a sender S. OT allows R to choose and receive one from a pair of messages provided by S without revealing her choice. In an 1-out-of-2 OT protocol, (OT_1^2), S holds a pair of messages (m_0, m_1); R holds a selection bit $s \in 0, 1$ and obtains m_s without revealing s to S and learns nothing about m_{1-s}.

Garbled Circuit. Yao's GC [17] is a cryptographic protocol that allows two parties Alice and Bob to jointly compute a function $y = f(a, b)$ on their private inputs a from Alice and b from Bob. In the end, one or both of them learn the output y. The function f needs to be represented as a Boolean circuit, called *netlist*, consisting of 2-input 1-output logic gates. Alice, called the *garbler*, garbles the circuit as follows. For each wire in the netlist, she assigns two k-bit[1] random keys corresponding to the values 1 and 0. For each gate, a garbled truth table is generated by encrypting the keys for output with the corresponding input keys. She also assigns a 1-bit random mask for each wire. The

[1] k is a security parameter, its value is set to 128 in recent works [21, 32]

masked value of a wire is XOR of the actual value and the mask bit. She then arranges the rows in the garbled table according to the masked value and sends them along with the keys corresponding to her input values to Bob, called the *evaluator*. Bob first obtains the keys corresponding to his input values obliviously through OT_1^2. He then uses these input keys to evaluate the encrypted tables gate by gate and learns the masked value of each wire in the netlist. This way, the final output that Bob learns is the masked values of the output wires. These masked values need to be XORed with the corresponding mask bits held by Alice to learn the actual values. We say that the final output is *XOR-shared* between Alice and Bob. They need to communicate their respective shares to the other party to learn the actual value of the output.

Garbled Circuit Optimizations. The GC protocol has gone through a number of optimizations: free-XOR [33], row reduction [34], half gate [35], and fixed-key cipher [32]. Among these optimizations, the most important one is free-XOR as it allows the evaluation of XOR, XNOR, and NOT gates without costly cryptographic encryption, which also translates to less communication time as the XOR gates do not need a transfer of the garbled tables. As a result, the primary optimization goal while generating the netlist for f is to minimize the number of non-XOR gates.

Enhancements for Malicious Security. Yao's GC is proven to be secure in the *honest-but-curious* security model [32, 36], where the participating parties follow the agreed on behavior but may want to deduce more information without violating the protocol. While this model is sufficient in a large number of applications, to ensure the security of the authentication scheme, we must employ a protocol that is secure in the *malicious* security model. In this model, the parties are allowed to deviate from the protocol to learn information about the other party's input or to force an incorrect result. Among various protocols that are secure in this model, the most efficient realization to date is the Authenticated Garbling protocol presented in Reference [20]. The critical enhancements in this protocol over the semi-honest version are the following. (i) The mask bit itself for each wire is XOR-shared between Alice and Bob. (ii) To ensure that none of them can alter their respective shares of the mask bit in the middle of the protocol execution, they authenticate their shares using Message Authentication Codes. (iii) They compute the garbled circuit in a distributed manner, where the garbled tables are XOR shared between Alice and Bob. The garbled circuit is *authenticated* in the sense that none of Alice or Bob alone can change the logic of the circuit. We employ this protocol in our authentication scheme.

3 AUTHENTICATION SCHEME

In this section, we outline our authentication scheme for secure mutual authentication of intrinsic PUFs. We first describe the threat model. Then, we explain the authentication function that forms the core of the scheme. Next, we provide our authentication protocol for generic PUFs with binary response. Finally, we show how this generic protocol can be extended to support the memory-based PUFs that generates the response as a set of integers.

3.1 Threat Model

Our model consists of three parties, the prover $\mathcal{P}$, the verifier $\mathcal{V}$ and the adversary $\mathcal{A}$. We assume no shared secret between $\mathcal{P}$ and $\mathcal{V}$ other than the CRP (a single CRP will suffice). $\mathcal{P}$ is equipped with a weak PUF. Our scheme allows some noise in the PUF response generated each time by $\mathcal{P}$ and therefore does not require $\mathcal{P}$ to be equipped with any error correction scheme. Further, $\mathcal{P}$ has no secret stored in its non-volatile memory and erases its volatile memory on exiting the protocol. $\mathcal{A}$ can access the non-volatile memory of $\mathcal{P}$ but not the volatile memory during the time of protocol execution. Consistent with the state of the art schemes, we assume $\mathcal{A}$ to be able to eavesdrop on the communication channel between $\mathcal{P}$ and $\mathcal{V}$. However, unlike previous schemes, we do not

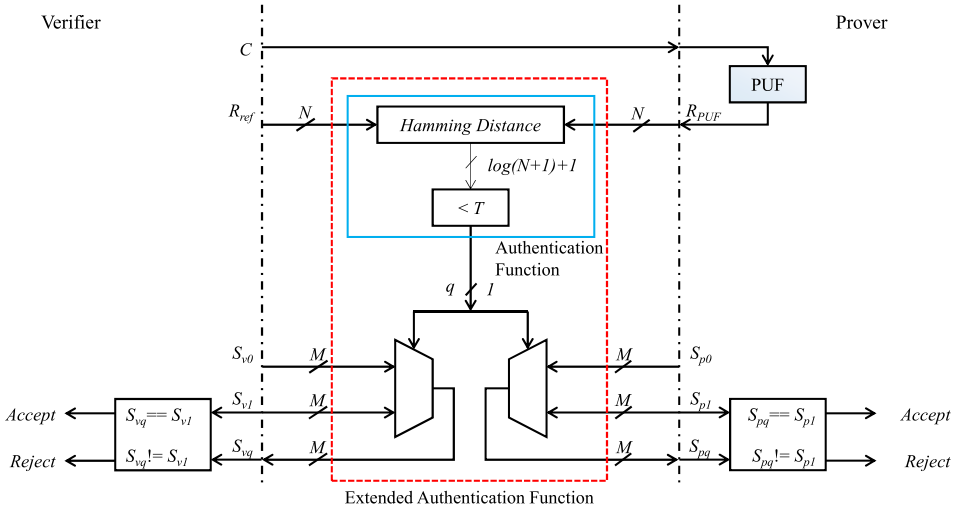

Fig. 1. Authentication protocol.

assume $\mathcal{V}$ to be trusted. The only distinction we make between $\mathcal{V}$ and $\mathcal{A}$ is the possession of the CRP.

3.2 Authentication Function

Figure 1 outlines the authentication protocol between $\mathcal{V}$ and $\mathcal{P}$. The steps enclosed within the solid (blue) box constitute the authentication function,

$$q = \mathcal{F}_{auth}(R_{ref}, R_{PUF}, T) = \begin{cases} 1, & HD(R_{ref}, R_{PUF}) < T \\ 0, & otherwise \end{cases}, \tag{2}$$

where $HD(.,.)$ denotes the Hamming distance. The input to the authentication function from $\mathcal{V}$ is the reference PUF response R_{ref}, an N-bit binary string stored in its database. The input from $\mathcal{P}$ is the PUF response R_{PUF}, an N-bit binary string extracted from the PUF. The first step of the function is computing the Hamming distance between R_{ref} and R_{PUF}. If PUF responses were free from noise or in presence of error correcting code, then the Hamming distance would be zero for a genuine PUF response. However, to allow a generic PUF without any error correction, we compare the resultant Hamming distance with a threshold value, T. Here, T is a publicly known parameter agreed on by $\mathcal{V}$ and $\mathcal{P}$ before the start of the protocol (during the initialization phase, Section 3.3). The choice of T will depend on the characteristics of the PUF and the acceptable error margin. The 1-bit output q of the comparator indicates whether or not the PUF response is close enough to the reference response saved at the verifier.

