Authenticache: Harnessing Cache ECC for System Authentication[*]

Anys Bacha
Computer Science and Engineering
The Ohio State University
bacha@cse.ohio-state.edu

Radu Teodorescu
Computer Science and Engineering
The Ohio State University
teodores@cse.ohio-state.edu

ABSTRACT

Hardware-assisted security is emerging as a promising avenue for protecting computer systems. Hardware based solutions, such as Physical Unclonable Functions (PUF), enable system authentication by relying on the physical attributes of the silicon to serve as fingerprints. A variety of PUF designs have been proposed by researchers, with some gaining commercial success. Virtually all of these systems require dedicated PUF hardware to be built into the processor or System-on-Chip (SoC), increasing the cost of deployment in the field.

This paper presents Authenticache, a novel, low-cost PUF design that does not require dedicated hardware support. Instead, it leverages on-chip error correction logic already built into many processor caches. As a result, Authenticache can be deployed and used by many off-the-shelf processors with minimal costs. We prototype, evaluate, and test the design on a real system, in addition to conducting extensive simulations. We find Authenticache to have high identifiability, as well as excellent resilience to measurement and environmental noise. Authenticache can withstand up to 142% of noise while maintaining a misidentification rate that is below 1 ppm.

1. INTRODUCTION

Security has become a critical design factor for computing systems today. As more of our personal data is collected, created, and consumed through interconnected devices, information security is becoming increasingly important. The rapid growth in mobile and wearable technology is driving the need for scalable designs that can autonomously safeguard digital content.

The mobility of today's computing devices makes them vulnerable to theft and tampering. Furthermore, the most ubiquitous form of secure transactions is marked by the generation and protection of private keys that can be used for encryption and authentication. As a result, the safeguarding of such keys represents an essential component of system security. Software-only security solutions may prove insufficient in preventing physical access attacks where partial information is maliciously recovered to reconstruct private keys.

In response to these challenges, researchers have proposed Physical Unclonable Functions (PUF), a security feature that intrinsically binds private keys to the physical attributes of the system. Silicon embedded PUFs share similarities with biometrics used for human identification in the sense that they possess fingerprints. They span a multitude of applications, including counterfeit detection, cryptographic key generation, memoryless key storage, and system authentication [1, 2, 3, 4, 5]. These designs exploit randomness caused by process variation in deep submicron silicon and are generally deployed in environments that use *challenge-response pairs* (CRP) for authentication. The mapping between challenges and responses is deemed difficult to reverse engineer since the relationship is governed by random physical properties.

A significant body of research has explored various forms of low-cost identification designs through silicon PUFs. For example, Lee et al. [6] proposed a circuit that arbitrates between a set of signals that traverse through a maze of cascaded switch blocks. Although the traversed paths are symmetrical by design, variation in the manufacturing process renders some paths faster than others. This forms the basis for generating a random output that is difficult to predict. Other delay-based designs [2, 7] compare the relative speed of ring oscillators in pairs using counters. The outcome of these tests vary from chip to chip due to process variation. Other techniques entail extracting randomness inherent in memory devices by examining default power-on states [8, 9] and inducing timing violations [10, 11].

Virtually all prior work we are aware of on silicon PUFs requires dedicated circuitry to be added to the processor, including custom made memory blocks [8, 9], arbitration logic [6], circuit delay monitors [2, 7], or support for special memory access modes [10]. The need for custom hardware support to enable security

[*]This work was supported in part by HP and the National Science Foundation under grant CCF-1253933.

MICRO-48, December 05-09, 2015, Waikiki, HI, USA
Copyright 2015 ACM. ISBN 978-1-4503-4034-2/15/12 ...$15.00
http://dx.doi.org/10.1145/2830772.2830814

can hinder the widespread adoption of PUFs, especially in mobile processors which have fast development cycles and relatively low profit margins.

In this paper we present Authenticache, a novel low-cost PUF design that does not require dedicated hardware support. Instead, it leverages on-chip error correction logic already built into the caches of modern processors. As a result, it can be deployed and used by many off-the-shelf processors with minimal cost.

Authenticache lowers the supply voltage of the chip to a level where on-chip caches exhibit errors that are corrected by the ECC logic. The inherently random distribution of these errors is used as the silicon-based fingerprint in our PUF design. Our solution relies on the observation presented in prior work [12, 13], that at low voltages, caches exhibit correctable errors that are randomly distributed and persistent. The random distribution means that different caches will exhibit errors in different lines. Within any given chip, the errors are persistent in the sense that, with high probability, the same lines will fail while under stress.

Authenticache associates the error distribution with an error map. Each point in the map is assigned a binary value where the cache lines that contain errors are set to "1" while the error free lines are cleared to "0." Starting from this map, Authenticache uses a challenge-response system to enable a large number of distinct challenges.

We demonstrate the robustness of Authenticache with a proof-of-concept prototype implementation, as well as extensive Monte Carlo simulations. Hardware-based PUFs are vulnerable to a combination of measurement error caused by noise, temperature variation, and circuit aging. Our evaluation shows that Authenticache is resilient to such perturbations and can tolerate up to 142% noise introduced into the system while maintaining a misidentification rate that is below 1 parts per million (ppm). It has near ideal uniqueness across a distribution of chips with an average inter-die variation of 49%. Finally, even without reusing authentication challenges, Authenticache can accommodate an average of 4.3K authentications per day over a 10 year lifespan when using a 4MB cache.

Overall, this paper makes the following contributions:

- Presents a novel security system that uses processor caches as Physical Unclonable Functions.
- Introduces a novel challenge-response design that employs error distributions to enable a large number of challenges with high noise resilience.
- Conducts a proof-of-concept implementation of Authenticache in firmware running on real hardware.
- Makes the observation that cache error maps can be dynamically constructed to serve as silicon-based fingerprints.
- Characterizes the robustness of the technique against environmental noise.

