Privacy-Preserving Authentication Protocols for IoT Devices Using the SiRF PUF

Jim Plusquellic, *Member, IEEE,* Eirini Eleni Tsiropoulou, *Senior, IEEE,* and Cyrus Minwalla

Abstract—Authentication between IoT devices is important for maintaining security, trust and data integrity in an edge device ecosystem. The low-power, reduced computing capacity of the IoT device makes public-private, certificate-based forms of authentication impractical, while other lighter-weight, symmetric cryptography-based approaches, such as message authentication codes, are easy to spoof in unsupervised environments where adversaries have direct physical access to the device. Such environments are better served by security primitives rooted in the hardware with capabilities exceeding those available in cryptography-only frameworks. A key foundational hardware security primitive is the physical unclonable function or PUF. PUFs are well known for removing the need to store secrets in secure non-volatile memories, and for providing very large sets of authentication credentials. In this paper, we describe two PUF-based mutual authentication protocols rooted in the entropy provided by a strong PUF. The security properties of the authentication protocols, called COBRA and PARCE, are evaluated in hardware experiments on SoC-based FPGAs, and under extended industrial-standard operating conditions. A codesign-based system architecture is presented in which the SiRF PUF and core authentication functions are implemented in the programmable logic as a secure enclave, while network and database operations are implemented in software on an embedded microprocessor.

Index Terms—Physical unclonable function, PUF-based Authentication

1 Introduction

Edge devices provide access points to core networks for IoT devices, which are often cheap, low-power, embedded systems deployed in unsupervised environments. IoT devices often lack the computing and power resources to leverage state-of-the-art software security primitives and protocols, and are therefore an attractive target for adversaries. A compromised IoT device represents a threat to the edge device and core network infrastructure because malicious actors can potentially gain unauthorized access and elevated privileges to back-end resources and/or supply network applications with compromised data.

The cryptographic primitives and protocols used in such systems are light-weight variants of compute-heavy asymmetric core network security functions. Authentication via message authentication codes and shared symmetric keys are often used due to resource and power constraints. Although software-based security and trust platform infrastructures are attractive from a developer's perspective, they cannot provide sufficient protection against active attackers who can employ techniques to exfiltrate device secrets. Stored secrets represent the root-of-trust in such systems, and their most significant vulnerability.

A hardware-based root-of-trust which derives device secrets from the random variations introduced by physical imperfections in modern microelectronic fabrication processes engenders a device with unclonable authentication credentials. The physical unclonable function or PUF embodies the class of hardware-based security functions, which are capable of providing significant enhancements to the security profile of a device over software-only approaches. Strong PUFs extend the capabilities of the profile by exponentially expanding on the number of challenge-response-pairs (CRPs) available per device. A large set of CRPs enable a strong form of authentication where credentials are used only once.

In this paper, two PUF-based, privacy-preserving, mutual authentication protocols are described and evaluated, called correlation-based robust authentication (COBRA) [1] and PUF-based authentication in resource constrained environments (PARCE) [2]. Both protocols are built on top of a strong PUF called the shift-register, reconvergent-fanout (SiRF) PUF [3], and consequently, both inherit the security properties of the SiRF PUF. The COBRA protocol utilizes only helper data bitstrings generated by the SiRF PUF for authentication, while PARCE utilizes a novel PUF-based encryption technique. The specific contributions of this paper are summarized as follows:

1) The security properties of COBRA and PARCE are evaluated in two distinct FPGA test-beds and experiments, the first while subjecting the devices to extended industrial-standard temperature and voltage variations, and the second while exercising the protocols in a realistic real-time, client-server-based execution environment.
2) The information leakage of both protocols is assessed, and a simple post-processing operation is proposed to significantly strengthen device

- *J. Plusquellic is with the Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, 87131.*

  E-mail: jimp@ece.unm.edu

- *E. E. Tsiropoulou is with University of New Mexico, Albuquerque, NM, 87131.*

- *C. Minwalla is with Bank of Canada, Ottawa, ON, Canada.*

Manuscript received Dec 7, 2022; revised xxx 99, 9999.

anonymity with no negative impact on the level of certainty associated with the authentication decisions made by the server and device.

1.1 Related Work

The primary threat to authentication protocols that utilize a strong PUF with an unprotected interface, i.e., no cryptographic primitives, is a model-building (MB) attack [4] [5] [6] [7] [8] [9]. Previously proposed methods attempt to defeat MB attacks by adding internal complexity to the strong PUF architecture [10] [11] [12], by obfuscating challenges [13] [14], by introducing reconfiguration options [15] [16], or by injecting noise into the responses [17] [18].

A wide range of light-weight, PUF-based authentication protocols proposed prior to 2015 are described and assessed in [19]. More recently, the authors of [20] propose an authentication protocol that utilizes only challenges in the message exchange, thereby eliminating the response bitstring. The protocol requires the server to search for a pair of challenges that produce the response bit from the device-generated challenge when the response bits from the challenge pair are XOR'ed. Although the protocol eliminates the response bitstring in the message exchange, it is not privacy-preserving and requires a reliable soft model of the PUF to be stored on the server given the noise amplification commonly observed when using XOR operations on response bits.

The authors of [21] propose a non-privacy-preserving, lightweight, PUF-based mutual authentication protocol which authenticates during the connection establishment phase of WiFi using three CRPs. XOR operations in combination with a router-generated nonce are used to encrypt challenges and responses between device and router, and to derive a set of CRPs for the next authentication. The protocol assumes error-free responses (no helper data scheme is proposed) and requires a secure hash function to ensure data integrity.

A PUF-based authentication and key management protocol for IoT is proposed in [22], improving upon on the attack resilience and performance overhead of the previous method [23]. Elliptic curve cryptography (ECC) is used to create shared keys among IoT nodes with the assistance of a verifier. The scheme requires a certificate management system (PKI or equivalent) and tamper-resistant hardware in devices 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 [24], which relies on zero knowledge proofs to implement a one-way device authentication technique. A PUF-based El-Gamal algorithm is proposed for message encryption as well as a PUF-based digital signature scheme. In parallel, Yu et al. [25] propose a scheme that is designed to prevent an adversary from obtaining sufficient CRPs to carry out model-building attacks. The proposed approach limits the number of authentications to the number of CRPs stored in the database, requiring either reuse or re-enrollment of the PUF.

A CRP noise-injection technique is proposed in [17] where stable responses are recorded during enrollment, while in-field response generation intentionally injects noise into the response bits. Given the very large CRP space of the PUF, noise injection is tolerable for some fraction of the responses. Wang et al. [18] [26] propose a PUF-based authentication protocol that utilizes internally generated signals from a free-running LFSR in a dynamic response mechanism to generate sub-challenges to a strong PUF, as a means of preventing attackers from obtaining valid CRPs in a MB attack. Authentication by the server is accomplished using a matching algorithm that is able to associate the dynamic response to an enrolled device. By contrast, the COBRA protocol avoids the need to obfuscate the challenge by eliminating the response bitstring altogether in the message exchange protocol. Given the exposed response bitstrings are central to MB attacks, their elimination defeats the traditional approach and requires an approach that uses helper data bitstrings instead. The machine learning (ML) algorithm would be tasked with predicting helper data bitstrings for unseen challenges. As we will show, the helper data bitstrings generated by SiRF PUF do not leak information about the response bitstring, and therefore, the model on which a ML algorithm would need to be based is unclear.

2 SiRF PUF Overview

The proposed COBRA and PARCE authentication protocols use the shift-register, reconvergent-fanout (SiRF) PUF as a hardware primitive for bitstring generation [3]. A block diagram of the SiRF PUF architecture is shown in Fig. 2. Symbol definitions used in the figures are given in Fig. 1. The SiRF netlist component defines a complex network of paths, which is synthesized into a region on the FPGA. The paths are composed of FPGA primitive components including shift registers, look-up tables (LUTs) and MUXs.

DV_x, $DV_{R/F}$, DVD, DVD_c, DVD_{co}: Arrays of 16-bit fixed-point delay values
p: LFSR seed, range and threshold parameters
SF: 2048-byte array of signed 8-bit integers
HD: 2048-bit array of binary values
v or **vec**: 771-bit array of binary values
$\mathbf{Chlng_a}\{v\}$: Set of challenge vectors ***v***
$\mathbf{Chlng_b}\{\mathbf{SF, p}\}$: One set of **SF** and **p**
F_x: FPGA device ID
XMR: odd integer constant, e.g., 1, 3, 5, 7; TMR = 3

Fig. 1: Symbols definitions.

Variations in the delays of paths through this network represent the source of entropy leveraged by the SiRF PUF. The structure of the physical paths is defined using the vector component of the SiRF challenge, labeled $\mathbf{Chlng_a}\{\mathbf{v}\}$, where vector **v** consists of 771 bits. The challenge bits connect to a set of MUXes distributed throughout the netlist, which enables different paths to be tested, i.e., the bits reconfigure the netlist. Each challenge vector creates a unique set of 32 paths through the netlist, which are drawn from a set of 2^{24} (16 million) possible path configurations. No placement or routing constraints are used during synthesis and place-and-route, i.e., the SiRF PUF does not require the paths to be identically-designed. The bias introduced by differences

in the lengths of the paths are removed by post-processing operations described in the following sections.

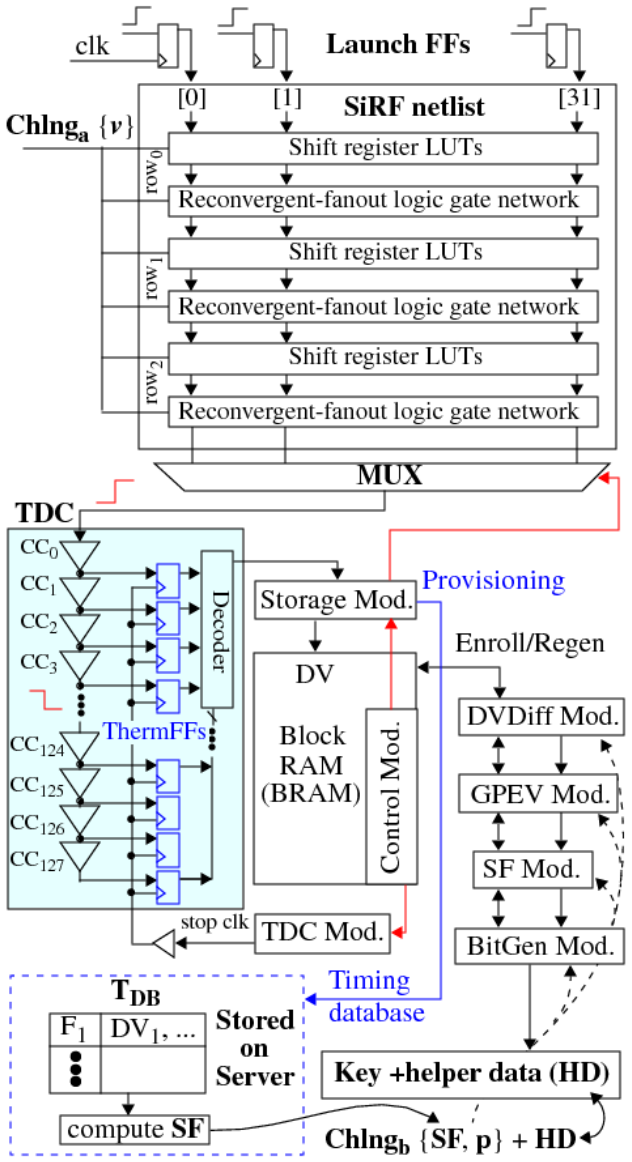


Fig. 2: Shift-register, reconvergent-fanout (SiRF) PUF architecture and algorithm.

