cryptography

MDPI

Article

Correlation-Based Robust Authentication (Cobra) using Helper Data Only

Jim Plusquellic[1*] and Matt Areno[2]

[1] University of New Mexico and IC-Safety; jimp@ece.unm.edu
[2] Trusted and Secure Systems; matt@trusecsys.com
* Correspondence: jimp@ece.unm.edu; Tel.: +240-475-1882

Received: date; Accepted: date; Published: date

Abstract: PUF-based authentication protocols have been proposed as a strong challenge-response form of authentication for IoT and embedded applications. A special class of so called strong PUFs are best suited for authentication because they are able to generate an exponential number of challenge-response-pairs (CRPs). However, strong PUFs must also be resilient to model-building attacks. Model-building utilizes machine learning algorithms and a small set of CRPs to build a model that is able to predict the responses of a fielded chip, thereby compromising the security of chip-server interactions. In this paper, response bitstrings are eliminated in the message exchanges between chips and the server during authentication and therefore, it is no longer possible to carry out model-building attacks in the traditional manner. Instead, the chip transmits a Helper Data bitstring to the server and this information is used for authentication instead. The server constructs Helper Data bitstrings using enrollment data that it stores for all valid chips in a secure database and computes correlation coefficients (CCs) between the chip's Helper Data bitstring and each of the server-generated Helper Data bitstrings. The server authenticates (and identifies) the chip if a CC is found that exceeds a threshold, which is determined during characterization. The technique is demonstrated using data from a set of 500 Xilinx Zynq 7020 FPGAs, subjected to industrial-level temperature and voltage variations.

Keywords: PUF-based authentication; Helper data correlation; hardware security

1. Introduction

Robust authentication and key generation are critically important to defining a root of trust and to providing data integrity and confidentiality in communications between internet-of-things (IoT) devices. Physical unclonable functions (PUFs) are proposed as replacements to traditional non-volatile memory (NVM) for storing keys and to using cryptographic primitives in authentication protocols [1-7]. PUFs are able to reproduce keys and bitstrings on-the-fly by measuring small changes in the signal behavior of an integrated circuit that occur because of the finite, non-zero tolerances that exist in manufacturing processes. A special class of so-called strong PUFs are able to generate an uncountable numbers of reproducible bits making it possible to construct unique response bitstrings for authentication protocols without the need to employ entropy-enhancing cryptographic primitives such as secure hash, thereby reducing energy and area overheads in IoT devices.

PUF-based authentication protocols that allow direct access to the embedded PUF through an unprotected interface [7], where challenges and responses are not obfuscated using cryptographic primitives, represent the most attractive usage scenario because such protocols are typically very simple and compact. The drawback of protocols with unprotected interfaces is that the embedded PUF is susceptible to model-building attacks. Model-building is typically carried out using machine learning (ML) algorithms where the goal of the adversary is build a model of the challenge-response behavior of the PUF and then later use the model to predict the PUF's response to any arbitrary challenge. If this can be accomplished, then the model can be used to impersonate the actual device. Although ML attacks require a large amount of computing effort, they represents a serious threat to PUF-based authentication protocols with unprotected interfaces.

In this paper, we propose a PUF-based, privacy-preserving, mutual authentication protocol with unprotected interfaces that is resilient to model-building attacks. The technique is demonstrated using the hardware-embedded delay PUF (HELP) [7-8] but is applicable to any PUF architecture that is able to produce **soft data**. Soft data refers to digital values that represent the magnitude of the signal being measured, e.g., path delays and frequencies. Soft data captures the inherent, random variations that occur in these signals from one chip to another, and represents the source of entropy for the PUF. The proposed methodology can therefore be applied to an enhanced version of the arbiter PUF [11], and to traditional weak PUFs, such as the RO PUF [12], with additional but simple enhancements to expand the challenge-response space from n to 2^n as required for authentication applications.

The PUF architecture defines a set of functions that convert soft data into bitstrings and keys to be used for encryption and authentication. The authentication protocol that we propose uses soft data, and the corresponding Helper Data bitstrings that are produced by the PUF architecture, as input to a correlation technique. We refer to the protocol as **Cobra** for **Correlation-based robust authentication**. The results presented in this paper show that by correlating Helper Data bitstrings, a server can correctly and securely authenticate a fielded chip. The significance of this claim is that there is no need to reveal the response bitstrings in the message exchanges between the chip and server and therefore, the traditional approach of applying machine-learning algorithms to the challenge-response-pairs (CRPs) is no longer possible.

In the Cobra protocol, the chip constructs and transmits a Helper Data bitstring to the server. The Helper Data bitstring transmitted is traditionally used by the server to identify weak bits in the response bitstring, but otherwise reveals no information about the response bitstring itself [7]. The server constructs a set of Helper Data bitstrings using enrollment data that it stores for all valid chips in a secure database. The chip's Helper Data bitstring is correlated to each of the server generated Helper Data bitstrings by bitwise AND'ing the two bitstrings and then counting the number of '1's in the AND'ed bitstring. (Alternatives to bitwise AND correlation include bitwise XNOR (discussed later) and traditional time and frequency domain digital signal processing forms of correlation.) The server authenticates the chip if **exactly one** of the Helper Data bitstrings constructed using enrollment data correlates, i.e., has a large number of '1's, to the chip's Helper Data bitstring. A similar process is carried out in reverse from server to chip to enable mutual (two-way) authentication but without the exhaustive search component. Therefore, no information regarding the PUF secrets is revealed to the adversary despite the fact that the Helper Data bitstrings are derived from random variations that occur within the PUF's circuit components.

The remainder of this paper is organized as follows. Section 2 provides background on PUF architectures and defines key concepts such as soft data, error avoidance methods, the HELP PUF architecture, Helper Data generation, methods of correlation and an analysis illustrating proof-of-concept. Section 3 describes the Cobra privacy-preserving, mutual authentication protocol. Experimental results are presented in Section 4 on data collected from a set of 500 Xilinx Zynq FPGAs, showing both the effectiveness and the limitations of Cobra on data collected across the range of industrial-level temperature and voltage specifications. A security analysis is presented in Section 5 and conclusions in Section 6.

2. Background

2.1. PUF Architectures and Soft Data

A wide range of PUF architectures have been proposed since the initial papers on PUFs were published [12][13]. The source of entropy (randomness) for the PUF is chip-to-chip and within-die process variations that occur between and within chips during production. The PUF architecture defines the mechanism that is used to measure small signal variations introduced by process variations effects. In some cases, the measurement process leverages the existing architectural features of the chip, e.g., the SRAM PUF measures its entropy source, i.e., the state of the individual SRAM cells, by simply applying power to the SRAM array [14]. For most PUFs, however, circuit components need to be added to the chip, e.g., the RO PUF requires a set of MUXs and counters to select and measure the frequencies associated with elements in the array of ROs [12]. In another example, the HELP PUF adds a launch-capture clocking mechanism to precisely time the delays of combinational logic paths [7-8].

For PUF architectures that measure and digitize the signal behavior associated with the entropy source, the digitized values provide additional information that can be leveraged in strong challenge-response-pair (CRP) forms of authentication. The digitized values represent the magnitude of the signal behavior, e.g., RO frequency or path delay, and are often used as input to mathematical processes defined by the PUF architecture. The goal of the mathematical operations is to isolate and amplify the random differences that occur among multiple copies of the individual circuit components. The digitized values are eventually converted to a bit and used in the response of the CRP. We refer to the digitized values as **soft data**.

Of the PUF architectures that create soft data (the RO and HELP PUFs are just two examples), the conversion to bits can be accomplished in a variety of ways. For example, the RO PUF typically selects a pair of ROs and then computes a difference by subtracting the soft data associated with the two ROs. The sign of the difference can then be used to generate a bit, with, e.g., negative differences producing a ‘0’ and positive differences producing a ‘1’. The HELP PUF also computes differences among pairings of path delays and uses a modulus operation to assign a ‘0’ or ‘1’ to the differences.

