Harmonizing PUFs for Forward Secure Authenticated Key Exchange with Symmetric Primitives

Harishma Boyapally[1,2], Durba Chatterjee[2], Kuheli Pratihar[2], Sayandeep Saha[3], Debdeep Mukhopadhyay[2], and Shivam Bhasin[1]

[1] Nanyang Technological University, Singapore
[2] Indian Institute of Technology Kharagpur, India
[3] Université catholique de Louvain
harishma.boyapally@ntu.edu.sg,durba.chatterjee94@gmail.com, its.kuheli96@gmail.com, sayandeep.saha@uclouvain.be, debdeep.mukhopadhyay@gmail.com,sbhasin@ntu.edu.sg

Abstract. Physically Unclonable Functions (PUFs) have been a potent choice for enabling low-cost, secure communication. However, in most applications, one party holds the PUF, and the other securely stores the challenge-response pairs (CRPs). It does not remove the need for secure storage entirely, which is one of the goals of PUFs. This paper proposes a PUF-based construction called Harmonizing PUFs (H_PUFs), allowing two independent PUFs to generate the same outcome without storing any confidential data. As an application of H_PUF construction, we present H-AKE: a low-cost authenticated key exchange protocol for resource-constrained nodes that is secure against replay and impersonation attacks. The novelty of the protocol is that it achieves *forward secrecy* without requiring to perform asymmetric group operations like elliptic curve scalar multiplications underlying traditional key-exchange techniques.

Keywords: Harmonizing PUFs · PUF-Throughput · AKE · Forward Secrecy.

1 Introduction

Secure communication is an age-old problem where two parties wish to exchange information in the presence of an eavesdropping adversary secretly. Symmetric and public-key encryption schemes enable parties to encrypt their messages, hiding them from adversaries. According to Kerckhoff's principle, although the schemes themselves are public, security is ensured by keeping only a key string secret. Such secret keys are typically stored in non-volatile device memory (NVM) in practical applications. However, with the continuous advances in technology, the adversarial models in cryptography have evolved significantly. Invasive and semi-invasive adversaries are a reality, and they can easily retrieve the secret

from device memory, as pointed out in several recent works [23,22,17]. A plausible solution to this issue is making the memory tamper-resilient, which might not be economical for small devices in IoT, CPS, and automotive sectors. Additionally, such low-end devices are the prime targets for attackers as they are easily accessible. *Physically Unclonable Functions* (PUFs) have been established as potential candidates to solve the above-mentioned secret storage problem. Based on uncontrollable manufacturing process variations, PUFs generate uniformly random bit(s) as the output response (making it suitable for cryptographic applications) when queried with an input bit-string (known as a challenge) [12,20]. The fascinating feature of a PUF is that the responses are *unpredictable* and depend solely upon the physical characteristics of the device on which it is instantiated. For the same PUFs instantiated on two different ICs, although the implemented circuit is the same, the responses are statistically independent or *unique*, thus working as a device-level fingerprint. These factors make PUF a strong candidate for on-the-fly secret generation, removing the need for highly fortified NVMs.

In PUF-based literature to design mutual authentication protocols between two PUF-enabled devices (nodes) [8,5,9,6,26], there is a requirement for a third party enabled by a server. Due to the unclonability and unique nature of PUF outputs, the server is expected to **securely** store the challenge-response pairs (CRP) corresponding to all the nodes in the network. It acts as a single point of failure and therefore is assumed to be a trusted entity with secure storage capabilities and requires to be *available* for the nodes to perform AKE, incurring an increase in latency. A survey of state-of-the-art works on PUF-based AKE protocols [10,6] motivate the requirement of building protocols that alleviate the requirement of complex crypto-primitives by rely on the ability of the nodes to compute light-weight symmetric operations. Additionally, the existing schemes achieve forward secrecy using asymmetric primitives that depend on elliptic curve discrete log, computational and decisional Diffie-Hellman problems [8,5,9,6,19,26]. If these protocol are not properly designed, there is a possibility of man-in-the-middle attacks which can be seen in Diffie-Hellman key exchange-based schemes. For a thorough analysis of the proposed protocol's forward secrecy, we model our adversary similar to the one discussed in one presented in a recent work [2], where they proposed a symmetric key based authenticated key exchange scheme. However, their scheme requires secure key storage, which we avoid using PUFs, making it more secure against key stealing from the device, since the secrets are generated on-the-fly.

This paper addresses the shortcomings associated with existing PUF-based literature. The contributions are summarized as follows:

- *Harmonizing PUFs*: We propose *Harmonizing PUFs* (H_PUF), a PUF-based construction that enables two distinct PUFs with independent output responses to eventually generate the same string. We solve the long-standing issue of asymmetry in PUF-based communication, where one party holds the PUF and stores the CRPs securely. We eliminate the need for storing the

responses securely while requiring only the challenge and so-called harmonizing vectors, both of which can be made public.

– *Forward Secure Authenticated Key Exchange*: As an application of the H_PUF construction, we propose a low-cost AKE protocol (H-AKE) with minimal assumptions, suitable for low-resource devices. Thus the protocol is best suited for node-to-node communication. It efficiently enables AKE on low-cost devices since there is no requirement to support public-key primitives contrary to traditional PUF-based AKEs. Throughout the communication, the need for a TTP is eliminated. As a trade-off, we require to store a challenge and a harmonizing vector making it a linear storage overhead. However, this information can be **publicly** stored, without any costly protections required for secure storage. Both authentication and key exchange are performed within one round of communication. The key feature of the H_PUF-based AKE is that it achieves forward secrecy by depending only on the H_PUF primitive and random oracles. To validate our proposals, we implement the H_PUF construction on FPGAs to analyse the computation overhead. We also provide the accuracy and overall communication overhead of the protocol.

The rest of the paper is organized as follows. In Section 2 we present the definition and security requirements of a H_PUF, followed by a straightforward construction. Section 3 presents a potential application of H_PUF for forward secure AKE with rigorous security evaluation and details of implementation overheads. We conclude in Section 4.

2 Harmonizing PUF (H_PUF)

In this Section, we introduce the concept of *Harmonizing PUFs* (H_PUFs) with a formal syntax and security definition along with a straightforward construction to realize H_PUFs. Eventually, we present a specific construction using a PUF primitive.

2.1 The Main Idea

A pair of PUFs $(\mathsf{P}_1, \mathsf{P}_2)$ take as input, a public challenge C and generate unpredictable and statistically independent responses (R_1, R_2) as output. On the other hand, a pair of H_PUFs $(\mathsf{H}_1, \mathsf{H}_2)$ are PUF-based constructions, that, given a challenge C and a unique pair of harmonizing vectors (V_1, V_2) generate the same outcome (R).

We define the syntax of H_PUF as follows. H_PUF is an ensemble of four algorithms.