2.1 Path Delays

Paths are tested using the **Launch FFs** shown along the top of Fig. 2. The rising (0-to-1) transitions introduced by the **Launch FFs** propagate through the netlist to the inputs of a 32-to-1 MUX shown along the bottom of the figure. The controlling state machine, labeled *Control Mod.*, selects and routes one of the paths to a time-to-digital converter (TDC). The TDC utilizes a sequence of carry-chain elements in the FPGA to measure the path delay at high resolution (approx. 18 ps). The carry chain extends the path-under-test, allowing the signal transition to propagate along the chain until the *TDC Mod.* asserts the *stop clk* signal. The rising edge of *stop clk* captures the current state of the *ThermFFs* within the TDC module, which record the position of the propagating signal in the delay chain.

The *Decoder* reads out a thermometer code from the *ThermFFs*, which is represented as a sequence of 1's in the initial portion of the TDC, followed by 0's in the remaining portion, and converts the number of 1's into a digitized delay value (DV). The carry chain is typically configured to a length of 128 stages, which limits the maximum path delay that can be measured to approximately 2.3 ns. In order to accommodate SiRF netlist path delays of up to 20 ns, the TDC includes components that extend its timing capability (see [27] for details).

The *Control Mod.* increases precision by collecting multiple integer-based delay values (DV) for each of the paths, and then averages them to create 16-bit fixed-point DVs, which are stored in the Block RAM (BRAM). The SiRF PUF post-processing algorithm is implemented as a sequence of modules shown on the right side of Fig. 2 and by the flow diagram in Fig. 3. All four of the modules apply linear operations to the DV, with several of them accepting input parameters **p** and ancillary data **SF**. The input parameters represent a second component of the challenge, labeled as $\mathbf{Chlng_b}$ along the bottom of the figure. The parameters expand the CRP space and increase the statistical quality of the generated bitstrings and keys.

2.2 SiRF PUF Algorithm

The *DVDiff* module selects pairs of DVs using a LFSR-based pseudo-random number generator, and creates delay value differences (DVD), as shown in Fig. 3. These pairs are drawn out of the rising edge delay value sets, DV_R, and the falling edge delay value sets, DV_F, stored in the BRAM. The pairing sequence is controlled by two LFSR seed parameters specified in **p**. The global-process and environmental variation (*GPEV*) module reads the DVD and applies a linear transformation to produce DVD_c. The linear transformation reduces chip-to-chip performance differences and undesirable environmentally-induced changes in the path delays. Elements in the DVD distribution are first standardized, i.e., the mean of the distribution is subtracted and then each value is divided by the range of the distribution. The standardized values are then multiplied by another component of **p** called the range parameter [3].

The DVD_c are then shifted vertically using a set of server-computed spread factors (**SF**) to remove the bias that exists because of non-identically-designed paths in the SiRF netlist. The **SF** significantly improve the statistical quality of the bitstrings by removing delay components from the DVD_c that do not contribute to the entropy of the PUF, a.k.a. the median component of each DVD_c. The **SF** are computed by the server using DV stored in a database (see T_{DB} in Fig. 2) for a sample of devices. The **SF** remove the median component and partition each of the DVD_c from the device sample equally over a 0 median value. The server transmits the **SF** to the device, and the device applies them to the DVD_c to produce DVD_{co}.

The *BitGen* module processes the DVD_{co} into a response bitstring. During enrollment, *BitGen* also generates helper data (**HD**), which will be used later during regeneration to reproduce the response bitstring while minimizing, ideally eliminating, any bit-flip errors. Unlike PUFs that use error correction codes, the SiRF PUF algorithm uses a thresholding method to generate the helper data bitstring. The

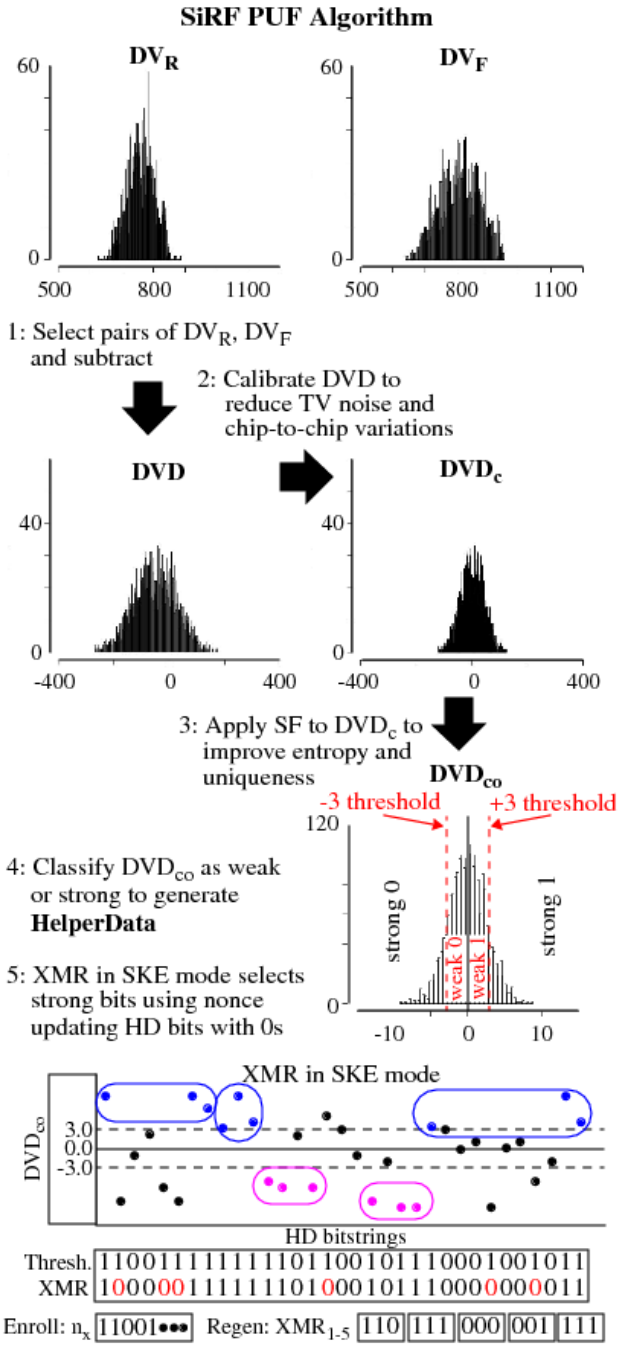


Fig. 3: SiRF PUF data post-processing algorithm.

threshold parameter is tunable and represents the last component of $\mathbf{p}$ in $\mathbf{Chlng_b}$. It divides each of the 0-1 bit regions into strong and weak regions as shown by the DVD_{co} distribution in Fig. 3.

The **HD** bitstring is constructed serially using a sequence of DVD_{co}, as shown by the example labeled *XMR in SKE mode* along the bottom of Fig. 3. A 1 is added to the **HD** bitstring when the corresponding DVD_{co} falls within one of the strong bit regions, and a 0 otherwise (the circled points relate to the XMR process, which is described in the following section). Although not shown, a response bitstring can also be generated depending on the bitstring generation mode. A bit value in this case is assigned based on whether the DVD_{co} is above (1) or below (0) the 0 line. Additional details related to the SiRF PUF algorithm are in [3].

3 COBRA and PARCE Bitstring Generation Algorithms

This section discusses the characteristics of the COBRA and PARCE bitstring generation algorithms and highlights their differences.

3.1 COBRA

COBRA leverages an observation made in our experiments, that the helper data (**HD**) bitstrings generated by the thresholding operation possess random but uniquely identifying information about the device. In contrast to the response bitstring, where bit values reflect whether the DVD_{co} is above (1) or below (0) zero, the **HD** bitstring classifies each response bit as either strong or weak. Given the threshold regions are symmetrically placed around zero completely decouples the **HD** bitstring from the response bitstring, i.e., the **HD** leaks no information about the bit values assigned to the DVD_{co}. This characteristic of the thresholding method will add to the difficulty of learning constituent path delays by ML algorithms, in cases where only the **HD** bitstrings are available, as is true in COBRA.

The basis of the claim that **HD** bitstrings posses uniquely identifying information about the device is rooted in the combination of SiRF PUF entropy and the algorithm's post-processing operations. As described earlier, the *GPEV* operation transforms the DVD to remove variations introduced by process and environmental factors. The compensated DVD_c produced by *GPEV* are very similar to the values measured under nominal conditions. Therefore, the strong/weak classification of the DVD_{co} carried out during bitstring generation are highly correlated when regenerated using the same challenge, and are insensitive to environmental conditions. An authentication technique can leverage this correlation by recording subsets of DV_R and DV_F for each device in a database on a server (given as T_{DB} in Fig. 2), and then request **HD** bitstring generation be performed by both the device and server independently using the same challenge. A simple matching operation can then be used to identify a specific device. Note that the response bitstring in this scenario is discarded.

3.2 PARCE

The authentication mechanism used by PARCE, in contrast to COBRA, makes use of the response bitstring, but does so in a non-traditional manner. The thresholding-based bitstring generation operation described in the previous section is altered to enable a special mode, called secure-key-encoding or SKE.

3.2.1 *Secure Key Encoding (SKE)*

SKE mode takes a random nonce as input and generates the **HD** bitstring needed to reproduce it. Therefore, SKE mode is a PUF-based encryption mechanism that encrypts the plaintext (the nonce) and produces an **HD** bitstring as the ciphertext. The encryption key is represented by the

entropy of the PUF. SKE mode also includes a reliability enhancement component that reduces the probability of bit flip errors when the ciphertext is decrypted. The reliability enhancement component is layered on top of the thresholding method discussed earlier. We refer to this component of SKE as XMR. XMR is modeled after the popular triple-modular-redundancy technique for detecting and correcting errors introduced by faults, except that the redundancy used here is represented in the sequence of strong bits produced by thresholding.

3.2.2 Thresholding

The enrollment operation carried out by SKE is illustrated along the bottom of Fig. 3 using a 3-bit XMR scheme. Here, each XMR box represents a bit generated from processing three bits of the response. The five boxes correspond to the five-bit portion of the PARCE authentication nonce, labeled *Enroll:* n_x, which represents the input that is encrypted by the SKE algorithm. SKE processes the n_x bits, one bit at-a-time, starting with the left-most '1' bit. SKE scans the DVD_{co} left-to-right searching for a strong bit match to the first nonce bit, which it finds at position 1. The algorithm continues to scan the DVD_{co} searching for two more instances of strong '1's, skipping strong '0's and DVD_{co} previously labeled as weak during thresholding.