2.2. Error Correction and Avoidance Methods

Nearly all PUF architecture need to deal with bit-flip errors, which are differences in the response bitstring that occur when the response is regenerated. Bit-flip errors are most probable when the magnitude of the difference between a pair of soft data values is close to zero. In these cases, regeneration of the bitstring, which takes place later in the field and under potentially adverse environmental conditions, can result in bits flipping from ‘0’ to ‘1’ or vise versa. Although authentication protocols can be designed to be tolerant to a small number of bit-flip errors, the number of bit-flip errors that can occur is too large in most PUF architectures to guarantee that authentication works correctly when regeneration is carried out in harsh environments.

To deal with this issue, nearly all PUF architectures define an **error correction** or **error avoidance** method to improve reliability during regeneration. Error correction is the more popular of the two reliability-enhancing methods. Error correction typically processes the PUF response bits into a final response bitstring using algorithms based on linear block codes [15] or Bose-Chaudhuri-Hochquenghen (BCH) codes [16]. However, nearly all of the error correction schemes ignore the magnitude of the difference in the soft data and use all of the PUF response bits to construct a smaller but reproducible bitstring response, including bits that have a high probability of changing value.

Error avoidance schemes, on the other hand, integrate a thresholding method that skips bits that are deemed unreliable. The reliability of a bit is often directly related to the soft data associated with the bit, and in particular, the distance of the soft data value to the bit-flip line. The bit-flip line is defined by the PUF architecture as a soft threshold between ‘0’ or a ‘1’. Unlike error correction methods, error avoidance methods can only be used with PUF architectures that produce soft data values, e.g., the RO [12], metal resistance [17], NVM [18] and HELP [8-10] PUFs are some examples. Notable exceptions here are memory-based PUFs, including SRAM, DRAM, FF and latch-based PUFs, which as originally proposed, are not capable to producing soft data.

In typical usage scenarios, an **enrollment** phase is carried out in which challenges are applied to the PUF in a secure facility and response bitstrings are generated for the first time. In addition to the response bitstring, PUF architectures that use either error correction or error avoidance methods also produce Helper Data. Helper Data is typically maintained in the secure facility and transmitted to the fielded device later during **regeneration** to enable the PUF to precisely reproduce the response bitstring. The form of the generated Helper Data varies dramatically depending on the error correction or avoidance method employed. A key contribution of the method proposed here relates to Helper Data, and in particular to Helper Data that is generated by error avoidance methods. The next two subsections discuss a simple error avoidance scheme used by the HELP PUF as well as

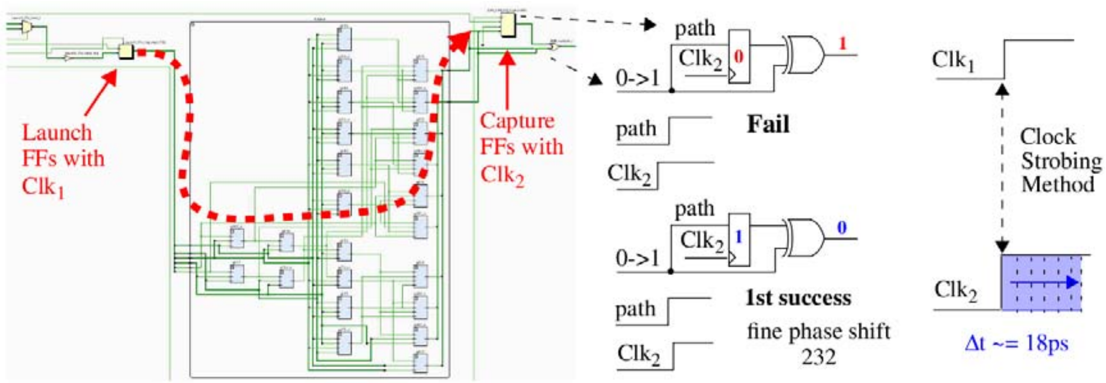

Figure 1 HELP Architecture illustrating Clock Strobing concept used to create path delay (Timing) soft data.

features of the generated Helper Data that are leveraged in the proposed Cobra protocol.

2.3. HELP PUF

To better exemplify the principles of the proposed technology, we use the HELP PUF architecture and data collected from a set of Zynq FPGAs in the illustrative examples that follow [9-10]. HELP uses a *launch-capture* technique to obtain accurate digital timing values of path delays through a combinational logic block. The combinational logic block for a full adder is shown in Fig. 1 but any functional unit can be used (the combinational logic from one column of the Advanced Encryption Standard is used in [7-10] and for the experimental results presented in Section 4). Logic signal transitions are launched from the *Launch FFs* shown on the left using Clk_1 and captured in the *Capture FFs* shown on the right using Clk_2.

The digital clock manager (DCM) on the FPGA is used to create the clocks, with *dynamic fine phase shift* enabled for Clk_2. Dynamic fine phase shift allows a state machine running in the programmable logic (PL) of the FPGA to increment the phase shift by increments of 18 ps (see right side of Fig. 1). A path through the full adder is timed by repeatedly applying a 2-vector sequence to the *Launch FFs* until the signal propagating along the highlighted path is successfully captured in the *Capture FF*. A successful capture occurs in this example when the '0' produced from the first vector V_1 of the 2-vector sequence is overwritten by the '1' produced by V_2 (see center portion of Fig. 1). When this occurs, the current fine phase shift value, which is an integer typically between 100 and 500, is recorded by HELP as the digitized timing value for this path. These timing values represents the soft data associated with HELP.

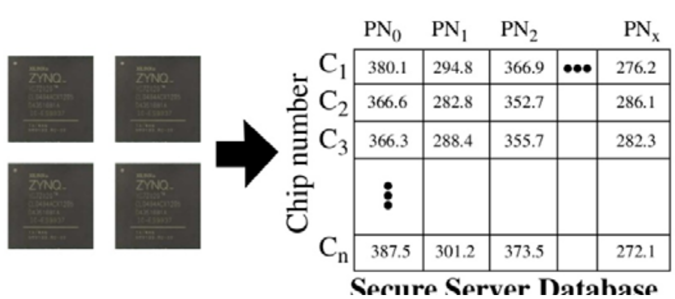

Figure 2 HELP PN database created during enrollment.

During enrollment, a set of timing values, called **PUF Numbers** or **PN**, for each chip are stored in the rows of a secure database as shown in Fig. 2. This data is consulted by the secure server to authenticate these chips after they are deployed in fielded systems. The storage of soft data on the server, in contrast to response bitstrings, is the first of several significant differences that exist between Cobra and the PUF-based protocols proposed by others [1-6].

As indicated above, the proposed Cobra protocol leverages Helper Data to carry out authentication. The Helper Data is derived from the PNs stored in the secure database on the server, and from an instance of HELP that is programmed into the programmable logic of an FPGA, which represents the fielded chip. The entropy that is associated with each chip is captured by the PN stored in the server database. An adversary carrying out machine learning attacks against the protocol would attempt to learn the timing information stored in the database by reverse-engineering the bitstring responses that are exchanged openly over the network. Once learned, the adversary can then impersonate the chip. Therefore, it is vital that the relationship between the PN and the response bitstring be obscured and remain hidden to make this task difficult or impossible for the adversary.