– $\mathsf{H\_PUF.Setup}\big(1^\lambda, n(\cdot), m(\cdot), l(\cdot)\big)$: It takes as input the security parameter $\lambda \in \mathbb{N}$, polynomials $n(\cdot)$, $m(\cdot)$ and $l(\cdot)$ and outputs two functions $b_H : \{0,1\}^{l(\lambda)} \times$

$\{0,1\}^{m(\lambda)} \times \{0,1\}^{m(\lambda)} \rightarrow \{0,1\}^{l(\lambda)}$ and $f_H : \{0,1\}^{l(\lambda)} \times \{0,1\}^{m(\lambda)} \rightarrow \{0,1\}^{m(\lambda)}$ and a family of PUFs defined as $\mathcal{P} = \{\mathsf{P} : \{0,1\}^{n(\lambda)} \rightarrow \{0,1\}^{m(\lambda)}\}$.

- $\mathsf{H\_PUF.Sample}\big(i, 1^{\lambda}\big)$: Takes as input the security parameter λ and outputs a PUF instance P_i sampled from the family of PUFs $\mathcal{P}$.
- $\mathsf{H\_PUF.Comp\_HV}\big(i, j, C_{i,j}\big)$: Takes as input two identifiers i, j, the $n(\lambda)$-bit challenge $C_{i,j}$ and uniformly samples harmonizing vector $V_i \xleftarrow{\$} \{0,1\}^{m(\lambda)}$ and computes
$$V_j = b_H\big(V_i, \mathsf{P}_i(C_{i,j}), \mathsf{P}_j(C_{i,j})\big)$$
Outputs V_i, V_j.
- $\mathsf{H\_PUF.Query}\big(i, j, C_{i,j}, V_i\big)$: It takes the challenge and harmonizing vector as inputs and outputs $R = f_H\big(\mathsf{P}_i(C_{i,j}), V_i\big)$.

Correctness. The construction is secure if
$$R = f_H\big(\mathsf{P}_i(C_{i,j}), V_i\big) = f_H\Big(\mathsf{P}_j(C_{i,j}), b_H\big(V_i, \mathsf{P}_i(C_{i,j}), \mathsf{P}_j(C_{i,j})\big)\Big)$$

We define the harmonizing PUFs H_i and H_j as
$$\mathsf{H}_i(C_{i,j}, V_i) = f_H(\mathsf{P}_i(C_{i,j}), V_i) \text{ and } \mathsf{H}_j(C_{i,j}, V_j) = f_H(\mathsf{P}_j(C_{i,j}), V_j)$$

For $k \in [1, 2]$, under the assumption that PUF P_k is *unpredictable*, the knowledge of C and V_k does not reveal $\mathsf{P}_k(C)$. To rephrase, $\mathsf{P}_k(C)$ is independent of C and V_1, V_2. At the same time, the shared secret R is unpredictable by a probabilistic poly-time (PPT) adversary, even with the knowledge of C and (V_1, V_2), without physical access to P_1, P_2, H_1 or H_2. We formalize these security guarantees below.

Definition 1. *Security Properties: Let $\mathcal{P} = \{P_i : \mathcal{C} \rightarrow \mathcal{R}\}$ be a family of independent and unique PUFs. We define a pair of H_PUFs $(\mathsf{H}_i, \mathsf{H}_j)$ as a composition with the PUF-based constructions of $(\mathsf{P}_i, \mathsf{P}_j)$ (as per the syntax) that take challenge $C \in \mathcal{C}$ and the harmonizing vector pair $(V_i, V_j) \leftarrow \mathcal{V} \times \mathcal{V}$ as input (where, $\mathcal{V}$ is an arbitrary set), such that the following conditions hold:*

- *Correctness: $\mathsf{H}_i(C, V_i) = \mathsf{H}_j(C, V_j) = R$ where, $R \in \mathcal{R}$*
- *For security parameter λ, let $k \in \{i, j\}$, r be a string chosen uniformly at random, negl be a negligible function and $\mathcal{D}$ be a PPT distinguisher. Then,*
  - *given $V_i, V_j \leftarrow \mathsf{H\_PUF.Comp\_HV}\big(i, j, C\big)$ and $P_k(C)$, $\mathcal{D}$ cannot distinguish between C and a uniformly sampled string C', i.e,*
$$\big|\Pr[\mathcal{D}(C, V_k, P_k(C))] - \Pr[\mathcal{D}(C, V_k, P_k(C))]\big| \leq \mathsf{negl}(\lambda)$$
  - *given challenge C, and harmonizing vector V_k, $\mathcal{D}$ cannot distinguish the PUF output $\mathsf{P}_k(C)$ from a uniformly random string, i.e,*
$$\big|\Pr[\mathcal{D}(C, V_k, \mathsf{P}_k(C)) = 1] - \Pr[\mathcal{D}(C, V_k, r) = 1]\big| \leq \mathsf{negl}(\lambda)$$
  This property is an extension of the unpredictability property of PUFs.

- *given challenge C, and both harmonizing vectors (V_i, V_j), $\mathcal{D}$ cannot distinguish H_PUF output from a uniformly random string, i.e,*

$$\left| \Pr[\mathcal{D}(C, V_i, V_j, R) = 1] - \Pr[\mathcal{D}(C, V_i, V_j, r) = 1] \right| \leq \mathsf{negl}(\lambda)$$

A Straightforward Construction. The abstract syntax of H_PUF, can be realized in several ways. Here we begin with one of the simplest possibilities. To adopt H_PUFs for building key exchange protocols, they can be instantiated in two phases: **Setup()** and **Query()**. Let $\mathsf{P}_1, \mathsf{P}_2$ be two PUF instances with challenge and response spaces $\mathcal{C} = \{0,1\}^n$ and $\mathcal{R} = \{0,1\}^m$ respectively and $C \in \mathcal{C}$.

- **Setup $(\mathsf{P}_1, \mathsf{P}_2, C)$:** Samples a uniformly random string $V_1 \leftarrow \{0,1\}^m$. Computes V_2 as

$$V_2 = \mathsf{P}_1(C) \oplus \mathsf{P}_2(C) \oplus V_1 \tag{1}$$

  Outputs the tuple (C, V_1, V_2) publicly. Note that V_2 is also a uniformly random string.
- **Query (P_i, C, V_i):** For $i \in [1,2]$, computes the shared secret as:

$$R = \mathsf{P}_i(C) \oplus V_i$$

Note that, the algorithms H_PUF.Setup(), H_PUF.Sample() and H_PUF.Comp_HV() from the syntax are performed offline in **Setup()**) and algorithm H_PUF.Query() is performed online in **Query()**.