The rest of this paper is organized as follows: Section 2 provides background information. Section 3 characterizes correctable errors in caches using experimental data from real hardware. Section 4 presents the design and algorithm for the proposed authentication system. Section 5 describes the Authenticache prototype. Section 6 presents the results of our evaluation. Section 7 details related work; and Section 8 concludes.

2. PUF BACKGROUND AND METRICS

2.1 PUF System Authentication

System identification marks one of the primary PUF applications [1, 2, 3]. PUFs can be deployed as a replacement for password-based authentication between a client and a server. Client devices with integrated PUF hardware undergo an enrollment phase after manufacturing. During enrollment, a large number of challenge-response pairs (CRP) are gathered for each client by characterizing their PUF responses to multiple stimuli (challenges). These CRPs are collected and stored in a secure database that will assume the role of an authenticating server. Once the manufactured devices complete enrollment, they become eligible for authentication in the field.

The authentication process begins with a client issuing a request to the server. The authenticating server responds by randomly selecting a challenge from its database and sending it back to the client. At this point, the client runs the received challenge through its PUF and responds with the extracted output. The server verifies the client's response against its database and grants access upon a match. The strength of the aforementioned protocol can be improved by embedding multiple challenges within the same authentication transaction as needed.

2.2 PUF Quality Metrics

Authenticache was designed using quality goals commonly referenced in prior work [14, 15, 16, 17] for evaluating PUF designs. These include metrics such as uniqueness, reliability, identifiability, uniformity, and bit-aliasing. We provide a high level overview of these metrics and highlight their importance.

2.2.1 *Uniqueness*

This metric evaluates the PUF's ability to uniquely identify a chip in a distribution of k chips. In other words, it captures the likelihood of mistaking one PUF's response for another's. This is measured as the inter-chip variation of different responses using equation (1).

$$Uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{r_i \oplus r_j}{n} \times 100\% \quad (1)$$

The equation computes the hamming distance between a pair of n-bit responses r_i and r_j that are extracted from chips i and j such that $i \neq j$. The ideal value for this metric is 50%.

2.2.2 *Reliability*

Practical PUF designs must account for measurement and environmental noise that can affect the quality of the hardware's response. Noise can cause bits within the CRP response to be incorrect. The reliability metric estimates the susceptibility of a PUF to such conditions by comparing n-bit responses r_i and r'_i for a given challenge C. The responses r_i and r'_i are obtained under ideal and noisy operating conditions respectively from the same chip i. The reliability information is computed over m challenge samples to denote the intra-chip variation where e represents the e-th sample. The ideal value for the reliability metric is 100%.

$$Reliability = 100\% - \frac{1}{m}\sum_{e=1}^{m}\frac{r_i \oplus r'_{i,e}}{n} \times 100\% \qquad (2)$$

2.2.3 Identifiability

The identifiability property measures the ability of a PUF to correctly identify valid responses. PUFs compensate for noise by using an identification threshold that specifies how many bits in a response can be different from the expected result before it is considered invalid. Exposing a PUF to noise creates a distribution of intra-die hamming distances between possible responses and correct ones. Ensuring high identifiability requires reducing the overlap between this intra-die distribution and the inter-die distribution for random responses.

The selection of the identification threshold requires balancing between two criteria. On one hand, the threshold must reduce False Rejections (rejection of valid responses) while accounting for noise. On the other, it must remain relatively low such that it doesn't introduce False Acceptances of invalid responses. As a result, the appropriate identification threshold t_{id} is determined such that the False Acceptance Rate (FAR) and False Rejection Rate (FRR) components are minimized [18]. FAR and FRR are computed using a cumulative binomial distribution function F_{bino} as shown in equations (3) and (4).

$$FAR(t_{id}) = F_{bino}(t_{id}; n, p_{inter}) \qquad (3)$$

$$FRR(t_{id}) = 1 - F_{bino}(t_{id}; n, p_{intra}) \qquad (4)$$

In these equations n is the number of bits in a CRP while p_{intra} and p_{inter} represent the intra-distance and inter-distance binomial probabilities respectively. A common method for minimizing the FAR and FRR components is to select a threshold that corresponds to the Equal Error Rate (ERR) at which FAR is approximately equal to FRR.

2.2.4 Uniformity

Uniformity measures the randomness of PUF responses for a given chip i. It is estimated by examining the distribution of 0's and 1's across a sample of n-bit responses. This is illustrated in equation (5).

$$Uniformity(i) = \frac{1}{n}\sum_{j=1}^{n} r_{i,j} \times 100\% \qquad (5)$$

The ideal value for uniformity is 50% which means there is no bias towards certain values that may make the PUF vulnerable to model building attacks.

2.2.5 Bit-aliasing

Bit-aliasing is similar to the uniformity metric. It estimates the predictability of a response by measuring the bias of a given bit position across a distribution of k chips. This is expressed in equation (6). Similar to uniformity, the ideal value for bit-aliasing is 50%, which implies there is no bias towards 0's or 1's in any of the CRP's bits.

$$Bit - aliasing(j) = \frac{1}{k}\sum_{i=1}^{k} r_{i,j} \times 100\% \qquad (6)$$

In addition to the aforementioned metrics, PUFs that are designed for system authentication must fulfill other characteristics. They must have the ability to generate a large number of unique CRPs to prevent exploits where snooped transactions are maliciously replayed for authentication purposes. As a result, CRPs are generally employed only once to minimize the susceptibility to such attacks. Furthermore, responses that are sourced from a PUF must not disclose information about the internal layout of the device. This is important for protection against model building attacks.