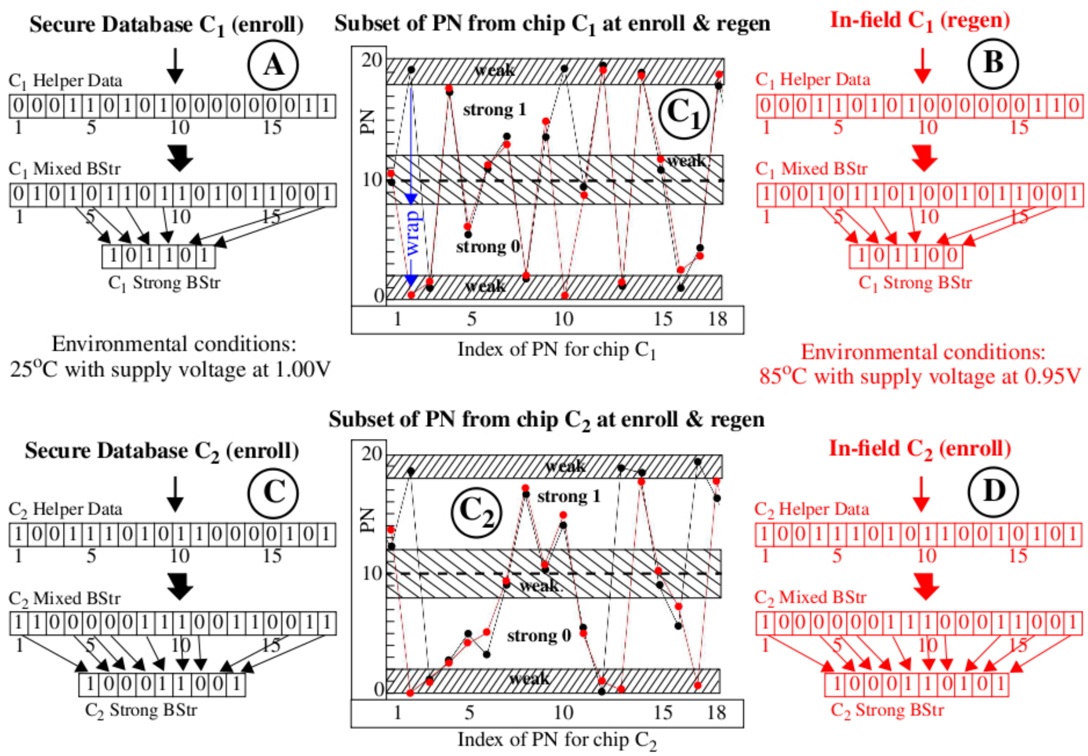

Figure 3 (Center) PN for chips C_1 and C_2 under enrollment (black points), and regeneration (red points), conditions, (top left A) Helper Data and response bitstrings for C_1 under enrollment conditions, (top right B) Helper Data and response bitstrings for C_1 under regeneration conditions, (bottom left C and right D) same for chip C_2.

2.4. Helper Data Generation using an Error Avoidance Technique

The illustration presented in Fig. 3 shows how bitstrings and Helper Data are generated using an error avoidance scheme called Margining, as a precursor to our discussion on the proposed Helper Data correlation method. The graphs labeled with "C_1" and "C_2" (in large circles) in the center of the figure plot a set of 18 PN (timing values) along the x-axis for two chips C_1 (top) and C_2 (bottom), with the black curves depicting PN obtained from the secure server database and the red curves depicting PN generated on-the-fly by these chips during regeneration in the field. The environmental conditions for data collected during enrollment and stored in the secure database are specified as 25°C under nominal supply voltage conditions (1.00V) while to fielded chips are subjected to high temperature (85°C) and low supply voltage conditions (-5% or 0.95V). The data shown are actual measurements obtained from two Zynq FPGAs used in our experiments.

The HELP PUF converts the PN into a bitstring by applying the following algorithm. First, a pseudo-random number generator selects pairs of PN and creates differences. A temperature-voltage compensation method called **TVComp** is then applied to the differences to compensate the measured timing values for changes introduced by environmental conditions (we omit the details of these operations [19] here to focus the discussion on the proposed correlation technique). Finally, a modulus operation is applied to the compensated differences to remove path length bias effects. The modulus operation is defined in the standard way as returning the positive remainder after dividing by the modulus. The value of the modulus is one of several parameters to the HELP algorithm. The graphs shown in Fig. 3 use a modulus of 20, which is reflected as the maximum value given on the y-axis.

The TVComp process implemented within HELP is very effective at compensating the chip regenerated PN (red values) but is not ideal. The red data points are vertically offset above and below the black (enrollment) data points because of **uncompensated temperature-voltage noise** (TVNoise). An interesting example labeled 'wrap' is shown for the 2nd data point in the upper "C_1" graph where TVNoise has caused the point to 'wrap' from the enrollment value near 20 back around to a value near 0 during regeneration. Despite these anomalies, the black and red curves in each graph track very closely, i.e., they are correlated. In contrast, the black curves from both graphs (for C_1 and C_2) are not correlated. This key observation serves as the basis for the innovation proposed within the Cobra protocol.

As mentioned earlier, the error avoidance scheme implemented within HELP is called Margining. The Margining scheme skips soft data values (PN in our example) for cases in which the probability of a bit-flip error is large. These highly probable bit-flip regions are labeled "weak" in the center graphs of Fig. 3 and are located adjacent to the bit-flip lines at 0, 10 and 20. In other words, PN that are within a distance of 2.0 of these bit-flip lines have the highest probability of changing value. For example, the PN labeled "wrap" represents a bit-flip error which is introduced by TVNoise. PN that are located in the "weak" regions are assigned '0' in the Helper Data bitstring. For example, the "C_1 Helper Data" bitstrings in the region labeled with the circled "A" in the figure begins with "000", which reflects that the status of the first 3 PN in the black curve of graph "C_1". In contrast, the 4th bit is '1' because the PN at position 4 in graph "C_1" falls within the "strong 1" region.

The "C_1 Mixed BStr" in region "A" of Fig. 3 records the bit value associated with each of the 18 PN, with PN < 10.0 assigned '0' and those >= 10.0 assigned '1'. This response bitstring corresponds one-to-one to the "C_1 Helper Data" bitstring and contains both strong and weak bits. The "C_1 Strong BStr" is constructed from the "C_1 Mixed BStr" by selecting only those bits identified as strong in the "C_1 Helper Data" bitstring. The region labeled "B" shows the corresponding bitstrings generated using the red (regeneration) data points from graph "C_1". The graphs and annotations labeled "C_2", "C" and "D" are completely analogous to "C_1", "A" and "B" except the PN and bitstrings are derived from second chip C_2.

The HELP authentication protocol from [7] proposes a DualHelperData scheme where both the Helper Data and Strong BStr are exchanged between the server modeled on the left side of Fig. 3 and the fielded chips modeled on the right side. As discussed earlier, exposing the Strong BStr to the adversary enables model-building attacks where the adversary attempts to reverse engineer the PN stored in the secure database using machine learning (ML) algorithms. Although attempts to model-build HELP have not been successful (see [20]), the exposure of the response bitstrings (Strong BStr) still represents a vulnerability that enables ML attacks. If it becomes possible to construct an ML attack that is able to deduce the relationships among the PN, then the response bitstrings to other challenges can be predicted, and the chip impersonated.

2.5. Poof-of-Concept: Helper Data Correlation

As a mitigation strategy, we propose an alternative authentication protocol that exchanges only the Helper Data bitstrings. Authentication is carried out in Cobra by correlating the Helper Data bitstrings generated using the enrollment data on the server with the Helper Data bitstring generated on-the-fly by the chip. The simplest correlation strategy is to compute a new bitstring by bitwise AND'ing the Helper Data bitstrings from the server and chip and then counting the number of '1's in the AND'ed version. We refer to the number of '1's as the **correlation coefficient (CC)** because it

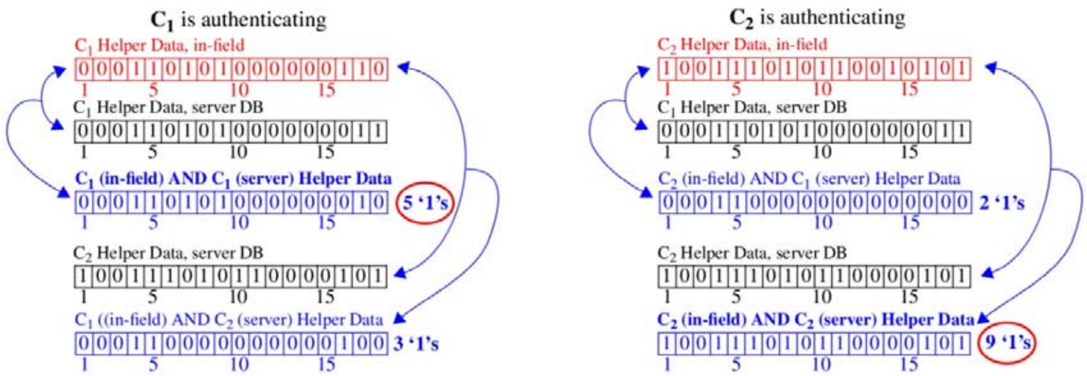

Figure 4 Illustration showing Helper Data bitstring correlation using "AND" operator with C_1 authenticating to the server on the left and C_2 on the right. Helper Data bitstrings are obtained from data in Fig. 3.

reflects the level of similarity that exists (among the '1's) between the two Helper Data bitstrings.