The correctness of this construction follows from Equation 1. Due to the unpredictability property of PUFs, any PPT adversary who has knowledge of the public information (C, V_1, V_2), cannot guess $\mathsf{P}_1(C)$, $\mathsf{P}_2(C)$ or R with more than negligible probability (as per Definition 1). Given uniformly random (V_1, V_2), adversary can only learn $\mathsf{P}_1(C) \oplus \mathsf{P}_2(C)$, but nothing about the secret R will be revealed (as $\mathsf{P}_1(C)$ and $\mathsf{P}_2(C)$ are individually unknown).

2.2 Possible Instantiations of H_PUFs

One may observe that the construction described in the last subsection does not specify the PUF being used. The H_PUF construction is general enough to be instantiated with any PUF. However, each construction's efficiency, applicability, and security depend on the underlying PUFs. It is feasible to construct a pair of H_PUFs from weak PUFs. Generating multi-bit responses is also straightforward and reliable, thanks to the properties of several known weak PUF constructions such as SRAM PUFs [14]. However, due to the limited challenge-response space of the weak PUFs, their applicability is mainly limited for key-generation purposes. On the other hand, strong PUFs, with their exponential size challenge-response space, enable the realization of diverse cryptographic applications. Without loss of generality, we mainly focus on strong PUF-based

Table 1. Comparison with state-of-the-art PUF-based AKE protocols for resource-constrained nodes. Hash, Mac, Enc, HashToPoint, ECMult, ECAdd, BPair, PRF are hashing, message authentication code, symmetric encryption/decryption, mapping to an elliptic curve point, elliptic curve point multiplication, elliptic curve point addition, Bilinear pairing, pseudorandom function operations. $\surd$ implies that the protocol is secure against replay/impersonation attacks. The security assumptions MAC, ECDLP, ECDHP, CDH, and ECCDH, are message authentication code, elliptic curve discrete log problem, elliptic curve Diffie-Hellman problem, computational Diffie-Hellman problem, and elliptic curve computation Diffie-Hellman problem, respectively.

	Computation Cost	**Messages Exchanged**	**Communication Overhead (in bits)**	**Replay**	**Impersonation**	**Forward Secrecy**	**TTP**
Aman [1] 2017	4 Hash + 12 Mac + 10 Enc	7	7872	$\surd$	$\surd$	MAC	Yes
Chatterjee [8] 2017	8 Hash + 6 HashToPoint + 4 ECMult	7	9312	Insecure	Insecure	ECDLP	Yes
Braeken [5] 2018	20 Hash + 7 ECMult + 4 ECAdd	5	1956	$\surd$	$\surd$	ECDHP, ECDLP	Yes
Chatterjee [9] 2019	14 Hash + 20 HashToPoint + 8 ECMult+16 ECAdd + 4 BPair	10	9856	$\surd$	$\surd$	ECDLP	Yes
Byun [6] 2019	6 Hash + 12 PRF + 3 MultGroup	3	1952	$\surd$	$\surd$	CDH	Yes
Li [19] 2020	12 Hash + 14 ECMult + 4 ECAdd	3	4160	$\surd$	$\surd$	ECCDH	No
Zheng [26] 2021	12 Hash + 4 Enc	13	2880	$\surd$	$\surd$	Insecure	Yes
H-AKE(Our)	**8 Hash**	**2**	**1216**	$\surd$	$\surd$	**H_PUF, Hash**	**No**

constructions and their potential applications in this paper. Looking forward, in Section 3, we leverage this property of strong PUFs, to build a forward secure authenticated key exchange protocol without depending on complex cryptographic operations. However, the ease of applicability for strong PUFs comes with a cost, as there are several usability issues associated with strong PUFs. Most importantly, they suffer from low throughput and vulnerability to model-building attacks.

Another issue that is relevant to both strong and weak PUFs is their reliability, which is often handled by using error-correcting codes [15] or Fuzzy Extractors [11]. Given these issues, it is rather evident that the straightforward construction presented in the last subsection needs to be augmented in several ways to be practically applicable. For this purpose we chose the throughput enhanced PUF construction presented in [3] for our experimental evaluation. In this work, machine learning based modelling attacks are out of scope, since the proposed protocol does not reveal the challenge-response information in the plain.

In the next section, we present an application of our H_PUF design by building an authenticated key exchange scheme between two resource-constrained devices. This scheme is forward secure without relying on computationally hard problems or public key primitives.

3 H-AKE: Authenticated Key-Exchange With Forward Secrecy

In this section, we first discuss existing and relevant PUF-based AKE protocols. Next, we present the system and threat model and the design goals. Then, we provide the details of our H_PUF-based AKE (H-AKE) protocol, followed by a detailed security analysis.

3.1 State-of-the-art PUF-based AKE Protocols

In the past two decades, PUFs have been broadly studied as hardware security primitives to generate on-the-fly keys eliminating the requirement of key storage by design. PUFs are extensively used to build authentication protocols [10,1,24], for IoT and CPS networks. Due to their lightweight nature and unpredictability property, they become good candidates to provide strong security. In literature, several PUF-based mutual authentication protocols [1,16,13,25,4] are proposed between a resource-constrained PUF-enabled node and a server. They require secure storage of secrets and to compute complex cryptographic operations on the server, making them unsuitable for AKE between two resource-constrained nodes, which is the focus of this work. A detailed survey on some recent node-to-node AKE protocols is presented in [10,6] and discussed their security against various attacks like replay, impersonation, and PUF modelling. The protocols proposed in [1,8,9] require a TTP to store the challenge response pair (CRP) information securely and assist the resource-constrained devices during the authentication, thus making them the root-of-trust. In [6], although the protocol eliminates the TTP assumption, it is not proven to be forward secure. The protocol proposed in [9] while eliminating secure CRP storage, requires the PUF-enabled devices to perform complex cryptographic operations like pairings. A 3-handshake PUF-based node-to-node AKE protocol is protocol is presented in [19], without the requirement of a TTP. However, it requires the nodes to perform 14 elliptic curve point multiplication operations. In a recent work [26], the resource-constrained devices perform only lightweight operations. However, it still requires a TTP that securely stores CRPs for authentication and exchanges 13 messages to establish the session key. Moreover, it is not forward secure.

Efficiency of H-AKE Protocol. In Table 1, we present a detailed comparison of the existing PUF-based protocols with H-AKE protocol. We can observe that H-AKE protocol performs well in terms of computation cost, messages exchanged, communication overhead, security properties, and computational assumptions to obtain forward secrecy. It requires each node to perform only four hash operations and performs mutual authentication and key exchange by exchanging two messages. The overall communication overhead is extremely low (only 1216 bits) compared to existing solutions. We obtain forward secrecy without depending on resource-intensive asymmetric key operations like elliptic curve scalar multiplications or bilinear pairings. The security of H-AKE depends only on the H_PUF primitive and hash functions.