Strong bits identified in the thresholding-only **HD** bitstring (labeled *Thresh.* in the figure) that are associated with mismatching strong response bits are changed from '1' to '0' during the scan, i.e., they are re-labeled as weak. A mismatching strong response bit occurs when the bit generated by the corresponding DVD_{co} is opposite in value to the current n_x bit. The updated thresholding-only **HD** bitstring is labeled as *XMR* in the figure, and shows three strong 0's have been relabeled as weak because they mismatch with the '1' specified by $n[1]_x$. The three strong '1's of the 3MR sequence needed to encode the $n[1]_x$ bit are circled and highlighted in the figure, and are located at positions 1, 7 and 8. Once $n[1]_x$ is fully encoded, the algorithm starts again with $n[2]_x$. A strong '1' is found at position 9 and the process continues.

3.2.3 Regeneration

Regeneration utilizes the XMR **HD** bitstring as input to reproduce the nonce n_x. The '1's in the XMR **HD** bitstring indicate which DVD_{co} to use in the reconstruction of the XMR_x sequences. Unlike enrollment, an n_x bit is generated for each XMR_x sequence using majority vote among the X bits in each XMR_x bit sequence. This characteristic of XMR adds resiliency to bitstring regeneration because an n_x bit will not flip unless $X/2+1$ or more of the X strong bits flip. For the XMR example, majority voting is able to correct two single bit flip errors in the *Regen:* XMR_x sequences, which enables exact reproduction of the n_x bitstring (note: the regenerated DVD_{co} are not shown).

The PARCE protocol leverages SKE mode by having the server transmit an authentication nonce, n_x, to the device along with a challenge. The device runs SKE enrollment to produce a XMR **HD** bitstring, which is then transmitted back to the server. The server runs SKE regeneration using the XMR **HD** bitstring and the DV data sets from its T_{DB} to produce a set of n'_x, which are then matched against the original n_x. The device from the T_{DB} which produces the best match, i.e., the fewest number of bit flip errors, is identified and authenticated by the server. The details of the protocol are described in the following sections.

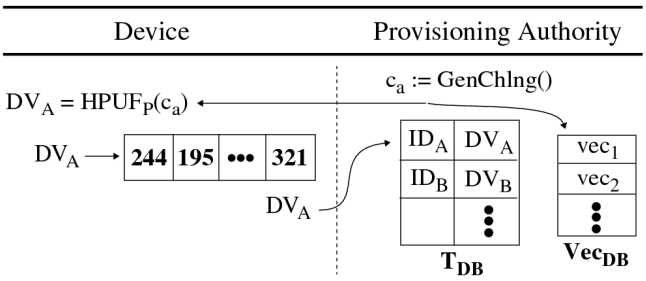


Fig. 4: The SiRF PUF provisioning process.

3.3 Comparison between COBRA and PARCE

Both protocols preserve privacy because the ID of the authenticating device is not sent to the server in the message exchange protocol. Instead, the server carries out an exhaustive search of the T_{DB} to identify the authenticating device. However, for scenarios in which privacy is not important, the authenticating device can transmit its ID as a means of accelerating the authentication process, by virtue of eliminating the database search operation. Moreover, for PARCE, the nonce that is sent by the server to the device can be encrypted using SKE because the identity of the authenticating device is known. This would eliminate the open transmission of n_x, reducing the amount of information available in a MB attack to that provided by the XMR **HD** bitstrings, similar to COBRA.

4 SiRF PUF Provisioning

Many PUFs require a provisioning phase to boot-strap the CRP-based authentication functions. For the SiRF PUF, the provisioning process involves collecting and storing timing data [28], in contrast to other PUFs which create a database of CRPs. The process is shown in Fig. 4, where a provisioning authority generates and transmits challenges to the device. The device applies the challenges to its hardware PUF in provisioning mode (HPUF_p), and transmits the DV to the provisioning authority. The DV and challenges are stored in T_{DB} and Vec_{DB} databases, respectively, along with the identity of the device. The T_{DB} is securely transferred to the server after provisioning (and updated periodically as new devices are added to the system), while the Vec_{DB} is transferred and stored on both the server and devices. Note that a separate programming bitstream is used for provisioning, and is not available to fielded devices.

The provisioning authority derives challenges, c_a, (which are equivalent to sets of $\mathbf{Chlng_a}\{\mathbf{v}\}$ referenced earlier), in advance, and when combined with the corresponding DV, they enable the device to carry out millions of authentications before re-provisioning is required. A large fraction of the challenges, c_a, are referred to as *common pool* challenges because they are applied to every device. The common pool challenges are used to preserve privacy in

COBRA and PARCE as we will discuss. However, other security functions including server authentication and session key generation can utilize challenges in the set c_a that are unique for each device.

5 COBRA AND PARCE PROTOCOLS

The message exchange diagram for the COBRA and PARCE protocols is presented in this section. Although both protocols have been described in previous work [1] [2], they are analyzed and compared in this paper using the same test beds and operating conditions, and in real-time. Moreover, the PARCE authentication scheme is extended beyond the earlier work in several ways. First, a database search operation is added to enable authentication to be carried out privately. Second, a bilateral, server authentication component is added to the original device-only protocol. Third, this work includes an experimental evaluation of PARCE, which was not included in [2]. And last, a new post-processing operation is added to both protocols that removes leakage related to the authenticating device's identity in the transmitted **HD** bitstrings.

As noted earlier, COBRA authenticates a remote party by correlating **HD** bitstrings, while PARCE authenticates by counting bit flip errors in the regenerated nonce. The server has access to the T_{DB}, which stores a relatively small set of approx. 10,000 DV per device. T_{DB} engenders the server with the ability to generate millions of uncorrelated bitstrings for authentication. The authentication model presented in the following occurs between the server, which implements a software version of the SiRF PUF algorithm, and devices, which include the PUF itself and a hardware implementation of the algorithm.

5.1 Message Exchange Diagrams

The message exchange diagram for the PARCE protocol is shown in Fig. 5. The diagram for COBRA is identical except for the nonce generation and transmission operations (which are omitted), and the SiRF PUF bitstring generation mode. As indicated, COBRA does not require the nonce n_x, and it utilizes the default bitstring generation mode based on thresholding, as opposed to PARCE which uses XMR and SKE mode. Last, the *BitFlip* function is replaced with an *XNOR* correlation function in COBRA. The PARCE protocol consists of 12 steps, described as follows with annotations in the figure.

1) The device requests authentication from a server.
2) The server generates a random 32-bit vector-selection seed t_x, random nonces n_x and q'_x. The seed t_x will be used by the server and device to select a set of vectors from the common pool of vectors stored in the Vecs_{DB}. The nonce n_x represents the authentication nonce. The nonce q'_x is used to specify the LFSR seed parameters to the SiRF algorithm once it is XORed with a corresponding device-generated nonce. The t_x, q'_x and n_x are transmitted to the device. The SiRF PUF parameter set $\mathbf{p}_x$ is constructed by combining q_x returned by the device with the range and threshold parameters discussed earlier. The $\mathbf{SF}_x$ are derived from DV_i stored in the T_{DB} using parameters $\mathbf{p_x}$, and then transmitted to the device.
3) The device XORs q'_x with its own TRNG-generated version to produce q_x, which is transmitted to the server. The q_x construction and exchange process prevents adversaries from engaging in chosen-message attacks on the device. The device extracts $\mathbf{vecs}_x$ from its Vec_{DB} and applies $\mathbf{vecs}_x$, $\mathbf{p}_x$, $\mathbf{SF}_x$ and n_x to its hardware PUF, HPUF_E, in enrollment mode. The PUF produces only a helper data bitstring, $\mathbf{HD}_x$, when configured in SKE mode.
4) The device transmits $\mathbf{HD}_x$ to the server.
5) The server performs authentication by searching its T_{DB} for a device i that can reproduce n_x. For each device i in the T_{DB}, the DV_i corresponding to the vectors $\mathbf{vecs}_x$, along with $\mathbf{p}_x$, $\mathbf{SF}_x$ and the device-generated $\mathbf{HD}_x$, are used as input to a software version of the PUF, SPUF_R, configured to run in regeneration mode. A correlation coefficient (CC) is computed for each device in the database using n'_x and n_x as input to a function called *BitFlips*. *BitFlips* counts the number of mismatching bits that exist between n_x and n'_x, and in the TMR_x sequences. The CC array is sorted once all CCs are computed, and then the two smallest CCs are used to compute a percentage-change coefficient, PCC (discussed further below). If the PCC exceeds a server-defined threshold, the server accepts the authentication request, otherwise it rejects and halts the process.
6) In cases where the server accepts the device authentication request, the device performs a similar process to authenticate the server. Here, the device generates a nonce n_y, and transmits it to the server.
7) The server generates nonces t_y and q'_y, and transmits q'_y to the device.
8) The device runs its TRNG to create q_y and transmits it to the server. The device also constructs $\mathbf{p_y}$.
9) The server constructs $\mathbf{p_y}$ and reads out vectors $\mathbf{vecs}_y$ from the Vec_{DB}. Since the device's identity is known, the vectors in this case are drawn from the unique set defined for each device. The T_{DB} is consulted to derive a set of spread factors $\mathbf{SF}_y$, which are used as input to the SPUF_E, along with $\mathbf{vecs}_y$, $\mathbf{p}_y$ and n_y to produce helper data $\mathbf{HD}_y$. The server transmits t_y, $\mathbf{SF}_y$ and $\mathbf{HD}_y$ to the device.
10) The device extracts vectors $\mathbf{vecs}_y$ and runs HPUF_R to generate n'_y and computes a CC using the *BitFlips* function.
11) Given the CC represents the number of bit-flip errors, the device accepts the authentication if the CC is below a device-defined threshold, otherwise it rejects.
12) The server is notified of the authentication decision and proceeds to engage with the device in some type of secure transaction for cases in which device accepts the server authentication request.