As an example, the left column of Fig. 4 shows an authentication attempt by chip C_1 while the right column shows an attempt by chip C_2. The top-most red-colored bitstrings in each column are the Helper Data bitstrings transmitted by the chips to the server. For each Helper Data bitstring received during authentication, the server carries out an exhaustive search operation using the PN stored its secure database. It constructs the black-colored Helper Data bitstrings in each column using the technique described in Fig. 3 (in fact, the bitstrings shown here are identical to those shown in Fig. 3).

For each Helper Data bitstring it constructs, the server bitwise AND's it with the received chip's Helper Data bitstring. For example, the AND'ed versions are labeled "C_1 (in-field) AND C_1 (server) Helper Data" and "C_1 (in-field) AND C_2 (server) Helper Data" in the left column of Fig. 4. As discussed, the bitwise AND operation is a form of correlation that acts to preserve more '1's in cases where the bitstrings are similar. The CCs (number of '1's) are reported to the right of AND'ed Helper Data bitstrings. The results for both authentication attempts show higher correlation for cases where the server-generated Helper Data bitstring is derived using PN collected earlier during enrollment from the same chip. In other words, the authenticating chip is correctly identified to the server using only the information provided by the CC.

This example uses only a small number of 18 PN from the larger set of 2048 that are produced by each iteration of the HELP algorithm [7]. The results shown in Fig. 5 expand the example illustration to the full length bitstrings and to PN collected across 9 TV corners using 500 Xilinx Zynq FPGAs. Each of the 9 curves plots the CCs for the 500 chips along the x-axis. The authenticating chip is labeled C_{250} and is highlighted in the center region of the figure.

The graphic illustration given in Fig. 6 shows how the curves in Fig. 5 are constructed. Here, the chip's Helper Data bitstring on the left, regenerated under 25oC, 1.00V, is transmitted to the server on the right. A second authentication request is also shown in red where the chip in this case is regenerating Helper Data with environmental conditions set to -40^{o}C, 1.05V. The server carries out an exhaustive search using data stored in the Enroll DB separately for each of these two authentication attempts and computes a set of 500 CCs. The CCs are plotted along the x-axis in Fig. 5 as the top-most and bottom-most curves. A similar process is carried out to construct the CCs for the remaining 7 curves in Fig. 5 but using PN from the other TV corners.

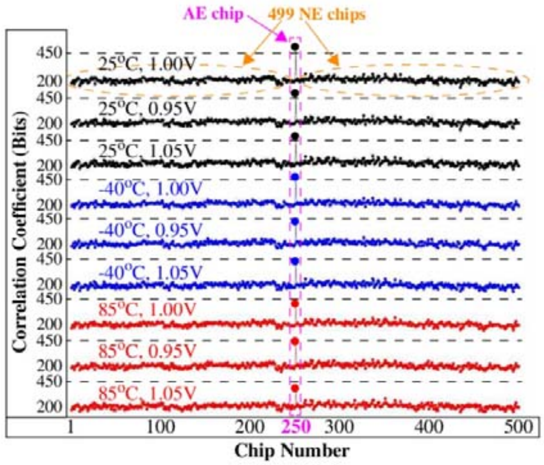

Figure 5 CCs (y-axis) for 500 chips (x-axis) across 9 TV corners obtained by correlating Helper Data bitstrings from the HELP PUF with a Margin of 3 and Modulus of 18. Peak at 250 occurs when Helper Data is derived using PN generated by the chip and matched with PN collected during enrollment for this same chip (called the authentic-enrolled or AE chip). All other CCs are derived by correlating

The CCs in Fig. 5 for C_{250} (referred to as the authentic-enrolled (**AE**) chip) vary from more than 450 bits in the top curve to approx. 380 bits in the bottom curve. Although the correlation is weakened when the chip is exposed to harsh environmental conditions, it still remains high with respect to the CCs produced by the remaining 499 chips (referred to as non-authentic-enrolled (**NE**) chips) from the DB. The largest value associated with a NE chip is approx. 260. The large margin between the AE and NE CCs suggests that it should be possible for the server to define a threshold to distinguish successful authentications from unsuccessful authentications with very high probability, e.g., any value between 260 and 380 works in this example.

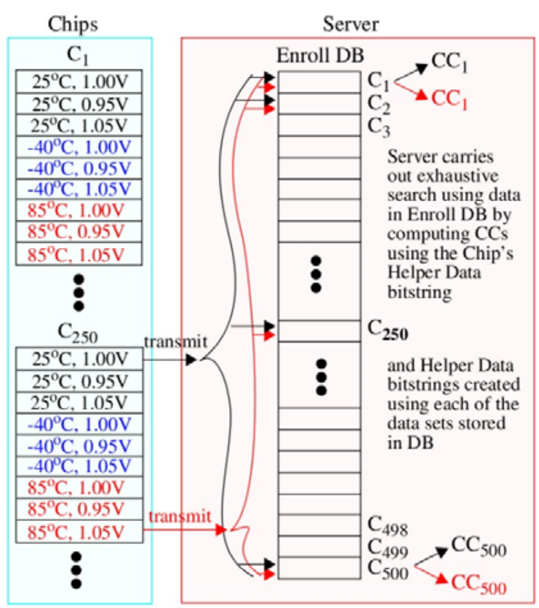

Figure 6 Graphic depicting process used to construct curves shown in Fig. 5. Each same-colored set of CCs displayed vertically are concatenated and shown along the x-axis in Fig. 5. This process represents exactly what the sever would do to identify and authenticate each Chip's Helper Data bitstring request transmitted to the server. Red-colored CCs correspond to the same C_{250} authenticating when the environmental conditions are 85°C, 1.05V as shown by the bottom-most curve in Fig. 5.

Note that our analysis considers only AE and NE authentication attempts. Two other possibilities include non-authentic-not-enrolled (**NN**) and non-authentic-counterfeit (**NC**) authentication attempts. Modeling NN authentication attempts is trivially accomplished by removing their data from the Enroll DB. It follows that attempts to authenticate by chips with no data in the Enroll DB would produce CCs similar to those produced for the 499 NE CCs shown in Fig. 5. Modeling NC authentication attempts is difficult without employing some type of machine learning algorithm if in fact an attack model can be devised. We leave this non-trivial task for future work.

We emphasize here that under the AND correlation sheme, it is possible for adversaries to construct Helper Data bitstrings with all '1's, which guarantees a large number of matches. However, large CCs would occur for ALL data sets in the secure DB, which, in turn, would be flagged by the server and result in a failed authentication attempt. This is true because the server allows only one CC to be above the threshold in order for an authentication attempt to be classified as successful.