We employ the Authenticated Garbling protocol [21] to ensure the privacy of the inputs of $\mathcal{V}$ and $\mathcal{P}$. We now take a closer look into the security of this protocol. As discussed in Section 2.2, the final output is XOR-shared between the two participating parties, and they have to authenticate their respective shares to prevent alteration in the middle of the protocol execution. The parameters of the authentication function $\mathcal{F}_{auth}$ are set such that the probability of success of an adversary $\mathcal{A}$ with an incorrect PUF response is infinitesimally small (2^{-128}, see Section 4.1 for details). Therefore, if the authentication function $\mathcal{F}_{auth}$ is evaluated through the Authenticated Garbling protocol, the adversary would already know the value of the output q (0, in this case) with success rate close to unity. Therefore, $\mathcal{A}$ could flip her share of the bit q, which would result in flipping the final value of q from 0 to 1 and making the honest party accept her as a legitimate $\mathcal{V}$

or $\mathcal{P}$. The Authenticated Garbling protocol is able to prevent alteration of the shares in the middle of the protocol execution. In this specific case, the final output is independent of the input of the honest party and is known to the adversary even before the execution starts. The Authenticated Garbling protocol would not be able to prevent this attack.

To prevent the above scenario, instead of directly using q as the output, it is used as the selector bit of two multiplexers to select from two pairs of random M bit strings. This constitutes the *extended* authentication function,

$$\{S_{vq},\ S_{pq}\} = \mathcal{F}_{ext\_auth}(R_{ref}, S_{v0}, S_{v1}, R_{PUF}, S_{p0}, S_{p1}, T). \tag{3}$$

The steps involved in this function is enclosed within the dotted (red) box in Figure 1. This extended authentication function is computed through the Authenticated Garbling protocol. Along with R_{ref} and R_{PUF}, the inputs to the extended authentication function from $\mathcal{V}$ is a pair of M-bit random nonces, S_{v0} and S_{v1}. Similarly, the inputs from $\mathcal{P}$ is a pair of M-bit random nonces, S_{p0} and S_{p1}. The 1-bit output q of the comparator is used as the selector bit of the two multiplexers with inputs $\{S_{v0}, S_{v1}\}$ and $\{S_{p0}, S_{p1}\}$. The outputs of the multiplexers S_{vq} and S_{pq} are revealed to $\mathcal{V}$ and $\mathcal{P}$, respectively. The final decision is made through the local comparison. Each party accepts the other party if and only if the received outputs entirely matches the S_{v1} (Verifier side) or S_{p1} (Prover side). Unlike the authentication function in Equation (2), the outputs of this extended authentication function depend on the inputs from both parties even with an incorrect PUF response. $\mathcal{A}$ will now have to correctly guess S_{v1} or S_{p1} that has a success rate of 2^{-M}.

3.3 Initialization

Similar to all PUF-based authentication protocols, we assume that the initialization is performed in a secure environment. $\mathcal{V}$ sends a set of challenges (at least one) to $\mathcal{P}$ and stores the responses sent back by $\mathcal{P}$ in her database. Note that the stored responses are considered secret owned by $\mathcal{V}$, while the challenges are public. $\mathcal{P}$ stores no secret data in its memory. $\mathcal{P}$ and $\mathcal{V}$ agree on the threshold value T that is publicly known and is stored on the non-volatile memory of both parties. In addition to these, the netlist of the authentication function required by the GC protocol is also stored on the non-volatile memory. The netlist can be generated by $\mathcal{V}$, $\mathcal{P}$, or an independent issuer. As explained in Section 2, the netlist is publicly known, independent of either $\mathcal{V}$ or $\mathcal{P}$ and only has to be generated once during the initialization phase.

3.4 Protocol for Binary Response

We now describe our protocol for PUFs that generate a binary string response. The protocol is denoted as π_{auth}. In the next section, we describe how this protocol can be extended to be compatible with the PUF response as a set of integer indices. In π_{auth}, the extended authentication function $\mathcal{F}_{ext\_auth}$ is computed through Authenticated Garbling [21]. This ensures that the inputs and outputs of this function remain private to the respective parties and is not revealed to the other party (or an eavesdropper). It also ensures the correctness of the output even if one of the parties are corrupted.

Authentication protocol π_{auth} between the verifier $\mathcal{V}$ and the prover $\mathcal{P}$:

Input: $\mathcal{V}$ inputs the N-bit reference response R_{ref}, and two M-bit random nonces S_{v0}, S_{v1}. $\mathcal{V}$ inputs the N-bit PUF response R_{PUF}, and two M-bit random nonces S_{p0}, S_{p1}. The challenge C and the threshold T is public and known by both parties.

Output: Both parties receive a one-bit output ACCEPT/REJECT indicating whether or not the other party is authenticated.

The protocol:

(i) $\mathcal{V}$ sends the challenge C to $\mathcal{P}$. If there is only one CRP, then C can be stored in a non-volatile memory of $\mathcal{P}$ and this step can be omitted.
(ii) $\mathcal{P}$ applies the challenge C to the PUF and generates the response R_{PUF}. In addition, $\mathcal{P}$ generates two random M-bit strings S_{p0} and S_{p1}.
(iii) $\mathcal{V}$ generates two random M-bit strings S_{v0} and S_{v1}.
(iv) $\mathcal{V}$ and $\mathcal{P}$ perform the Authenticated Garbling protocol on the extended authentication function $\mathcal{F}_{ext\_auth}$ (Figure 1) with $\mathcal{V}$ as the garbler and $\mathcal{P}$ as the evaluator. The inputs to the authentication function from $\mathcal{V}$ are the N-bit reference PUF response R_{ref} and the two M-bit random nonces S_{v0} and S_{v1}. Similarly, the inputs from $\mathcal{P}$ are the N-bit PUF response R_{PUF} and two M-bit random nonces S_{p0} and S_{p1}.
(v) $\mathcal{P}$ sends his share of S_{vq} to $\mathcal{V}$ so that $\mathcal{V}$ can XOR it with her share and learn the actual value.
(vi) Similarly, $\mathcal{V}$ sends her share of S_{pq} to $\mathcal{P}$ so that $\mathcal{P}$ can XOR it with his share and learn the actual value.
(vii) $\mathcal{V}$ locally compares S_{vq} with S_{v0} and S_{v1}. If $S_{vq} = S_{v1}$, then $\mathcal{V}$ accepts $\mathcal{P}$. Otherwise, $\mathcal{V}$ rejects $\mathcal{P}$ and aborts.
(viii) $\mathcal{P}$ locally compares S_{pq} with S_{p0} and S_{p1}. If $S_{pq} = S_{p1}$, then $\mathcal{P}$ accepts $\mathcal{V}$. Otherwise, $\mathcal{P}$ rejects $\mathcal{V}$ and aborts.

3.5 Extension for Integer Response

As we discussed in Section 2, for the DRAM PUF we need to translate the authentication based on Jaccard similarity to the authentication based on Hamming distance. For this purpose, we utilize LSH [22, 23].

