cryptography

MDPI

Article

Lightweight Mutually Authenticated Key Exchange with Physical Unclonable Functions

Cyrus Minwalla [1,*,†], Jim Plusquellic [2,*,†] and Eirini Eleni Tsiropoulou [2,†]

[1] Financial Technology Research, Bank of Canada, Ottawa, ON K1A 0G9, Canada
[2] Department of Engineering, University of New Mexico, Albuquerque, NM 87131, USA; eirini@unm.edu
* Correspondence: cminwalla@bank-banque-canada.ca (C.M.); jplusq@unm.edu (J.P.)
† These authors contributed equally to this work.

Abstract: Authenticated key exchange is desired in scenarios where two participants must exchange sensitive information over an untrusted channel but do not trust each other at the outset of the exchange. As a unique hardware-based random oracle, physical unclonable functions (PUFs) can embed cryptographic hardness and binding properties needed for a secure, interactive authentication system. In this paper, we propose a lightweight protocol, termed PUF-MAKE, to achieve bilateral mutual authentication between two untrusted parties with the help of a trusted server and secure physical devices. At the end of the protocol, both parties are authenticated and possess a shared session key that they can use to encrypt sensitive information over an untrusted channel. The PUF's underlying entropy hardness characteristics and the key-encryption-key (KEK) primitive act as the root of trust in the protocol's construction. Other salient properties include a lightweight construction with minimal information stored on each device, a key refresh mechanism to ensure a fresh key is used for every authentication, and robustness against a wide range of attacks. We evaluate the protocol on a set of three FPGAs and a desktop server, with the computational complexity calculated as a function of primitive operations. A composable security model is proposed and analyzed considering a powerful adversary in control of all communications channels. In particular, session key confidentiality is proven through formal verification of the protocol under strong attacker (Dolev-Yao) assumptions, rendering it viable for high-security applications such as digital currency.

Keywords: physical unclonable functions; mutual authentication; key exchange protocol; secret-free; quantum-safe; lightweight

Citation: Minwalla, C.; Plusquellic, J.; Tsiropoulou, E.E. Lightweight Mutually Authenticated Key Exchange with Physical Unclonable Functions. *Cryptography* **2024**, *8*, 46. https://doi.org/10.3390/cryptography8040046

Academic Editor: Christoforos Ntantogian

Received: 13 August 2024
Revised: 11 October 2024
Accepted: 15 October 2024
Published: 19 October 2024

1. Introduction

Consumer devices increasingly communicate sensitive information directly to each other. Often, these communications are over untrusted channels where default security controls are either minimal (relying on a pre-shared secret) or nonexistent (information is sent in the clear). In such scenarios, passive attackers can steal secrets, while active attackers can gain control of the channel and engage in malicious activities. This problem is particularly prevalent in IoT systems where security may be weak or absent [1].

Establishing a secure channel is a two-stage process where parties typically first authenticate each other and then cooperate to generate a shared secret, which is used to encrypt the sensitive information traveling over the untrusted channel. Mutual authentication protocols tie the final shared secret to the authentication process, resulting in an authenticated key exchange. Doing so ensures that fraud and malfeasance can be traced back to the malicious participant, a property that is especially desirable for high-value and time-sensitive applications such as digital payments.

Strong PUFs embed a physical source of entropy that is a building block for a true random number generator and/or a one-way function. Secrets of a specific PUF instantiation are inextricably tied to the silicon; ergo, credentials cannot be stolen, and impersonating a

device requires modeling the physical characteristics of the nanometer-sized devices and wires. As such, strong PUFs can play an important role as a hardware root of trust that is impervious to a variety of typical attack vectors, including supply-side threats, and can therefore mitigate large-scale system-wide risks in mission-critical applications.

Contributions

In this work, we propose a light-weight PUF-based mutual authentication and key exchange (PUF-MAKE) protocol that utilizes a set of hardware-based security primitives derived from a strong physical unclonable function (PUF). Facilitated by a trusted authority, the protocol explicitly authenticates both parties and establishes a shared session key between them. Additionally, the authentication challenge and response are updated on every successful authentication, and the server's cache of per-device PUF challenges can be refreshed asynchronously, without requiring the devices to re-enroll or even connect to the system. Furthermore, the authentication completes in three rounds and is considered lightweight as it neither uses asymmetric cryptography primitives, nor does it require devices to carry any information about other devices.

The foundation of the proposed protocol is a composable security model where the protocol is as secure as its underlying building blocks [2]. The scheme is centered around two roots of trust: the PUF as a local root of trust and the enrollment process as a system root of trust. Both roots of trust must be compromised before an attacker can gain access to the secure channel. One-time use keys preserve the secrecy of future sessions. Quantum hardness is achieved by relying on mature, well-understood primitives, such as AES and SHA3, that are known to be quantum-safe. The chosen primitives can be replaced by equivalent block cipher or hash functions without impacting the protocol sequence or security goals. By relying on hardware PUF primitives, the scheme reduces the reliance on fixed, long-lived public-private key pairs or a public key infrastructure (PKI). While the proposed scheme is suited for high-security applications such as digital currency transactions, it can also be used in any type of client-server architecture to implement an authentication layer for secure communications or data transfer.

The remainder of the paper is organized as follows: Section 2 presents a review of related work, including the state of the art. Following that, the underlying primitives are covered in Section 3 and the proposed protocol in Section 4. Section 5 presents the formal security analysis of the protocol, while Section 6 covers other security aspects not discussed in the formal model. Section 7 describes the experimental methodology and performance analysis. Section 8 summarizes the findings and future work.

2. Related Work

Authenticated key exchange (AKE) is a mechanism that serves the dual purpose of (a) authenticating two parties with respect to each other and (b) building a secure communications channel between two parties over untrusted networks (e.g., the Internet) where active adversaries can be expected. In cases where (a) is satisfied for both parties, the protocol is considered to be mutually authenticated key exchange (MAKE). Early work by Diffie et al. [3] explores authenticated key exchanges as part of a station-to-station (STS) encryption protocol. The work is notable for establishing design principles for a secure protocol, with later work by Bellare and Rogaway [4] further formalizing this approach and extending it to arbitrary two-party protocols.

Bellare et al. [5,6] later proposed an authenticated key exchange protocol based on the standard model of an asymmetric encryption algorithm coupled with a trap-door function, but the scheme proved susceptible to chosen cipher-text attacks. Okamoto [7] improved upon this approach by introducing a pseudo-random function as a replacement for the trap-door function, proving IND-CCA indistinguishability under the standard model. In parallel, Law et al. [8] proposed an efficient protocol titled MQV (the author's initials concatenated) combining Diffie-Hellman for forward secrecy and elliptic curve cryptography for efficient key generation. Authentication, however, was implicit as entities were not identified,

leaving the scheme vulnerable to impersonation attacks. Security of the protocol was later upgraded by Krawczyk [9] by introducing a Schnorr signature scheme that serves the dual purpose of including identities and a building block to construct a challenge-response sequence for a three-round protocol. LaMacchia et al. [10] improved upon Krawczyk's scheme with the NAXOS protocol by making stronger adversarial assumptions.

Hardware-based MAKE constructions utilize PUFs as a local root of trust. An extensive literature review on previously proposed light-weight PUF-based authentication protocols is given in [11]. Idriss et al. [12] propose a lightweight PUF-based mutual authentication and secret message exchange protocol. The protocol requires no cryptographic primitives and exchanges only challenges between the server and device. The proposed protocol authenticates during the connection establishment phase of WiFi using 3 CRPs. XOR operations in combination with a router-generated nonce are used to encrypt challenges and responses between device and router and to derive the next set of CRPs for the next authentication. The protocol is one-way, i.e., it does not authenticate the device. It also assumes that the router has a soft model of the PUF and can generate a response to any randomly generated challenge during the refresh CRP operation. Mahalat et al. [13] propose a scheme for secure WiFi authentication of IoT devices. This approach was later expanded to a PUF-based authentication and key sharing scheme that utilizes Pedersen's commitment scheme coupled with Shamir's secret sharing [14]. The mutual authentication and key sharing scheme proposed between user (server) and sink nodes can be implemented more easily using CRP-strong PUF-based schemes without the mathematical complexity of the secret sharing schemes [15].

A PUF-based authentication and key management protocol for IoT is proposed in [16], improving upon the attack resilience and performance overhead of the previous method [17]. The scheme relies on elliptic curve cryptography (ECC) and requires a trusted setup with tamper-resistant hardware to protect secret keys. Similarly, a controlled PUF that utilizes ECC is proposed as a lightweight authentication and key generation protocol for IoT nodes in [18], relying on zero knowledge proofs for device authentication; however, the authentication is one-way, and the server is not authenticated. A PUF-based ElGamal algorithm is proposed for message encryption as well as a PUF-based digital signature scheme.

A lightweight authentication protocol is proposed in [19] that is designed to prevent an adversary from obtaining sufficient CRPs to carry out model-building attacks. The protocol gives the device control over its own authentication by requiring the server to send half of a response to an authentication challenge to the device initially along with the challenge. The device does not respond with its half of the response unless the server's response component is validated by the device. The main drawback is the limited number of authentications that are possible, which is constrained by the number of CRPs stored in the database. In [20], the authors propose a crossover ring oscillator (RO) PUF cloning technique that enables a group of IoT devices to all generate the same (shared) key, thereby eliminating the key distribution problem for devices engaging in multi-party shared key encrypted communication.