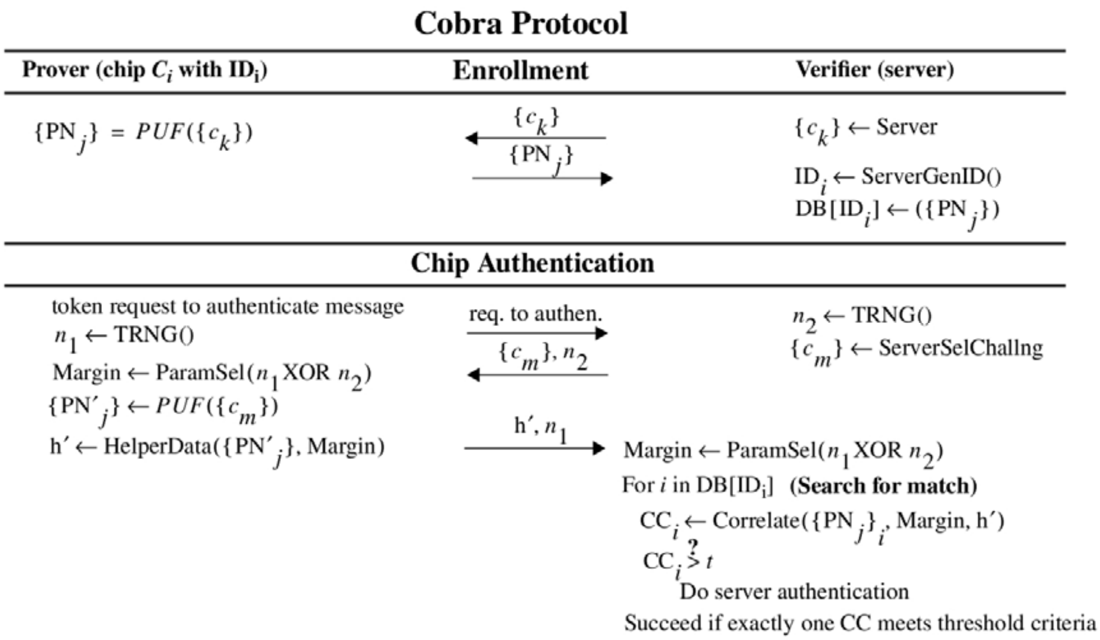

Figure 7 Cobra Protocol: Enrollment (top) and Chip Authentication (bottom) operations and message exchanges of proposed Helper Data bitstring correlation technique for implementing a privacy-preserving, mutual authentication protocol between chip (left) and server (right).

3. The Cobra Protocol

In this section, we describe the general structure of the proposed Cobra protocol. A graphical illustration of the Enrollment and Authentication operations, including the message exchanges between the chip and server, are presented in Fig. 7 along the top and bottom, respectively. Enrollment is performed in a secure facility using a confidential FPGA programming bitstream that allows access to the PUF's soft data. The server on the right generates challenges $\{c_k\}$ and transmits them to the chip on the left. The chip then applies the challenges to its PUF to generate the set $\{PN_j\}$, which is returned to the server. The server generates a chip identifier ID_i and stores the soft data set $\{PN_j\}$ under ID_i in its secure database *DB*.

The first phase of authentication is called Chip Authentication. Here, a fielded chip *i* requests authentication to the server (note, no chip *ID* is transmitted to the server in order to preserve privacy). The server generates a nonce n_2, selects a set of challenges $\{c_m\}$ and transmits them to the chip. The nonce n_2 is used to select a *Margin* parameter, and the challenges $\{c_m\}$ are a subset of the challenges $\{c_k\}$ used during enrollment. The chip applies the challenges $\{c_m\}$ to its PUF, along with a *Margin*, which is selected using the function *ParamSel*(n_1 XOR n_2), and generates a Helper Data bitstring h' using the Margining scheme described earlier. Both h' and n_1 are transmitted to the server.

The server then carries out a search by processing soft data sets $\{PN_j\}_i$ it stores in its database for each chip i. The routine *Correlate* produces a Helper Data bitstring h internally (not shown) using each of the stored data sets $\{PN_j\}_i$ and the same *Margin* that was used by the chip. The Helper Data bitstring h' is then correlated with h. Correlation based on a bitwise AND operation was shown in the previous section, but other possibilities exist including bitwise XNOR and/or standard waveform correlation methods. The output of *Correlate* is a correlation coefficient CC_i that is then compared to a threshold t. The authentication is considered successful if **exactly one CC** is larger than the threshold. The 'exactly one CC' constraint implements a countermeasure against simple adversarial attacks which use Helper Data bitstrings constructed with all 1's. This issue is discussed in detail in Section 5. The threshold is determined in advance using a characterization process in a secure facility. The goal of characterization is to select a threshold that unambiguously distinguishes authentic-enrolled chips (**AE**) from non-authentic-enrolled (**NE**), non-authentic-not-enrolled (**NN**) and non-authentic-counterfeit (**NC**) chips.

The last phase of the Cobra mutual authentication protocol is for the chip to authenticate the server. This phase is not shown in Fig. 7 but is similar except the message exchanges are reversed and the search process is omitted. Moreover, the AND-based correlation scheme used by the server to authenticate the chip cannot be used because the chip does not have access to the secure database. Instead, the chip uses XNOR correlation, which, as we discuss further below, requires matches to both 0's and 1's in the Helper Data bitstring received from the server and the bitstring produced on-the-fly by the fielded chip.

Server authentication is not performed unless chip authentication succeeds, in which case, the server has identified the chip's soft data set $\{PN_j\}_i$. The server uses this $\{PN_j\}_i$ to generate another Helper Data bitstring, which is transmitted to the chip. Note that the Helper Data bitstrings generated during server authentication are distinct from those generated during chip authentication because the challenge subset $\{c_m\}$ and nonces n_1 and n_2 are selected differently in this phase. Although the nonces select only the *Margin* parameter in this example, other parameters can be introduced to further expand the CRP space, as described below (and in reference [7]).

4. Experimental Results

This section carries out a worst case analysis using a larger set of the HELP CRP space and discusses the security properties of the Cobra protocol in more detail. Similar to the preliminary analysis presented in Section 2.5, the data analyzed here is collected from a set of 500 chip-instances under enrollment and 9 temperature-voltage (TV) corners.

4.1. HELP Challenge Space

The full CRP space of the HELP algorithm is defined by 1) sets of challenges (2-vector sequences) and corresponding *Path-Select-Masks* where each challenge set produces 4096 PN and 2) a set of parameters, consisting of two *LFSR seeds*, two floating point parameters called the *reference mean* and *range*, and a *Modulus* and *Margin* as discussed earlier in Section 2.4. The challenges and *Path-Select-Masks* create a CRP space with size exponentially related to the size of the functional unit used as the entropy source [21]. The parameters increase the CRP space by approx. 2^{20}, i.e., there are 2^{20} 2048-bit bitstrings that can be generated by varying these parameters for each set of challenges and *Path-Select-Masks*. Only one set of challenges and *Path-Select-Masks* are used for the analysis carried out in this paper and instead we focus on analyzing Helper Data bitstrings produced by varying only the parameters. Although this represents only a small subset of the entire CRP space, our results show that the Cobra technique works well across a statistically significant sample.

The two *LFSR seed* parameters allow up to 2048 distinct bitstrings to be generated, each of length 2048 bits. The *reference mean* and *range* increase the number of distinct bitstrings by a factor of approx.

128. The analysis performed here analyzes the Helper Data bitstrings generated using 10 distinct *LFSR seeds*, one combination of the *reference mean* and *range* and a set of 11 *Moduli* and 3 *Margins*.

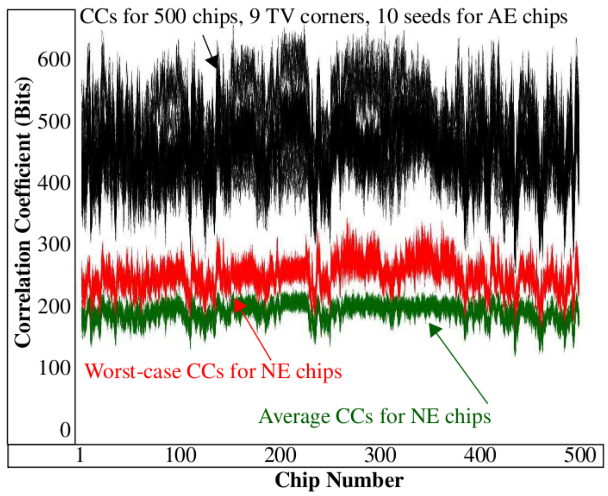

Figure 8 AE chip (black), worst-case NE chip (red), average-case NE chip (green) CCs with a Margin of 3 and Modulus of 18.