LSH is a family of functions that map the input domain to the output domain (hash) with the following condition: *The probability that hashes of two inputs are equal (collision) is higher for similar inputs than non-similar ones.* The similarity between inputs can be quantified using different similarity metrics such as Jaccard or Cosine. More formally, if the collision probability $Pr_{\mathcal{H}}(h(x) = h(y))$ for a hash family $\mathcal{H}$ is a monotonically increasing function of the similarity, $Sim(x, y)$, the hash family $\mathcal{H}$ is a valid LSH family

$$Pr_{\mathcal{H}}(h(x) = h(y)) = f(Sim(x, y)), \tag{4}$$

where $f(.)$ is a monotonically increasing function. To be consistent with Reference [12], which presents the most efficient DRAM PUF to date, we use the Jaccard similarity index $\mathcal{J}$ [37] in the authentication method presented in this article. The Jaccard similarity between two given sets x, $y \subseteq \Omega = \{1, 2, \ldots, |\Omega|\}$ is defined as

$$\mathcal{J} = \frac{|x \cap y|}{|x \cup y|}. \tag{5}$$

Minwise hashing (MinHash) [38] is the LSH that preserves the Jaccard similarity and is defined as follows:

$$h_{\pi}^{min}(x) = \min(\pi(x)), \tag{6}$$

where $\pi : \Omega \rightarrow \Omega$ is a random permutation applied to the given set x. The hash is the minimum value of the permuted set. For example, given the set $x = \{1, 2, 5\} \subset \Omega = \{1, 2, 3, 4, 5\}$ and the random permutation

$$\pi : 1 \rightarrow 5, 2 \rightarrow 3, 3 \rightarrow 1, 4 \rightarrow 2, 5 \rightarrow 4,$$

set x is mapped to $\pi(x) = \{5, 3, 4\} = \{3, 4, 5\}$, hence $h_\pi^{min}(x) = 3$. For any two sets x and y, by an elementary probability argument (see Reference [39]) it can be shown that

$$Pr\{h_\pi^{min}(x) = h_\pi^{min}(y)\} = \frac{|x \cap y|}{|x \cup y|} = \mathcal{J}. \tag{7}$$

MinHash is a valid LSH since the function $f(.)$ in Equation (4) is equal to $f(\alpha) = \alpha$ that is a monotonically increasing function. Please note that MinHash maps an input set to an integer value $h_\pi^{min}(x) : S \subseteq \Omega \mapsto i \in \{1, 2, \ldots, |\Omega|\}$. However, from the computation and storage consumption perspective, it is preferable to map the output integer to only a 1-bit output. This can be realized by the universal hash functions $h_{univ} : \mathbb{N} \mapsto \{0, 1\}$. One of the popular approaches is to take only the least significant bit of MinHash [40]. Therefore, 1-bit MinHash can be realized as $h_\pi^{min,1bit}(x) = h_{univ}(h_\pi^{min}(x)) = h_\pi^{min}(x) \bmod 2$. To identify whether $h_\pi^{min,1bit}(x)$ is a valid LSH or not, we need to compute the collision probability. The probability that two sets x and y have equal MinHash values is $\mathcal{J}$ (Equation (7)) in which case $h_\pi^{min,1bit}(x) = h_\pi^{min,1bit}(y)$. If $h_\pi^{min}(x) \neq h_\pi^{min}(y)$ (with probability $1 - \mathcal{J}$), then there is a 50% chance that 1-bit hashes would collide (symmetry between outcome events). As a result

$$Pr\left\{h_\pi^{min,1bit}(x) = h_\pi^{min,1bit}(y)\right\} = \mathcal{J} \times 1 + (1 - \mathcal{J}) \times \frac{1}{2} = \frac{\mathcal{J} + 1}{2}. \tag{8}$$

Consequently, 1-bit MinHash is also a valid LSH. One can repeat the computation ℓ times with ℓ different random permutations to create an ℓ-bit LSH embedding. We denote the ℓ-bit LSH embedding of a set x as $LSH_{min}^{\ell}(x)$. Given two sets x and y with similarity of $Sim(x, y) = \mathcal{J}_0$, the number of bit-matches between $LSH_{min}^{\ell}(x)$ and $LSH_{min}^{\ell}(y)$ is $\frac{\mathcal{J}_0+1}{2} \times \ell$ on average. The uncertainty comes from the fact that LSH is a probabilistic embedding. More precisely, $E[NumBitMatch(LSH_{min}^{\ell}(x), LSH_{min}^{\ell}(y))] = \frac{\mathcal{J}_0+1}{2} \times \ell$, where $E[.]$ is the expected value of a random variable. One can express this formula using HD,

$$E_{HD}^{\ell}(x, y) = E\Big[HD(LSH_{min}^{\ell}(x), LSH_{min}^{\ell}(y))\Big] = \frac{1 - \mathcal{J}_0}{2} \times \ell. \tag{9}$$

For instance, if $\mathcal{J}_0 = 0.9$, then the Hamming distance between the 64-bit LSH embeddings of x and y ($E_{HD}^{64}(x, y)$) is 3.2. Therefore, if the Jaccard similarity of 0.9 or higher is accepted as the verified response, then one can similarly verify the hash of a response if it passes the HD threshold of 3.

4 SECURITY ANALYSIS

We analyze the security of the proposed scheme in two stages. First, we analyze the security of the authentication function (Section 3.2) to show that an adversary is not able to learn the PUF response or pass the authentication with an incorrect input in polynomial (in terms of the number of response bits) number of attempts. Then we analyze the security of the protocol (Section 3.4) to prove that it does not leak any information other than the function output. Note that the steps to extend the protocol to support integer response (Section 3.5) is performed locally and therefore do not affect the security. Moreover, since the protocol is symmetric, we only analyze the security for the scenario where the adversary $\mathcal{A}$ is posing as the prover $\mathcal{P}$ to a legitimate verifier $\mathcal{V}$.

4.1 Security of the Authentication Function

The security of the Hamming distance–based schemes relies on the fact that the Hamming distance sums up the difference between the bits of the two input binary strings and thus holds no information about individual bits. While this is true for a single execution, in the case of multiple executions, it is possible to deduce information about individual bits of one input by adaptively

varying the other input and observing the changes in the resulting Hamming distance. Any authentication scheme based on the Hamming distance should ensure that the authentication keys cannot be learned in a polynomial (in terms of the number of bits in the keys) number of executions.

We start with a more generic version of the authentication function,

$$Q = \mathcal{F}^{\mu}_{gen\_auth}(R_{ref}, R_{PUF}, T), \tag{10}$$

where Q is the left most μ bits ($1 \leq \mu \leq log(N) + 1$)[2] of $HD(R_{ref}, R_{PUF}) - T$. The authentication function $\mathcal{F}_{auth}$ provided in Equation (2) is a special case of $\mathcal{F}_{gen\_auth}$ with $\mu = 1$, since the output of the comparator is essentially the carry bit of the subtraction result.

Hamming distance has been employed in biometric authentication protocols [15, 16] that deals with noisy signatures similar to the PUF responses. However, these protocols generally output the Hamming distance itself and the comparison with the threshold is performed locally. The authentication function employed in these protocols is another special case of the one provided in Equation (10) with $\mu = log(N) + 1$ and $T = 0$. The security analysis of these protocols prove that a malicious adversary cannot obtain more information than what can be generated from the information she has (i.e., her input and received function output). However, even if the protocol is assumed to be secure by definition, we show in this section that it is possible for the adversary to deduce the input of the honest party in $O(N)$ attempts. This is why we start with the generic authentication function given in Equation (10) and show that it is secure in the special case of

$$\mathcal{F}_{auth}(R_{ref}, R_{PUF}, T) \equiv \mathcal{F}^{\mu=1}_{gen\_auth}(R_{ref}, R_{PUF}, T). \tag{11}$$

Let us define another variable,

$$\nu = log(N) + 1 - \mu. \tag{12}$$

Thus, ν is the number of bits hidden from the subtraction result of $HD(R_{ref}, R_{PUF}) - T$.