Zheng et al. [21] propose a peer-to-peer protocol in which two PUF devices can authenticate and generate a shared session key without the need to store CRPs and instead uses a one-time pad to generate a unique key mask for each pair of customer devices. Therefore, every enrolled device must store information on every other device. Another drawback relates to the use of ECC, where the protocol shows that the fuzzy extractor 'Rep' operation is performed by the device, which according to [22] is too expensive for lightweight authentication applications. Last, the security of their protocol is reliant on the security guarantees of the Diffie-Hellman key exchange. Although our protocol uses some of the same cryptographic primitives, it does not use ECC and is therefore more lightweight.

MAKE protocols between an IoT device and a secure server and between two IoT devices is proposed in [23]. The server is used in both protocols to authenticate the first device using CRPs stored on the server during provisioning. The device does not store

any CRPs, requiring challenges and helper data (for a fuzzy extractor error correcting scheme) to be sent in the clear to the device. The proposed scheme, although secure against model-building attacks, requires a centralized authority and an on-line connection to carry out any type of authentication. The proposed device-server scheme is very similar to early work on PUF-based protocols described in [11]. The device-to-device scheme is unique but requires the server to pass an authentication credential, i.e., a CRP, for the second device to the first device, which, if compromised, might allow the first device to authenticate as the second device. The MAKE protocol proposed in this paper distributes the workload across multiple authentication servers, reducing the bottleneck that would result with a large scale deployment, and does not treat even authenticated devices as trusted.

In [24], the author proposes a PUF-based authentication protocol that leverages Paillier homomorphic encryption or ElGamal encryption with a plaintext equality test as a means of obfuscating the CRPs from the gateway routers and the verifier. The method addresses CRP depletion, machine learning, and impersonation attacks commonly carried out against authentication protocols. However, the device experiences high computational loads and power consumption when authenticating because of the modular operations associated with homomorphic encryption, in comparison with the linear operations employed in our proposed protocol.

3. Underlying Primitives

3.1. Physical Unclonable Function (PUF)

The protocol relies on a strong PUF hardware primitive as a local root of trust. In particular, the properties of a strong PUF variant, called a super-high information complexity (SHIC) PUF, as introduced by Ruhrmair [25], are essential to the security of the protocol. An idealized SHIC PUF has the following properties:

1. It contains an exponential amount of response-relevant random information tied to an entropy source with a very high information density.
2. The read-out speed of PUF responses is limited to low values. Moreover, the limitation is an inherent property of the PUF measurement process and not an arbitrary delay injected into the read-out circuit.
3. The CRPs are mutually independent. Furthermore, the pairwise mutual information within any two responses is zero.

The protocol assumes that each participant is equipped with an instance of a SHIC PUF that is enrolled in the system. Furthermore, the authentication server, a component that facilitates authentication between two participants, is also equipped with a SHIC PUF to establish a PUF-based mutually authenticated secure channel with the issuing authority.

3.2. Key-Encryption-Key (KEK)

Key-encryption-key, or KEK, is traditionally used in reference to a master key that a device stores and uses to decrypt boot images at start-up and to generate other keys, e.g., ephemeral session keys and authentication bitstrings during system operation. Given its central role, it defines the root-of-trust in most systems. KEKs also need to be reproducible at any instance in time, potentially over the lifetime of the device, and are therefore also referred to as long-lived keys (LLK). The role of a KEK as the root-of-trust also imposes strong security constraints on the system to maintain its secrecy because an attack that is able to extract the KEK compromises the entire system.

A SHIC PUF is hardened against KEK extraction attacks as it stores only the challenges and helper data needed to reproduce the KEK, not the KEK itself, and can go a step further by harnessing the ability to generate an exponential number of KEKs. A SHIC PUF enables Alice and Bob to authenticate and generate a shared session key. In particular, a strong PUF's LLK generation function provides an exponential number of unique KEKs as a means of meeting the one-time use constraint associated with authentication while simultaneously providing the ability to reliably regenerate any one of its KEKs on-demand.

4. The PUF-MAKE Protocol

The protocol assumes a minimum of four entities, namely an issuing authority (IA), an authentication server (AS), and a pair of devices (Alice and Bob) who wish to establish a secure channel with each other to exchange information. An example application is one of digital currency, where the IA is the central authority, AS is an intermediary, and Alice and Bob are devices that exchange electronic funds for goods and services. For example, Alice can be a customer wishing to buy something wirelessly from a brick-and-mortar or online store (Bob), or Alice and Bob may be two customer devices communicating through a Bluetooth or IR link to exchange funds. The AS in this scenario can be a trusted intermediary providing value-added services, ranging from a small embedded system in a coffee shop to a bespoke enterprise server installation tailored to high-traffic applications. Notably, the scale and scope of deployment are not restricted by protocol primitives.

The protocol encapsulates two core security properties: First, physical access to the PUF is required to participate in authentication and secure channel creation, thus establishing a hardware root of trust. Second, authentication tokens (AT) are used only once for a single successful authentication, thus enabling forward secrecy.

4.1. MAKE Enrollment Protocol

The enrollment process involves three entities: the device (Alice), IA and AS, as shown by the vertical partitions in the message exchange diagram of Figure 1. The IA utilizes a CRP database, CRP_{DB}, generated during PUF provisioning (or alternatively, it can generate the PUF CRPs on-the-fly using a soft PUF model and soft PUF data [15]). The provisioning process is not shown because it varies depending on the SHIC PUF utilized. The IA interacts directly with the device to create a set of authentication tokens, AT. The requirement for the device to be involved in this three-party protocol prevents certain attacks that are discussed in Section 5. The IA runs on a multi-core server in a central, secure facility.

The AS is initialized and periodically refreshed with a compact representation of the AT, a core component of PUF-MAKE's CRP-based authentication and session key generation process. The right side of Figure 1 shows the format of the database called AuTk_{DB} that stores AT. An AT is defined as including a customer device ID, e.g., ID_A, a hashed version of a KEK response, HK_A, a PUF challenge Chlng_A and helper data HD_A. This information enables AS to validate Alice and Bob's devices as enrolled, PUF-instantiated devices via an interactive protocol. The AS can also run on a multi-core server in a central, secure facility, or as a fielded device.

The ordered sequence of message exchanges and operations that comprise the enrollment process followed during the initialization phase are described below, in correspondence with the numbered annotations shown in Figure 1.

1. The IA and AS mutually authenticate (MA) using a privacy-preserving PUF-based (P3B) protocol and then generate a session key (SKG), called SK_I. An example P3B protocol suitable for PUF-MAKE is described in [26].
2. The same MA and SKG process is carried out between Alice and the IA to authenticate and generate a shared session key SK_A.
3. The IA selects a CRP for Alice from its CRP_{DB} (constructed during provisioning), extracts Chlng_{Ax}, and encrypts it using AES-256. The ciphertext C_1 is transmitted to Alice.
4. Alice decrypts C_1 to obtain Chlng_{Ax}. Alice applies the challenges to her hardware HPUF in enrollment mode to generate an authentication key, KK_{Ax} and helper data HD_{Ax}. She hashes KK_{Ax} using a SHA-3 hashing function to produce HK_{Ax}, and encrypts HK_{Ax} and HD_{Ax} with SK_A to produce ciphertext C_2.
5. Alice transmits ciphertext C_2 to IA. On the first iteration of the enrollment process, Alice also stores the tuple [Chlng_{A1}, HD_{A1}] to persistent memory. She will use this challenge and helper data to reproduce HK_{Ax} during her first in-field authentication, discussed below.

6. IA decrypts C_2, fetches the response component of the challenge $Chlng_{Ax}$ from its CRP database, and assigns the response to KK'_{Ax}. IS hashes KK'_{Ax} and compares the output HK'_{Ax} with the decrypted HK_{Ax} received from Alice. If they match then Alice's HK_{Ax} is validated. IA then encrypts packet C_3 with SK_I, containing Alice's ID, ID_A, and AT components, namely HK_{Ax}, $Chlng_{Ax}$ and HD_{Ax}.
7. IA transmits the C_3 to AS.
8. AS decrypts the C_3 and stores ID_A, HK_{Ax}, $Chlng_{Ax}$ and HD_{Ax} in its $AuTk_{DB}$ database. The AT elements in this table will be used by AS for in-field authentication operations carried out on behalf of Alice (and Bob) and to enable Alice and Bob to generate a shared session key.

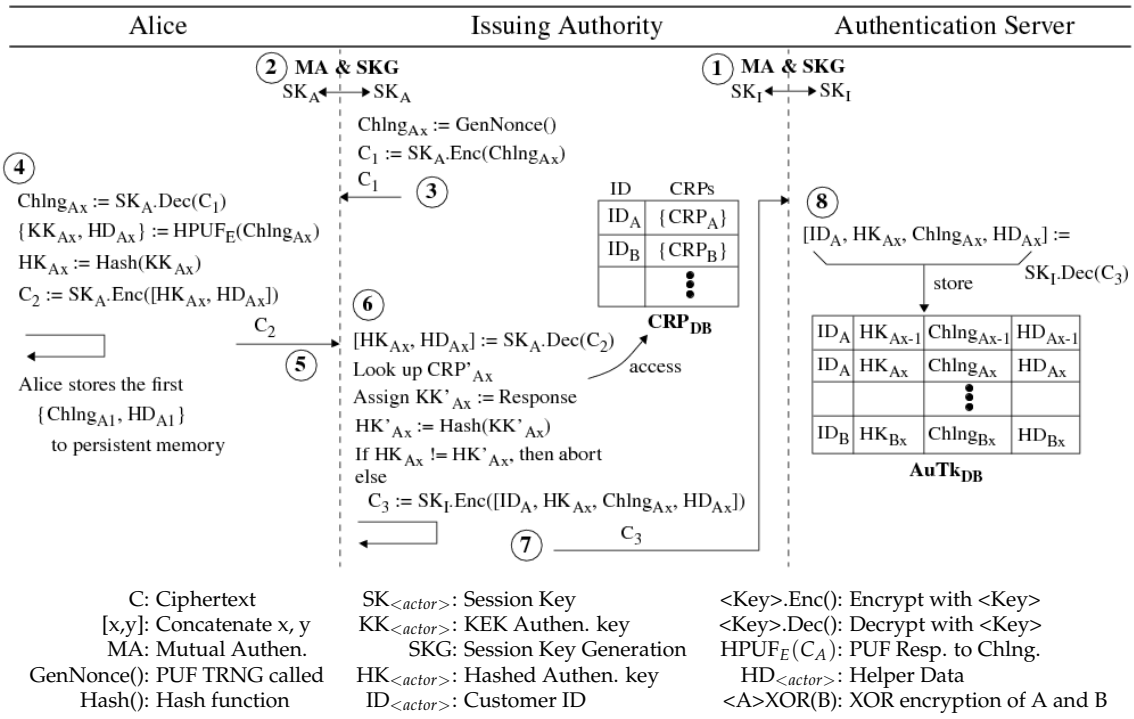

Figure 1. Message exchange diagram for PUF-MAKE Enrollment.