4.2. Illustration of Worst Case Scenario

The approach taken for the analysis of the worst case is illustrated using Figs. 8 and 9. The data presented is derived using only one *Modulus* and *Margin* in this section, and is expanded to the larger set in Section 4.3. The black curves in Fig. 8 plot the CCs computed by the server when each of the 500 AE chips identified along the x-axis authenticates. These curves are constructed as we illustrated for C_{250} in Fig. 6 but now include data for all 500 chips and 9 TV corners shown on the left in that figure, and for the 10 *LFSR seeds*. Therefore, there are 90 curves, each with 500 AE CCs. Similarly, the red curves in Fig. 8 plot the worst-case (largest) NE CC among the remaining 499 CCs in each authentication attempt while the green curves plot the average NE CC.

Note that the curves shown in Fig. 8 are in a different format than the curves shown in Fig. 5. In particular, the CCs in the black curves of Fig. 8 correspond to the AE chip only and the red curves plot only one CC (the largest, worst case value) from the 499 NE CCs in each of the curves of Fig. 5. Therefore, only two points from each of the curves in Fig. 5 are plotted in Fig. 8, and appear as column pairs in the black and red curves. The two points in each pair represent the worst case (smallest) separation between the AE and NE CCs and visually portray how close a NE chip gets to being falsely authenticated as the AE chip on the server. In summary, the number of black and red CC pairs is given by 500 * 9 TV corners * 10 *LFSR seeds* = 45,000. Note that each pair corresponds to 500 authentication attempts so in total, there are 45,000*500 = 22,500,000 (**22.5M**) authentication attempts.

As indicated, the key feature of this graph is the distance (separation) between pairs of points in the black and red curves. This separation is key to the server's ability to distinguish between AE and NE, NN and NC chip authentications. Unfortunately, a hard threshold (horizontal line) between the black and red points cannot be drawn without some AE authentications failing (false negatives) and some NE authentications succeeding (false positives).

To deal with this issue, we propose a new metric that computes the *percentage change CC*, called *PCC*, as follows. First, a set of CCs are computed using the chip's Helper Data bitstring against each of the server computed Helper Data bitstrings. Then the two largest CCs are identified and plugged in Eq. 1 to obtain the *PCC* value. The server successfully authenticates the chip if the *PCC* is larger than a hard threshold value, and fails otherwise. In cases where the authentication is successful, the soft data in the Enroll DB associated the largest CC, $CC_{largest}$, identifies the authenticating chip.

$$\text{PCC} = \frac{(CC_{\text{largest}} - CC_{\text{2nd\_largest}})}{CC_{\text{largest}}} \qquad \textbf{Eq. 1.}$$

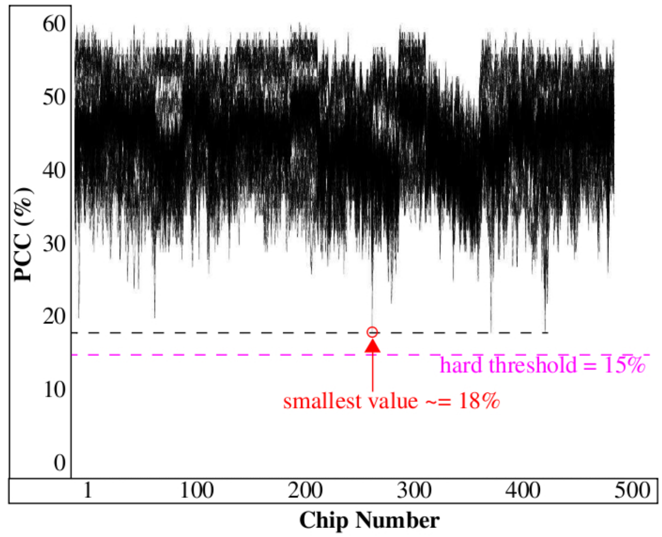

Figure 9 CCs as percentage change using AE chip and worst-case NE chip from Fig. 8.

The curves in Fig. 9 plot a closely related metric, defined as PCC_{AE_WC} in Eq. 2, using the CCs from Fig. 8. Here, CC_{AE} is associated with an AE chip obtained from the black curves in Fig. 8 and $CC_{worst\_case\_NE}$ is the other point in the pair obtained from the red curves. Note that in practice, we would not know the authenticate chip and therefore this analysis is somewhat artificial. However, it turns out for every CC pair in Fig. 8, $CC_{largest} = CC_{AE}$ and $CC_{2nd\_largest} = CC_{worst\_case\_NE}$. Therefore, Eq. 1 and Eq. 2 produce the same results for this set of CCs. In fact, the PCC_{AE_WC} would be negative if any of the CC_{AE} is not the largest CC among the 500 generated by the server for the authentication attempt. The smallest PCC_{AE_WC} present is circled in Fig. 9 and is approx. 18%. This indicates that all of the CC_{AE} are significantly larger than the NE CCs across the 22.5M authentications. Expressed in terms of bits, the smallest CC from Fig. 8 is approx. 300 bits. Therefore, the smallest separation between AE and NE CCs is approx. 300*0.18 = 54 bits. A **hard threshold** can now be defined, e.g., at 15% as shown in Fig. 9, that enables the server to properly identify and authenticate chips with high probability.

$$\text{PCC}_{\text{AE\_WC}} = \frac{(CC_{\text{AE}} - CC_{\text{worst\_case\_NE}})}{CC_{\text{AE}}} \qquad \textbf{Eq. 2.}$$

4.3. Validation using the Larger CRP Space

An analysis reporting PCC_{AE_WC} is expanded in this section to a set of 11 *Moduli* and 3 *Margins*. The analysis is carried out using bitwise AND correlation as described in the previous sections and is repeated using bitwise XNOR correlation. XNOR correlation further restricts the matching criteria over AND correlation to count matches to both '1's and '0's in the two Helper Data bitstrings. Bitwise XNOR correlation relaxes the criteria used for a successful authentication in the Cobra protocol from 'exactly one' to any CC that exceeds the threshold. This is possible because bitwise XNOR correlation measures the degree of matching between all bits in the Helper Data bitstrings, in contrast to bitwise AND which counts only the number of matching '1's. Interestingly, the two forms of correlation behave differently and are somewhat complementary as we discuss below and further in Section 5.

The PCC_{AE_WC} values are shown in Fig. 10 for Helper Data bitstrings derived using bitwise AND correlation along the top row and bitwise XNOR correlation along the bottom row. The bar heights represent the worst case PCC_{AE_WC} for a set of *Moduli* given along the x-axis and *Margins* given along the y-axis. Note that only one PCC_{AE_WC} is reported for each *Margin-Moduli* combination, and in particular, the value that corresponds to the circled point in Fig. 9 labeled 'smallest value'. Negative height bars represent that at least one authentication failure has occurred, i.e., at least one CC_{AE} is not the largest CC among the 500 CCs computed as we discussed in the previous section.