6 ZED AND ZYBO EXPERIMENTS

The ZED and ZYBO Avnet FPGA boards are used in two different experiments to provide data for evaluating the

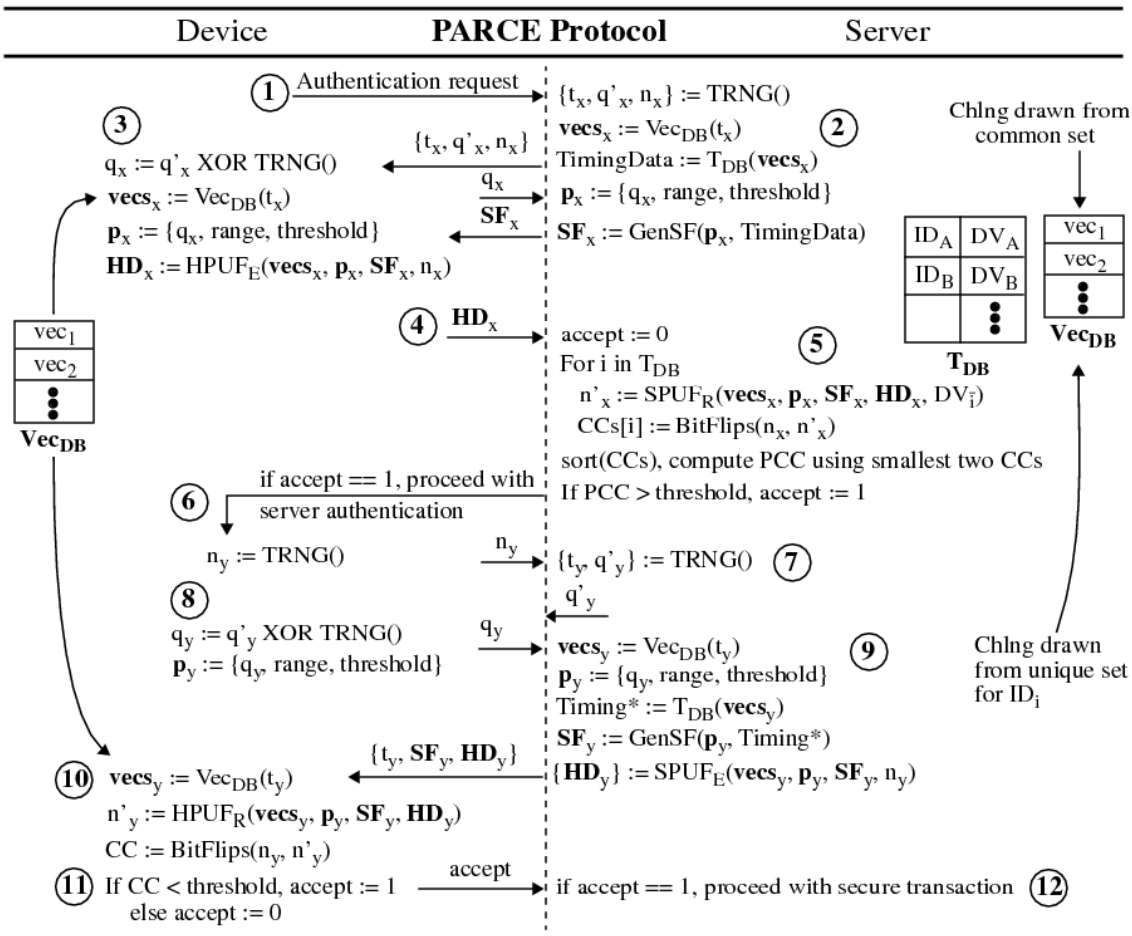


Fig. 5: PARCE privacy-preserving mutual authentication protocol.

COBRA and PARCE security properties. Provisioning (timing) data is collected from 20 copies of the ZED board, on which 6 identical instantiations of the SiRF PUF are created. The 120 instances are subjected to 16 extended industrial standard temperature-voltage (TV corner) conditions, which are referred to as TV_0 through TV_{15} in Table 1. The timing data associated with 4096 paths is collected from each board under each of the 16 TV corners, and a software program is used to emulate the COBRA and PARCE protocols.

A set of 35 ZYBO boards are used in the second experiment, on which 4 identical instantiations of the SiRF PUF are created, for a total of 140 instances. The SiRF PUF provisioning process is used to collect timing data, which is stored in the T_{DB} and used in a real-time evaluation of the protocols. A programmable logic implementation of the SiRF PUF is used to produce the HD bitstrings utilized in the protocols, e.g., for the HPUF$_{E/R}$ functions in Fig. 5, while other database-related and network communication-based components of the message exchange protocol delegated to the devices are implemented in a C program running under Linux. The server runs a multi-threaded C program which executes all message exchange diagram operations (including the SiRF PUF algorithm) in software. The experiment is configured to enable 12 ZYBO devices to carry out authentications concurrently over a wired ethernet network. The server searches the T_{DB} data of all 140 devices in each authentication operation.

TABLE 1: Temperature-voltage (TV) corners definitions.

Acronym	TV Corner	Used As
TV_0	25^o C, 1.00 V	Server T_{DB} data
$TV_{1,2,3}$	25^o C, 0.95, 1.00, 1.05 V	In-field Device data
$TV_{4,5,6}$	0^o C, 0.95, 1.00, 1.05 V	In-field Device data
$TV_{7,8,9}$	-40^o C, 0.95, 1.00, 1.05 V	In-field Device data
$TV_{10,11,12}$	85^o C, 0.95, 1.00, 1.05 V	In-field Device data
$TV_{13,14,15}$	100^o C, 0.95, 1.00, 1.05 V	In-field Device data

6.1 Experimental Evaluation of COBRA

For COBRA, the server correlates the device-generated $\mathbf{HD}_x$ with the $\mathbf{HD}_y$ bitstrings computed using the timing data stored in the T_{DB} for all devices y. Correlation refers to determining the degree of similarity of two HD bitstrings. A correlation coefficient, CC$_{\text{Cobra}}$, is computed for each pairing of **HD** bitstrings using Eq. 1. The sum represents the number of matching bits over the 2-bit columns of the **HD** bitstrings, NB represents the number of bits, and $\mathbf{HD}_x$ and $\mathbf{HD}_y$ represent the pair of **HD** bitstrings. Given this definition, the authentic device is expected to posses a significantly larger value of CC$_{\text{Cobra}}$ than other enrolled devices in the database.

$$\mathbf{CC_{Cobra}} = \sum_{i=1}^{NB} \overline{HD_x[i] \oplus HD_y[i]} \tag{1}$$

Authentication critically depends on the uniqueness property of the **HD** bitstrings to accurately identify the authenticating device and to detect forged authentication

```
1011110001001101100100011100001111010011011010000110000000011110  FPGA0, TV0, Chlng0
1011110001001101110000010100001111010011110010001110000000111110  FPGA0, TV1, Chlng0  (a)
11111111111111111010111101111111111111111010111110111111111011111  CCXNOR = 57 (1833)

0000000100001101010000101100000011011001100011000010001001110011  FPGA0, TV0, Chlng1
0110001110001101010001101101001001011000101010001010001001110010  FPGA0, TV15, Chlng1 (b)
10011101011111111111101111101101111110110110111111111111111110  CCXNOR = 51 (1687)

1011110001001101100100011100001111010011011010000110000000011110  FPGA0, TV0, Chlng0
1010000001000011011000000001111000110010110001010011011100101010  FPGA1, TV0, Chlng0  (c)
11100011111100010000111000000000001001001000111011111000110110101  CCXNOR = 30 (1189)

1011110001001101100100011100001111010011011010000110000000011110  FPGA0, TV0, Chlng0
0000000100001101010000101100000011011001100011000010001001110011  FPGA0, TV0, Chlng1  (d)
01000010101111110010110011111110011110101000110111011110110010010  CCXNOR = 37 (1108)
```

Fig. 6: Authentication scenarios illustrated using COBRA **HD** bitstrings obtained from FPGA hardware experiments.

attempts by adversaries, as well as the randomness property to prevent MB and impersonation attacks. The scenarios important to assess regarding uniqueness are shown in Fig. 6 (a) through (d). Each of the scenarios includes two 64-bit segments from a pair of **HD** bitstrings under evaluation. Scenarios (a) and (b) represent cases where the server should accept the authentication attempt while (c) and (d) represent cases that should be rejected. For (a), the server generated an **HD** bitstring for $FPGA_0$ using $Chlng_0$ that matches with the device generated **HD** bitstring on 57 of 64 bits (and on 1833 bits of the full-length 2048-bit bitstrings not shown).

Scenario (b) is similar except the device generated its **HD** bitstring under adverse environmental conditions, namely $\{100^o$ C, 1.05 V$\}$. Although the number of matching bits is smaller at 51 (1687), it is still larger than the number of matching bits under scenarios (c) and (d). Scenario (c) shows the results when the device **HD** bitstring for $FPGA_0$ is correlated by the server with the **HD** bitstring from another device in the T_{DB}. Scenario (d) portrays an artificial scenario where the device uses a different challenge, $Chlng_1$, as a means of illustrating that **HD** bitstrings from the same device are distinguishable.

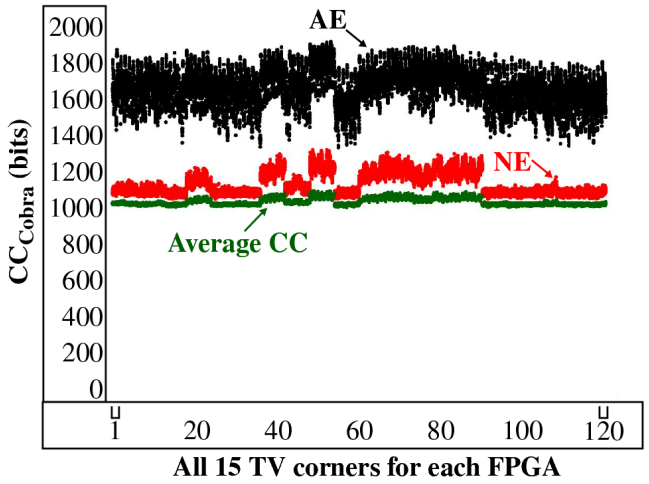


Fig. 7: COBRA CC_{Cobra} results for 120 SiRF PUF instances on the ZED boards tested at 15 TV corners (x-axis) with 10 challenges.

The curves shown in Fig. 7 show that the CC_{Cobra} by itself can be used to distinguish the genuine (authentic) device from other (non-authentic) devices enrolled in the database. The analysis uses the 2048-bit **HD** bitstrings generated from a single iteration of the SiRF PUF algorithm. Here, the CCs computed by the server are plotted for the authentic-enrolled (AE) device in black and the largest non-authentic-enrolled (NE) device in red. The average CC_{Cobra} computed using all device CCs from each authentication are shown in green. The nomenclature 'enrolled' refers to PUF devices whose timing data is stored in the T_{DB}, in contrast to 'not enrolled', a scenario discussed further below.

The graph shows the CCs values for 10 separate challenges superimposed, with the results for all 15 TV corners adjacent along the x-axis for each of the 120 FPGAs. There are no instances in which the CC_{Cobra} for a NE device is larger than the corresponding AE CC_{Cobra}, and therefore, all accept decisions made by the server are done correctly using only the top ranked CCs in each authentication.

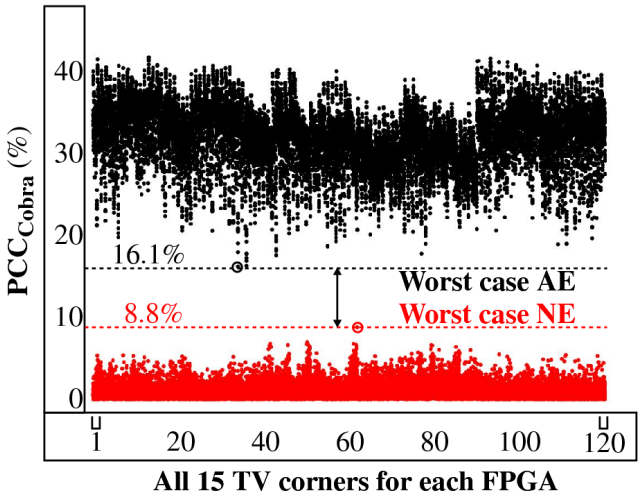


Fig. 8: PCC_{Cobra} results for 120 SiRF PUF instances on the ZED boards, tested at 15 TV corners with 10 challenges.