Table 2. Frequently Used Notations in the Protocol

Symbol	Definition
Id_j	Identity of Node_j
H_j	H_PUF embedded in Node_j
nonce_i	Nonce for i-th session
C_i	Challenge for i-th session
R_i	Shared secret for i-th session
$V_{j,i}$	Harmonizing vector of Node_j for challenge C_i
$\mathcal{H}$	Collision resistant hash function
K_i	Session key for i-th session

3.2 Formal modelling and Design Goals

Table 2 presents some frequently used notations. The system model, threat model, and design goals of our H-AKE scheme are described below.

System Model:

- Our system consists of N resource-constrained nodes ($\mathsf{Node}_1, \ldots \mathsf{Node}_N$) where each node Node_i is embedded with the PUF instance $\mathsf{P}_i : \{0,1\}^n \rightarrow \{0,1\}^{2n}$.
- A pair of H_PUFs $(\mathsf{H}_1, \mathsf{H}_2)$ defined as per Definition 1, are PUF-based constructions of $(\mathsf{P}_1, \mathsf{P}_2)$. Then, for any $C_i \in \{0,1\}^n$, there exists a pair of harmonizing vectors $(V_{1,i}, V_{2,i}) \in \{0,1\}^{2n} \times \{0,1\}^{2n}$ and shared secret R such that the following condition holds:

$$\begin{aligned} \mathsf{H}_1(C_i, V_{1,i}) &= \mathsf{P}_1(C_i) \oplus V_{1,i} \\ &= R \\ &= \mathsf{P}_2(C_i) \oplus V_{2,i} = \mathsf{H}_2(C_i, V_{2,i}) \end{aligned} \tag{2}$$

- Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^{4n}$ be a collision-resistant one-way hash function modelled as a random oracle.
- The PUF-enabled resource-constrained nodes are capable of performing hash and XOR operations.

Threat Model:

We assume that the initial one-time setup is performed in a trusted environment. Any information exchanged in this phase is not accessible to any third-party/adversary. We assume that the adversary can control the communication channel actively or passively. It can collect publicly stored challenge and harmonizing vector pairs. The adversary cannot mount a physical attack on the device without tampering with the H_PUF circuit. It can mount the following attacks:

– *Replay and Impersonation*: We assume an active eavesdropping adversary that may attempt to replay messages from a previous session or impersonate a legitimate node without possessing either H_PUFs.

– *Perfect Forward Secrecy*: We assume an adversary that learns the H_PUF's response in j-th session, using which it tries to learn the session keys of any i-th session for $i < j$. For this work, we do not assume the adversary gains physical access to either H_PUF device, to learn the response j-th session and it is out of scope for this work.

Design Goals: The proposed H-AKE has the following goals:

– *Correctness:* Any two legitimate nodes that generate a harmonizing vector pair during the setup phase should succeed in authentication and secure session-key establishment without needing to store any CRP or reliance on a TTP.

– *Security:*
  1. Knowledge of past session keys should not reveal the current session key.
  2. Security against replay and impersonation attacks.
  3. The protocol should be forward secure.

3.3 Proposed Protocol

The H_PUF-based AKE protocol (H-AKE) consists of the following two phases:

Setup Phase: This is a one-time operation performed in a trusted environment. In this stage, for challenge C_1, Node_1 and Node_2 (with identifiers Id_1 and Id_2 respectively), generate their corresponding harmonizing vectors $(V_{1,1}, V_{2,1})$ to compute shared secret as $R_1 = \mathsf{H}_1(C_1, V_{1,1}) = \mathsf{H}_2(C_1, V_{2,1})$. They proceed as follows:

– Node_1: Randomly samples a challenge $C_1 \leftarrow \{0,1\}^n$ and corresponding harmonizing vector $V_{1,1} \leftarrow \{0,1\}^{2n}$. Computes the shared secret as $R_1 = \mathsf{H}_1(C_1, V_{1,1}) = \mathsf{P}_1(C_1) \oplus V_{1,1}$. Stores the tuple $\langle C_1, V_{1,1} \rangle$ locally and sends $\langle C_1, R_1 \rangle$ to Node_2.
– Node_2: Computes harmonizing vector as $V_{2,1} = \mathsf{H}_2(C_1, R_1) = \mathsf{P}_2(C_1) \oplus R_1$ (refer to Equation 2). Stores the tuple $\langle C_1, V_{2,1} \rangle$ locally.

This step is exactly the same as the setup phase of H_PUF (refer to Section 2). As per our threat model, the communication between the two nodes is secure against any adversary. Therefore, the shared secret is not revealed. The nodes do not require costly secure storage capabilities; rather, the challenge and harmonizing vector can be stored locally on the device or in a public cloud. The nodes participate in forward-secure AKE using this public information while preserving security against replay and impersonation attacks. Below, we present the details of H-AKE protocol for the 1-st session identified by a nonce nonce_1. Note, that the nonce in AKE schemes are similar to time stamps, where each nonce uniquely identifies a session.

Online Phase: In this phase, both the nodes mutually authenticate each other and generate a secure session key using the public challenge and harmonizing vectors. Without loss of generality, let us assume that Node_1 initiates the AKE phase with Node_2 for the 1-st session with challenge C_1.

– Node_1: In the 1-st session with nonce nonce_1, it generates

$$R_1 = \mathsf{H}_1(C_1, V_{1,1})$$

and computes

$$X_1 = \mathcal{H}(R_1||\mathsf{nonce}_1||\mathsf{Id}_1||\mathsf{Id}_2)$$

It sends the tuple $\langle X_1, \mathsf{nonce}_1, \mathsf{Id}_1, \mathsf{Id}_2\rangle$ to Node_2.

– Node_2: On receiving $\langle X_1, \mathsf{nonce}_1, \mathsf{Id}_1, \mathsf{Id}_2\rangle$, it generates

$$R_1 = \mathsf{H}_1(C_1, V_{2,1})$$

- It checks if

$$X_1 = \mathcal{H}(R_1||\mathsf{nonce}_1||\mathsf{Id}_1||\mathsf{Id}_2)$$

to verify that Node_1 is authentic.

  * Samples a uniformly random string $r_1 \xleftarrow{R} \{0,1\}^{2n}$, new challenge $C_2 \xleftarrow{R} \{0,1\}^n$ and corresponding harmonizing vector $V_{2,2} \xleftarrow{R} \{0,1\}^{2n}$ for the next (2-nd) session.
  * Generates the new shared secret R_2 for the 2-nd session as

$$R_2 = \mathsf{H}_1(C_2, V_{2,2})$$

and computes

$$Y_1 = \mathcal{H}(R_1||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_1) \oplus (R_2||r_1)$$
$$Z_1 = \mathcal{H}(C_2||R_2||r_1||\mathsf{nonce}_1||\mathsf{Id}_1||\mathsf{Id}_2)$$

  * Sends the tuple $\langle Y_1, Z_1, C_2, \mathsf{Id}_1, \mathsf{Id}_2\rangle$ to Node_1 and establishes the session key as

$$K_1 = \mathcal{H}(R_1||r_1||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_1)$$

- Else, it rejects Node_1.