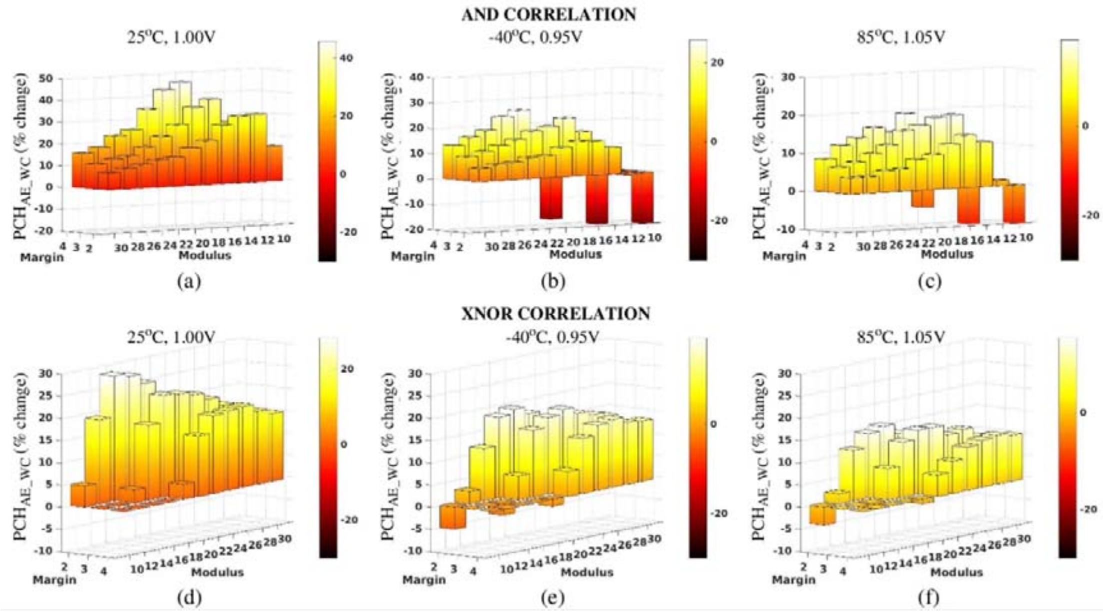

Figure 10 Worst case Correlation Coefficient (CC) differences between AE and NE chips expressed as percentage change using Eq. 2 under nominal environmental conditions, 25°C, 1.00V in (a) and (d), and worst case environmental conditions, -40°C, 0.95V in (b) and (e), and 85°C, 1.05V in (c) and (f) for 22.5M authentication attempts for Margins 2 through 4 and Moduli 10 through 30. Top row gives results using bitwise AND correlation and bottom row gives results using bitwise XNOR correlation. Positive bars indicate server identifies AE chip correctly in all 22.5M authentications while negative bars indicate at least one false authentication with an NE chip occurred.

The bar graphs in each of the three columns in Fig. 10 gives the results for three TV corners, namely 25°C, 1.00V in (a) and (d), -40°C, 0.95V in (b) and (e) and 85°C, 1.05V in (c) and (f). The latter two columns represent worst case TV corners, i.e., the results for the remaining 6 TV corners (not shown) produce bars that are larger. From these results, it is clear that the *Margin-Moduli* combinations with negative bar heights cannot be used in the Cobra authentication protocol. However, *Moduli* between 18 to 22 for *Margin* 3 and for 22 and 24 for *Margin* 4 produce bar heights greater than 15% when using AND correlation. For XNOR correlation, the behavior is reversed regarding the *Margin* where *Margins* of 2 and 3 produce better results (note that the bar graph orientation in the top row is rotated 180° in the bottom row). Here, the *PCCs* exceed 15% for *Moduli* of 16 and 18 for a *Margin* of 2 and for *Moduli* 20 and 22 for a *Margin* of 3.

The bar graphs in the left column for TV corner 25°C, 1.00V show the *PCCs* generated under nominal conditions. Unlike the results for the other TV corners, the bar heights for all *Moduli* and *Margins* are positive, indicating that the CC_{AE}s are the largest among the sets of 22.5M authentications carried out in each analysis.

5. Security Analysis

The effectiveness of the proposed Helper Data bitstring correlation method is directly related to two components of the soft data processing algorithm carried out within the PUF architecture. The first is temperature-voltage compensation, referred to as TVComp earlier. Under the HELP algorithm, TVComp scales and shifts the path delays (PN) measured on-the-fly by the chip to make them as similar as possible to the values measured during enrollment under nominal conditions. The proposed correlation techniques depend heavily on the effectiveness of TVComp. For other PUF architectures, e.g., the ARB, RO, NVM and metal PUFs, a similar form of TV compensation can be applied as a means of enabling correlation-based authentication as described here for HELP.

A second critically important component of the correlation methods is related to the Margining scheme portrayed within the center graphs of Fig. 3. Each of the two response bit regions, with '0' assigned to the region between 0-10 and '1' assigned to the region between 10-20, also contain two strong-weak region boundaries. Bit assignments within the Helper Data bitstrings depend only on these four strong-weak region boundaries and are independent of response bit boundaries at 0 and 10. In other words, a Helper Data bit can be assigned '0' or '1' in either of the response bit regions with equal probability. The symmetrical placement of the strong-weak region boundaries within each response bit region eliminates leakage in the Helper Data bitstrings and is the basis for the claimed improvements to model-building resistance of the correlation methods.

As mentioned earlier, when the AND correlation method is used by Cobra to generate *PCCs*, an authentication is deemed successful only in cases where **exactly one** server-computed *PCC* is above the threshold. The requirement of 'exactly one *PCC*' represents a countermeasure to adversarial attacks in which Helper Data bitstrings are artificially constructed with all '1's or a large fraction of '1's. The server is effectively screening for an outlier, i.e., one *PCC* that is significantly larger than all the others it computes during the exhaustive search process. A successful impersonation attack then requires the adversary to construct a Helper Data bitstring that is consistent with a matching server-generated version **at all bit positions**, i.e., both the '0's and '1's must be correlated. Otherwise authentication fails because more than *PCC* is above the threshold.

From the results shown in Fig. 10 (a), the AND correlation method performs best, i.e., produces the largest *PCCs*, when the fraction of '1's in the Helper Data bitstrings is small. The fraction of '1's (and '0's) is determined by the ratio of the *Margin* and *Modulus*. The Margining scheme requires the *Modulus* >= 4**Margin* + 2. As an example, when the *Margin* is 4, the *Modulus* must be at least 18. Assuming the PN are evenly distributed across the range defined by the *Modulus* (which is not valid for individual PN but is valid across a large collection of PN), the fraction of '1's is approximately equal to the sum of the two strong bit regions divided by the *Modulus*. For AND correlation, the best results are obtained when the strong response bit regions are size 1 or 2. In particular, the largest CC_{AE} is nearly 45% larger than $CC_{worst\_case\_NE}$ in Fig. 10 (a) for a *Margin* of 4 and a *Modulus* of 18. The fraction of '1's in this case is 2/18 ~= 11%.

The strong-weak bit regions acts as selection functions for the correlation methods. The AND correlation method is a one-sided selection function because only the 'strong bit' side of the boundary impacts the *PCC*. From the results, AND correlation performs best when the selection regions are asymmetrically skewed toward narrow strong bit regions.

In contrast, both sides of the strong-weak boundaries affect the XNOR *PCC*. This fundamental difference in the two correlation functions is reflected in the *PCCs* computed using different *Margins*. For AND correlation in Fig. 10 (a), the largest *PCCs* occur using a *Margin* of 4 for the majority (not all) of the *Margin-Moduli* combinations, while best case for the XNOR correlation occurs for a *Margin* of 2. The bar graphs in the second row are rotated 180^{o} to more clearly illustrate this characteristic. For example, the best results for XNOR occur for a *Margin* of 2 and for *Moduli* of 14 and 16. For these *Margin-Moduli* combinations, the size of weak and strong bit regions are nearly equal (they are equal for a *Modulus* of 16). Therefore, the two-sided XNOR selection function appears to work better for *Margin-Moduli* combinations that divide the entire region more evenly into weak and strong regions. Given the distinctive behavior of the AND and XNOR correlation functions, the Cobra protocol can in fact leverage both *PCCs* as a means of reducing the probability of false negative and false positive authentication decisions.

The overall decrease in the *PCC* magnitudes under adverse environmental conditions (Figs. 10 (b-c) and (e-f)) is caused by TVNoise. From the results, it is clear that the level of sensitivity of the *PCCs* to TVNoise is higher for smaller *Margins* and *Moduli*.

5.1. Overhead of the Cobra Protocol

The largest component of the overhead associated with the Cobra protocol is related to the PUF architecture. Reference [10], Table II gives the resource requirements for the HELP algorithm and original HELP protocol as 2,350 LUTs and 1,454 FFs with an additional 706 and 3,494 LUTs needed to implement the functional unit (entropy source) using an instance of AES SBOX and AES *sbox_mixcol*, resp. An instance of *sbox_mixcol* is used to generate the data presented in this paper. It consists of 4 AES SBOXs and 1 32-bit copy of AES mixed column implemented in a hazard-free combinational logic style. Cobra eliminates the need to generate the response bitstring and therefore, is slightly smaller by approx. 100 LUTs. For example, the size of the implementation used for experiments in this paper is 5,750 LUTs and 1,454 FFs, excluding the LUTs and FFs used by the Xilinx IP. Total size with Xilinx IP is 6,770 LUTs and 3,271 FFs, plus 1 MMCM and 1 25-bit multiplier.