3. CORRECTABLE ERRORS IN CACHES

Caches are among the most vulnerable on-chip structures [19, 20, 21, 22, 23]. They are optimized for density and therefore use the smallest transistors available in a given technology to promote large on-die storage capacities. SRAM designs rely on balanced device parameters to ensure stable operation. This stability is undermined by variation in the manufacturing process that leads to various parametric failures as a result of transistor strength mismatches within SRAM cells. The distribution of such failures conforms to process variation effects that are dominated by random phenomena such as fluctuations in transistor dopant density. This causes a random distribution of errors, making them suitable for PUF applications.

Authenticache uses the pattern of errors found in on-chip caches at low voltages as a basis for its PUF implementation. Prior work [12, 13] has shown that these errors can be triggered by lowering the chip's supply voltage at constant frequency. These errors are flagged and corrected by the on-chip ECC hardware. The locations of these errors are logged by the processor and can be accessed by software in many systems.

In order to characterize the errors in on-chip caches, we conducted experiments on a hardware platform that uses an Intel Itanium II 9560 processor [24] similar to the one used in [12]. The supply voltage (V_{dd}) of the cache was gradually lowered from a nominal setting ($\approx$ 0.8V) while it ran a built-in self-test meant to test each line in the processor's L2 caches. We combined these L2's to construct a 4MB cache similar to what can be found in mobile processors, such as Apple's A8 [25].

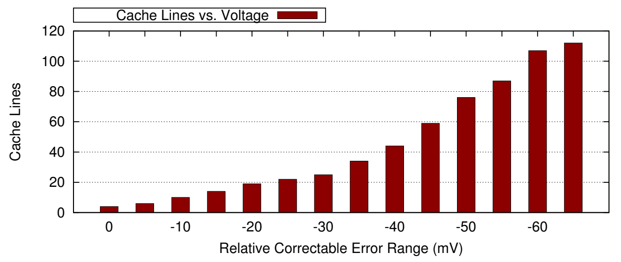


Figure 1: Number of distinct cache lines that trigger correctable errors as a function of voltage relative to the V_{dd} of the first correctable error (V_{corr}) in a 4MB cache.

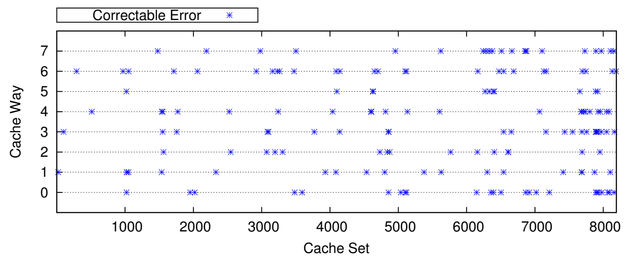


Figure 2: Distribution of correctable error locations at the minimum safe V_{dd} in a 4MB cache.

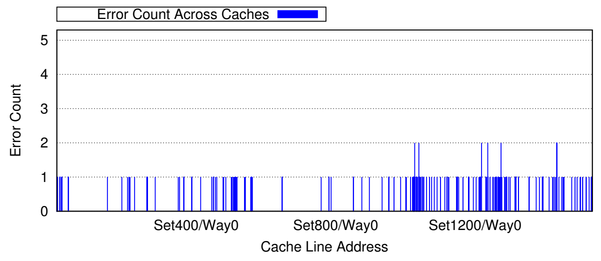


Figure 3: Addresses of correctable errors at minimum safe V_{dd} in 8 L2 caches of 768KB each. For each address we show the total number of errors across all 8 caches.

Figure 1 shows the relative supply voltage range over which cache lines exhibit correctable errors. We can see that once the V_{dd} crosses the correctable error range starting with V_{corr}, the number of unique cache lines reporting correctable events increases steadily to 122 over a 65mV reduction in V_{dd}. This corresponds to an average rate of 2 cache lines/mV. We note that these are benign errors that are corrected by the ECC hardware and do not affect the correct execution of the system. We show that this number of errors is sufficient to build PUFs that can generate a large number of CRPs as well as being resilient to noise.

Figure 2 shows how the errors are distributed in a 4MB cache. We can see that the distribution is uniform across all cache sets and ways. Our experiments also confirm that the errors are randomly distributed across different caches. Figure 3 illustrates the overlap in correctable error locations across eight different core caches (768KB each). We observe that even after superimposing the error locations from the eight different caches, only six addresses are repeated. We find that each one of these duplicated lines overlaps with a similar location in just one other cache.

To quantify the applicability of this data to Authenticache's challenge-response design, we conducted experiments using L2 caches of the same processor to determine the inter-die and intra-die components on hardware. We computed the inter-die distribution of 64-bit responses across the eight core caches consisting of the error locations shown in Figure 3. We found the inter-die distribution in this case to be approximately 44% (6% within the ideal value of 50%). We attribute this relative skew of the inter-die component from the ideal case to possible systematic process variation effects in addition to the small sample size.

Intra-die variation in responses to the same challenge captures the effects of noise on PUF reliability. To induce high levels of noise we conducted experiments at two different temperatures: normal and high, with a difference of 25 °C between the two. We collected responses to the same set of 64-bit challenges at each temperature. We measured an intra-die variation in responses of less than 6%. The absence of overlap between the inter-die and intra-die distributions indicates that our system can correctly distinguish between different chips even when exposed to changes in operating conditions, such as temperature.

Finally, while conducting these experiments, we observe that the correctable errors raised by the system are persistent. In other words, the vast majority of low-voltage errors are reproducible and will re-occur with high probability when self-tests are conducted at the same voltage level. Error persistence is crucial to our design since it allows our cache-based PUF to reliably generate correct responses to authentication challenges.

4. THE AUTHENTICACHE SYSTEM

Authenticache leverages the distribution of cache errors at low voltages as a foundation for its PUF, requiring no dedicated hardware support. The challenge-response function is formulated by mapping the error distribution into a 3D representation (x, y, z) of the cache layout at different voltages, as illustrated in Figure 4. The (x, y) dimensions represent a geographic mapping of the cache addresses, including sets and ways on a bi-dimensional plane. The z dimension represents different V_{dd} levels. Each point in the 3D depiction is associated with a binary value. Cache lines that are error free are given a value of "0," whereas cache lines that exhibit errors are set to "1."