– Node_1: On receiving $\langle Y_1, Z_1, C_2, \mathsf{Id}_1, \mathsf{Id}_2\rangle$, it computes

$$(R_2'||r_1') = \mathcal{H}(R_1||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_1) \oplus Y_1$$

- It checks if

$$Z_1 = \mathcal{H}(C_2||R_2'||r_1'||\mathsf{nonce}_1||\mathsf{Id}_1||\mathsf{Id}_2)$$

to verify that Node_2 is authentic and integrity of data is maintained.

* Computes harmonizing vector $V_{1,2}$ corresponding to challenge C_2 for the 2-nd session as

$$V_{1,2} = \mathsf{H}_1(C_2, R_2') = \mathsf{P}_1(C_2) \oplus R_2'$$

and establishes the session key as

$$K_1 = \mathcal{H}(R_1||r_1||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_1)$$

- Else, it rejects Node_2.

After session key establishment, both parties erase the shared secrets from device memory, thus concluding the 1-st session. In the protocol, the nodes start with the shared secret R_1, which they use for authentication and session key generation. During this process, they secretly agree on a new challenge (C_2), shared secret (R_2) and harmonizing vectors ($V_{1,2}, V_{2,2}$) for the 2-nd session. It can be noted that the harmonizing vectors are not sent in plaintext during the communication, yet they do not require secure storage on the node devices. This process updates the long-term secret, which here is the shared secret generated by H_PUFs in each session while performing an authenticated key exchange. Thus a rogue node cannot force a legitimate node to update its challenge and harmonizing vectors, without passing the authentication. At the same time, an adversary gaining knowledge of R_2 in 2-nd session cannot learn the session key K_1 or R_1 of the previous session, without breaking the collision-resistance property of hash function ($\mathcal{H}$) or without violating the property of random oracle.

We illustrate H-AKE for i-th session with nonce nonce_i in Figure 1. The session key is established in one round, and each node performs 1 XOR and four hash operations. Note that traditional, as well as PUF-based AKE protocols, depend on elliptic curve discrete log problem or decisional/computational Diffie-Hellman assumptions to achieve forward secrecy (as shown in Table 1). However, H-AKE protocol is forward secure by depending on the H_PUF and collision-resistant hash functions modelled as random oracles.

3.4 Security Analysis

In this section, we formally prove the security of H-AKE against the security definition. The proof uses the well-known Canetti-Krawczyk [7] and extended Canetti-Krawczyk [18] models for AKE. Then we discuss robustness against replay and impersonation attacks. Finally, we show that H-AKE protocol is forward secure.

Security Definition for H_AKE. Our security definition involves m parties $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_m$ and an adversary $\mathcal{A}$. The parties and $\mathcal{A}$ are PPT algorithms. Each party $\mathcal{P}_i$ for $i \in [1, n]$ has its own H_PUF instance H_i, a challenge and harmonizing vector pair corresponding to the remaining parties. We assume that all the challenges are unique. All the uncorrupted parties follow the protocol and respond accordingly.

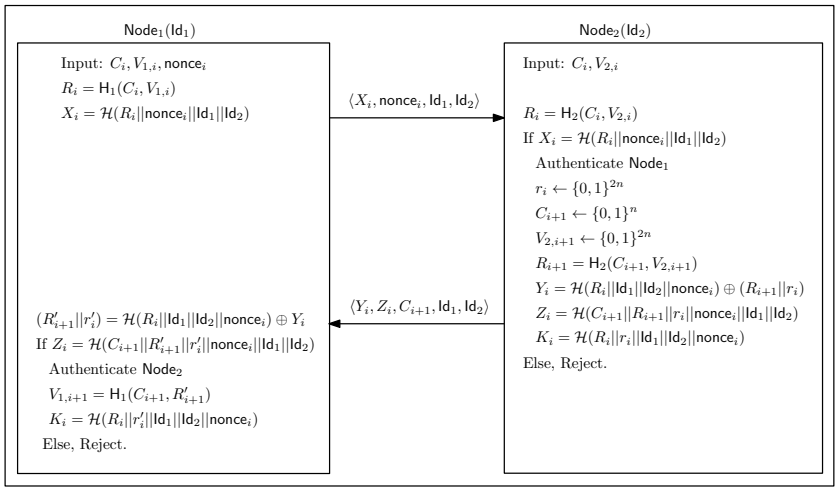

Fig. 1. H_PUF-based Forward Secure Authenticated Key Exchange

$\mathcal{A}$ has complete control over the network, it can decide to drop, modify, store and replay the messages. The adversary $\mathcal{A}$ can activate different $(\mathcal{P}_i, \mathcal{P}_j)$ pairs of its own choice to run multiple instances of the protocol. We define a protocol instance between $(\mathcal{P}_i, \mathcal{P}_j)$ as $\mathsf{sid} = \langle i, j, C_{i,j}, V_i, V_j\rangle$. For a pair of parties, only one protocol instance is initiated by the adversary and a new instance is initiated only if the past instance is **finished** either in **accept** or **reject**.

We define the following queries:

- **send**(i, m): $\mathcal{A}$ can use this query to send message m to the party $\mathcal{P}_i$. If the adversary sends the message $\langle \top, j\rangle$, then P_i initiates the protocol instance with the first message and acts as an *initiator*, while party $\mathcal{P}_j$ acts as a *responder*.
- **reveal**(i, j): $\mathcal{P}_i$ reveals the session key to $\mathcal{A}$ it established with P_j if the protocol instance is accepted and finished. Else, it will send $\emptyset$.
- **corrupt**(i, j): On receiving this query P_i responds with H_PUF response related to party P_j. Then the challenge and corresponding responses are marked as corrupted for both parties.
- **test**(i, j): This query is made only once throughout the game. The query is handled by $\mathcal{P}_i$ as follows: If all the protocol instances are accepted and finished, then it responds with a failure symbol $\perp$. Else, it flips a fair coin b, samples a random element $k_0 \xleftarrow{\$} \mathcal{K}$, where $\mathcal{K}$ is the range of session key space, sets $k_1 = k$, where k is the real session key between $\mathcal{P}_i$ and $\mathcal{P}_j$, and returns k_b.

We define our security game for AKE protocol as follows:

Security Games for H_AKE: In this game, the challenger $\mathcal{C}$ sets up a protocol environment with m parties $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_m$. If party $\mathcal{P}_j$, receives the

message $m = \langle \top, j \rangle$, it initiates the protocol (if there does not exist a protocol instance between $\mathcal{P}_i$ and $\mathcal{P}_j$ that has not **finished**), and if it receives a protocol message it responds to it according to the protocol specification. $\mathcal{C}$ performs the setup phase, which includes generating the challenge and harmonizing vector pairs for all pairs of parties.