Although the CC_{Cobra} metric by itself is effective in distinguishing the authentic device from other devices, it does not allow a fixed threshold to be used by the server for accept-reject decisions. An alternative metric, called

the percentage change coefficient, $\text{PCC}_{\text{Cobra}}$, addresses this deficiency and is defined in Eq. 2. $\text{PCC}_{\text{Cobra}}$ measures the fractional difference between consecutive CCs in the sorted list of CCs, and in particular, the CCs at positions x and $x+1$. Here, the expectation is that the AE device, represented by CC_1, is significantly larger and stands apart from the closest NE device, represented by CC_2.

$$\mathbf{PCC_{Cobra}} = \frac{CC_x - CC_{x+1}}{CC_x} * 100\% \tag{2}$$

The $\text{PCC}_{\text{Cobra}}$ results are plotted in Fig. 8 in the same format as Fig. 7. The AE PCC are computed using the two largest CCs for each authentication, while the NE PCC are computed using CCs ranked 2nd and 3rd in the sorted list. The 'Worst case AE' and 'Worst case NE' PCCs are highlighted illustrating the smallest margin is 7.3%. The server can set the threshold close to a lower bound that remains secure, i.e, well above the worst case NE, e.g., at 15%, as a means of creating a fixed pass-fail criteria for authentications, and retry on the off-chance of a false negative (reject) authentication decision.

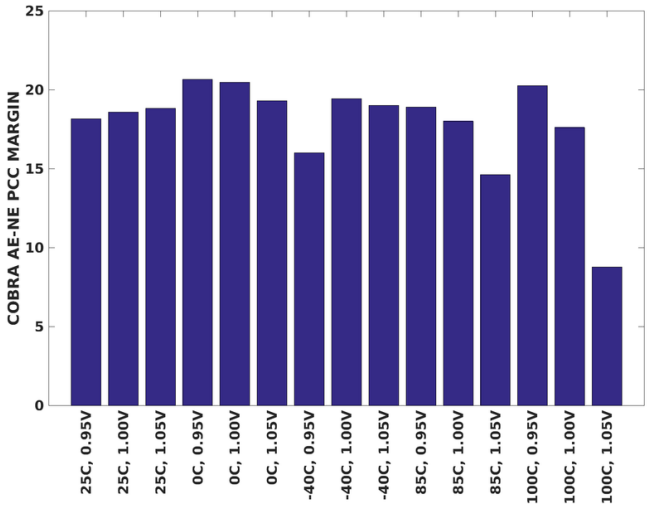


Fig. 9: COBRA worst case margins (differences) between the AE and NE PCCs for each TV corner.

The bar graph shown in Fig. 9 depicts the worst case margins for an authentication i, i.e., (smallest AE_i) − (largest NE_i), per TV corner. It is clear from the height of the right-most bars that the worst case margins occur at TV corner 100^oC, 1.05 V. Although the difference is small, the margin for the worst case AE-NE authentication *pair* in these results is slightly larger at 8.78% than the worst case margin shown in Fig. 8 between the circled points for two different authentications.

As indicated, the ZED experimental results described above use 2048-bit **HD** bitstrings created from one iteration of SiRF PUF algorithm. In the ZYBO experiments, we extend the analysis to 4096- and 6144-bit **HD** bitstrings to determine if the worst case margin improves (increases) when using additional entropy (in the form of additional **HD** bits) from the authenticating devices.

The graphs shown in Fig. 10(a)-(c) display the AE (black), NE (red) and average NE (green) points for 100,000 COBRA authentications carried out using 12 ZYBO boards, with each device authenticating approx. 8,300 times. The results using 2048-bit **HD** bitstrings in Fig. 10(a) show a large number of false-reject decisions, i.e., AE data points that fall below the 15% threshold defined in the ZED experiments. Reducing the threshold to, e.g., 10%, would reduce the number of false-rejects but it also reduces the worst case AE-NE margin to approx. 5%.

The results shown in Fig. 10(b) for 4096-bit **HD** bitstrings improve on the 2048-bit results, by reducing the number of false-rejects to 14, as well as the worst case NE magnitudes. However, the best results are shown in Fig. 10(c), which uses 6144-bit **HD** bitstrings. All of the AE points exceed the 15% threshold, and the points for the worst case NE and average NE curves are reduced even further, to an upper bound of 5%. In general, the trends in the graphs show that the AE-NE margins increase as the number of **HD** bits increase, which enables the server to tune the level of certainty it requires for a successful authentication.

We also tested the COBRA protocol using randomly generated **HD** bitstrings, as one possible strategy to evaluate authentication attempts by devices that are not included in the database (referred to as the 'not enrolled' scenario earlier). The CCs and PCCs produced were similar to the average case NE values shown in the result graphs of this section, confirming that such authentication attempts are rejected by COBRA.

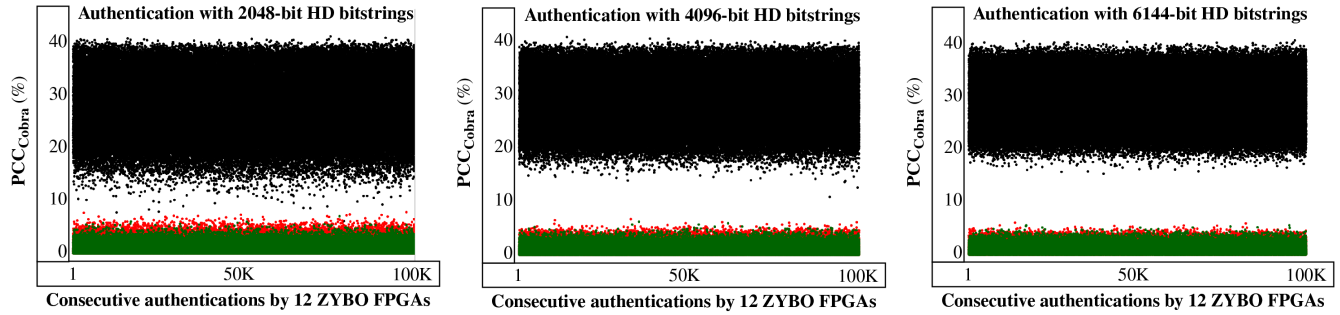
6.1.1 Statistical analysis of COBRA HD Bitstrings

The COBRA protocol as described provides anonymity because no IDs are transmitted between the server and device. However, the **HD** bitstrings themselves may leak information about the identity of the device that an adversary can leverage. In this section, we show that leakage does in fact occur and propose a simple change to the SiRF algorithm that removes the source of the leakage.

In our analysis, we observed that the mean value of within-die variations i.e. the source of entropy, varies among the FPGAs, where some FPGAs posses a larger overall level of within-die variation relative to a fixed noise level. The average level of entropy impacts the number of bits classified as strong and weak during bitstring generation, and therefore, variations that occur across FPGAs will be apparent in the number of 1's in the **HD** bitstrings, which can leak information about the identity of the authenticating device.

The disparities in the number of 1's in the **HD** bitstrings is shown in Fig. 11 using data collected from the ZED board experiments under TV_0 (nominal) conditions. The number of 1's in the **HD** bitstrings is reported as a percentage called the 'HD-1 Percentage' along the y-axis for each of the FPGAs (x-axis). The mean, minimum and maximum HD-1 percentages are shown by the black and blue curves, respectively, from an analysis of the **HD** bitstrings produced by applying 10 challenges to each FPGA. Although groups of devices possess similar HD-1 percentages, which benefits anonymity, the large range from 10% to 65% may allow an adversary to derive a smaller list of authenticating device candidates.

The most straightforward way of eliminating this source of leakage is to determine a 'personalized' threshold constant (PTC) for each device. The threshold parameter from



(a) COBRA PCC results using 2048-bit **HD** bitstrings.

(b) COBRA PCC results using 4096-bit **HD** bitstrings.

(c) COBRA PCC results using 6144-bit **HD** bitstrings.

Fig. 10: ZYBO experiments: COBRA AE and NE PCC results.

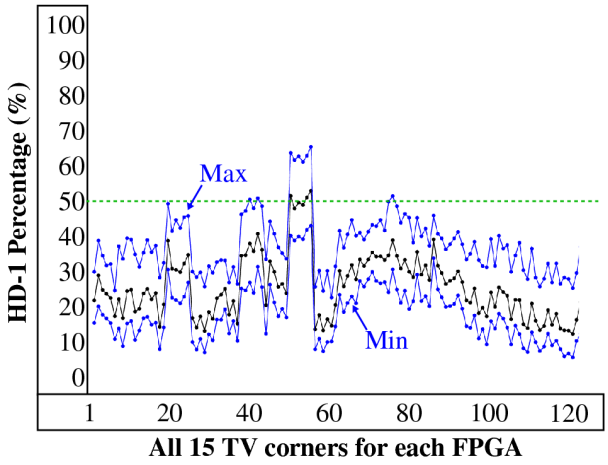


Fig. 11: Original HD-1 percentages in the COBRA 2048-bit **HD** bitstrings for 120 ZED boards and 10 challenges.

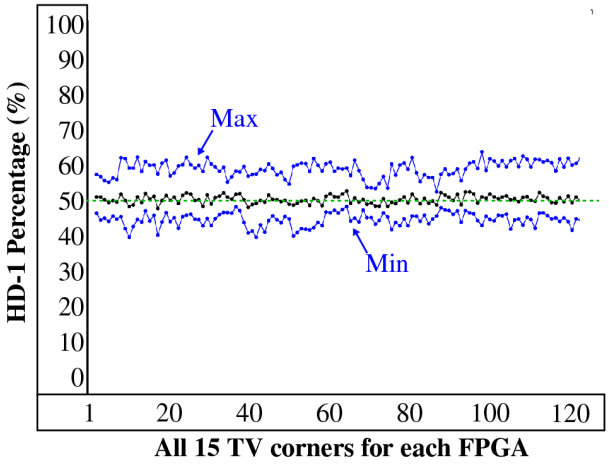


Fig. 12: New HD-1 percentages after applying PTCs in the 2048-bit **HD** bitstrings for 120 ZED boards and 10 challenges.

Section 2 defines the sizes of the strong and weak bit regions. A personalized threshold constant for each device can be chosen such that the number of strong and weak bits in the **HD** bitstrings are nearly equal, e.g. 50%. The drawback of reducing the threshold parameter is that the response bitstring can become less resilient to bit flip errors for devices which require a larger value, which is the case for nearly all of the devices shown in Fig. 11. However, COBRA does not utilize response bitstrings, and therefore, the proposed operation has no negative consequences.

The PTC value can be readily determined during provisioning by applying a set of challenges and computing the fraction of 1's in the **HD** bitstrings. We developed a search algorithm that iteratively scales the PTC value for each device until the average HD-1 percentage converges to a value close to 50% for each device. The personalized threshold constant can be stored in a non-secure NVM by the device since it leaks no information about the individual bits within the **HD** bitstrings.

The average HD-1 percentages obtained after computing and applying the PTC to the ten **HD** bitstrings used in Fig. 11 are shown in Fig. 12. The HD-1 percentages hover around the target 50% value. Although it is possible to fine tune the PTC values further to obtain percentages even closer to 50%, this is not necessary because the region delineated by the minimum and maximum percentages nearly fully encapsulates the average values, and therefore, variations in the HD-1 percentages introduced by the challenges masks the identity of the device carrying out the authentication. The COBRA authentication results presented earlier, as well as the results for PARCE presented in the next section, are obtained with PTCs applied.