4.1 Challenge and Response

One of our design objectives is to provide a large number of CRPs to ensure sufficient authentication transactions are available for the lifetime of the chip. A naïve approach to constructing the PUF function would be to formulate challenges that directly check for the presence of cache errors at specific geographic locations and supply voltages. However, this approach has the downside of disclosing the error information which lim-

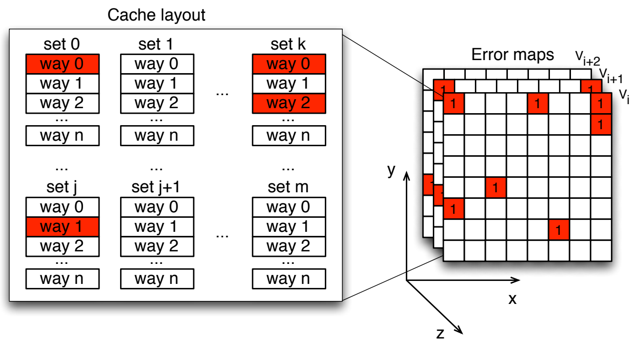


Figure 4: The mapping of cache line errors to Authenticache's error map.

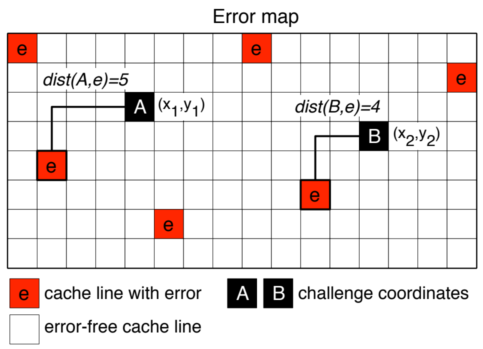


Figure 5: Authenticache error map example for $Challenge(A, B)$.

its the number of challenges to the number of errors observed by the cache. The responses would also be heavily biased towards "0," leading to poor uniformity since the error-free cache lines greatly outnumber the ones with errors. To circumvent these limitations, we design a challenge-response mechanism that indirectly leverages the location of cache errors.

Our solution selects arbitrary cache line pairs and measures the Manhattan distance from each of these cache lines to the closest cache line that contains an error. The challenge can be summarized with the question, *"Which of the two points, A or B, is closest to an error?"* More formally, let A and B be two cache lines that represent cache map coordinates $P_1(x_1, y_1, V)$ and $P_2(x_2, y_2, V')$ respectively. The challenge is defined by equations (7) and (8):

$$Challenge(A, B) = (P_1(x_1, y_1, V), P_2(x_2, y_2, V')) \quad (7)$$

$$Response = \begin{cases} 0, & \text{if } dist(A, e_1) \leq dist(B, e_2) \\ 1, & \text{if } dist(A, e_1) > dist(B, e_2) \end{cases} \quad (8)$$

where e_1 represents the closest error to point P_1, and e_2 the closest error to point P_2. The Manhattan distance between the points is computed as shown in equation (9).

$$dist(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2| \quad (9)$$

Figure 5 illustrates an example of an Authenticache challenge using cache lines A and B. In this example, we assume the challenge is conducted at the same supply voltage, thus $V = V'$. Since the Manhattan distance from point B to the closest error ($dist(B, e) = 4$) is smaller than the distance from point A to its closest error ($dist(A, e) = 5$), the response to this challenge is "1." Note that this represents only one bit of the entire challenge. The typical challenges we use range between 64 and 512 bits, requiring as many pairs of randomly chosen cache lines.

4.2 Challenge Diversity and Storage

The use of two arbitrary coordinates in the cache map to generate a challenge greatly increases the number of unique CRPs available to the system. With this approach, the number of possible challenges for a cache with n lines can be viewed as a complete graph G where each cache line is a vertex. Therefore, the amount of possible CRPs is equivalent to the number of edges in the fully connected graph G, as shown in equation (10).

$$Possible\ CRPs = \sum_{k=1}^{n-1} k = \frac{1}{2}n(n-1) \quad (10)$$

Traditional PUF implementations require CRPs to be stored on a server. Since the number of unique CRPs required for authentication over the lifetime of a given chip is very large (millions-billions), the storage space requirements can be substantial. Considering that each authentication server manages a large number of clients, the storage demands can become prohibitive. The Authenticache PUF design allows for a compact representation of each client's information. Instead of storing individual CRPs in a database, the Authenticache server only keeps the client's error data. Therefore, CRPs are generated in an on-demand fashion based on these error maps.

4.3 Authentication

Authenticache is designed to ensure safe and efficient interaction between a client and server in the field. The authentication process associated with our design is summarized in Figure 6. After receiving an authentication request, the server begins by selecting a supply voltage V_i that identifies the cache address information (cache sets and ways) to be used in constructing the error map. To mitigate the susceptibility of our design to model building attacks, the physical error layout is applied to a hash function that produces a logical error map based on a derived key, K_A. The server formulates a challenge using the produced logical map and issues it to the client. Upon receipt of the challenge, the client extracts the address information using the same key K_A, tests the cache at supply voltage V_i, and responds to the server. When the client's response is received, the server compares it with its computed response and makes an authentication decision.