AKE_Game: $\mathcal{A}$ is allowed to ask $q-1$ **reveal** queries and one **test** query to a protocol instance where, neither the *initiator* nor the *responder* is corrupted, under the following conditions

- if $\mathcal{P}_i$ is an initiator, then **reveal** query cannot be made on any responder party receiving the message from $\mathcal{P}_i$, and all the responders must be **uncorrupted**.
- if $\mathcal{P}_i$ is the responder, the **reveal** query cannot be made on the initiator party and it should be **uncorrupted**.

At the end of the game, $\mathcal{A}$ outputs a bit b' and wins the game if $b = b'$ where b is the bit chosen during the **test** query.

Replay_Game: $\mathcal{A}$ may make upto q **send** and **corrupt** queries. It wins the game in a fresh session if there are at least two responders that accept the same message, if there is a responder oracle that accepts a message from an uncorrupted sender that has not been sent by the latter or if there is an initiator party that accepts without having a corresponding responder from which the message originated.

ForwardSecrecy_Game: If the adversary $\mathcal{A}$ issues the query **corrupt**(i, j) either during a protocol instance or after the protocol instance is finished, then it wins the game if it can a) distinguish between a random string chosen from the session key space and the actual session key or b) distinguish between random string from $\{0,1\}^{2n}$ and corresponding shared secret, corresponding to any accepted and finished protocol instances in the past (other than the instance, where $\mathcal{A}$ made the **corrupt** query), under the conditions that $\mathcal{A}$ did not make **reveal** or **corrupt** query on that instance before it was accepted and finished.

Definition 2. *The H-AKE protocol described in Section 3.3 is said to be secure in the AKE_Game for i ($\in \mathbb{N}$)-th session and security parameter λ, if any PPT adversary $\mathcal{A}$, cannot distinguish between the actual session key $K_i = \mathcal{H}(R_i||r_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i)$ from uniformly random string $K' \xleftarrow{R} \{0,1\}^{4n}$, i.e,*

$$\Big| \Pr[\mathcal{A}(\mathcal{H}(R_i||r_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i) = 1] - \Pr[\mathcal{A}(K') = 1] \Big| \leq \mathsf{negl}(\lambda)$$

Theorem 1. *H-AKE is secure as per Definition 2, under the assumptions that a) (H_1, H_2) that form an H_PUF pair as per Definition 1 are independent and unpredictable and b) $\mathcal{H}$ is a collision-resistant one-way hash function modelled as a random oracle.*

Proof. Let $\mathcal{A}$ be a PPT adversary against the H_PUF based AKE protocol for target session with nonce nonce_i and challenge C_i. Note that $\mathcal{A}$ is restricted

from creating two different sessions with the same nonce. $\mathcal{A}$ can distinguish the session key K_i from a uniformly random string only in two ways:

- *Key Replication Attack:* By forcing to establish another session with the same session key K_i as the target session, without querying the random oracle $\mathcal{H}$ on $(R_i||r_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i)$.
- *Forging Attack:* By eavesdropping on the communication to gain enough information to query the random oracle $\mathcal{H}$ on $(R_i||r_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i)$.

Since, two sessions cannot have the same nonce, $\mathcal{A}$ can force to establish two sessions with the same session key only by forcing collision on the random oracle $\mathcal{H}$, which is possible only with negligible probability. So, $\mathcal{A}$ must perform a forging attack. Under the assumption that H_1 and H_2 are defined as per Definition 1, $\mathcal{A}$ can distinguish between R_i (shared secret) from a uniformly random string with no more than negligible advantage. Further, under the assumption that r_i is sampled uniformly at random, $\mathcal{A}$ can distinguish Y_i from $\mathcal{H}(R_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i) \oplus S$ for some random $S \xleftarrow{R} \{0,1\}^{4n}$ only with negligible advantage. Since, we model the one-way hash function $\mathcal{H}$ as a random oracle, $\mathcal{A}$ cannot learn R_i, R_{i+1} and r_i from X_i and Z_i with more than negligible probability. Finally, $\mathcal{A}$ can distinguish between Y_i and $S' \oplus (R_{i+1}||r_1)$ for some random $S' \xleftarrow{R} \{0,1\}^{4n}$ only with negligible advantage. So the adversary cannot mount a forging attack.

Therefore, $\mathcal{A}$ cannot distinguish between K_i and a uniformly random string with more than a negligible advantage, proving that the protocol is secure as per Definition 2.

Below, we prove the protocol's security against replay and impersonation attacks. Next, we prove that it is forward secure.

Replay Attacks: To establish the session key in j-th session, Node_1 sends $\langle X_j, \mathsf{nonce}_j, \mathsf{Id}_1, \mathsf{Id}_2\rangle$ and Node_2 responds with $\langle Y_j, Z_j, C_{j+1}, \mathsf{Id}_1, \mathsf{Id}_2\rangle$. An eavesdropping PPT adversary $\mathcal{A}$ in i-th ($i > j$) session can mount a replay attack in three ways, either by a) replaying Node_1's message, Node_2's message from j-th session or by replaying either's message in j-th session to another node, say Node_3 in i-th session.

Case a) $\mathcal{A}$ sends $\langle X_i, \mathsf{nonce}_i\rangle = \langle X_j, \mathsf{nonce}_j\rangle$ in i-th session. Node_2 accepts $\mathcal{A}$ as an authentic node if

$$\mathcal{H}(R_i||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2) = X_i = X_j = \mathcal{H}(R_j||\mathsf{nonce}_j||\mathsf{Id}_1||\mathsf{Id}_2)$$

As per the security definition, $\mathcal{A}$ is restricted from establishing two sessions with the same nonce. So, $\mathcal{A}$ can replay only by forcing collision on the random oracle $\mathcal{H}$, the probability of which is only negligible.