In the following analysis, we examine the effect of ν on the security of the authentication function. We assume that the adversary $\mathcal{A}$ with input R'_{PUF} is posing as the prover $\mathcal{P}$ and interacts with an ideal simulator Σ that computes the function $\mathcal{F}^{\mu}_{gen\_auth}(R_{ref}, R'_{PUF}, T)$. Σ holds R_{ref} and is secure by definition, i.e., it does not leak any information regarding R_{ref}, except the output of the function. $\mathcal{A}$ tries to deduce R_{ref} by adaptively updating her input R'_{PUF}.

- Initially, we set $\nu = 0$ and $T = 0$. To deduce R_{ref} in the minimum number of attempts, $\mathcal{A}$ should update her input with the finest resolution. Therefore, initially, she flips a single bit of her input R'_{PUF} at index n in 2 consecutive attempts. If the values of the output Q provided by Σ increases, then the nth bit of R'_{PUF} at the first attempt was equal to the nth bit of R_{ref}, since flipping that bit resulted in an increased Hamming distance. In this setting, $\mathcal{A}$ will require $2N$ attempts to deduce all N bits of R_{ref} (Thus, the protocols of References [15, 16] are not secure since they can be broken in $O(N)$ attempts).
- If ν is set to 1, i.e., the least significant bit of the Hamming distance is hidden from $\mathcal{A}$, then flipping only 1 bit will not produce any difference in the values of Q. Therefore, $\mathcal{A}$ is forced to flip 2 bits of R'_{PUF}. Since there are 4 possible combinations, she will need 4 attempts to learn 2 bits of R_{ref}, and a total of $4 \times N/2$ attempts to learn all N bits.
- We now consider an arbitrary value of ν. The Hamming distance between two W-bit binary strings is $log(W) + 1$ bits. However, if we keep the first string fixed, for only one of the 2^W possible instances of the second string the most significant bit (bit index $log(W) + 1$) of the

[2]For fractional values of $log(.)$, we take the nearest integer smaller than $log(.)$, i.e., $\lfloor log(.) \rfloor$. For simplicity, we denote $\lfloor log(.) \rfloor$ as $log(.)$ throughout the article.

Hamming distance will be set to 1 when the second string is bit-by-bit inverse of the first one. Similarly, to observe a change in the $(v+1)$-st bit of the Hamming distance, $\mathcal{A}$ will require to alter groups of 2^v bits together, and thus will need 2^{2^v} attempts. In total, she will need $2^{2^v} \times N/2^v$ attempts to learn all the bits of R_{ref}.

- Finally, for $v = log(N)$, thus $\mu = 1$ as employed in the authentication function of Equation (2), $\mathcal{A}$ will need $2^{2^{log(N)}} \times N/2^{log(N)} = 2^N$ attempts to learn all the bits of R_{ref}. Thus the number of attempts is exponential in N.

Effect of the Threshold T. So far we have assumed the threshold $T = 0$, which is ideal but not practical due to the noise present in the PUF response. For $T > 0$, it will suffice for $\mathcal{A}$ to learn any $N - T$ out of N bits correctly, irrespective of the bit indices that are correct. Since the distribution of the bits of the PUF responses is unknown to $\mathcal{A}$, the process of forming these N bit binary strings follows a binomial distribution [41] that is used to model the number of successes when sampling with replacement. Let w be the number of bits guessed correctly. Assuming each bit of R_{ref} has equal probability (= 1/2) of being 0 or 1,

$$Pr(w \geq N - T) = \frac{1}{2^N} \sum_{n=N-T}^{N} \binom{N}{n}. \tag{13}$$

Note that the nominator in Equation (13) is a polynomial in N and the denominator is an exponential in N. Therefore, the number of attempts required by $\mathcal{A}$ to authenticate with a high probability is still exponential in N. To ensure 128 bit security, we need

$$Pr(w \geq N - T) \leq 2^{-128}. \tag{14}$$

We set T as a fraction t of N, i.e., $T = \lceil tN \rceil$; $0 < t < 1$. Solving inequality 14 for $t = 0.05, 0.1, 0.15$ yields $N = 181, 237, 320$, respectively. The threshold fraction t, which denotes the maximum fraction of mismatched bits between an authentic pair of R_{PUF} and R_{ref}, is the only parameter that controls the total execution time of the protocol.

Minimum Set Size for PUFs with Integer Responses. For the case of PUFs with integer responses, along with the minimum bit-length for LSH encoding, we need to set a minimum size of the sets so that the probability of an adversary successfully guessing the response set is sufficiently low. Let S be the set guessed by $\mathcal{A}$, u be the number of elements guessed correctly, and v be the number of elements guessed incorrectly. The Jaccard similarity, $\mathcal{J}$ can be computed as

$$\mathcal{J} = \frac{|S| - v}{|S| + v}. \tag{15}$$

If the maximum number of incorrect guess is v_M, then the minimum value of Jaccard similarity, $\mathcal{J}_{min}$, is given by

$$\mathcal{J}_{min} = \frac{|S| - v_M}{|S| + v_M}. \tag{16}$$

For a given $\mathcal{J}_{min}$, we have

$$v_M = \frac{1 - \mathcal{J}_{min}}{1 + \mathcal{J}_{min}} \times |S|. \tag{17}$$

The process of choosing integers to form the sets of responses follows the multivariate hypergeometric distribution [42] that models sampling without replacement. Since each element (the

integer indices) is represented only once in the urn,

$$Pr(\mathcal{J} \geq \mathcal{J}_{min}) = Pr(v \leq v_M) = Pr(u \geq |S| - v_M) = \sum_{n=|S|-v_M}^{|S|} \frac{\binom{|\Omega|-|S|}{|S|-n} \times \binom{|S|}{n}}{\binom{|\Omega|}{|S|}}. \tag{18}$$

To ensure 128-bit security we need

$$Pr(\mathcal{J} \geq \mathcal{J}_{min}) \leq 2^{-128}. \tag{19}$$

In the DRAM PUF construction of Reference [12] each logical PUF has the size of 32KB, which gives $|\Omega| = 32 \times 8 \times 2^{10}$. Solving inequality 19 for $\mathcal{J}_{min} = 0.9$ yields $|S| \geq 10$.

4.2 Security of the Authentication Protocol

We now analyze the security of the authentication protocol, π_{auth}. We again assume that the adversary $\mathcal{A}$ with input R'_{PUF} is posing as the prover $\mathcal{P}$. $\mathcal{A}$ interacts with an ideal simulator Σ. Unlike the previous section, in this case, Σ does not communicate with the verifier $\mathcal{V}$, and therefore have no knowledge about the reference response R_{ref}. We prove the security of π_{auth} by showing that Σ is able to generate a view that is indistinguishable from the view of the adversary in a real execution of the protocol, π_{auth} with $\mathcal{V}$. This would imply that $\mathcal{A}$ learns no information about the input of $\mathcal{V}$ from the real protocol [43]. In this analysis, we utilize the hybrid model presented in Reference [44]. According to this model, if a protocol is proven to be secure in the right setting, it suffices to assume that the parties have access to a trusted party that computes that functionality. Since the Authenticated Garbling protocol of Reference [21] is proven to be secure in the malicious model, we assume that the parties have access to a trusted party that computes this functionality. The ideal simulator Σ works as follows:

(i) Σ sends the challenge, C to $\mathcal{A}$. Since C is considered public, Σ does not need communication with $\mathcal{V}$ to learn it.
(ii) Σ receives the the response R'_{PUF}, two random M-bit strings S_{p0} and S_{p1} from $\mathcal{A}$.
(iii) Σ generates a N-bit random string R'_{ref} and two random M-bit strings S_{v0} and S_{v1}.
(iv) Σ performs the Authenticated Garbling protocol on the extended authentication function $\mathcal{F}_{ext\_auth}$ through a trusted party. The inputs to the extended authentication function from Σ is the N-bit random string R'_{ref} and two the M-bit random nonces, S_{v0} and S_{v1}.
(v) Σ receives $\mathcal{A}$'s share of S_{vq}.
(vi) Σ sends its share of S_{pq} to $\mathcal{A}$.