Changing the supply voltage of the chip can be a

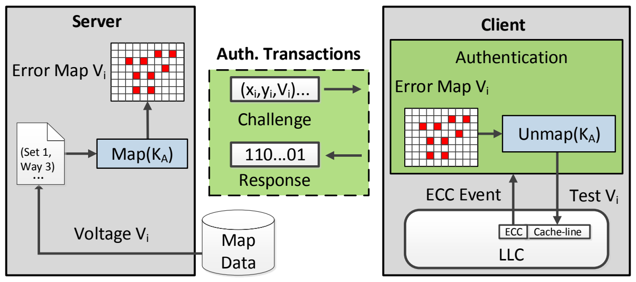


Figure 6: Authentication process between a client and server using error maps.

relatively slow process. Therefore, in order to accommodate performance sensitive applications, the full set of challenge bits can be restricted to a single supply voltage level. This way, a client device would only need to set the voltage once per challenge-response transaction. In cases where multiple voltage settings per challenge-response transaction are desired, the challenge bits can be re-arranged by the client in decreasing order of the supply voltage to minimize the delay associated with V_{dd} transitions.

4.4 Threat Model

We assume attackers could potentially intercept CRP transactions. In order to thwart replay and model building attacks, we do not allow challenges to be reused, regardless of the cache line ordering in the challenge. For instance, once challenge $C(A, B)$ is consumed, both $C(A, B)$ and $C(B, A)$ can no longer be used. As a result, the authentication server keeps track of previously issued challenges. While this requires additional storage over time, the amount needed is proportional to the number of authentications performed rather than the number of possible authentications. We expect this to result in lower storage overhead compared to traditional PUF authenticating servers.

Gaining physical access to the device presents another vector of attack. This approach assumes that an attacker has the appropriate knowledge about the chip and the ability to bypass the firmware. Under such circumstances, an attacker could collect the physical error map associated with the device for multiple voltages. Authenticache protects against physical access attacks in two ways. First, PUF access requires firmware level privileges. Second, Authenticache uses a hashing function that remaps the physical error locations. A successful attack would require the mapping of the physical error locations as well as successful extraction of the remapping key.

4.5 Adaptive Error Remapping

To mitigate the risk of model building attacks, we devised a secure mechanism that enables on-demand reconstruction of the logical error maps. This is accomplished by dedicating a fraction of the available V_{dd} settings to serve the purpose of supplying Authenticache's hash function with new keys. This process is

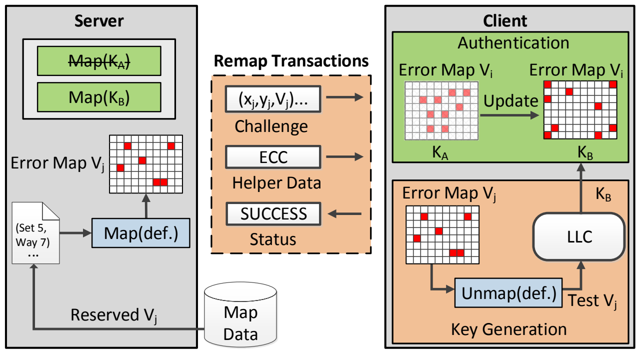


Figure 7: Illustration of the error map update process between a client and server through new keys.

described in Figure 7. To initiate a map update, the authenticating server identifies one of the reserved voltages, shown as V_j in Figure 7. The server uses the aforementioned voltage in conjunction with a default cache line mapping to generate a challenge out to the client. To ensure precise key derivation on the client side, the server provides error-correcting helper data along with the issued challenge. Once the client receives a map update request, it computes the response and combines it with the helper data to form the new key, K_B, that can be employed in future error maps. The client concludes this process by acknowledging the server with a completion status. It performs this without disclosing the response associated with the original challenge, so it is kept secret. For added security, multiple challenges involving different reserved V_{dd} settings can be chained for producing the final key.

5. AUTHENTICACHE PROTOTYPE

The client side of the proposed authentication system was prototyped on an HP Integrity Server that uses Intel Itanium 9560 processors. We implemented all the core functionality of the Authenticache client including initiation of the authentication process through the OS, firmware functionality for taking control of the processor during authentication, support for lowering the chip voltage, routines for self-testing and error handling, and the Authenticache PUF algorithm.

Figure 8 depicts the overall architecture of the prototype. It outlines the main components of the System Firmware and how our solution integrates into the existing framework. We describe this section using x86 terminology based on the readers' familiarity with this architecture. The equivalent Itanium interfaces employed in the actual prototype can be found in [26, 27].

5.1 Shadowed Execution

Client authentication is initiated through the transfer of control from the OS layer into a secure region in System Firmware. This is achieved through a special interrupt called System Management Interrupt (SMI). This mechanism enables System Firmware to conduct platform management tasks transparently from the OS. Therefore, once a client application running in user space is ready to initiate an authentication transaction, it indirectly generates an SMI through a loadable kernel module, prompting the interrupted core to enter System Management Mode (SMM). Once in SMM mode, various resources become shadowed from the OS and other less privileged entities. At this point, the processor that is currently residing in SMM will act as a master throughout the authentication process and in turn, synchronize the remaining cores to be in the same mode. This synchronization is achieved by broadcasting a set of interrupts to the remaining cores.

Upon entrance into the SMI handler that is part of System Firmware, all the processors are halted with the exception of the master which coordinates the remainder of the authentication process. This prevents any user code from running on the system during authentication. Employing SMM mode for authentication has distinct advantages. It has the advantage of serving as a protection layer through its trusted execution environment. For instance, execution in this mode is mapped to a special memory range that is known as System Management RAM (SMRAM). This range is only available while in SMM mode. It can be configured with a variety of memory attributes, including the ability to designate critical regions as uncacheable. This makes the design more robust against contention based exploits such as prime and probe [28, 29, 30]. Moreover, this approach has the benefit of ensuring a quiesced system state to reduce the impact of noise effects prior to the manipulation of the cache voltage settings.