A refresh phase is periodically carried out between the IA and AS to replenish AT stored by the AS. Although the protocol as shown stores CRP on IA in a database, the number of elements defining the CRP space for each device can be significantly increased using a more compact soft data representation of PUF CRPs as described in [15]. In such cases, the process described here is one of expansion from IA-stored PUF soft data to AS CRPs, which further justifies the need for a refresh phase.

4.2. PUF-MAKE In-Field Interactive Authentication Protocol

At the end of enrollment, Alice and Bob each store a challenge that they will use to generate a response, HK_{A1}, for the first authentication request with AS. The in-field version of the protocol is sequenced such that Alice and Bob must first authenticate to AS with their existing challenge, and, as part of successful authentication, they receive and store a new challenge for the next authentication cycle, guaranteeing forward secrecy. The $AuTk_{DB}$ maintained by AS is designed to facilitate multiple simultaneous transactions while retaining security against attacks. Each authentication uses a one-time credential that is replaced at the end of the authentication cycle. The session key is jointly derived by both participants, creating a binding partnership between devices for the duration of the session.

The Interactive Authentication message exchange diagram for MAKE is shown in Figure 2. The diagram shows the sequence of operations that occur when Alice and Bob

wish to establish an authenticated and encrypted channel with each other, e.g., to exchange electronic cash for goods and services.

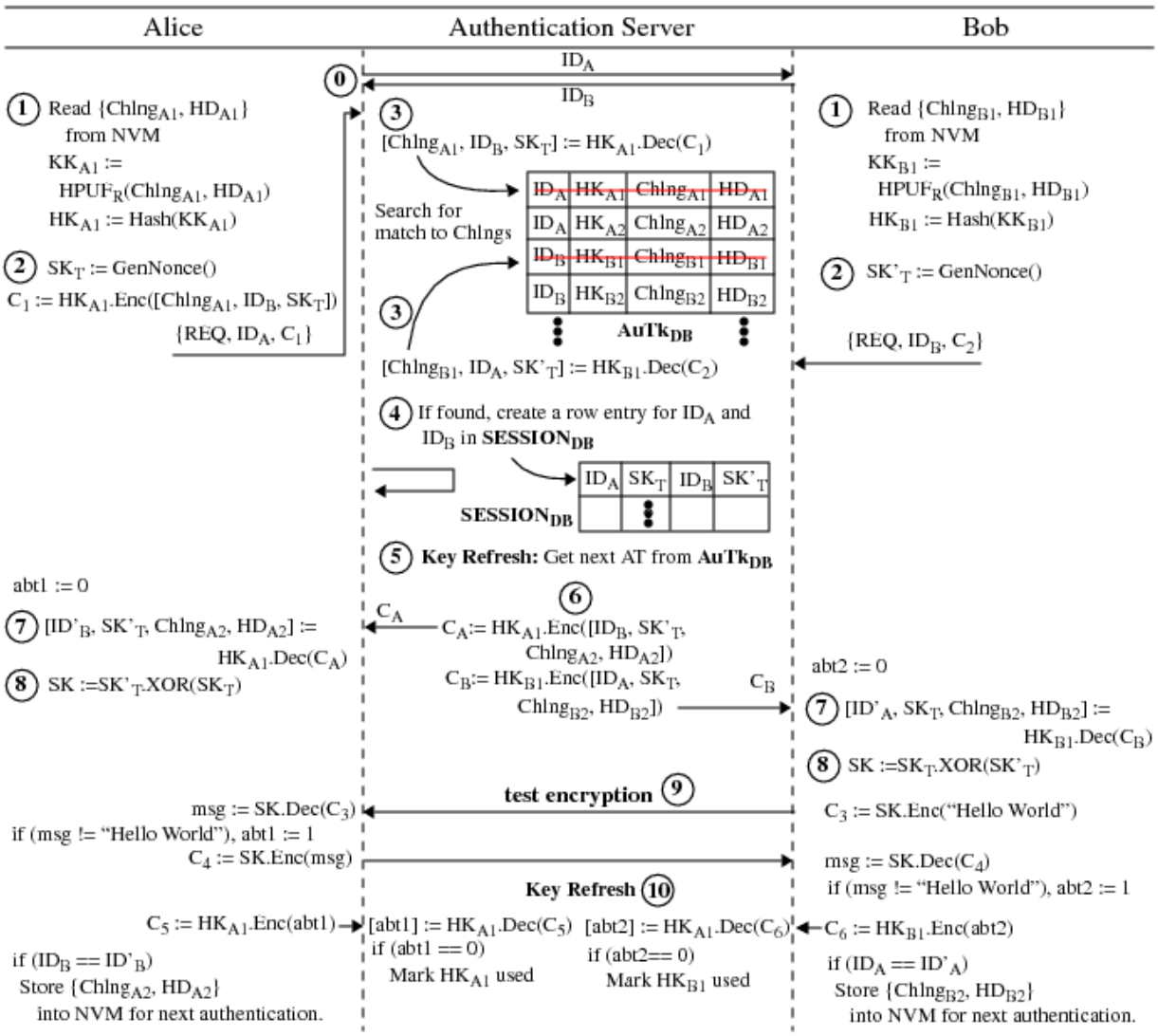

Figure 2. Message exchange diagram for in-field authentication between Alice and Bob. The strike-outs, indicated with red lines in the $AuTk_{DB}$, identify used AT that are deleted from the database.

0. The protocol begins with Alice and Bob sending a request to communicate to each other using an unencrypted packet with identifiers ID_A and ID_B.
1. Alice reads $Chlng_{A1}$ from NVM and runs her PUF, $HPUF_R$, in regeneration mode to regenerate response KK_{A1}. She then calculates a hash HK_{A1} of KK_{A1} using a suitable hash function. Bob performs the same sequence of operations.
2. Alice and Bob generate session key shards, SK_T and SK'_T, using their PUF-based true-random-number-generators (TRNGs).
3. Alice assembles packet C_1 by concatenating authentication artifacts, $Chlng_{A1}$, Bob's device ID, ID_B, and the session shard, SK_T, which are then AES-encrypted with HK_{A1} as the key. Alice sends an authentication request to AS, along with ID_A and C_1 as metadata. The server matches ID_A to the list of known devices in the database $AuTk_{DB}$ populated by the issuing authority (IA) during enrollment. Upon match, the server uses the associated HK_{A1} to decrypt C_1 and disassembles the $Chlng_{A1}$, ID_B and SK_T fields. Authentication is a success if the extracted $Chlng_{A1}$ matches the one stored for ID_A in the database; otherwise, this process repeats using each of the

remaining ID_A elements. If no matches are found, the protocol aborts. Note that both HK_{A1} and $Chlng_{A1}$ must be correct for authentication to succeed. If HK_{A1} does not match any of those stored in the $AuTk_{DB}$, the packet will not be decrypted correctly and $Chlng_{A1}$ will be random, causing a mismatch. Alternatively, if HK_{A1} is correct and $Chlng_{A1}$ is not, the packet will decrypt correctly but $Chlng_{A1}$ will fail to match. In either case, authentication fails and the server aborts the connection. Bob performs this same set of operations with AS.

4. If Alice's authentication succeeds, AS adds ID_A and SK_T to a $SESSION_{DB}$ database. Similarly, if Bob's authentication succeeds, AS adds ID_B and SK'_T to the $SESSION_{DB}$. In both cases, AS first searches for a match to ID_B and ID_A supplied by Alice and Bob, respectively, to determine if a row already exists, and, if so, adds Alice or Bob's information to the matching row element instead of creating a new row. Once Alice and Bob's IDs and SK_T are both present, AS proceeds to the next step; otherwise, it stalls the thread waiting for Alice or Bob to complete the transaction. A time-out can also be included that aborts the entire transaction and deletes the $SESSION_{DB}$ element.
5. The first step of Alice and Bob's key refresh operation is for AS to fetch new AT elements from the $AuTk_{DB}$, e.g., it selects rows associated with HK_{A2} and HK_{B2} elements. The key update mechanism ensures that Alice and Bob are protected against both impersonation and replay attacks.
6. AS constructs a packet C_A for Alice by encrypting Bob's authentication artifacts ID_B, SK'_T and Alice's next challenge $Chlng_{A2}$ and HD_{A2} with Alice's HK_{A1} AES key. AS constructs a packet C_B for Bob in a similar fashion and transmits the packets to Alice and Bob.
7. Alice decrypts her C_A with HK_{A1} to obtain Bob's ID ID_B, Bob's SK'_T and her next authentication challenge. Bob does the same with his packet.
8. Both Alice and Bob XOR the SK_T and SK'_T shards to obtain a shared session key SK, which they can use to encrypt communications between them.
9. Alice and Bob validate their shared key by performing a test encryption operation, where Bob encrypts a test message "Hello World" with SK and transmits it to Alice. Alice decrypts and validates. If the comparison fails, she sets her $abt1$ flag to 1. She then encrypts msg with her version of SK and transmits it to Bob. Bob decrypts and compares it to "Hello World". If this fails, he sets $abt2$ to 1. Alice also encrypts her $abt1$ flag with HK_{A1} and transmits to AS as encrypted packet C_5. Bob carries out the same operation with $abt2$.
10. If Alice and Bob succeed in the test encryption comparison operation, they possess a valid shared key and can encrypt communications between them. They notify AS with C_5 and C_6, and AS marks the current AT as used. Alice and Bob also update their NVMs with the next Chlng and HD information under the condition that the ID' values match the original ID values, which indicates they have received valid next challenge information.