Since none of R'_{ref} from Σ and R'_{PUF} from $\mathcal{A}$ are authentic, $\mathcal{A}$ observes with a high probability that the authentication has failed, according to the analysis in Section 4.1. Since her own input R'_{PUF} is not authentic, this is the expected result from her side. Therefore, she is not able to distinguish between a real execution of the protocol π_{auth} with $\mathcal{V}$ and an ideal execution by the simulator Σ, which does not communicate with $\mathcal{V}$. This proves that the π_{auth} does not leak any information about the inputs of $\mathcal{V}$.

Note that the security proof could be stronger if we assumed $\mathcal{A}$ "corrupts" $\mathcal{P}$ as opposed to "posing as" $\mathcal{P}$, in which case she would posses the actual PUF response R_{PUF}. However, R_{PUF} is just a noisy version of R_{ref}. Therefore, $\mathcal{A}$ would learn nothing new from the protocol in that case.

Table 1. The Numbers of Non-XOR Gates in the Generated Netlist for Different Values of Threshold Fraction t

t	N	T	M	Number of non-XOR gates					Size(KB)
				Hamming	Comparator	MUX	Total	Total†	
				$N-1$	$log(N)+1$	$2M$	$N+log(N)+2M$	Gen. by Yosys	
0.05	181	10	128	180	8	256	444	439	42
0.10	237	24	128	236	8	256	500	494	51
0.15	320	48	128	319	9	256	584	582	64

† The synthesis tool perform optimization on the entire circuit to reduce the number of gates.

5 IMPLEMENTATION

The primary target of the authentication scheme is to support authentication by any generic off-the-shelf intrinsic PUF that do not require dedicated hardware resources. Therefore, we implement the entire protocol in software and simulate the PUF as a black box.

5.1 Generating GC Netlist

To execute securely through the Authenticated Garbling [21] protocol (or any garbled circuit-based protocol), the extended authentication function shown in Figure 1 needs to be represented as a netlist of Boolean logic gates. Due to free-XOR [33] described in Section 2.2, optimizing the netlist for GC requires minimizing the number of non-XOR gates. We design the function in Verilog HDL and compile by Yosys Open Synthesis Suite [45] with the TinyGarble [46] circuit synthesis library for GC to generate the netlist. Even though TinyGarble is the most efficient tool to generate the GC netlist, the GC execution protocol supported by this framework is only secure in the semi-honest setting. Therefore we garble/evaluate the generated netlist through the realization of the Authenticated Garbling protocol provided in the EMP-toolkit [47]. This protocol is secure in the malicious setting as demanded by our authentication scheme.

Table 1 shows the number of non-XOR gates required by the different components of the extended authentication function for different values of threshold fraction t. The number of bits in the PUF response N and the threshold T are set according to the computations in Section 4.1. M is set to 128, the value of the security parameter k employed in recent works [21, 32] in SFE. Note that the total number of non-XOR gates in the netlist generated by Yosys is smaller than the sum of the number of non-XOR gates in all components. This is because the synthesis tools preform optimizations on the entire circuit to minimize the cost function, which in this case is the number of non-XOR gates.

We have created a parser to automatically convert the netlist from TinyGarble in the format supported by the EMP-toolkit. The size of the netlist files for different settings is shown in Table 1. The netlist needs to be generated only once and stored on the non-volatile memory of both parties.

5.2 Implementing LSH

As we discussed in Section 3.5, to create a 1-bit LSH from a set, we need to perform a random permutation on the input set. Computing an l-bit LSH, in fact, requires ℓ random permutations that accounts for the costliest part of computing the LSH embedding. However, recent advances in Minwise hashing [48–50] make it possible to extract more than one bit from a single permuted set. The general idea behind these methods is that one can partition the universal set (Ω) into N_p different pieces and perform Minwise hashing for each partition separately. In other words, once we have permuted the input set, we output the minimum value in each section as the MinHash value and *mod* 2 as its 1-bit LSH. Figure 2 illustrates this idea for a sample set of $S = \{0, 3, 7, \ldots, |\Omega| - 2\}$.

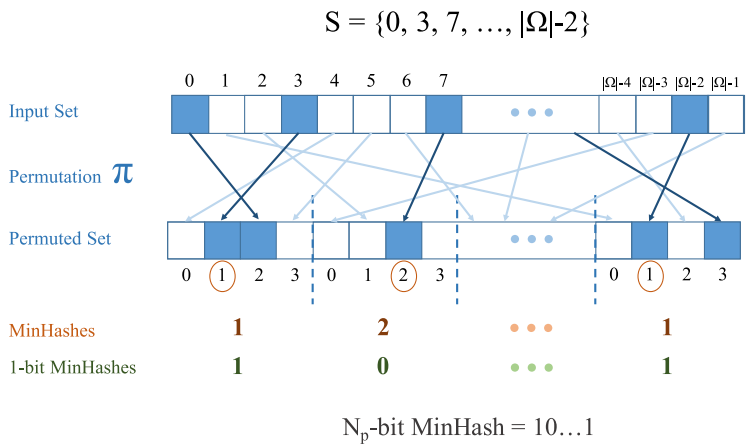

Fig. 2. Extracting multiple LSH bits from single random permutation π.

Table 2. Computational Time and Memory Utilization Complexity for Two Implementations of LSH

	Time	Memory
Pre-computing Permutations	$O(\lvert S\rvert \frac{\ell}{N_p})$ Read Ops	$O(\lvert\Omega\rvert \frac{\ell}{N_p})$
HW-based PRP	$O(\lvert S\rvert \frac{\ell}{N_p})$ AES Ops	$O(\lvert S\rvert)$

This technique enables us to extract N_p 1-bit LSH from single permutation that in turn, reduces the computation and memory usage by almost a factor of N_p. As a result, the overall number of random permutations required to generate a ℓ-bit LSH is $\lceil \frac{\ell}{N_p} \rceil$.

We present two different approaches to compute the permuted version of a set. The first approach is to pre-compute all $\lceil \frac{\ell}{N_p} \rceil$ permutations and store them in memory. To permute a set S, one needs to perform only $|S|$ memory accesses. This approach is very fast ($O(|S|)$ read operations) but it needs to compute and store all random permutations that take $O(|\Omega|)$ of memory. The second approach only requires $O(|S|)$ of memory but the computation is slower that we describe next.