5.2 Self-Test and Error Handling

The error handler is responsible for capturing the correctable error distribution from the processor cache. It serves two primary purposes throughout the different phases of platform operation: self-testing and error monitoring. During system calibration, the error handler is tasked with performing a series of self-tests through cache sweeps. The handler responds to ECC events that occur during this process, compiles the error rate information, and shares it with the voltage control system. This enables the voltage control system to make adaptive decisions about the lowest V_{dd} available during the authentication process at runtime. In addition to performing cache sweeps, the error handler is responsible for conducting targeted cache line testing. It receives a list of cache lines to self-test for a given challenge request. It accomplishes this by generating test bit patterns that are written into the designated cache lines followed by read transactions to the same set of lines. Finally, the module is provisioned with emergency detection capability. It monitors emergency events by tracking abrupt changes in the error rate and reacts by instructing the voltage control system to immediately raise the supply voltage if a pre-defined threshold is exceeded.

5.3 Voltage Control

The voltage control system is responsible for making dynamic voltage adaptation decisions based on feedback from the authentication and error handling modules.

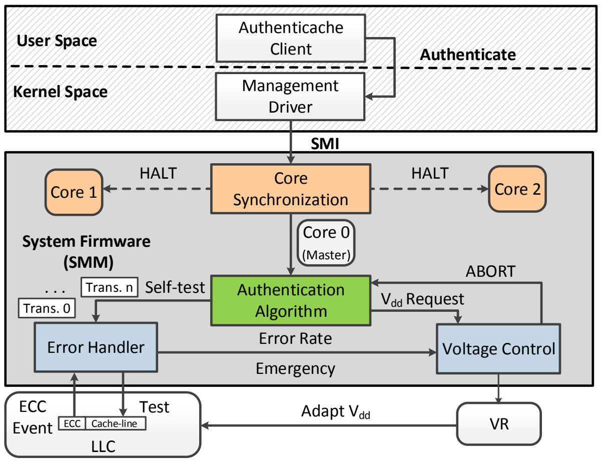


Figure 8: Main components and interfaces of the Authenticache prototype.

Voltage control can be activated during two distinct phases of operation: boot time and runtime.

During boot time, the voltage control system establishes a voltage *floor* that represents the lowest safe V_{dd} at which all triggered errors are correctable. Challenges are not permitted to use V_{dd} levels below this *floor* setting. This safeguards against malicious exploits that issue challenges containing unsafe voltages intended to crash the system. The $floor$ setting is different from chip to chip due to process variation. It is determined in tandem with the error handling module by progressively lowering the V_{dd} while performing built-in self-tests on the cache. The system periodically recalibrates its V_{dd} $floor$ to account for environmental changes, such as aging and temperature variation.

The second phase of operation for the voltage control system is during runtime. The primary role of this module throughout this phase is to service new V_{dd} requests issued by the main authentication algorithm. Once ownership of the processors is transferred to the OS, SMM mode marks the only path for invoking the voltage control system. This restriction protects against unauthorized access to the chip's low-V_{dd} error profile. In the event that an invalid V_{dd} setting is received, an $ABORT$ signal is issued back to the requester and the transaction is ignored. Otherwise, if the requested V_{dd} is valid, the voltage control system proceeds with setting the supply voltage to the appropriate level.

5.4 Authentication

The authentication algorithm is the centerpiece that coordinates the operation of the various entities within Authenticache. It begins by taking a challenge received from user space and sorting the individual bits in descending order according to their V_{dd}. This minimizes the number of V_{dd} transitions and their respective delays by grouping challenge bits that have the same voltage level. Since our prototype focuses on challenges with single V_{dd} settings as a proof-of-concept, we leave the optimization of multiple V_{dd} usage for future work.

As a next step, the module consumes the updated challenge and segments it into multiple atomic transactions that are bound by a maximum payload size. The different challenge bits embedded within these transactions are converted into their respective cache line addresses in preparation for self-testing. Once the cache line set and way information is determined, the voltage control system is instructed to lower the V_{dd} to match the setting dictated by the current challenge. The V_{dd} setting is followed by a transaction to the error handler that self-tests the neighboring cache lines in search of the closest error. The region of cache lines that are explored for each bit of the challenge is processed in an outward and clockwise fashion. As such, self-testing for each challenge bit is performed over its Von Neumann neighborhood starting with a range $r = 1$, $r = 2$, etc. until an error is triggered. Once a correctable error is discovered, the Manhattan distance is recorded for comparison. In the event that an *ABORT* is signaled at any point in the process, the algorithm simply terminates and resumes execution back to the OS with an error code indicating a failure.

6. EVALUATION

6.1 Methodology

The evaluation is based on the Authenticache prototype as well as extensive Monte Carlo simulations. In the case of Monte Carlo, each cache configuration was simulated with 100 distinct error maps where every map was evaluated against 50K noise profiles. For hardware based experiments, we used eight L2 caches from an Intel Itanium II 9560 processor [24]. The caches are based on 32nm CMOS technology and are protected with SECDED ECC. To evaluate the effect of temperature variation on our solution, we conducted experiments at normal and high temperatures, with a difference of 25 °C. The high temperature was achieved on hardware by slowing down the enclosure fan speeds of the server while running a power virus on all the processor cores. We examined the robustness of the system in adapting to environmental conditions as well as the impact of our technique on performance. We begin this discussion by characterizing the effects of noise on failure rates and the sensitivity of cache lines to self-tests at low V_{dd}.

6.2 Identifiability in Noisy Environments

A large body of research has demonstrated the impact of noise on the reliability of modern microprocessors [31, 32, 33, 34, 35, 36, 37]. Sources of noise include: static voltage drops across the power distribution network, dynamic voltage noise, temperature, and circuit aging induced by Negative Bias Temperature Instability (NBTI) and Hot-Carrier Injection (HCI) phenomena. In the case of Authenticache, we are concerned with this noise changing the error profile of the cache with respect to what was measured post-manufacturing. We classify these effects in two categories: the introduction of new errors and the masking or removal of errors identified in the recorded error map.