5. Security Analysis

This section describes the formal static analysis of the PUF-MAKE protocol to demonstrate authentication properties. An automated solver is used to prove that the confidentiality and integrity properties of the generated session key are maintained as described. We build on the principle of universal composability to achieve the security goals of the protocol. Under a composable security model, the protocol is proven to be secure under a set of adversarial assumptions if the underlying building blocks are proven to be secure under those same assumptions and the protocol steps themselves preserve confidentiality and integrity of sensitive materials [2]. Two distinct security sub-models are defined, namely the hardness of the PUF primitive and the hardness of the protocol. The hardness of the PUF primitive is discussed in reference to prior work as follows: To ensure that the protocol is secure, we provide a formal proof of correctness generated by a formal

prover/verification program conducting adversarial analysis on the protocol. The proof guarantees that confidentiality of the session key is preserved under all circumstances.

5.1. Formal Model

We start with a formalization of the physical strong-PUF primitive following notation from [27], additionally introducing a measurement time parameter as required for a SHIC PUF:

Definition 1. *Let a physical system, Σ be instantiated from a design $\hat{\Sigma}$ such that:*

- *Σ holds a state $X \epsilon \{0,1\}^l$, which can be sampled by a physical probe P at measurement points defined by vector $z \epsilon \{0,1\}^k$.*
- *Σ enables $y = f(X, P_z) \rightarrow \{0,1\}^n$ to be measured from z, where $f(X, P_z)$ denotes the measurement process.*

Definition 2. *Let $f : \{0,1\}^l \times \{0,1\}^k \rightarrow \{0,1\}^n$ be a physical one-way function where:*

1. *$\exists y$ that represents a measurement made by P of the physical source Σ.*
2. *y is sampled in finite, positive, non-zero time t, which is tied to the intrinsic physical characteristics of Σ. The sampling time t is deterministic ($O(1)$) based on the design of Σ.*
3. *Recovering z from y requires $\Omega(\exp(l))$ queries to Σ, or expressed formally, the likelihood of guessing by brute-force is:*

$$Pr[A'(f(X, P_r))outputs\ X\ or\ P_r] < \frac{1}{exp(l)} \tag{1}$$

4. *Similarly, for two distinct physical instantiations, Σ_i and Σ_j, the likelihood that a measurement vector z generates an identical response is:*

$$Pr[f(X_i, P_z) = f(X_j, P_z)] < \frac{1}{exp(l)} \tag{2}$$

5. *Simulating y, given X and P and non-zero Δt, requires $O(exp(l))$ operations, where each operation requires duration t to complete.*

Here, z is equivalent to a challenge bitstring, while y is equivalent to the corresponding response bitstring. It is noted that for practical instantiations, t may be on the order of seconds to satisfy the requirements of a SHIC PUF. Furthermore, it should not be based on an arbitrary delay injected into the circuit, as such a delay would be trivial to bypass. In the case where l could be exponentially large but n and k are bounded by P and f, respectively, an adversary could construct a look-up table of vectors z and y while ignoring X altogether. To counteract this, a strong PUF must embed an exponential search space for both challenges and responses, which can be expressed as follows:

Definition 3. *Building on Definitions 1 and 2, and given a set of challenges, $Chlg \subseteq \{0,1\}^*$, and responses, $Resp, \subseteq \{0,1\}^*$, a PUF is a physical system S that maps physical stimuli defined by challenges $Chlg_i$ to measured responses $Resp_i$ such that:*

1. *The PUF function $f(Chlg_i) \rightarrow Resp_i$ maps a unique unbounded challenge to a unique unbounded response.*
2. *The design is collision-resistant: $\forall R : Pr[Resp_i = Resp_j] \leq \frac{1}{2^N}$, where $i \neq j$ and N is the number of response bits in $Resp_*$;*
3. *For a distinct physical device, Σ', two non-identical challenges, $Chlg_i$ and $Chlg_j$, will always generate responses $Resp_i = f(Chlg_i)$ and $Resp_j = f(Chlg_j)$ such that $HD(Resp_i, Resp_j) = N/2$, where $HD()$ is the hamming distance and N is the bit cardinality of $Resp_i$ and $Resp_j$, i.e., 50% of the bits are different between the two responses.*

4. *Two distinct physical devices based on the same design (Σ_i and Σ_j), a unique, identical challenge, $Chlg$, will generate responses $Resp_i = f_i(Chlg)$ and $Resp_i = f_j(Chlg)$ such that $HD(Resp_i, Resp_j) = N/2$.*

5.2. Protocol Model

The protocol described in Figure 2 is a three-round interactive protocol between Alice and Bob, mediated by a third party, AS, that provides authentication services to both. Algorithm 1 describes the model in BAN notation, while predicates derived from the protocol are listed in Algorithm 2. Here, f denotes the hardware PUF function, while $Chlg_x$ and Rsp_x are the challenges and associated responses, respectively, as per the model.

Algorithm 1: Formal description of the protocol

Common Input: $Chlg_x$ is well-formed

1. $A \rightarrow B : REQ, A$
2. $B \rightarrow A : ACK, B$
3. A computes:
   - $SK_t := \{0,1\}^l$
   - $HK_{A1} := Hash(f(Chlg_{A1}))$
   - $C_1 := \text{Enc}(HK_{A1}, \{Chlg_{A1}, B, SK_t\})$
4. $A \rightarrow AS : REQ, A, C_1$
5. B computes:
   - $SK'_t := \{0,1\}^l$
   - $HK_{B1} := Hash(f(Chlg_{B1}))$
   - $C_2 := \text{Enc}(HK_{B1}, \{Chlg_{B1}, B, SK'_t\})$
6. $B \rightarrow AS : REQ, B, C_2$
7. AS computes:
   - $\{HK_{A1}, HK_{B1}\} := Lookup(A), Lookup(B)$
   - $\{B, SK_t, Chlg_{A1}\} := \text{Dec}(HK_{A1}, C_1)$
   - $\{A, SK'_t, Chlg_{B1}\} := \text{Dec}(HK_{B1}, C_2)$
   - Verify $Match(Chlg_{A1}, HK_{A1}) = Match(Chlg_{B1}, HK_{B1}) = true$
   - For A, $Lookup(Chlg_{A2}, HK_{A2})$. For B, $Lookup(Chlg_{B2}, HK_{B2})$.
   - $C_A := \text{Enc}(HK_{A1}, \{B, SK'_t, Chlg_{A2}\})$
   - $C_B := \text{Enc}(HK_{B1}, \{A, SK_t, Chlg_{B2}\})$
8. $AS \rightarrow A : C_A$, $AS \rightarrow B : C_B$
9. A computes:
   - $\{B, SK'_t, Chlg_{A2}\} = \text{Dec}(HK_{A1}, C_A)$
   - $SK := SK_t \oplus SK'_t$
10. B computes:
   - $\{A, SK'_t, Chlg_{B2}\} = \text{Dec}(HK_{B1}, C_B)$
   - $SK := SK'_t \oplus SK_t$
11. $A \rightarrow B : \text{Enc}(SK, Test)$
12. $B \rightarrow A : \text{Enc}(SK, Confirm)$

Algorithm 2: Assertions derived from the PUF and protocol properties

Assertions

1. SK_t and SK'_t are fresh, HK_{Ax} and HK_{Bx} are one-time use.
2. $HK_A1 \neq HK_{A2} \wedge HK_{B1} \neq HK_{B2}$ (Definition 3.2).
3. $HK_A1 \neq HK_{B1} \wedge HK_{A2} \neq HK_{B2}$ (Definition 3.3).
4. $Hash(f(Chlg_x))$ obfuscates $Resp_x$, thwarting model-building attacks (Definition 2.5).
5. A believes AS is authentic since AS can produce HK_{A1} to decrypt C_1 ($Assumption$: IND-CCA secure encryption).
6. B believes AS is authentic since AS can produce HK_{B1} to decrypt C_2 ($Assumption$: IND-CCA secure encryption).
7. AS believes A is authentic since A knows $Chlg_{Ax}$ and possesses a physical PUF that can produce HK_{Ax} from $Chlg_{Ax}$. (Definition 3.1).
8. AS believes B is authentic since B knows $Chlg_{Bx}$ and possesses a physical PUF that can produce HK_{Bx} from $Chlg_{Bx}$. (Definition 3.1).
9. A believes B is authentic since AS asserts B and B has shared session key SK
10. B believes A is authentic since AS asserts A and A has shared session key SK
11. Confidentiality of SK is always preserved (Formal Verification Proof).
12. An adversary cannot produce HK_x even knowing $Chlg_x$ unless it possesses the specific physical PUF (Definition 2.4).
13. HK_A and HK'_A are unlinkable by an adversary (Definition 3.2).
14. An adversary cannot recover SK or HK_x unless the AS is dishonest or compromised.