6.2 Experimental Evaluation of PARCE

As discussed earlier, the encoding component of SKE is a PUF-based encryption technique, where plaintext information is encrypted by a PUF-entropy-based encryption key to produce an **HD** bitstring, which represents the ciphertext. The plaintext can be a nonce sent in the clear by an authentication server to the device, or it can be, e.g., a password or an account number, that the device wishes to send encrypted to the server (or vise versa) across an untrusted network. The message exchange diagrams for these two scenarios are shown in Fig. 13. Note that the latter encryption-based scenario can also serve as a key sharing mechanism where the device and server encrypt and decrypt, respectively, a shared secret. In this paper, we evaluate PARCE under Scenario$_A$ and leave the evaluation of Scenario$_B$ for a future work.

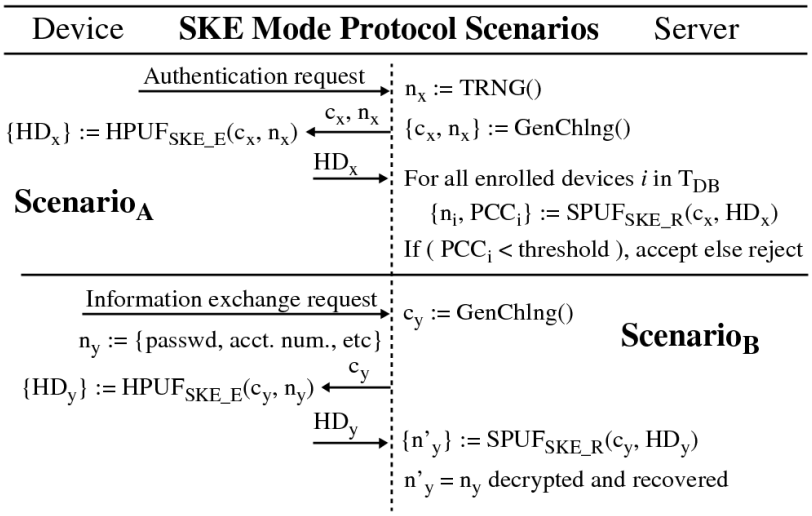


Fig. 13: PARCE application scenarios: $Scenario_A$, authentication and $Scenario_B$, light-weight encryption.

6.2.1 PARCE Accept-Reject Decision Criteria

In step 5 of Fig. 5, the accept-reject authentication decision made by the server is based on the results from the *Bit-Flips* function. *BitFlips* compares the n'_x regenerated by the server for each enrolled device in T_{DB} with the original server-generated n_x. Given the authenticating device **HD** is optimized by the device to reproduce n_x, when the server uses its DV from the database, the number of mismatches between n'_x and n_x should be zero in the ideal case. In contrast, the DV for other devices will be inconsistent with the device regenerated DV, resulting in mismatches. The most straightforward method of identifying the authenticating device is to count the number of mismatches, and select the ID_x from the database with the smallest number.

However, the XMR technique provides additional information in the **HD** bitstring that can be leveraged to increase the robustness and certainty of the authentication decision. The TMR_x sequences produced during regeneration also record mismatches, but on a bit-by-bit basis, in contrast to bits in n'_x which are obtained from majority voting across the TMR_x sequences. The example set of TMR_x sequences labeled *Regen: TMR_x* along the bottom of Fig. 3 show TMR sequences where two bit flips have occurred. Given the bit flips occur within the TMR_x sequences, we refer to them as 'minority bit flips' (MBF), and the total number of occurrences across all TMR_x sequences as NMBF. For the same reasons given for n'_x, the NMBF is expected to be very small for the AE device and large for the NE devices.

The *BitFlips* routine does not use the MBF directly because they do not take into account bit flip errors that occur in the n'_x bitstring. When an n'_x bit mismatch occurs, the majority voting operation applied to the TMR sequence produces the wrong n'_x bit value, and therefore the number of mismatches should be represented by the number of majority bits in the TMR sequence, not the number of minority bits. The *BitFlips* routine therefore uses mismatches in both the n'_x bitstring and TMR sequences to compute the 'true' number of bit flip errors that have occurred. The term NTBF is used in reference to the true number of bit flips, and represents the correlation coefficient (CC) for PARCE as given by Eq. 3.

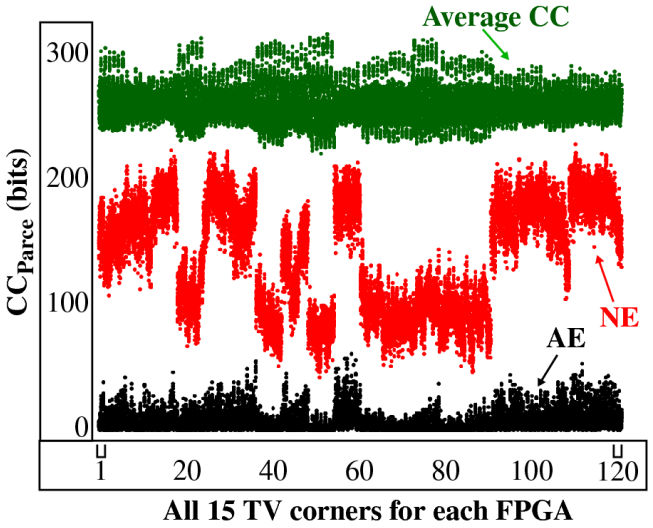


Fig. 14: CC_{Parce} results for 120 SiRF PUF instances tested at 15 TV corners with 10 challenges in the ZED experiments. The points from different challenges are superimposed.

$$\mathbf{CC_{Parce}} = NTBF_x \tag{3}$$

Like COBRA, the accept-reject decision in PARCE is based on a percentage change coefficient (PCC). The PCC expression is given by Eq. (4) and is designed to measure the relative difference between the two top ranked NTBF values in the sorted array. For PARCE, the top ranked values are the smallest NTBF values at positions 1 and 2. A device authentication request is accepted if the PCC_{Parce} is larger than a server-defined threshold.

$$\mathbf{PCC_{Parce}} = (NTBF_{x+1} - NTBF_x)/NTBF_{x+1} * 100\% \tag{4}$$

The experimental results from the two FPGA experiments again focus on the the AE and NE PCCs. The AE PCC is computed using the NTBF values ranked 1st and 2nd in the sorted array, while the NE PCC is computed using the NTBF values ranked 2nd and 3rd.

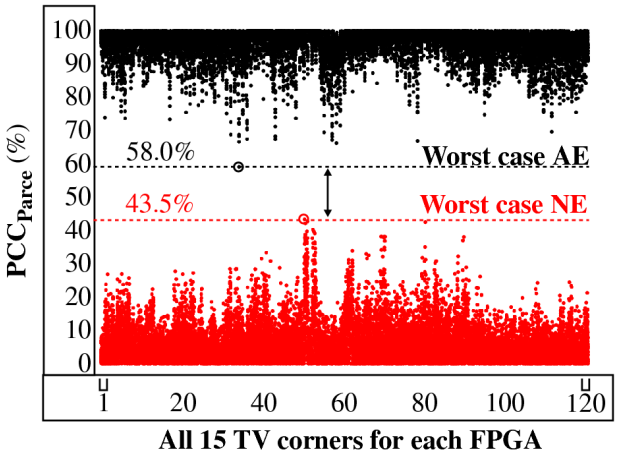


Fig. 15: PCC_{Parce} results for 120 SiRF PUF instances tested at 15 TV corners with 10 challenges in the ZED experiments.

The CC_{Parce} results are plotted in Fig. 14 for the ZED experiments. The SiRF PUF XMR algorithm configured in SKE mode with XMR set to 5 (5MR) is run for one iteration, which is able to encode 103 bits of a randomly generated n_x on average. Similar to the COBRA results presented earlier, the CC_{Parce} can be used to distinguish the AE device from the other devices in all 18,000 authentications (120 device, 15 TV corners and 10 challenges), but the threshold needs to be customized for each device.

On the other hand, the PCC_{Parce} results shown in Fig. 15 allow a fixed threshold to be defined by the server. The worst case AE and NE PCCs are highlighted at 58.0% and 43.5% and occur for two different authentications. The server can use a conservative threshold of 55.0% for the accept-reject decision.

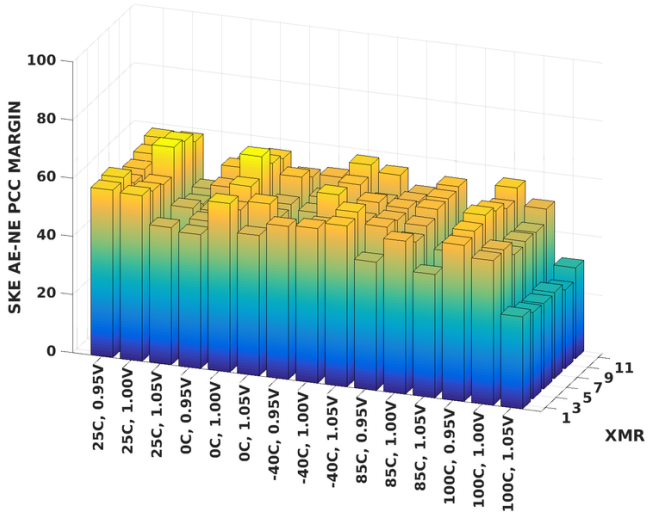


Fig. 16: PARCE worst case margins (differences) between AE and NE PCCs for each TV corner (x-axis) and XMR (y-axis) in the ZED experiments.

The bar graph in Fig. 16 present the worst case margins, $(\text{smallest AE}_i) - (\text{largest NE}_i)$, for an authentication i across the 10 challenges from the ZED experiments as a function of temperature-voltage (x-axis) and XMR (y-axis). The smallest margins are associated with a worst case TV corner (100^oC, $1.05V$), where the overall smallest margin is 27.10% for 9MR. It should be noted that even though the devices used in the study are commercial grade, no authentication failures occurred.

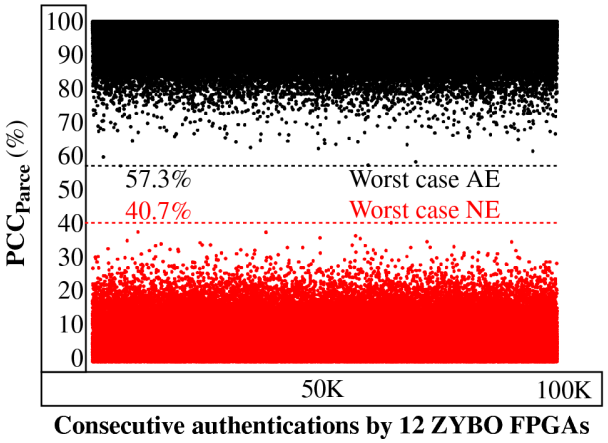


Fig. 17: PCC_{Parce} results from the ZYBO experiments showing AE PCCs (black) and NE PCCs (red) using a 256-bit n_x with XMR set to 5.