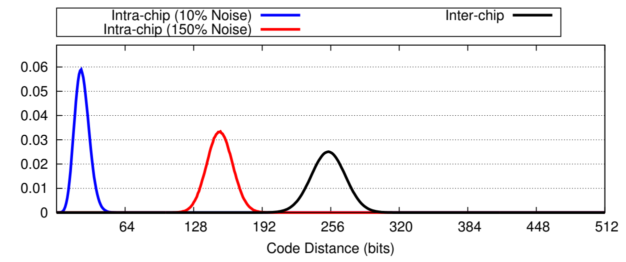


Figure 9: Hamming distance distribution for PUF responses from a 4MB cache with 512-bit challenges.

Unexpected new errors can occur as the system is exposed to noise sources such as voltage fluctuations and circuit aging. Having new cache lines added to the error distribution can lead to incorrect authentication. We measure PUF errors introduced by noise as the Hamming distance from the noisy response to the expected correct response. Figure 9 shows the distribution of Hamming distances for 512-bit challenges with 10% and 150% injected noise. These are Monte Carlo simulations of randomly generated error maps with 10% and 150% unexpected errors injected into the system relative to the number of existing errors (e.g. if the baseline cache has 100 errors, we add 150 new errors in the 150% case) . We also show the inter-chip Hamming distance distribution for randomly generated error maps relative to the current cache's distribution.

The 10% noise distribution represents what we would expect to encounter during normal operation. This distribution shows virtually no overlap with the inter-chip variation distribution, meaning the probability of misidentification at 10% noise is extremely low. Even with 150% of injected noise, the amount of overlap between the intra-die and inter-die components is small (around 2×10^{-6} or 2 ppm failure rate).

Overall, we find Authenticache to be remarkably resilient to injected noise. This is primarily due to the fact that Authenticache uses the error map indirectly by computing distances to errors, rather than using the location of the errors as part of the challenge. Injected errors induced through noise would have to occur in specific regions of the cache in order to flip the result of the challenge. Moreover, several bits within a given response would need to be affected in order to lead to incorrect authentication.

The second effect of noise Authenticache could face is error masking or removal. In this case, cache lines that are part of the error map fail to manifest when tested in response to a challenge. We expect this type of error to be induced predominantly through inaccuracies in measurement as cache lines undergo self-testing during the enrollment phase. For instance, noisy conditions during enrollment could cause an otherwise robust cache line to trigger an error. This error will be recorded in the error map, but may never trigger during authentication transactions.

We evaluate the robustness of Authenticache in the presence of new errors being introduced as well as ex-

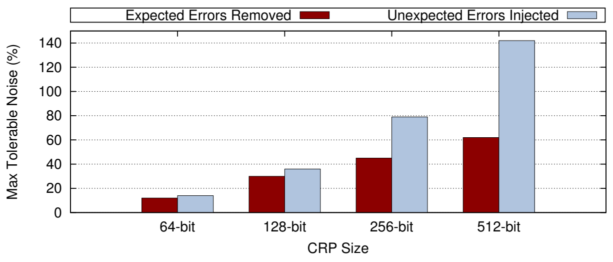


Figure 10: Maximum tolerable noise for maintaining a failure rate less than 1 ppm across multiple CRP sizes.

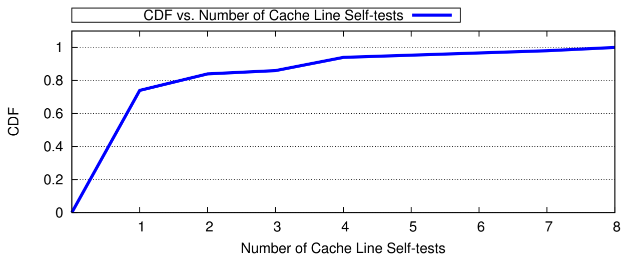


Figure 11: Cumulative distribution function of cache lines from the error map triggering correctable errors as a function of number of self-test attempts.

pected errors being removed from the error map. We collect data through Monte Carlo simulations where we consider 50K noise profiles across different chip configurations. We assume a 4MB cache size with 100 errors each in randomly generated distributions. The simulated cache closely resembles the characteristics of our prototype.

Figure 10 shows the maximum tolerable noise in percentage of errors either added or removed from the error map. The maximum acceptable noise is defined as any noise that results in authentication failure with a probability lower than 10^{-6} or 1 ppm – a threshold considered acceptable in prior work such as [8]. We show the maximum threshold for a range of CRP sizes from 64 to 512 bits.

Overall, Authenticache is quite robust against environmental noise. It can maintain a failure rate that is under 1 ppm while tolerating up to 142% and 79% of unexpected errors injected using 512-bit and 256-bit CRP lengths respectively. Similarly, we can tolerate up to 62% and 45% of error cache lines removed for the same CRP sizes respectively. We observe that sensitivity to noise increases inversely proportional to the CRP size. This data suggests that Authenticache is more susceptible to system errors being removed than injected. However, for CRP sizes above 128-bit, the maximum tolerable noise is much larger than one would expect to experience in the field.

6.3 Cache Error Persistence

Having cache lines that consistently and repeatedly trigger errors during self-test is fundamental to the design of Authenticache. We call this characteristic error persistence. Given that most cache errors encountered during low-V_{dd} self-test are caused by process variation, we expect them to exhibit high persistence. We conduct tests on our hardware prototype to verify this assumption.

We set the supply voltage to the minimum safe V_{dd} (*floor*) at which the system functions correctly and run targeted self-tests across 50 cache lines with known errors from different cores. We record the number of self-tests required to trigger each error. Figure 11 shows a cumulative distribution of all the cache lines we test as a function of the number of self-tests before an error manifests in each line.