6. Informal Security Analysis

The PUF device acts as a local root of trust, and as such, its security properties are crucial to the overall security posture of the scheme. All possible attacks to the device and underlying primitives are analyzed for completeness.

6.1. Challenge Response Search-Space

A SHIC PUF is a physical random oracle [28], such that the entropy pool of the challenge-response space is vast, yet the response is reproducible at the bit level for a given challenge, irrespective of the number of times the same challenge is presented. Our version of the protocol uses the SHIC PUF design from [29] and associated primitives from [26] to implement the protocol, although any SHIC PUF could be used.

6.2. Model-Building

A strong PUF can have its responses modeled and predicted using machine learning techniques, allowing a functional clone of a PUF device to be created by an adversary [30]. The results presented in [31,32] conclusively demonstrate it is possible to model certain PUF designs using evolutionary models or logistic regression (multi-layer perceptron) given only a subset of the CRPs. In particular, Arbiter XOR PUFs are shown to be broken by these techniques.

The SiRF PUF used in our experiments embeds certain properties that make model-building difficult. Primarily, the CRP search space is shown to possess LPN (Learning Parity with Noise) hardness, as discussed in [26]. Additionally, the delay in the response read-out is rooted in the process that measures and digitizes path delays at high precision, and cannot be circumvented. Finally, raw responses are never used without masking with one-way functions in CRP exchanges, a concept upon which PUF-MAKE extends.

The PUF-MAKE protocol uses obfuscated challenges and responses for authentication. Raw responses are ephemeral and discarded once the hash is computed. This requires the attacker to physically hack the hardware to extract the response bitstring, and to do so in a short interval of time before the response is deleted from memory. In particular, an attack

of this nature would require an attacker to isolate the PUF read-out circuitry and probe the precise memory location where the response bits are stored.

The IA stores sufficient CRP information that, if compromised, would allow any PUF in the system to be cloned to the limit of provisioning information, at least until it is discovered that such an attack occurred and before the CRP_{DB} is replenished with new sets of CRP. For this reason, the IA is treated as a high-security infrastructure component that is kept in a highly controlled enterprise environment with explicit network segmentation and permission controls to restrict access. Similar arguments hold true for the AS, albeit the amount of information that would be compromised is smaller. AS needs only to connect periodically to the IA to refresh its authentication database. This interaction can be orchestrated carefully between two trusted entities using state-of-the art security, or dedicated channels, in addition to PUF-based authentication.

6.3. Unlinkability

We show the unlinkability of two distinct MAKE sessions through the following properties: Definition 2.4 guarantees that a SHIC PUF embeds sufficient entropy to generate an exponential number of challenge-response pairs, such that one-time use is possible for any reasonable time frame. Additionally, Definition 3.3 guarantees that any two keys generated by the same PUF are unlinkable to each other. Finally, Definition 3.4 guarantees that keys generated by two distinct physical instantiations of a PUF design will also be unlinkable to each other. Since all keys used in the protocol are derived from the PUF and only used for one iteration of the protocol, keys used in successive authentications are unlinkable.

6.4. Fault Injection

An attacker could attempt to flip one or more bits in the CPU during sensitive computations to change the control flow. The microprocessor component of system-on-chip (SoC) FPGAs is synthesized onto standard silicon processes and lacks specific tamper-resistant features. In the present approach, the difficulty of performing fault injection attacks is increased by implementing protocol steps at the FPGA synthesis level as Verilog state machines. In doing so, the attacker would be forced to first gain knowledge of the physical design; otherwise, randomly injecting faults would, with high probability, just cause state machine execution to halt or produce useless information. Execution of cryptographic primitives (AES, SHA-3) can be ported to hardware as well, and other necessary software functions (e.g., database query) can be transferred to a secure processing environment such as ARM TrustZone.

In terms of the core PUF primitives, fault injection is mitigated by the natural tamper-evident property of the PUF itself. Unlike cryptographic primitives, in which fault injection can be used to weaken the security, for instance by reducing the number of rounds in an AES encryption operation, fault injection performed during the PUF's bitstring generation operations would only prevent the key material from being correctly generated or reproduced and result in a failed authentication attempt.

6.5. Stored Secret Exfiltration

An attacker could attempt to extract sensitive information stored on the PUF either at rest or during execution of the protocol. Timing and side-channel attacks have been exploited effectively against PUFs in the past [31]. All devices in the system (AS, Alice, and Bob) except for the IA are PUF devices. Each PUF device must be powered, activated, and presented an explicit challenge to elicit a corresponding response. Given the delicate nature and nanometer-sized features of the devices and wires embodying within-die variations, which represent the source of entropy for the PUF, it is impossible to ascertain the response by examining the silicon. Information stored in local non-volatile memory is encrypted at rest via a PUF-based KEK key. An attacker could attempt to exfiltrate the AES secret key during encryption/decryption operations via differential power analysis (DPA) techniques.

While this is possible, it is noted that the window of opportunity to exfiltrate the key and make use of it in a single transaction is small, and the power transient information would be completely different for the next authentication cycle since each key is only used once before replacement.

An attacker could manipulate the communications channel to force Alice to make repeated authentication attempts with the same HK_A. For the present work, we used the AES version implemented in the OpenSSL library, which is susceptible to side-channel attacks such as differential power analysis [33] on common processors, including ARM platforms. This attack can be mitigated by switching to a hardened hardware version of AES, with suitable care taken during implementation to prevent leakage [34]. Note that even if a key is compromised, it can only compromise a single authentication cycle as the scheme uses a brand new challenge for every authentication attempt, achieving perfect forward secrecy.

6.6. Bitstream Manipulation

An attacker could steal the device and attempt to read out all the CRPs by repeating the provisioning process. This attack is prevented by disabling the provisioning operation in the bitstream of fielded devices. An adversary can attempt to reverse engineer the in-field bitstream and re-enable the read-out functionality. However, the FPGA bitstream is encrypted in persistent storage on the device, and therefore the adversary would need to extract the bitstream decryption key from the FPGA to obtain an unencrypted bitstream and then go through a reverse engineering process to make changes to enable provisioning. Although possible, the process is designed to be very difficult by the FPGA manufacturer. Note that an ASIC version of the PUF entirely obviates this attack vector.

6.7. Cryptographic Primitives

The practical instantiation of the protocol relies on the mathematical operation XOR, the AES cryptographic primitive for symmetric encryption/decryption, and SHA-3 for hashing operations. Note that any symmetric encryption and hashing primitive are compatible with the protocol. Cryptographically secure implementations of AES and SHA-3 used in the experimental prototype are sourced from OpenSSL. The XOR operation is used for performance and is interchangeable with a symmetric block cipher primitive for added security if desired. The protocol is future-proof in that it does not rely on specific properties of AES and SHA-3, and any symmetric block cipher plus hash function can be used instead, in case these primitives happen to be broken in the future.

6.8. Enrollment

A local adversary could eavesdrop on the MAKE enrollment process. During initial enrollment, the customer device and IA carry out a P3B mutual authentication and session key generation process with IA. Therefore, it is not possible for an adversary to collect and manipulate plaintext information or insert new information between the customer device and IA. The same process is carried out between IA and AS during enrollment and refresh operations, and the same protections are afforded against MITM. Furthermore, enrollment would typically occur in a trusted environment with additional enterprise security controls in place.

6.9. Correctness of In-Field Authentication

The correctness of the in-field protocol was verified and proven by the use of a cryptographic protocol verification tool. A formal verification tool, Verifpal, was used to verify protocol correctness and ensure that security goals specified on the outset were satisfied by the protocol's construction. Verifpal is an automated formal verification system that employs deductive reasoning to emulate both a passive and an active attacker [35]. It emulates a Dolev-Yao threat model [36] for the active attacker. Under this model, the attacker is a powerful adversary with a high computational capability, full knowledge of

the protocol, and the ability to observe and manipulate all communications channels in use. To simulate this capability, the prover allows the attacker to perform the following actions:

- Perform all attacks available to a passive attacker
- Control all communications channels specified in the model
- Collect packets over multiple runs of the protocol
- Inject, manipulate, replay, or delete messages at any point with information gathered from previous runs

In the present work, the use of the tool was explicitly focused on proving channel security, namely that Alice and Bob are able to exchange session key shards securely and confidentially, without the attacker learning the secret key. Figure 3 illustrates the model constructed under Verifpal, mirroring Steps 0–7 in Figure 2, with all PUF-specific operations excised for clarity.

Confidentiality and equivalence are tested by specifying queries at the end of the model description, which the attacker attempts to falsify through repeated runs of the protocol. The protocol is deemed secure if confidentiality of the secret key is preserved and the only attack possible is a denial of service. Note that an adversary in complete control of the communications channels can always prevent Alice and Bob from receiving the necessary information required to authenticate each other, thus causing a denial of service, irrespective of security protections inherent to and external to the protocol.

Queries tested by Verifpal are depicted in Figure 4. The first two confidentiality queries establish secrecy of the session key shards, while the equivalence establishes that packets sent across the communications channel were not tampered with (denial of service). As expected, Verifpal successfully verifies the confidentiality statements and fails to verify the equivalence statement. Therefore, secrecy of the session was formally proven under all active and passive channel attacks. PUF-related information and key update steps could not be tested due to the tool's inability to model PUF-based primitives. As such, it was not possible to formally prove key freshness, even though each HK_x is used only once and static analysis confirms freshness guarantees.