5.2. Analysis of Cobra's Challenge-Response Space

As outlined earlier in Section 4.1, the full CRP space of HELP is defined by two components; 1) the values associated with a set of six parameters including 2 11-bit *LFSR seeds*, two floating point parameters called the *reference mean* and *range*, and a *Modulus* and *Margin*, and 2) the challenges and *Path-Select-Masks*. The challenges and *Path-Select-Masks* are used to select two sets of 2,048 PN from the larger set $\{PN_j\}$ stored in the secure database (see Fig. 7). The number of distinct 2048-bit Helper Data bitstrings that can be generated by varying the parameters is given by the product: 2048 * 128 * 2 * 2 = 2^{20} or 1 million. HELP first creates 2048 PN differences (PND) by subtracting unique pairs of elements from the two sets of 2,048 PNs. The factor 2,048 in the above product represents the number of ways unique sets of 2,048 PND can be created using the *LFSR seeds*. HELP applies a procedure called TVComp that shifts and scales the PND using two floating point parameters called the *reference mean* and *range*. The factor 128 in the product represents a conservative estimate on the number of ways the PND can be modified to produce unique response and Helper Data bitstrings by varying these parameters. Finally, the factors of 2 represent a conservative estimate on the number of *Margins* and *Moduli* that can be applied, as we discussed in Section 4.2. Note that 2^{20} represents the number of unique Helper Data bitstrings that can be produced using the 4,096 PN selected by one set of challenges and *Path-Select-Masks*.

The challenges and *Path-Select-Masks* represent the larger component of the CRP space. In [21], we used two sets of 7,500 PN as the enrollment data (15,000 PN), where each PN can be stored as a 16-bit fixed-point value, resulting in less than 32 KB of storage in the secure database per fielded device. Although the number of distinct PND that can be created from the two sets of 7,500 PN is $7{,}500^2$ ~= 56 million, the number of distinct response and Helper Data bitstrings that can be produced is much larger because of the Distribution Effect (which is the main topic of [21]). The challenges and *Path-Select-Masks* allow any two subsets of 2,048 PN from the 7,500 sets to be selected. The Distribution Effect is an artifact of the TVComp process which transforms any given PND into one of approx. 100 different compensated PND (PND_c). More importantly, it is not possible to predict the value of a PND_c unless the entire set of 2,048 PND are known. Although the Distribution Effect only increases the number of distinct PND_c to approx. 5.6 billion, the number of different distributions of 2,048 PND is characterized as (*7,500* select *2,048*) which is a very large exponential.

The entire CRP space with 15,000 PN per fielded device would then be lower bounded by the product of $2^{20} * 2^{32} = 2^{52}$, which accounts for both the parameters and challenge components. The large diversity of HELP's CRP space prevents replay attacks, and adds significantly to the difficulty of model-building attacks if in fact such attacks are possible using only Helper Data bitstrings.

6. Conclusions

A privacy-preserving, mutual PUF-based authentication protocol called Cobra is described in this paper. Cobra exchanges and correlates Helper Data bitstrings instead of PUF response bitstrings as a means of authenticating the chip and server. Helper Data is derived from the PUF response bitstrings and therefore the Helper Data bitstrings inherit the randomness and uniqueness characteristics associated with the PUF's source of entropy. By eliminating PUF response bitstrings in the message exchanges between the chip and server, attacks such as model-building are much more difficult to carry out. Cobra is demonstrated on a statistically significant set of FPGAs using the HELP algorithm, and a simple thresholding method is proposed that the server and chip can use for authentication. Although the HELP algorithm is used in this paper, the method is applicable to any PUF that produces soft data, i.e., digitized values that capture the magnitude of signal behavior such as delay or metal resistance. Future work will investigate the application of the Helper Data correlation method to other PUF architectures and will evaluate more traditional forms of correlation that are used in digital signal processing applications.

Author Contributions: "Conceptualization, Jim Plusquellic; Methodology, Jim Plusquellic; Validation, Matt Areno; ; Writing-Original Draft Preparation, Jim Plusquellic and Matt Areno;

Funding: "This research received no external funding"

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

References

1. L. Bolotny and G. Robins, "Physically Unclonable Function-based Security and Privacy in RFID Systems", *PerCom*, 2007, pp. 211-220.
2. G. Hammouri, E. Ozturk, and B. Sunar, "A Tamper-Proof and Lightweight Authentication Scheme", *Pervasive and Mobile Computing*, Vol. 4, Issue 6, 2008, 807-818.
3. A.-R. Sadeghi, I. Visconti, and C. Wachsmann, "Enhancing RFID Security and Privacy by Physically Unclonable Functions", *Information Security and Cryptography*, 2010, pp. 281-305.
4. U. Kocabas, A. Peter, S. Katzenbeisser, and A. Sadeghi, "Converse PUF-Based Authentication", *TRUST*, 2012, pp. 142-158.
5. M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching", *Symposium on Security and Privacy Workshop*, 2012, pp. 33-44.
6. A. Aysu, E. Gulcan, D. Moryama and P. Schaumont, "Compact and Low-power ASIP Design for Lightweight PUF-based Authentication Protocols", *IET Information Security*, Vol. 10, Issue 5, 2016, pp. 232-241.
7. W. Che, M. Martin, G. Pocklassery, V. K. Kajuluri, F. Saqib, and J. Plusquellic, "A Privacy-Preserving, Mutual PUF-Based Authentication Protocol", *Cryptography* 2017.
8. J. Aarestad, P. Ortiz, D. Acharyya and J. Plusquellic, "HELP: A Hardware-Embedded Delay-Based PUF", *Design and Test of Computers*, Mar., 2013, pp. 17-25.
9. W. Che, V. K. Kajuluri, M. Martin, F. Saqib and J. Plusquellic, "Analysis of Entropy in a Hardware-Embedded Delay PUF", *Cryptography*, 2017.
10. W. Che, F. Saqib and J. Plusquellic, "Novel Offset Techniques for Improving Bitstring Quality of a Hardware-Embedded Delay PUF", *Trans. on VLSI*, 2018.
11. http://ece-research.unm.edu/jimp/pubs/ARB_PUF.pdf, *UNM publication*, Mar. 2013.
12. B. Gassend, D. E. Clarke and M. van Dijk, S. Devadas, "Silicon Physical Unknown Functions", *Conference on Computer and Communications Security*, 2002, 148-160.
13. K. Lofstrom, W. R. Daasch, D. Taylor, "IC Identification Circuits using Device Mismatch", *International Solid State Circuits Conference*, 2000, pp. 372-373.
14. J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA Intrinsic PUFs and their use for IP Protection", *Cryptographic Hardware and Embedded Systems* (CHES), 2007, pp. 63-80.
15. Y. Dodis, L. Reyzin, A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data", *Advances in cryptology* (EUROCRYPT), 2004, pp. 523-540.
16. C. Bosch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi and P. Tuyles, "Efficient Helper Data Key Extractor on FPGAs", *Workshop Cryptographic Hardware and Embedded Systems* (CHES), pp. 181-197.
17. R. Helinski, D. Acharyya and J. Plusquellic, "A Physical Unclonable Function Defined Using Power Distribution System Equivalent Resistance Variations", *Design Automation Conference*, 2009, pp. 676-681.
18. W. Che, S. Bhunia and J. Plusquellic, "A Non-Volatile Memory based Physically Unclonable Function without Helper Data", *International Conference on Computer-Aided Design*, 2014.

19. http://ece-research.unm.edu/jimp/HOST/index.html, see video tutorials labeled 'HELP' and 'HELP Protocol'
20. W. Che, M. Martinez-Ramon, F. Saqib, J. Plusquellic, "Delay Model and Machine Learning Exploration of a Hardware-Embedded Delay PUF", *Symposium on Hardware-Oriented Security and Trust (HOST)*, 2018.
21. W. Che, V. K. Kajuluri, F. Saqib and J. Plusquellic, "Leveraging Distributions in Physical Unclonable Functions", Cryptography, MDPI, 2017.

1. © 2018 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).