For the ZYBO experiments, the XMR value is fixed at 5, and the server is configured to generate a random 256-bit n_x. The device iterates three times on average to encode the n_x, with each iteration processing 2048 DVD_{co}. The total time per authentication is less than 2 seconds. The experiment is set up to enable simultaneous authentications to be carried out by 12 ZYBO boards, each of which authenticates approx. 8300 times with the multi-threaded server application, for a total of 100,000 authentications.

The PCC_{Parce} results for the ZYBO experiments are shown in Fig. 17. All of the AE PCCs are larger than the NE PCCs, which indicates that all authentications succeeded in correctly identifying the AE device. The worst case PCCs for AE and NE are 57.3% and 40.7%, respectively, which enables the same 55.0% accept-reject threshold determined in the ZED experiments to be used here.

6.3 Server Authentication

The server authentication process performed by the device needs to utilize the CC_{Cobra} and CC_{Parce} defined by Eqs. 1 and 3 to make accept-reject decisions. This is true because the device does not have access to the T_{DB} database used by the server to compute PCCs. As indicated earlier, the CCs are equally effective as the PCCs for making accept-reject decisions, but the threshold for each device will vary. Customized thresholds can be computed during the provisioning process (along with the PTC discussed earlier), and the devices can store them in a non-volatile memory. The CC result graphs generated from the ZED and ZYBO experiments are not shown because the results are nearly identical to the CC graphs given in Figs. 7 and 14. No failures were observed in any of the server authentications.

7 Implementation Characteristics

Customer devices run the PUF-Cash protocol on mobile, hand-held devices and therefore, a low-power implementation is important in a practical application. A block diagram

depicting the mapping of the protocol's components is shown in Fig. 18. The prototype system is implemented on a Xilinx 7 series SoC, which incorporates an ARM Cortex A9 microprocessor (PS side) with a programmable logic region (PL side) on the same device. All of the core security functions are implemented in the low-power PL side, and only the network and database components are implemented in a C program that runs on the microprocessor. In this section, we summarize the resource utilization, power consumption, and performance of the SiRF PUF and protocols on this system architecture.

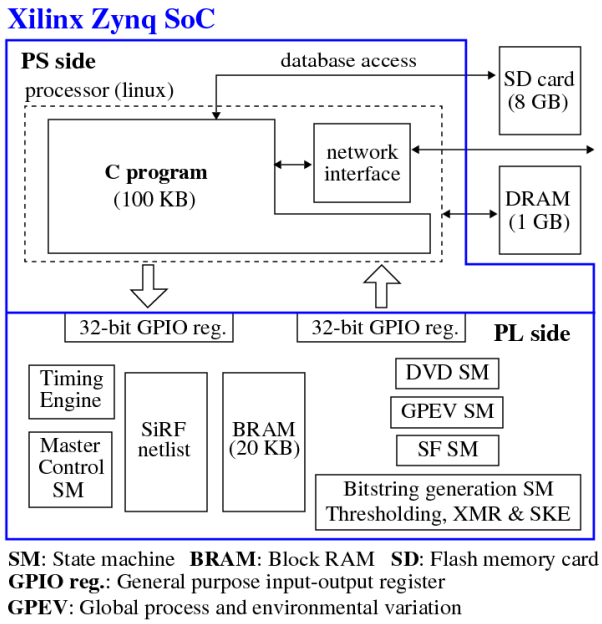


Fig. 18: System architecture of the SiRF PUF and protocols on the Xilinx Zynq 7010 SoC.

7.1 Resource Utilization and Power

Table 2 gives the SiRF PUF resource utilization on a Xilinx Zynq 7010 FPGA. The *SiRF Engine* and *Netlist* components are listed separately, with the latter referring to the top-most component of Fig. 2, and the former referring to the remaining components. The resource utilization given for the *Netlist* relates to the SiRF PUF entropy source configured with three rows of modules (see Fig. 2). Additional rows add linearly to the resource utilization, e.g., LUTs increase from 796 to 1062, but exponentially to the number of paths, as described in the next section. The *SiRF Engine* component includes implementations of device and server authentication, session key generation and a TRNG.

TABLE 2: SiRF PUF resource utilization on the Zynq 7010.

Resource	SiRF Engine	Netlist	Available	% Used
LUT	5842	796	17,600	37.72
LUTRAM	60	96	6000	2.60
FF	4377	32	35,200	12.53
BRAM	5	-	60	8.33
DSP	2	-	80	2.50
BUFG	2	-	32	6.25

Xilinx Vivado's power analysis tool estimates power consumption at 1.43 W for the processor side and 27 mW for the programmable logic side, which indicates that most of the power consumption occurs in the microprocessor.

7.2 Performance Analysis

The average run time of the SiRF algorithm for either enrollment or regeneration of a 256-bit response or 2048-bit **HD** bitstring is 1.4 s, which includes the network transfer time to retrieve the challenge components from the server. Larger bitstrings can be generated by running additional iterations of the algorithm, where each additional iteration adds only 100 ms to the overall run time.

8 Security Analysis

8.1 Challenge-Response-Pair (CRP) Analysis

The engineered netlist with an array of 3×8 modules as shown in Fig. 2 possesses 2^{24} distinct rise and fall paths. A characterization process (see [3] for details) is carried out on a sample of devices that groups the set of 2^{24} paths into compatibility sets, where compatibility is defined as a subset of paths whose delay changes linearly, within a margin, across different temperature and supply voltage conditions. Therefore, the set of 2^{24} paths are partitioned into multiple compatibility sets, where the cardinality of each set is approx. 20,000 paths. This yields approx. 838 distinct, non-overlapping compatibility sets (all paths are compatible with some subset of paths from the initial set). Assuming each compatibility set consists of 10,000 DV_R and 10,000 DV_F, the number of distinct path delay differences (DVD) is equal to 100,000,000. Therefore, the total number of unique path combinations (and corresponding bits) across all 838 compatibility sets is approx. 84 billion (2^{36}).

8.2 Learning Parity with Noise (LPN) Hardness

The underlying guarantees of the protocol stem from the hardness of the PUF primitive. Here we start from the position that the PUF is a physical random oracle that is constructed to achieve LPN hardness [29]. To demonstrate LPN hardness, we must show that both challenges C and responses R are chosen uniformly from $\{0,1\}^n$, and the mapping between challenges and responses can be expressed as $\forall i, r_i = f(c_i) + e_i$, where $f()$ represents the PUF transformation operating on challenge bit c_i summed with the noise term e_i. Here, each input challenge C is generated via an LFSR using a 32-bit random PRNG seed defined by vectors v. If each seed is used only once, we can assert that challenges are chosen uniformly over the set $\{0,1\}^n$.

In prior work [3], we experimentally verified the statistical quality of individual responses by subjecting a broad range of responses to the NIST statistical test suite analysis [30]. Statistical analysis on compensated path delays is also conducted to demonstrate that $f(C)$ is an unbiased distribution for large numbers of C, therefore $(C, f(C) + e)$ is random for large numbers of CRPs as well, thus achieving LPN hardness. Note that LPN hardness does not guarantee protection against MB attacks. However, LPN hardness does guarantee that the number of challenges required to build a sufficiently predictive model will be quite large, a property that we exploit in the following section.

8.3 Model-Building Resistance

The authors of [4] and [9] conclusively demonstrate that a strong PUF can be modeled with a subset of CRPs via evolutionary models or logistic regression (multi-layer perceptron). In particular, Arbiter XOR PUFs were shown to be broken, which is relevant to the proposed SiRF architecture as it has an arbiter-style structure. While a thorough model building analysis is still pending, we can make a preliminary claim that SiRF is likely to resist such attacks due to the following properties:

8.3.1 Ambiguous Delay and Entropy

Unlike a typical Arbiter XOR PUF, the SiRF has a reconvergent fan-out property. This means that at specific junctions, the output fans out before reconverging at downstream nodes. This introduces ambiguity in which path is actually timed on a given device for a specific challenge, i.e., the fanout branch which dominates the timing will vary across devices and challenges. This type of physical path structure does not exist in most other PUFs, e.g., the XOR Arbiter PUF. Additionally, the determination of which of the 2048 DVD_{co} are weak (and discarded) and which are strong is probabilistic, because of uncompensated temperature-voltage variation in the measured DV. This adds uncertainty regarding the classification of some bits as weak or strong during enrollment. Moreover, the bit assignment of the remaining weak bits are not observable at all on a given device, but may in fact be classified as strong on other devices.

8.3.2 Brute Force Attack on Entropy Source

In an ideal strong PUF, challenge response pairs are unique and responses are guaranteed to be uncorrelated with each other. A brute force attack could be conducted by generating all possible challenges and reading out all possible responses. The time required to generate a response is intrinsic to the internal operations of the PUF architecture, e.g., digitization, compensation and other post-processing operations. Section 8.1 describes a valid selection space of 2^{36} challenge pairs and corresponding response bits. In the current architecture, the response-bit generation rate is measured at 0.5 milliseconds. Therefore, a brute-force application of all possible challenges to extract the entire set of response bits of the PUF would take more than a year.

If additional security is required, the source of entropy can be increased by adding another row of modules to the block diagram shown in Figure 2, exponentially increasing the number of distinct rise and fall paths. Each additional row adds a factor of 64 to the number of paths for each output. With four rows and 32 outputs, this expands the selection space to 2^{42} and brute-force read-out time to 68 years. The response measurement time makes a brute-force attack economically infeasible. Additionally, the attack would need to be repeated for every PUF in the system.

8.4 Protocol security

We prove the following properties for protocol security:

8.4.1 Adversary

We assume that a powerful adversary that supplies a counterfeit device with the intent that the server authenticates the counterfeit as a legitimate device. Additionally we assume that the adversary is able to observe all interactions on the communications channel between the device and the server (Fig. 5). Then, in Step 2, the adversary would have access to the challenge vector seed t, spread factors **SF**, parameters **p** and nonce n. Similarly, in Step 4, the adversary has access to the helper data bitstring **HD** associated with each PUF response. The adversary may replace one or more of these variables with modified versions to generate the desired PUF response and helper data, **HD**, which would be accepted by the server.

8.4.2 Brute Force Attack on the Protocol

We observe that t, the q component of **p** and n are generated randomly via a server and device TRNG. t is a 32-bit value, n is a 256-bit random nonce, q is a 22-bit seed for the LFSR. The **SF** is 16,384 bits and is generated by a dedicated function that takes **p** and DV as input. The output, **HD**, is 2048 bits for the parameters chosen in this work. Under the assumption that **HD** bits are indistinguishable from random bits, an adversary attempting to directly guess the output would be required to guess all randomly and independent chosen bits of **HD** correctly with a probability of 2^{-2048}, which is considered infeasible.

8.4.3 Packet Injection

The adversary may inject modified versions of t, q', n and **SF** to prompt a specific response from the device. As the attacker only has helper data **HD** to correlate responses against, the probability of guessing the right output based on an input is 2^{-2048}, which is highly unlikely. Additionally, injecting randomly generated values would cause authentication to fail as the server generates **HD** responses using it's own versions of t, n and **SF** and DV recorded during the provisioning step.