6.10. Freshness and Replay

The key update mechanism relies on a unique (Chlg, HK) tuple, which is used only once per authentication and renewed on every authentication, mitigating all replay attacks. This use-only-once implementation practice is also utilized during enrollment, during the P3B MA and SKG operations, and one-time nonces are used at critical points throughout the protocol to ensure that all aspects of the in-field authentication are impervious to replay attacks.

Step 6 in Figure 2 refreshes HK_x after each successful authentication, guaranteeing freshness and preventing replay attacks on Alice and Bob. Nonetheless, an attacker may delete AS's response to Alice in Step 6. While this would cause the authentication attempt to fail, it would also prevent Alice from receiving the challenge information needed to generate HK_{A2}, thus causing a synchronization error in the next iteration where Alice would use HK_{A1} as the session key with AS whereas the AS expects to use HK_{A2}. The protocol prevents this attack via Step 10, where AS only updates the database entry for Alice once Alice confirms her receipt of HK_{A2}.

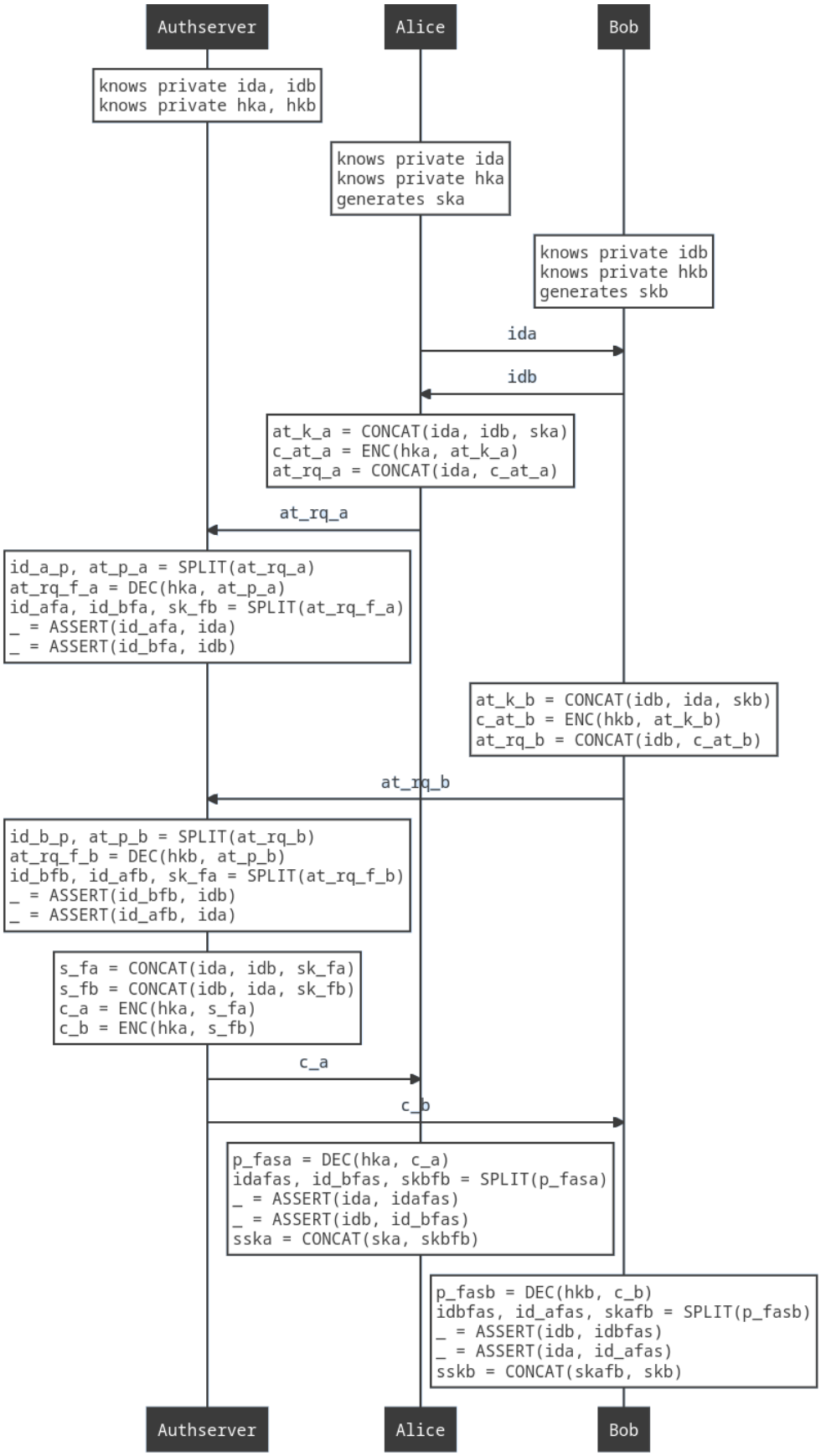

Figure 3. A formally verified model of in-field authentication.

```
queries[
    confidentiality? ska
    confidentiality? skb
    equivalence? sska, sskb
]
```

Figure 4. Established queries for formal verification.

6.11. AS Spoofing and Key Exfiltration

An adversary could attempt to spoof the AS to manipulate Alice and Bob transactions. This attack would be detected by the AS at Step 3 during decryption of C_1 or C_2. A packet injected by the attacker using an unknown key would be detected by Alice and/or Bob during decryption. For an attacker to succeed, HK_x keys would need to be leaked at either the device level or at the AS. Given that the AS is a trusted infrastructure component protected by enterprise security controls such as network segmentation and access control, the latter vector is deemed mitigated. In addition to typical enterprise-level controls, the AS is also equipped with a SHIC PUF; ergo, the AuTk_{DB} database is encrypted in persistent storage by a KEK key produced by the PUF at boot-up. The adversary would need to extract an in-memory copy of the KEK key to obtain a plaintext copy of AuTk_{DB} as PUFs do not store keys in persistent storage.

7. Experimental Results

The proposed protocol is implemented using a co-design methodology, with both software (C code) and hardware (Verilog) components defining various aspects of the protocol. The PUF and KEK algorithms are implemented entirely in Verilog, and synthesized to a Xilinx Zynq 7010 system-on-chip (SoC) FPGA embedded on the Digilent ZYBO Z7-10 board [37]. The PUF uses high-speed general-purpose input-output (GPIO) registers to exchange challenges and KEK bitstrings with software components of the protocol running on the embedded ARM microprocessor. The Linux operating system running on the microprocessor provides a TCP-IP network stack to enable communications between protocol entities.

The IA component is implemented on a Dell PowerEdge T440 server equipped with 32, 1.8 GHz processors and 128 GB of main memory. The implementation test bed possesses the following characteristics:

- The programmable logic (PL) component of the Zynq 7010 FPGA board is programmed with an instance of a SiRF PUF and KEK key generation algorithm, namely, those described for the SiRF PUF in [29]. A C program running under Linux coordinates communications between network entities and the PUF hardware. A microSD card provides NVM storage.
- The AS is also implemented on a ZYBO Z7-10 board, with the PL component programmed with an instance of the PUF. The C program implementation of AS is a multi-threaded application enabling concurrent communication through sockets with multiple customer devices, and with the IA. The AS utilizes sqlite3 databases to implement the AuTk_{DB} and SESSION_{DB} components of the MAKE protocol.
- The IA is implemented as a multi-threaded application with socket communication channels to customer devices and to AS. It also utilizes a sqlite3 database to implement the CRP_{DB} component.
- An openSSL implementation of the AES encryption algorithm is used for all protocol encryption and decryption operations. AES is configured to use a 256-bit key and CBC mode.
- An openSSL implementation of the SHA-3 256-bit hashing algorithm is used for all protocol hashing operations.

A series of experiments are carried out to evaluate the scalability of the protocol in which Enrollment and InField protocol operations are run with different numbers of AT. The time required to carry out the various steps of the authentication protocol is measured by having Alice and Bob's device perform repeated authentications. Four experiments are performed with the AS AuTk$_{DB}$ database configured with 10, 100, 1000, and 10,000 AT. The PUF KEK bitstrings collected over the duration of the run are analyzed to determine the statistical quality of the bitstrings.

7.1. KEK Bitstring Statistical Analysis

The Entropy and MinEntropy statistics associated with the KEK bitstrings produced during the execution of the '10,000 AuTk experiment' are plotted as a function of time in Figure 5. The results for the 10,000 KEK bitstrings generated by Alice and Bob are plotted as two superimposed curves and are analyzed in groups according to the number generated during each 1 min time interval (x-axis) over the duration of the run. The duration of the protocol run is ∼270 min. Therefore, each group includes ∼37 KEK bitstrings. The equations for Entropy and MinEntropy are given by Equations (3) and (4), where p and q are the probabilities of 0 and 1 occurring in a KEK bitstring X of length 256 bits, respectively, and $max(p, q)$ returns the larger of p or q. Given the bits in the bitstring are either 0 or 1, it follows that $q = 1 - p$.

$$\mathbf{H}(\mathbf{X}) = -p \times log_2(p) - q \times log_2(q) \tag{3}$$

$$\mathbf{H}_\infty(\mathbf{X}) = -log_2(max(p, q)) \tag{4}$$

The ideal value for Entropy and MinEntropy is 1.00, which is nearly achieved for Entropy with an average value of 0.997, computed using all 10,000 256-bit bitstrings concatenated as a 2,560,000-bit bitstring. MinEntropy varies over the range of 0.92 to 0.95, which indicates that in the worst case, each KEK bit generates on average 0.93 bits of Entropy. According to the literature, the Entropy and MinEntropy statistics obtained in these experiments indicate the KEK bitstrings are of cryptographic quality.