Case b) Node_1 accepts $\mathcal{A}$ as an authentic node in i-th session if

$$Z_i = \mathcal{H}\big(C_{i+1}||T||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2\big)$$

where,

$$T = \big(R'_{i+1}||r'_i\big) = \mathcal{H}\big(R_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i\big) \oplus Y_i$$

$\mathcal{A}$ replays $\langle Y_j, Z_j, C_{j+1}\rangle$ from j-th in i-th session. Now, by replacing Z_i with Z_j and Y_i with Y_j, we have

$$Z_i = Z_j = \mathcal{H}(C_{j+1}||R_{j+1}||r_j||\mathsf{nonce}_j||\mathsf{Id}_1||\mathsf{Id}_2) \tag{3}$$

and

$$\begin{aligned}
&\mathcal{H}\big(C_{i+1}||T||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2\big)\\
&\quad= \mathcal{H}\Big(C_{i+1}||\mathcal{H}\big(R_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i\big) \oplus Y_j||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2\Big)\\
&\quad= \mathcal{H}\Big(C_{j+1}\Big\|\\
&\qquad\qquad \mathcal{H}\big(R_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i\big) \oplus \mathcal{H}(R_j||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_j) \oplus \Big\|(R_{j+1}||r_j)\\
&\qquad\qquad \Big\|\mathsf{nonce}_i\Big\|\mathsf{Id}_1\Big\|\mathsf{Id}_2\Big)
\end{aligned} \tag{4}$$

The adversary wins in the Replay_Game only by forcing Equations. 3 and 3 to be equal. The probability of this is same as that of finding collisions in the hash function $\mathcal{H}$. Under the assumption that $\mathcal{H}$ is collision-resistant, the adversary wins only with negligible probability

Case c) For simplicity let us assume that $\mathcal{A}$ replays the message from Node_1 as an initiator node to Node_2 in j-th session as the initiator message to Node_3 in its i-th session. Let us also assume that adversary forces both the sessions to have the same nonce (i.e, $\mathsf{nonce}_i = \mathsf{nonce}_j$). However from Case a) and Case b), we can notice that even if the nonces are same, since the identifiers are different for each node $\mathcal{A}$ succeeds only by finding collisions, since the identifier of the nodes are unique. Thus the probability of it winning is still negligible.

Therefore, the protocol is secure against replay attacks.

Impersonation Attacks: In this attack, we consider two cases where a PPT adversary $\mathcal{A}$ eavesdrops on the communication channel to learn enough information to impersonate either Node_1 or Node_2 in the i-th session. From Theorem 1, we can say that for any $j \neq i$ a PPT adversary cannot learn R_j, R_{j+1} or r_j by eavesdropping on the j-th session.

To impersonate Node_1 in i-th session, $\mathcal{A}$ generates $\langle X, \mathsf{nonce}\rangle$ such that $X = \mathcal{H}(R_i||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2)$ without querying the random oracle $\mathcal{H}$ on $(R_i||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2)$. This amounts to a collision on $\mathcal{H}$, violating the collision-resistance property.

To impersonate Node_2 in i-th session, $\mathcal{A}$ generates $\langle Y, Z\rangle$ such that

$$Z = \mathcal{H}\Big(\big(\mathcal{H}(C_i||R_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i) \oplus Y\big)||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2\Big)$$

without querying the random oracle $\mathcal{H}$ on $(R_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i)$. Again, this amounts to a collision on $\mathcal{H}$, violating the collision-resistance property.

Therefore, the adversary $\mathcal{A}$ succeeds to impersonates Node_1 or Node_2 only with negligible probability.

Forward Secrecy: Let there be a PPT adversary $\mathcal{A}$ with the knowledge of shared secret (R_{i+1}) corresponding to $(i+1)$-th (current) session. From the past sessions, $\mathcal{A}$ also gained the knowledge of $(X_j, Y_j, Z_j, \mathsf{nonce}_j)$ for $j \in [1, i]$. For i-th session, $\mathcal{A}$ can trivially learn the first $2n$ bits of $\mathcal{H}(R_i||\mathsf{ID}_1||\mathsf{ID}_2||\mathsf{nonce}_i)$ (refer to Figure 1). From these strings, $\mathcal{A}$ learns the past shared secret R_j with probability with which $\mathcal{H}$ can be invertible. However, under the assumption that this hash function is one-way and H_PUFs are unpredictable, the probability that $\mathcal{A}$ succeeds to learn R_j from R_{i+1} is only negligible. Additionally, under the assumption that $\mathcal{H}$ is collision-resistant and modelled as random oracle and H_PUFs are unpredictable, for $\mathcal{A}$

$$\mathcal{H}(R_i||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i) \approx \mathcal{H}(s||\mathsf{Id}_1||\mathsf{Id}_2||\mathsf{nonce}_i)$$
$$\mathcal{H}(R_i||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2) \approx \mathcal{H}(s||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2)$$
$$\mathcal{H}(C_{i+1}||R_{i+1}||r_i||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2) \approx \mathcal{H}(C_{i+1}||R_{i+1}||t||\mathsf{nonce}_i||\mathsf{Id}_1||\mathsf{Id}_2)$$

where $a \approx b$ implies that a is statistically indistinguishable from b and s, t are chosen uniformly at random from $\{0,1\}^{2n}$.

Therefore, without the knowledge of R_i and r_i, under the assumption that H_PUFs are independent and unpredictable, and hash function $\mathcal{H}$ is modelled random oracle and is collision-resistant, as per Theorem 1, $\mathcal{A}$ cannot learn past session keys or past shared secrets with more than negligible probability.

3.5 Implementation Details of AKE Protocol

Finally, we use SHA-256 from the *libgcrypt* library to implement the hash function $\mathcal{H}$ and realize the AKE protocol as a hardware-software co-design. Except for H_PUF, all the remaining components are implemented in the software. The communication between the software and hardware components is realized using Universal Asynchronous Receiver/Transmitter (UART). The total latency to generate the final H_PUF response is 780μ seconds. We employ BCH(15,7,2) error correcting code [21] for 100% accurate AKE protocol implementation. The overall communication overhead of the protocol is 1216-bits (as nonce, Id_1 and Id_2 are 256, 32 and 32-bit strings respectively) and the overall latency is 0.169 seconds.

4 Conclusion

State-of-the-art PUF-assisted communication protocols require at least one party or TTP to store the CRP database securely. It often becomes a usability and security issue as having a truly secure memory for low-end devices is challenging. This paper addresses this long-standing issue by introducing a PUF-based

construction called Harmonizing PUFs (H_PUFs). H_PUFs enable two distinct PUFs (and therefore two different devices holding them) to eventually generate the same output with assistance from some public data. To establish the utility of H_PUF, we present a lightweight AKE protocol that does not require heavy public-key computation or TTP and solely relies on the H_PUF and a hash function. The protocol performs mutual authentication and secure session key establishment within a single round of communication. The protocol requires each node to perform only four hash operations, and the overall communication overhead is only 1216 bits. We achieve forward secrecy without depending on asymmetric group operations involved in traditional Diffie-Hellman key exchange. We show that the H_PUF-based AKE protocol (H-AKE) fares well when compared to the existing PUF-based solutions in terms of computation and communication overheads as well as provides forward secrecy with minimal assumptions. It is important to note that the idea of H_PUF is not limited to strong PUFs or AKE protocols only but can also be utilized for key generation or use cases based on weak PUFs.