8.4.4 Response Prediction

Finally, the adversary may attempt a model building attack by gathering traces from multiple successful executions of the protocol. Let us suppose 10,000 successful executions of the protocol were traced over a period of 10,000 - 20,000 seconds. Upon termination, the adversary would possess 10,000 tuples of $< \{t, \mathbf{p}, n, \mathbf{SF}\}, \mathbf{HD} >$, with $\{t, \mathbf{p}, n, \mathbf{SF}\}$ as input and **HD** as output. For the reasons outlined earlier, this PUF is likely to be resistant to such an attack.

8.5 Discussion

In this section, we compare and contrast features and results of the COBRA and PARCE protocols as a means of developing a usage scenario that provides the best overall security profile. Both protocols are demonstrated to convey a high level certainty to the server and device regarding authentication accept-reject decisions and both have similar performance characteristics, in particular, individual authentications can be completed in less than 2 seconds. And both protocols provide device anonymity by virtue of the database search carried out by the server, and through

the use of personalized threshold constants as a means of reducing leakage in the HD bitstrings.

On the other hand, COBRA reveals less information to an adversary than PARCE. Although the n_x used by PARCE is not a response bitstring in the traditional sense, it is constructed using PUF response bits and is therefore vunerable to a MB attack. In contrast, a MB attack on COBRA would need to be developed using only the HD bitstrings. It is unclear how one would craft a ML model that is capable of learning constituent delay elements of the tested paths without having access to the response bitstrings, given that response bitstrings represent the centerpiece of most previously proposed MB attacks. We note that assessing the MB resistance of the protocols is required for completeness in the evaluation of the protocols' security profile, but the complexity of the analysis precludes its inclusion in this paper, and we instead defer this analysis to a future work.

Given these considerations and results, the most attractive strategy is to use COBRA for device authentication and either COBRA or PARCE for server authentication, given that MB attacks are not possible in the server authentication component of the protocol. This is true because device authentication 'gates' server authentication, i.e., it is not performed unless device authentication succeeds. Moreover, the enhanced set of resources available on the server allow it to detect repeated attempts to authenticate. A second attractive scenario is to use both COBRA and PARCE, in that order, for device and/or server authentication. This type of methodology requires the adversary to develop impersonation methods for both protocols, significantly increasing the difficulty of a successful attack.

9 Conclusion

The COBRA and PARCE PUF-based, privacy-preserving, mutual authentication protocols are evaluated and compared using common test beds in this paper. The test beds utilize ZED and ZYBO boards, both incorporating Xilinx Zynq SoC-based FPGAs. The ZED board experiments assess the protocols for resiliency across industrial-standard environmental conditions while the ZYBO board experiments assess real-time performance characteristics of the protocols. All authentications carried out in both protocols, totaling 18,000 and 100,000 for the ZED and ZYBO boards, respectively, succeeded in identifying the authenticating device and provided a large margin in the decision metric.

The PARCE protocol exchanges a challenge, a nonce and a helper data bitstring with the authenticating agent (device or server) while COBRA exchanges only a challenge and a helper data bitstring. Both protocols are privacy-preserving, and utilize a novel personalized threshold constant to minimize leakage of the device identity in the helper data bitstrings. Both protocols avoid utilizing cryptographic primitives for protecting the device interface against attacks, and both protocols allow the server and device to tune to the number of elements that are correlated during authentications. The latter feature enables the authenticating agent to define the level of certainty it wishes to achieve regarding an accept-reject decision.

Neither PARCE or COBRA use the PUF's response bitstrings in the message exchange protocol, which deters standard approaches to MB attacks. COBRA exposes less information in the message exchange, and is therefore a more secure protocol for device authentication where adversaries can engage in a wide range of attacks. A series application of both protocols further increases resilience against impersonation attacks. Future work will investigate MB attacks, and other applications, including the use of the SKE algorithm for implementing a PUF-based, light-weight encryption mechanism.

References

[1] J. Plusquellic and M. Areno, "Correlation-based robust authentication (cobra) using helper data only," *Cryptography*, vol. 2, no. 3, 2018. [Online]. Available: https://www.mdpi.com/2410-387X/2/3/21

[2] D. Heeger and J. Plusquellic, "Analysis of iot authentication over lora," in *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2020, pp. 458–465.

[3] J. Plusquellic, "Shift register, reconvergent-fanout (sirf) puf implementation on an fpga," *Cryptography*, vol. 6, no. 4, 2022. [Online]. Available: https://www.mdpi.com/2410-387X/6/4/59

[4] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "Puf modeling attacks on simulated and silicon data," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1876–1891, 2013.

[5] G. T. Becker, "On the pitfalls of using arbiter-pufs as building blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1295–1307, 2015.

[6] Y. Wen and Y. Lao, "Enhancing puf reliability by machine learning," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[7] H. Awano, T. Iizuka, and M. Ikeda, "Pufnet: A deep neural network based modeling attack for physically unclonable function," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–4.

[8] Y. Xu, Y. Lao, W. Liu, Z. Zhang, X. You, and C. Zhang, "Mathematical modeling analysis of strong physical unclonable functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4426–4438, 2020.

[9] J. Delvaux, "Machine-learning attacks on polypufs, ob-pufs, rpufs, lhs-pufs, and puf-fsms," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2043–2058, 2019.

[10] Y. Lao and K. K. Parhi, "Statistical analysis of mux-based physical unclonable functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 5, pp. 649–662, 2014.

[11] Q. Wang, M. Gao, and G. Qu, "A machine learning attack resistant dual-mode puf," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 177–182. [Online]. Available: https://doi.org/10.1145/3194554.3194590

[12] S. V. S. Avvaru, Z. Zeng, and K. K. Parhi, "Homogeneous and heterogeneous feed-forward xor physical unclonable functions," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2485–2498, 2020.

[13] Y. Gao, D. C. Ranasinghe, G. Li, S. F. Al-Sarawi, O. Kavehei, and D. Abbott, "A challenge obfuscation method for thwarting model building attacks on pufs," *IACR Cryptology ePrint Archive*, vol. 2015, p. 471, 2015. [Online]. Available: https://eprint.iacr.org/2015/471

[14] S. S. Zalivaka, A. A. Ivaniuk, and C.-H. Chang, "Reliable and modeling attack resistant authentication of arbiter puf in fpga implementation with trinary quadruple response," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1109–1123, 2019.

[15] Y. Lao and K. K. Parhi, "Reconfigurable architectures for silicon physical unclonable functions," in *2011 IEEE International Conference on Electro/Information Technology*, 2011, pp. 1–7.

[16] J. Ye, Y. Hu, and X. Li, "Opuf: Obfuscation logic based physical unclonable function," in *2015 IEEE 21st International On-Line Testing Symposium (IOLTS)*, 2015, pp. 156–161.

[17] A. Wang, W. Tan, Y. Wen, and Y. Lao, "Nopuf: A novel puf design framework toward modeling attack resistant pufs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 6, pp. 2508–2521, 2021.
[18] Y. Wang, C. Wang, C. Gu, Y. Cui, M. O'Neill, and W. Liu, "A generic dynamic responding mechanism and secure authentication protocol for strong pufs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 9, pp. 1256–1268, 2022.
[19] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, "A survey on lightweight entity authentication with strong pufs," *ACM Comput. Surv.*, vol. 48, no. 2, oct 2015. [Online]. Available: https://doi.org/10.1145/2818186
[20] T. Idriss and M. Bayoumi, "Lightweight highly secure puf protocol for mutual authentication and secret message exchange," in *2017 IEEE Int. Conf. on RFID Technology Application*, 2017, pp. 214–219.
[21] M. H. Mahalat, S. Saha, A. Mondal, and B. Sen, "A puf based light weight protocol for secure wifi authentication of iot devices," in *2018 8th Int. Symp. on Embed. Comp. and System Design*, 2018, pp. 183–187.
[22] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty, D. Mahata, and M. M. Prabhu, "Building puf based authentication and key exchange protocol for iot without explicit crps in verifier database," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp. 424–437, 2019.
[23] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A puf-based secure communication protocol for iot," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 3, pp. 1–25, 2017.
[24] J. R. Wallrabenstein, "Practical and secure iot device authentication using physical unclonable functions," in *2016 IEEE 4th Int. Conference on Future Internet of Things and Cloud*, 2016, pp. 99–106.
[25] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on pufs for lightweight authentication," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 146–159, 2016.
[26] Y. Wang, C. Wang, C. Gu, Y. Cui, M. O'Neill, and W. Liu, "A dynamically configurable puf and dynamic matching authentication protocol," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1091–1104, 2022.
[27] D. Owen Jr, D. Heeger, C. Chan, W. Che, F. Saqib, M. Areno, and J. Plusquellic, "An autonomous, self-authenticating, and self-contained secure boot process for field-programmable gate arrays," *Cryptography*, vol. 2, no. 3, p. 15, 2018.
[28] W. Che, M. Martin, G. Pocklassery, V. K. Kajuluri, F. Saqib, and J. Plusquellic, "A privacy-preserving, mutual puf-based authentication protocol," *Cryptography*, vol. 1, no. 1, 2017.
[29] C. Jin, C. Herder, L. Ren, P. H. Nguyen, B. Fuller, S. Devadas, and M. Van Dijk, "Fpga implementation of a cryptographically-secure puf based on learning parity with noise," *Cryptography*, vol. 1, no. 3, 2017. [Online]. Available: https://www.mdpi.com/2410-387X/1/3/23
[30] "A statistical test suite for random and pseudorandom number generators for cryptographic applications," *https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf, accessed on Sept. 10, 2022.*, 2010.

Jim Plusquellic is a Professor in Electrical and Computer Engineering at the University of New Mexico. He received both his M.S. and Ph.D. degrees in Computer Science from the University of Pittsburgh. Professor Plusquellic received an "Outstanding Contribution Award" from IEEE Computer Society in 2012 and 2017 for co-founding and for his contributions to the Symposium on Hardware-Oriented Security and Trust (HOST).

Eirini Eleni Tsiropoulou is currently an Assistant Professor at the Department of Electrical and Computer Engineering, University of New Mexico. Her main research interests lie in the area of cyber-physical social systems and wireless heterogeneous networks, with emphasis on network modeling and optimization, resource orchestration in interdependent systems, reinforcement learning, game theory, network economics, and Internet of Things. Five of her papers received the Best Paper Award at IEEE WCNC in 2012, ADHOCNETS in 2015, IEEE/IFIP WMNC 2019, INFOCOM 2019 by the IEEE ComSoc Technical Committee on Communications Systems Integration and Modeling, and IEEE/ACM BRAINS 2020. She was selected by the IEEE Communication Society - N2Women - as one of the top ten Rising Stars of 2017 in the communications and networking field. She received the NSF CRII Award in 2019 and the Early Career Award by the IEEE Communications Society Internet Technical Committee in 2019.

Cyrus Minwalla is a Security Lead with the Financial Technology Research group at the Bank of Canada. He was selected as NRC's Top Scientist Under 40 in 2017 and received the Bank of Canada's Award of Excellence in 2020 and 2022. Cyrus's research interests include digital currencies, cryptography, embedded devices, and Internet-of-Things. He received his B.A.Sc. and Ph.D. degrees in Computer Engineering from York University in Canada.