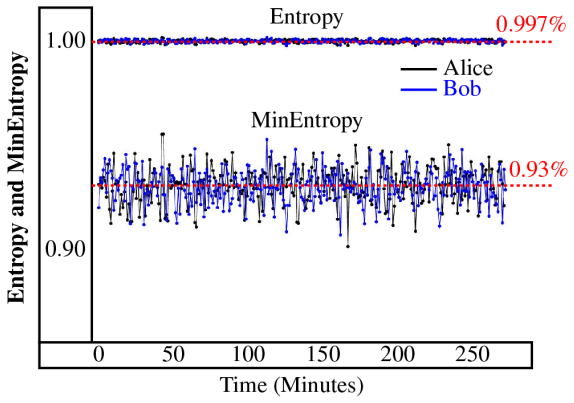

Figure 5. Entropy and MinEntropy of 10,000 KK$_A$ that are generated over a 4.5 h run of the MAKE protocol.

The inter-authentication bitstring hamming distances (HD$_{IAB}$) of the KEK bitstrings are plotted in Figure 6, also as a function of one-minute time intervals. The sequences of KKs produced by Alice and Bob are analyzed separately. HD$_{IAB}$ is computed by pairing the bitstrings within each group from the same device under all combinations. The number of bit-wise differences is summed across all pairing combinations and then divided by the total number of bits in the bitstrings from the group. Equation (5) gives the expression for HD$_{IAB}$, converted to a percentage as shown in the figure. Here, NBS represents the number

of bitstrings (approx. 37 per group), NB is the number of bits per bitstring (256), TNB is the total number of bits in each bitstring group, and BS denotes the bitstrings themselves.

$$\mathbf{HD_{IAB}} = \frac{\sum_{i=1}^{NBS} \sum_{j=i+1}^{NBS} \sum_{k=1}^{NB} (BS_{i,k} \oplus BS_{j,k})}{TNB} * 100 \quad (5)$$

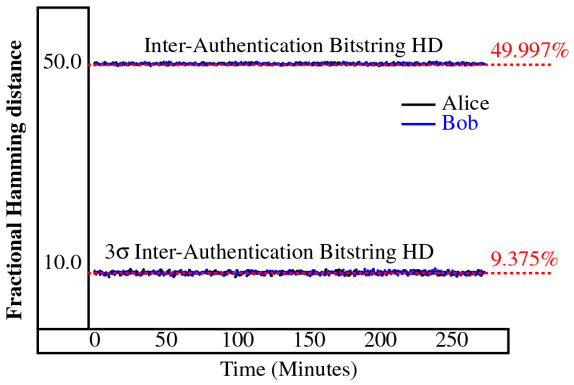

Figure 6. Intra-chip Hamming Distance statistics for 10,000 KK from Alice and Bob's devices (superimposed curves), produced in one minute time intervals over a 4.5 h duration.

The HD_{IAB} values are very close to the ideal value of 50%, with the overall mean value across all bitstring pairing combinations (10,000 × 9999/2 pairings) given as 49.997%. The $3\sigma_{IAB}$ values for each group are also depicted in the lower portion of the graph. The expected value is given by the binomial expression in Equation (6). With NB = 256, the expected value is 9.375%. The data plotted in the figure is a very good match to the expected value.

$$3\sigma_{IAB} = \frac{3 \times \sqrt{NB \times 0.25}}{NB} \times 100 \quad (6)$$

Alice and Bob's 256-bit bitstrings are concatenated to form bitstrings of length larger than 50,000 bits and evaluated using the NIST statistical test suite. All applicable NIST tests for bitstrings of this size passed as well as the p-value-of-the-p-value tests. A separate set of experiments is carried out on a set of 120 PUF instances on FPGAs in which KEK regeneration is evaluated across extended industrial range temperatures, from −40 °C to 100 °C. The results obtained indicate that the probability of a bit-flip error is less than 1×10^{-8}, i.e, one chance in 100 million.

The MAKE enrollment operations take approx. 1.75 s per AT generation. For example, the total amount of time to generate 10,000 AT for both Alice and Bob and for the AT to be transmitted and stored in the $AuTk_{DB}$ on the AS is 4.87 h.

The run times for the MAKE In-Field protocol operations are given in Figure 7, partitioned into a sequence of 7 protocol steps along the x-axis. The ordering of the steps given here is consistent with, but not identical, to the sequence of operations shown in Figure 2. The bar heights give the run times in seconds of 4 experiments, each carrying out different numbers of Alice-Bob authentications as shown along the y-axis.

The run times are nearly constant at ∼1.8 s per authentication for the first three experiments, with the $AuTk_{DB}$ database enrolled with 10 through 1000 AT, respectively. The run time increases to ∼2.2 s for the 10,000 AT experiment. From the bar graph, the increase is attributed to operations carried out in step 4, where AS performs a database search before it can return the C_A and C_B packets to Alice and Bob. With the $AuTk_{DB}$ database populated with 10,000 AT, the search process adds ∼0.4 s to the overall runtime.

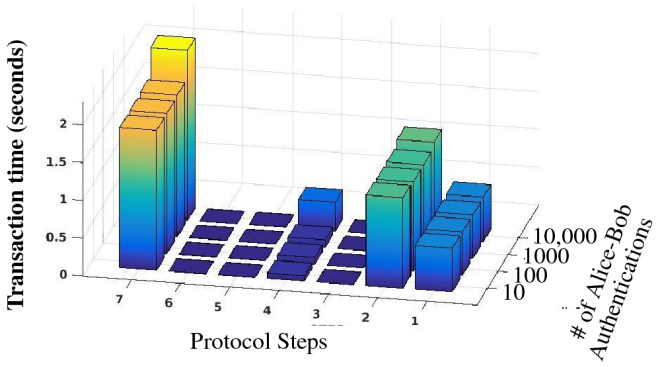

Figure 7. Transaction times of PUF-MAKE In-Field protocol steps: (1) SK_T generation; (2) KK_A generation and hash to HK_A; (3) C_1, C_2 creation and transmission to AS; (4) DB search, C_A, C_B creation and transmission from AS; (5) C_A, C_B extraction and SK creation; (6) Encrypt-transmit-decrypt test message with SK; and (7) total authentication time.

Although only one AS device is used in our experiments, the proposed system architecture can support an arbitrary number of authentication servers (AS), which enables the protocol to scale up to millions of customer devices without incurring additional database search time penalties. The $AuTk_{DB}$ database in this type of distributed environment would be populated with local customer device authentication tokens (AT) to enable value-exchange operations to occur with local vendors, in combination with AT from popular global retailers, such as Amazon.

7.2. Communication Complexity

The device-side communication complexity of in-field authentication can be expressed as $1N_{PUF} + 1N_H + 3N_{ENC} + 2N_{DEC}$, which denote PUF, hash, block cipher encryption, and decryption operations, respectively. Note that the storing and retrieving of Chlng and helper data from the NVM, as well as the XOR operation used in session key construction, have only minor computation costs and are therefore not included. Similarly, the computational complexity of the AS per iteration of the protocol can be expressed as: $2N_{DBS} + 1N_{DBC} + 2N_{ENC} + 2N_{DEC}$, where DBS refers to a DB search operation and DBC refers to a DB record creation operation. Encryption and decryption operations are constant time, $O(1)$, hashing with SHA-3 is a linear operation with a complexity of $O(n)$. The generation of the PUF response, N_{PUF}, is $O(n)$, where n is the number of response bits generated. The database search operation is characterized as $O(n \cdot log(n))$ for a SELECT query applied to a single table containing n records. Note that in terms of absolute time, N_{PUF} is the dominant operation. On the instantiated PUF device, the response regeneration time to a PUF challenge takes 1 s, as shown in the performance analysis section.

The total communication complexity is presented in Table 1 and compares favorably to a recently proposed scheme. Zheng et al. [21] present the complexity analysis of other, more expensive, peer-to-peer authentication protocols, which are omitted here. Note that the SiRF PUF uses an error avoidance algorithm that runs in linear time, which is included in the N_{PUF} metric, in contrast to the 'Rep' operation of ECC, which is unspecified but can be on order of $O(n^2)$ for an n-bit response. The symbol N_{ECCR} in the table refers to the heavier-weight 'Rep' operation of ECC, while N_{ECCG} refers to the lighter-weight 'Gen' operation.

Table 1. Comparison of complexity between the state of the art (SOTA) and our proposed approach.

Scheme	Single Node	Server
SOTA [21]	$1N_{PUF} + 3N_H + 1N_{ECCR} + 2N_{ENC} + 2N_{MAC}$	$2N_{ENC} + 2N_{DBS} + 1N_{ECCG}$
This work	$1N_{PUF} + 1N_H + 3N_{ENC} + 2N_{DEC}$	$2N_{DBS} + 1N_{DBC} + 2N_{ENC} + 2N_{DEC}$

7.3. Resource Utilization

The number of LUTs utilized to implement the SiRF PUF authentication, session key generation, KEK LLK, and TRNG operations on the Xilinx Zynq 7010 system-on-chip (SoC) device is 5842, while the AES encryption and SHA-3 hashing algorithms consume 6937, which is approximately 77% of the 17,600 LUTs available. Additional details regarding programmable logic resource utilization are available in [38].

A C program running on the ARM Cortex A-9 processor co-located on the Xilinx Zynq 7010 SoC device implements all network, database, and bookkeeping operations on customer devices. The size of the binary is approximately 150 KB. The Chlng and HD stored in an NVM by customer devices during enrollment is 4 bytes and 512 bytes, respectively. The Chlngs for the SiRF PUF are represented as a 4 byte seed to an LFSR, which is used to look up challenge vectors from a $Chlng_{DB}$, which is approximately 1 megabyte in size [38]. The HD is sufficient to generate a 256-bit KEK LLK. The size of each AT in the AuTk_{DB} database on the server is given as 4 bytes for the ID, 32 bytes for the HK, 4 bytes for the Chlng, and 512 bytes for the HD.