Utilizing HW-Based Pseudo-Random Permutation (PRP). We propose to employ PRP such as Advanced Encryption Standard (AES) to efficiently perform the permutation. Since almost all modern processors have AES-NI in their instruction set, the fast HW-based permutation can be implemented using AES. More precisely, one of the parties randomly generates a 128-bit key (K) and announce it publicly. Each time a party wants to permute a set, he computes $\text{AES}_K(S(i))$ for $i = 1, 2, \ldots, |S|$, where $S(i)$ denotes the ith member of S. This approach does not support scenarios where $|\Omega| > 2^{128}$ but this limit is far beyond our requirements. Please note that AES only operates on 128-bit input blocks, whereas, the members of set S are represented with $\log_2(|\Omega|)$ number of bits. However, this is not an issue since instead of partitioning the universal set Ω, we partition the output range of AES. This results in same hash results since AES is a uniform random permutation (each input has equal chance to take any output value).

Overall Cost and Complexity. We summarize the complexity of computational time and memory utilization for both aforementioned approaches in Table 2.

Table 3. Timing Evaluation of the Authentication Protocol in the Two Settings for Different Values of the Threshold Fraction t

(a) Setting A: Both $\mathcal{V}$ and $\mathcal{P}$ on powerful platform.

Stage	$t = 0.05$				$t = 0.10$				$t = 0.15$			
	Verifier		Prover		Verifier		Prover		Verifier		Prover	
	cc	ms	cc	ms	cc	ms	cc	ms	cc	ms	cc	ms
Set up	1.70E+08	77.25	1.64E+08	74.37	1.67E+08	76.01	1.61E+08	73.01	1.68E+08	76.37	1.59E+08	72.24
Func. Indep.	9.96E+06	4.52	1.70E+07	7.71	9.99E+06	4.53	1.72E+07	7.81	1.12E+07	5.08	2.09E+07	9.48
Func. Dep.	2.43E+06	1.10	1.74E+06	0.79	2.12E+06	0.96	1.57E+06	0.71	2.45E+06	1.11	2.02E+06	0.92
Online	2.44E+05	0.11	2.92E+06	1.33	3.16E+05	0.14	2.86E+06	1.30	3.65E+05	0.17	3.46E+06	1.57
Total	1.83E+08	82.99	1.85E+08	84.19	1.80E+08	81.66	1.82E+08	82.83	1.82E+08	82.72	1.85E+08	84.20

(b) Setting B: $\mathcal{V}$ on powerful platform and $\mathcal{P}$ on resource-constrained platform.

Stage	$t = 0.05$				$t = 0.10$				$t = 0.15$			
	Verifier		Prover		Verifier		Prover		Verifier		Prover	
	cc	μs	cc	μs	cc	μs	cc	μs	cc	μs	cc	μs
Set up	6.82E+08	309.88	4.34E+08	295.99	6.85E+08	311.18	4.34E+08	296.17	6.89E+08	313.25	4.35E+08	296.30
Func. Indep.	1.36E+08	61.78	1.02E+08	69.33	1.99E+08	90.66	1.44E+08	98.19	2.05E+08	93.24	1.48E+08	100.99
Func. Dep.	1.84E+07	8.35	4.50E+06	3.06	1.91E+07	8.70	4.89E+06	3.33	2.07E+07	9.40	5.71E+06	3.89
Online	5.39E+06	2.45	3.78E+06	2.58	5.44E+06	2.47	3.28E+06	2.23	5.57E+06	2.53	3.85E+06	2.63
Total	8.41E+08	382.46	5.44E+08	370.97	9.09E+08	413.01	5.87E+08	399.93	9.21E+08	418.42	5.92E+08	403.81

6 EVALUATION

6.1 System Settings

We employ two platforms to evaluate the authentication protocol. Platform 1 is an Intel Core i7-2600 CPU @3.4GHz with 12GB of memory running Ubuntu 14.04. Platform 2 is an Intel Atom E3815 @1.46GHz with 2GB of memory running Lubuntu 16.04.1. The protocol is evaluated in the following two settings:

- Setting A: To assess the best case capability, we run the protocols for both $\mathcal{V}$ and $\mathcal{P}$ on Platform 1.
- Setting B: To emulate the practical scenario of a verifier with high computational power and a resource-constrained prover, we run the protocol for $\mathcal{V}$ on Platform 1 and the protocol for $\mathcal{P}$ on Platform 2.

6.2 Evaluation of the Authentication Protocol

The GC protocol is run through the realization of the Authenticated Garbling [21] in the EMP-toolkit [21]. We first evaluate the timing of the protocol involving only the extended authentication function, without LSH. The Authenticated Garbling protocol consists of three phases: (1) *Set up*: generate correlated randomness between $\mathcal{V}$ and $\mathcal{P}$ that are used during the last (online) phase for information-theoretic authentication of different values. (2) *Function-independent pre-processing*: independent of the inputs from $\mathcal{V}$ or $\mathcal{P}$ or the extended authentication function. (3) *Function-dependent pre-processing*: dependent of the authentication function, but not the inputs. (4) *Online phase*: dependent on both the extended authentication function and the inputs from $\mathcal{V}$ and $\mathcal{P}$. Tables 3(a) and 3(b) show the the number of clock cycles and time on both $\mathcal{V}$ and $\mathcal{P}$ at each stage of the protocol in the two settings for different values of the threshold fraction t. Each timing measurement is averaged over 10 instances. The total time on the prover side for $t = 0.1$ is 399ms in Setting B, which emulates the real life scenarios. To put this time into context, the interrogation

time of the DRAM PUF [12] is in the range of minutes. Therefore, the protocol execution time is negligible compared to the response generation time of the PUF.

6.3 Evaluation of Protocol for Integer Response

To evaluate the timing of our protocol while authenticating the DRAM PUF with integer responses we need to add the time that LSH computation takes. The time and memory utilization of two different approaches for hash computation is illustrated in Table 2. For the actual timing results, we use the HW-based PRP method. As we discussed, we partition the output space of AES into N_p different pieces. As can be seen from Table 2, by increasing N_p, one can reduce the overall computation time. However, there is an upper bound limit on N_p [48] that depends on $|\Omega|$ and $|S|$. In our setting, the limit is 300 hashes. Therefore, having $\ell < 300$ means that we can create ℓ-bit LSH embedding using a single permutation ($\lceil \frac{\ell}{N_p} \rceil = 1$). The overall hash computation time is therefore bounded by the number of AES invocations that is $|S|$.

The DRAM PUF presented in References [12] employed logical PUF constructions on 32KB segments of the DRAM. To simulate this construct, we take Ω as the set of integers from 0 to $(32 \times 8 \times 2^{10} - 1)$. We consider the PUF response and the reference response stored at the verifier to be sets of 300 integers that are the subset of Ω. The total number of clock cycles for LSH are 3.13E+05 on Platform 1 and 1.04E+06 on Platform 2. These translate to 92μs and 867μs, respectively.

7 RELATED WORK

Most of the work on PUF-based remote authentication involve strong PUFs [4–9] These works require regular updates of the key and therefore the PUF is required to have a large CRP database. The authors in Reference [51] surveyed the state-of-the-art authentication schemes based on strong PUFs and show that most of them lack formal security analysis and adequate security against various attacks.

There are a few works that deal with remote authentication using weak PUFs. The work in Reference [12] propose a lightweight authentication scheme based on the DRAM PUF construction. In their scheme, the verifier chooses a certain decay time, which acts as the challenge in this case, and sends it to the prover. The prover generates the corresponding response and sends it back to the verifier. The verifier then computes the Jaccard similarity between the received response and the response saved in it CRP database. If the Jaccard index is larger than a certain threshold, then the verifier accepts the prover. The drawback of this scheme is that the decay times have to be monotonically increasing for subsequent authentication sessions. Since there is only limited number of possible decay times, after a certain number of authentication sessions all the PUF responses will be exposed and the PUF will not be useful for further secure authentication sessions.