References

1. Aman, M.N., Chua, K.C., Sikdar, B.: Mutual authentication in iot systems using physical unclonable functions. IEEE Internet Things J. **4**(5), 1327–1340 (2017). https://doi.org/10.1109/JIOT.2017.2703088, https://doi.org/10.1109/JIOT.2017.2703088
2. Avoine, G., Canard, S., Ferreira, L.: Symmetric-key authenticated key exchange (SAKE) with perfect forward secrecy. In: Jarecki, S. (ed.) Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12006, pp. 199–224. Springer (2020). https://doi.org/10.1007/978-3-030-40186-3_10, https://doi.org/10.1007/978-3-030-40186-3_10
3. Boyapally, H., Chatterjee, D., Pratihar, K., Saha, S., Mukhopadhyay, D.: PUF-COTE: A PUF construction with challenge obfuscation and throughput enhancement. IACR Cryptol. ePrint Arch. p. 1005 (2022), https://eprint.iacr.org/2022/1005
4. Boyapally, H., Mathew, P., Patranabis, S., Chatterjee, U., Agarwal, U., Maheshwari, M., Dey, S., Mukhopadhyay, D.: Safe is the new smart: Puf-based authentication for load modification-resistant smart meters. IEEE Trans. Dependable Secur. Comput. **19**(1), 663–680 (2022). https://doi.org/10.1109/TDSC.2020.2992801, https://doi.org/10.1109/TDSC.2020.2992801
5. Braeken, A.: PUF based authentication protocol for iot. Symmetry **10**(8), 352 (2018). https://doi.org/10.3390/sym10080352, https://doi.org/10.3390/sym10080352
6. Byun, J.W.: End-to-end authenticated key exchange based on different physical unclonable functions. IEEE Access **7**, 102951–102965 (2019). https://doi.org/10.1109/ACCESS.2019.2931472, https://doi.org/10.1109/ACCESS.2019.2931472
7. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels (2001). https://doi.org/10.1007/3-540-44987-6_28, https://doi.org/10.1007/3-540-44987-6_28

8. Chatterjee, U., Chakraborty, R.S., Mukhopadhyay, D.: A puf-based secure communication protocol for iot. ACM Trans. Embed. Comput. Syst. **16**(3), 67:1–67:25 (2017). https://doi.org/10.1145/3005715, https://doi.org/10.1145/3005715
9. Chatterjee, U., Govindan, V., Sadhukhan, R., Mukhopadhyay, D., Chakraborty, R.S., Mahata, D., Prabhu, M.M.: Building PUF based authentication and key exchange protocol for iot without explicit crps in verifier database. IEEE Trans. Dependable Secur. Comput. **16**(3), 424–437 (2019). https://doi.org/10.1109/TDSC.2018.2832201, https://doi.org/10.1109/TDSC.2018.2832201
10. Delvaux, J., Peeters, R., Gu, D., Verbauwhede, I.: A survey on lightweight entity authentication with strong pufs. ACM Comput. Surv. **48**(2), 26:1–26:42 (2015). https://doi.org/10.1145/2818186, https://doi.org/10.1145/2818186
11. Dodis, Y., Reyzin, L., Smith, A.D.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3027, pp. 523–540. Springer (2004). https://doi.org/10.1007/978-3-540-24676-3_31, https://doi.org/10.1007/978-3-540-24676-3_31
12. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Atluri, V. (ed.) Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002. pp. 148–160. ACM (2002). https://doi.org/10.1145/586110.586132, https://doi.org/10.1145/586110.586132
13. Gope, P., Sikdar, B.: Lightweight and privacy-preserving two-factor authentication scheme for iot devices. IEEE Internet Things J. **6**(1), 580–589 (2019). https://doi.org/10.1109/JIOT.2018.2846299, https://doi.org/10.1109/JIOT.2018.2846299
14. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic pufs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4727, pp. 63–80. Springer (2007). https://doi.org/10.1007/978-3-540-74735-2_5, https://doi.org/10.1007/978-3-540-74735-2_5
15. Hamming, R.W.: Error detecting and error correcting codes. The Bell System Technical Journal **29**(2), 147–160 (1950). https://doi.org/10.1002/j.1538-7305.1950.tb00463.x
16. Idriss, T., Bayoumi, M.A.: Lightweight highly secure PUF protocol for mutual authentication and secret message exchange. In: IEEE International Conference on RFID Technology & Application, RFID-TA 2017, Warsaw, Poland, September 20-22, 2017. pp. 214–219. IEEE (2017). https://doi.org/10.1109/RFID-TA.2017.8098893, http://doi.ieeecomputersociety.org/10.1109/RFID-TA.2017.8098893
17. Khan, M.N.I., Ghosh, S.: Comprehensive study of security and privacy of emerging non-volatile memories. CoRR **abs/2105.06401** (2021), https://arxiv.org/abs/2105.06401
18. LaMacchia, B.A., Lauter, K.E., Mityagin, A.: Stronger security of authenticated key exchange (2006), http://eprint.iacr.org/2006/073
19. Li, S., Zhang, T., Yu, B., He, K.: A provably secure and practical puf-based end-to-end mutual authentication and key exchange protocol for iot. IEEE Sensors Journal **21**(4), 5487–5501 (2021). https://doi.org/10.1109/JSEN.2020.3028872

20. Lim, D., Lee, J.W., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. IEEE Trans. Very Large Scale Integr. Syst. **13**(10), 1200–1205 (2005). https://doi.org/10.1109/TVLSI.2005.859470, https://doi.org/10.1109/TVLSI.2005.859470
21. Peterson, W., Weldon, E.: Error-correcting Codes, Second Edition. MIT Press (1972)
22. Rivest, R.: Illegitimi non carborundum. Invited keynote talk given at CRYPTO (2011)
23. Skorobogatov, S.P.: Semi-invasive attacks: a new approach to hardware security analysis. Ph.D. thesis, University of Cambridge, UK (2005), https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.614760
24. Wallrabenstein, J.R.: Practical and secure iot device authentication using physical unclonable functions. In: Younas, M., Awan, I., Seah, W. (eds.) 4th IEEE International Conference on Future Internet of Things and Cloud, FiCloud 2016, Vienna, Austria, August 22-24, 2016. pp. 99–106. IEEE Computer Society (2016). https://doi.org/10.1109/FiCloud.2016.22, https://doi.org/10.1109/FiCloud.2016.22
25. Zheng, Y., Cao, Y., Chang, C.: Udhashing: Physical unclonable function-based user-device hash for endpoint authentication. IEEE Trans. Ind. Electron. **66**(12), 9559–9570 (2019). https://doi.org/10.1109/TIE.2019.2893831, https://doi.org/10.1109/TIE.2019.2893831
26. Zheng, Y., Chang, C.: Secure mutual authentication and key-exchange protocol between puf-embedded iot endpoints. In: IEEE International Symposium on Circuits and Systems, ISCAS 2021, Daegu, South Korea, May 22-28, 2021. pp. 1–5. IEEE (2021). https://doi.org/10.1109/ISCAS51556.2021.9401135, https://doi.org/10.1109/ISCAS51556.2021.9401135