8. Conclusions

Herein, we propose and evaluate a light-weight PUF-based mutual authentication and key exchange (MAKE) protocol designed for high-security applications. The protocol is composed of three distinct phases, namely provisioning, enrollment, and in-field interactive authentication. Evaluation was performed on a set of three FPGAs that incorporate programmable logic instantiations of strong (SHIC) PUF and KEK algorithms and a desktop server. Run times of the interactive in-field authentication operations are upper bound at 2.2 s with an authentication token database containing 20,000 authentication tokens. The computational complexity was analyzed and compared favorably to the state of the art. A detailed security analysis was conducted, in particular channel security for the in-field authentication protocol was formally proven to be cryptographically secure under the Dolev-Yao adversary model, making PUF-MAKE a good candidate for high-security applications such as digital currency transactions for goods and services.

Author Contributions: All authors contributed equally to all aspects of this research, including experimental setup, data collection and manuscript editing.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guin, U.; Singh, A.; Alam, M.; Cañedo, J.; Skjellum, A. A Secure Low-Cost Edge Device Authentication Scheme for the Internet of Things. In Proceedings of the 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), Pune, India, 6–10 January 2018; pp. 85–90. [CrossRef]
2. Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. In Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, Las Vegas, NV, USA, 14–17 October 2001; pp. 136–145. [CrossRef]
3. Diffie, W.; Van Oorschot, P.C.; Wiener, M.J. Authentication and authenticated key exchanges. *Des. Codes Cryptogr.* **1992**, *2*, 107–125. [CrossRef]
4. Bellare, M.; Rogaway, P. Entity Authentication and Key Distribution. In *Advances in Cryptology—CRYPTO' 93*; Springer: Berlin/Heidelberg, Germany, 1993; pp. 232–249.
5. Bellare, M.; Rogaway, P. Provably Secure Session Key Distribution: The Three Party Case. In Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing. Association for Computing Machinery, Las Vegas, NV, USA, 29 May–1 June 1995; pp. 57–66.
6. Bellare, M.; Pointcheval, D.; Rogaway, P. Authenticated Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology—EUROCRYPT 2000*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 232–249.
7. Okamoto, T. Authenticated Key Exchange and Key Encapsulation in the Standard Model. In *Advances in Cryptology—ASIACRYPT 2007*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 474–484.

8. Law, L.; Menezes, A.; Qu, M.; Solinas, J.; Vanstone, S. An Efficient Protocol for Authenticated Key Agreement. *Des. Codes Cryptogr.* **1995**, *28*, 119–134. [CrossRef]
9. Krawczyk, H. HMQV: A High Performance Secure Diffie-Hellman Protocol. In *Advances in Cryptology—CRYPTO 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 546–566.
10. LaMacchia, B.; Lauter, K.; Mityagin, A. Stronger Security of Authenticated Key Exchange. In Proceedings of the International Conference on Provable Security, Wollongong, Australia, 1–2 November 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–16.
11. Delvaux, J.; Peeters, R.; Gu, D.; Verbauwhede, I. A Survey on Lightweight Entity Authentication with Strong PUFs. *ACM Comput. Surv.* **2015**, *48*, 1–42. [CrossRef]
12. Idriss, T.; Bayoumi, M. Lightweight highly secure PUF protocol for mutual authentication and secret message exchange. In Proceedings of the 2017 IEEE International Conference on RFID Technology & Application (RFID-TA), Warsaw, Poland, 20–22 September 2017; pp. 214–219. [CrossRef]
13. Mahalat, M.H.; Saha, S.; Mondal, A.; Sen, B. A PUF based Light Weight Protocol for Secure WiFi Authentication of IoT devices. In Proceedings of the 2018 8th International Symposium on Embedded Computing and System Design (ISED), Cochin, India, 13–15 December 2018; pp. 183–187. [CrossRef]
14. Mahalat, M.H.; Karmakar, D.; Mondal, A.; Sen, B. PUF Based Secure and Lightweight Authentication and Key-Sharing Scheme for Wireless Sensor Network. *J. Emerg. Technol. Comput. Syst.* **2021**, *18*, 1–23. [CrossRef]
15. Che, W.; Martin, M.; Pocklassery, G.; Kajuluri, V.K.; Saqib, F.; Plusquellic, J. A Privacy-Preserving, Mutual PUF-Based Authentication Protocol. *Cryptography* **2017**, *1*, 3. [CrossRef]
16. 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.* **2019**, *16*, 424–437. [CrossRef]
17. Chatterjee, U.; Chakraborty, R.S.; Mukhopadhyay, D. A PUF-based secure communication protocol for IoT. *ACM Trans. Embed. Comput. Syst. (TECS)* **2017**, *16*, 1–25. [CrossRef]
18. Wallrabenstein, J.R. Practical and Secure IoT Device Authentication Using Physical Unclonable Functions. In Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud, Vienna, Austria, 22–24 August 2016; pp. 99–106. [CrossRef]
19. Yu, M.D.; Hiller, M.; Delvaux, J.; Sowell, R.; Devadas, S.; Verbauwhede, I. A Lockdown Technique to Prevent Machine Learning on PUFs for Lightweight Authentication. *IEEE Trans.-Multi-Scale Comput. Syst.* **2016**, *2*, 146–159. [CrossRef]
20. Zhang, J.; Qu, G. Physical Unclonable Function-Based Key Sharing via Machine Learning for IoT Security. *IEEE Trans. Ind. Electron.* **2020**, *67*, 7025–7033. [CrossRef]
21. Zheng, Y.; Liu, W.; Gu, C.; Chang, C.H. PUF-Based Mutual Authentication and Key Exchange Protocol for Peer-to-Peer IoT Applications. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 3299–3316. [CrossRef]
22. Van Herrewege, A.; Katzenbeisser, S.; Maes, R.; Peeters, R.; Sadeghi, A.R.; Verbauwhede, I.; Wachsmann, C. Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-Enabled RFIDs. In Proceedings of the Financial Cryptography and Data Security, Kralendijk, Bonaire, 27 Februray–2 March 2012; Springer: Berlin/Heidelberg Germany, 2012; pp. 374–389.
23. Zerrouki, F.; Ouchani, S.; Bouarfa, H. T2S-MAKEP and T2T-MAKEP: A PUF-based Mutual Authentication and Key Exchange Protocol for IoT devices. *Internet Things* **2023**, *24*, 100953. [CrossRef]
24. Tun, N.W.; Mambo, M. Secure PUF-Based Authentication Systems. *Sensors* **2024**, *24*, 5295. [CrossRef] [PubMed]
25. Rührmair, U.; Sölter, J.; Sehnke, F. On the Foundations of Physical Unclonable Functions. Cryptology ePrint Archive, Paper 2009/277. 2009. Available online: https://eprint.iacr.org/2009/277 (accessed on 8 August 2024).
26. Plusquellic, J.; Tsiropoulou, E.E.; Minwalla, C. Privacy-Preserving Authentication Protocols for IoT Devices Using the SiRF PUF. *IEEE Trans. Emerg. Top. Comput.* **2023**, *11*, 918–933. [CrossRef]
27. Rührmair, U.; Busch, H.; Katzenbeisser, S. Strong PUFs: Models, Constructions, and Security Proofs. In *Towards Hardware-Intrinsic Security: Foundations and Practice*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 79–96. [CrossRef]
28. van Dijk, M.; Rührmair, U. Physical Unclonable Functions in Cryptographic Protocols: Security Proofs and Impossibility Results. Cryptology ePrint Archive, Report 2012/228 2012. Available online: https://ia.cr/2012/228 (accessed on 10 August 2024).
29. Plusquellic, J. Shift Register, Reconvergent-Fanout (SiRF) PUF Implementation on an FPGA. *Cryptography* **2022**, *6*, 59. [CrossRef]
30. Rührmair, U.; Sehnke, F.; Sölter, J.; Dror, G.; Devadas, S.; Schmidhuber, J. Modeling Attacks on Physical Unclonable Functions. In Proceedings of the 17th ACM Conference on Computer and Communications Security, Chicago IL, USA, 4–8 October 2010; pp. 237–249. [CrossRef]
31. Rührmair, U.; Xu, X.; Sölter, J.; Mahmoud, A.; Majzoobi, M.; Koushanfar, F.; Burleson, W. Efficient Power and Timing Side Channels for Physical Unclonable Functions. In Proceedings of the Cryptographic Hardware and Embedded Systems, Busan, Republic of Korea, 23–26 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 476–492.
32. Delvaux, J. Machine-Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF-FSMs. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 2043–2058. [CrossRef]
33. Ramsay, C.; Lohuis, J. *TEMPEST Attacks against AES*; Fox-IT: Fremont, CA, USA, 2017.
34. Das, D.; Sen, S. Electromagnetic and Power Side-Channel Analysis: Advanced Attacks and Low-Overhead Generic Countermeasures through White-Box Approach. *Cryptography* **2020**, *4*, 30. [CrossRef]

35. Kobeissi, N.; Nicolas, G.; Tiwari, M. Verifpal: Cryptographic Protocol Analysis for the Real World. In Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop, New York, NY, USA, 9 November 2020; p. 159.
36. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
37. Digilent Corporation. *ZYBO-Z7 Reference Manual*; Digilent Corporation: Pullman, WA, USA, 2021.
38. Bean, B.; Minwalla, C.; Tsiropoulou, E.E.; Plusquellic, J. PUF-based Digital Money with Propagation-of-Provenance and Offline Transfers between Two Parties. *J. Emerg. Technol. Comput. Syst.* **2024**, *20*, 1–26. [CrossRef]