To the best of our knowledge, there have been two weak PUF-based authentication schemes that limit the exposure of the response and thus allow unlimited mutual authentication sessions: the Reverse Fuzzy Extractor presented in Reference [13] and the Trapdoor Computational Fuzzy Extractors presented in Reference [14].

Fuzzy Extractor [24] is used to correct noisy PUF data with the help of helper data that are generated during the enrollment phase. The main idea in Reference [13] is moving the task of reconstruction of the PUF response from the prover to the verifier. In their reverse fuzzy extractor, the helper data are generated at the prover every time the PUF is interrogated as opposed to generating it only once during enrollment phase in a regular fuzzy constructor. Since the reconstruction of the response requires more computational power than a generation of helper data, this setting improves the overall efficiency of a system with resource-constrained prover and a strong verifier. During authentication, only the helper data are transferred from the prover to the verifier. A

legitimate verifier would have the reference PUF response and would be able to regenerate the noisy PUF response generated at that session. Thus authentication is performed without communicating the PUF response. This allows unlimited use of the same response in different authentication sessions. One drawback of this scheme is that it requires a large number of bits to ensure security (1785 bits to ensure 128-bit security) that may not be available from many intrinsic PUFs. Moreover, the helper data inevitably reveal some information about the PUF response [51]. In a regular fuzzy extractor, it is generated only once, while in the reverse fuzzy extractor it is generated in every authentication session resulting in more probability of information leak.

The work in Reference [14] presents a computational fuzzy extractor that can correct $O(m)$ errors in polynomial time. Unlike Reference [13] the confidence information is not exposed in this work. As a result, it does not require a large number of response bits and can ensure 128-bit security with 128-but PUF response. A limitation common to both References [13] and [14] is that their proof of correctness must assume certain properties of the distribution of the PUF response (e.g., it can provide confidence information) that requires empirical verification. The work in Reference [14] demonstrated by experimental results that these assumptions hold for Ring Oscillator (RO)-based PUF. However, there is no guarantee that these assumptions will hold for any generic PUF design (or even for a different realization of the RO-based PUF). Especially, the intrinsic PUFs has the possibility of having a skewed distribution. Lastly, neither of these two schemes support authentication with PUFs providing integer responses, like the DRAM PUF.

Similar to the PUF responses, biometric keys tend to be noisy. Thus, Hamming distance–based authentication has been popular in the field of biometric authentication [15, 16, 52–55]. While the majority of these work adopt the honest-but-curious security model there have been a number of works that are claimed to be secure in the malicious setting [15, 16]. However, these protocol outputs the Hamming distance in plain text. The security proof of the protocols is based on the premise that a protocol is secure if it does not leak any information to one party other than what can be deduced from her input and the protocol output. However, as we have shown in this article, revealing the Hamming distance allows an adversary to learn the input of the honest party in a linear number of attempts.

One interesting work in this field is the *bin*HDOT protocol presented in Reference [56]. At the end of this protocol, one party holds a set of variables: $\{Z_l\}_{l=1\ldots L}$ and the other party holds only one of them Z_H where H is the Hamming distance between their inputs. Thus the final result is shared between them. The article suggests that this can be turned to a secure Hamming distance threshold comparison (similar to the one presented in this article) by setting $\{Z_l\} = 1$ for $i < T$ and 0 otherwise. However, while the *bin*HDOT protocol is proven to be secure in the malicious setting, this threshold comparison protocol is only secure in the honest-but-curious model. The authors in Reference [56] outline some possible strategies to achieve security in the malicious setting. However, no complete protocol has been developed. This is an example of the fact that while designing custom secure protocols may seem efficient, ensuring the security of these protocols is a daunting task. In our design, we first developed a secure authentication function and then executed it with a standard SFE protocol to prevent the possibility of security breach.

8 CONCLUSION

We have presented a novel PUF-based mutual authentication scheme that allows a verifier and a prover to authenticate each other without revealing their respective data. As a result, our scheme allows an unlimited number of authentication even by employing a weak PUF with limited CRP space. This is particularly useful for mutual authentication of IoT devices that usually have limited resources and thus work better with intrinsic PUFs. The core of our scheme is an authentication function that is computed securely through an SFE protocol secure in the malicious setting. To the

best of our knowledge, this is the first work that employs secure function evaluation techniques in remote PUF-based authentication. We have shown that our scheme is practical for both SRAM and DRAM PUFs, two of the most efficient and popular variants of intrinsic PUFs. Implementation of our scheme on an embedded processor shows its practicality in a real-life scenario.

REFERENCES

[1] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. 2010. Memory leakage-resilient encryption based on physically unclonable functions. In *Towards Hardware-Intrinsic Security*. Springer, 135–164.

[2] Sergei Petrovich Skorobogatov. 2005. *Semi-invasive Attacks: A New Approach to Hardware Security Analysis*. Technical Report. University of Cambridge.

[3] M. Rostami, F. Koushanfar, and R. Karri. 2014. A primer on hardware security: Threat models, metrics, and remedies (unpublished).

[4] Lars Kulseng, Zhen Yu, Yawen Wei, and Yong Guan. 2010. Lightweight mutual authentication and ownership transfer for RFID systems. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'10)*. IEEE.

[5] Stefan Katzenbeisser, Ünal Kocabaş, Vincent Van Der Leest, Ahmad-Reza Sadeghi, Geert-Jan Schrijen, and Christian Wachsmann. 2011. Recyclable pufs: Logically reconfigurable pufs. *J. Cryptogr. Eng.* 1, 3 (2011), 177–181.

[6] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. 2010. Enhancing RFID security and privacy by physically unclonable functions. In *Towards Hardware-Intrinsic Security*. Springer.

[7] Mehrdad Majzoobi, Masoud Rostami, Farinaz Koushanfar, Dan S. Wallach, and Srinivas Devadas. 2012. Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching. In *Proceedings of the Symposium on Security and Privacy Workshops (SPW'12)*. IEEE.

[8] Daisuke Moriyama, Shin'ichiro Matsuo, and Moti Yung. 2013. PUF-based RFID authentication secure and private under complete memory leakage. In *Proceedings of the Conference of the International Association for Cryptologic Research (IACR'13)*.

[9] Aydin Aysu, Ege Gulcan, Daisuke Moriyama, Patrick Schaumont, and Moti Yung. 2015. End-to-end design of a PUF-based privacy preserving authentication protocol. In *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES'15)*. Springer.

[10] Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. 2007. FPGA intrinsic PUFs and their use for IP protection. In *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES'07)*. Springer.

[11] Charles Herder, Meng-Day Yu, Farinaz Koushanfar, and Srinivas Devadas. 2014. Physical unclonable functions and applications: A tutorial. In *Proceedings of the IEEE* 102, 8 (2014), 1126–1141. IEEE.

[12] Wenjie Xiong, André Schaller, Nikolaos A. Anagnostopoulos, Muhammad Umair Saleem, Sebastian Gabmeyer, Stefan Katzenbeisser, and Jakub Szefer. 2016. Run-time accessible DRAM PUFs in commodity devices. In *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES'16)*. Springer.

[13] Anthony Van Herrewege, Stefan Katzenbeisser, Roel Maes, Roel Peeters, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. 2012. Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs. In *Proceedings of the International Conference on Financial Cryptography and Data Security (ICFCDS'12)*. Springer.

[14] Charles Herder, Ling Ren, Marten van Dijk, Meng-Day Yu, and Srinivas Devadas. 2017. Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions. *IEEE Trans. Depend. Secure Comput.* 14, 1 (2017), 65–82.

[15] Julien Bringer, Hervé Chabanne, and Alain Patey. 2013. Shade: Secure hamming distance computation from oblivious transfer. In *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, 164–176.

[16] Mehmet Sabir Kiraz, Ziya Alper Genç, and Suleyman Kardas. 2015. Security and efficiency analysis of the Hamming distance computation protocol based on oblivious transfer. *Secur. Commun. Networks* 8, 18 (2015), 4123–4135.

[17] A. Yao. 1986. How to generate and exchange secrets. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS'86)*.

[18] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. 2014. Non-interactive secure computation based on cut-and-choose. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 387–404.

[19] Luís TAN Brandão. 2013. Secure two-party computation with reusable bit-commitments, via a cut-and-choose with forge-and-lose technique. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 441–463.

[20] Xiao Wang, Alex J Malozemoff, and Jonathan Katz. 2017. Faster secure two-party computation in the single-execution setting. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 399–424.
[21] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. 2017. Authenticated garbling and efficient maliciously secure two-party computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 21–37.
[22] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing (STOC'98)*. 604–613.
[23] Piotr Indyk and David Woodruff. 2006. Polylogarithmic private approximations and efficient matching. In *Theory of Cryptography*. Springer, 245–264.
[24] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. 2004. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04)*. Springer.
[25] Ulrich Ruhrmair, Srinivas Devadas, and Farinaz Koushanfar. 2011. Security based on physical unclonability and disorder. In *Introduction to Hardware Security and Trust* (2011).
[26] Roel Maes, Vladimir Rozic, Ingrid Verbauwhede, Patrick Koeberl, Erik Van der Sluis, and Vincent van der Leest. 2012. Experimental evaluation of physically unclonable functions in 65 nm CMOS. In *Proceedings of the European Solid-State Circuits Conference (ESSCIRC'12)*. IEEE.
[27] Florian Kohnhäuser, André Schaller, and Stefan Katzenbeisser. 2015. PUF-based software protection for low-end embedded devices. In *Proceedings of the India-China Technology Transfer Centre (ICTTC'15)*. Springer.
[28] Geert-Jan Schrijen and Vincent van der Leest. 2012. Comparative analysis of SRAM memories used as PUF primitives. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'12)*. EDA Consortium.
[29] Christoph Keller, Frank Gurkaynak, Hubert Kaeslin, and Norbert Felber. 2014. Dynamic memory-based physically unclonable function for the generation of unique identifiers and true random numbers. In *PRoceedings of the International Symposium on Circuits and Systems (ISCAS'14)*. IEEE.
[30] Sami Rosenblatt, Srivatsan Chellappa, Alberto Cestero, Norman Robson, Toshiaki Kirihata, and Subramanian S. Iyer. 2013. A self-authenticating chip architecture using an intrinsic fingerprint of embedded DRAM. *J. Solid-State Circ.* 48, 11 (2013), 2934–2943.
[31] M. Naor and B. Pinkas. 2005. Computationally secure oblivious transfer. *J. Cryptol.* 18, 1 (2005), 1–35.
[32] M. Bellare, V. Tung Hoang, Sriram K., and P. Rogaway. 2013. Efficient garbling from a fixed-key blockcipher. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'13)*.
[33] V. Kolesnikov and T. Schneider. 2008. Improved garbled circuit: Free XOR gates and applications. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP'08)*. Springer.
[34] M. Naor, B. Pinkas, and R. Sumner. 1999. Privacy preserving auctions and mechanism design. In *Proceedings of the Conference on Electronic Commerce (CEC'99)*. ACM.
[35] S. Zahur, M. Rosulek, and D. Evans. 2014. Two Halves Make a Whole: Reducing Data Transfer in Garbled Circuits using Half Gates. *Cryptology ePrint Archive*. Retrieved from http://eprint.iacr.org/2014/756.
[36] B. Kreuter, A. Shelat, B. Mood, and K. R. B. Butler. 2013. PCF: A portable circuit format for scalable two-party secure computation. In *Proceedings of the USENIX Security Symposium (USENIX Security'13)*.
[37] Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *The Compression and Complexity of Sequences*. 21–29.
[38] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. 1998. Min-wise independent permutations. In *ACM Symposium on Theory of Computing (STOC'98)*. 327–336.
[39] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. 2000. Min-wise independent permutations. *J. Comput. System Sci.* 60, 3 (2000), 630–659.
[40] J. Lawrence Carter and Mark N. Wegman. 1977. Universal classes of hash functions. In *ACM Symposium on Theory of Computing (STOC'77)*. 106–112.
[41] George P. George P. Wadsworth and Joseph G. Bryan. 1960. *Introduction to Probability and Random Variables*. McGraw–Hill, New York.
[42] Shelby J. Haberman. 1976. *Discrete Multivariate Analysis: Theory and Practice*.
[43] Oded Goldreich. 2009. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press.
[44] Ran Canetti. 2000. Security and composition of multiparty cryptographic protocols. *J. Cryptol.* 13, 1 (2000), 143–202.
[45] Clifford Wolf. [n.d.]. Yosys Open SYnthesis Suite. Retrieved from http://www.clifford.at/yosys/.
[46] E. M. Songhori, S. U. Hussain, A. Sadeghi, T. Schneider, and F. Koushanfar. 2015. TinyGarble: Highly compressed and scalable sequential garbled circuits. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'15)*.
[47] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. 2016. EMP-toolkit: Efficient MultiParty computation toolkit. Retrieved from https://github.com/emp-toolkit.

[48] Anshumali Shrivastava. 2017. Optimal densification for fast and accurate minwise hashing. *arXiv preprint arXiv:1703.04664* (2017).

[49] Anshumali Shrivastava and Ping Li. 2014. Densifying one permutation hashing via rotation for fast near neighbor search. In *Proceedings of the International Conference on Machine Learning (ICML'14)*. 557–565.

[50] M. Sadegh Riazi, Beidi Chen, Anshumali Shrivastava, Dan Wallach, and Farinaz Koushanfar. 2016. Sub-linear privacy-preserving search with untrusted server and semi-honest parties. *arXiv preprint arXiv:1612.01835* (2016).

[51] Jeroen Delvaux, Roel Peeters, Dawu Gu, and Ingrid Verbauwhede. 2015. A survey on lightweight entity authentication with strong PUFs. *ACM Comput. Surv.* (2015).

[52] Margarita Osadchy, Benny Pinkas, Ayman Jarrous, and Boaz Moskovich. 2010. Scifi-a system for secure face identification. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP'10)*. IEEE, 239–254.

[53] Julien Bringer, Hervé Chabanne, and Bruno Kindarji. 2011. Identification with encrypted biometric data. *Secur. Commun. Netw.* 4, 5 (2011), 548–562.

[54] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. 2009. Privacy-preserving face recognition. In *Proceedings of the International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 235–253.

[55] M. Sadegh Riazi, Neeraj K. R. Dantu, L. N. Vinay Gattu, and Farinaz Koushanfar. 2016. GenMatch: Secure DNA compatibility testing. In *Proceedings of the 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST'16)*. IEEE, 248–253.

[56] Ayman Jarrous and Benny Pinkas. 2009. Secure hamming distance based computation and its applications. In *Proceedings of the International Conference on Applied Cryptography and Network Security*. Springer, 107–124.

Received July 2007; revised August 2018; accepted August 2018