The results confirm that indeed cache errors exhibit high persistence. We observe that 74% of the cache lines in the error map trigger an error during the first self-test attempt while 94% of the lines trigger on their fourth attempt. All cache lines in our error map trigger an error by their eighth attempt.

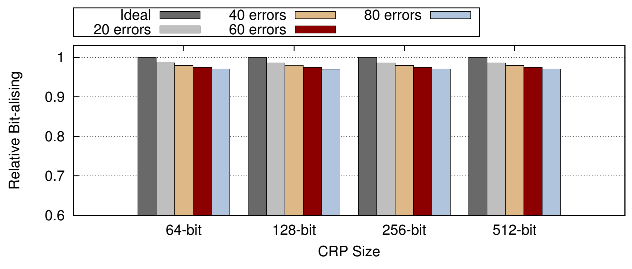
In order to ensure that all errors are triggered, the Authenticache PUF could test every cache line in the challenge at least eight times. This conservative approach, however, would increase the PUF's performance overhead. A more efficient alternative is to exploit Authenticache's noise tolerance and perform a smaller number of self-tests. In fact, all CRP sizes with the exception of the 64-bit CRP can deliver a failure rate that is less than 1 ppm with a single self-test attempt per cache line. This is because all CRP sizes larger than 64 bits can tolerate more than the 26% masked errors occurring in the single-attempt self-test scenario (Figure 10). Other configurations in this optimization space are possible (e.g. 2 self-tests per line for a 6% error masking rate, etc.).

6.4 Aliasing and Uniformity

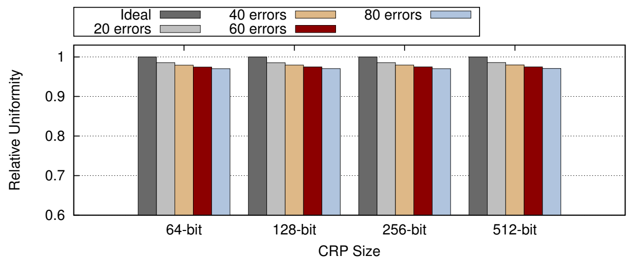
To evaluate the randomness and predictability of our design, we consider several Monte Carlo experiments that examine bit-aliasing and uniformity characteristics. We use a distribution of 100K CRPs across multiple chip configurations consisting of different error counts. In general, we observe that the bit-aliasing and uniformity results represented by Figures 12a and 12b respectively are within approximately 1% of their ideal values (49% on average).

While the differences are relatively small, we note a downward trend as the number of errors in the error map increases. This is due primarily to our PUF's slight bias towards "0." In other words, when the distances to the closest errors are equal, the PUF's outcome is "0" rather then "1" as indicated by equation (8). At higher error densities the distances between challenge points and errors are smaller. This increases the probability that given two coordinates in a challenge, the distances to their closest errors are equal.

Finally, we do not observe any notable changes in aliasing or uniformity as we vary cache sizes from 4MB to 64KB, provided we maintain the same error density. Similarly, we do not see significant differences as a result of changes in the CRP size.



(a) Bit-aliasing



(b) Uniformity

Figure 12: Bit-aliasing and uniformity of Authenticache relative to their ideal values for a 4MB cache using different numbers of errors and CRP sizes.

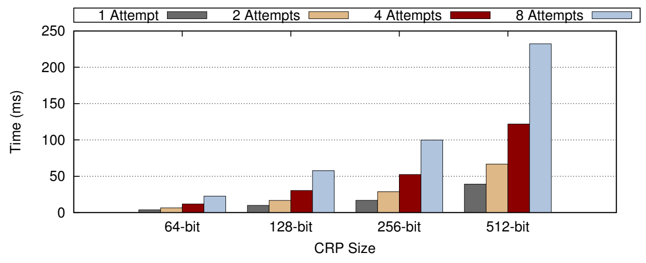


Figure 13: Runtime as a function of CRP size while using different per cache line self-test attempts in a 4MB cache.

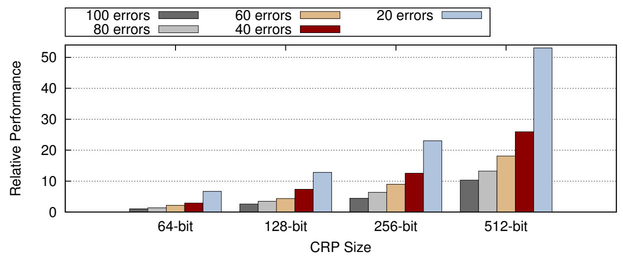


Figure 14: Performance as a function of the number of errors relative to a cache with 100 errors, for a 4MB cache.

6.5 Performance Overhead

The performance overhead of Authenticache is primarily a function of challenge size, number of self-test attempts, cache size, and number of errors in the PUF's error map. Figure 13 shows Authenticache's runtime for a single authentication as a function of CRP size based on measurements from our prototype system. As expected, the performance overhead increases mostly linearly with the CRP size. Larger CRPs have the advantage of being more robust against noise. Smaller CRPs, on the other hand, are faster to process. Similarly, the overhead increases linearly with the number of self-test attempts. Thus, it is expected that for each design the number of self-test attempts would be chosen based on the amount of environmental noise a system may be exposed to. Overall, a robust 512-bit CRP with a conservative 4 self-test attempts per cache line completes in less than 125ms. This is negligible overhead relative to the communication cost involved in connecting to the authentication server.

Figure 14 illustrates the performance overhead as a function of CRP size and number of errors in the error map for a 4MB cache. The performance is relative to a baseline with 100 errors and 64-bit CRP. We note that performance overhead increases as the number of errors in the error map goes down. This is due to the increased average distance between challenge coordinates and the closest errors, as the error map becomes more sparse. Identifying errors that are more distant requires additional cache lines to be tested. Thus, increasing the overall performance overhead.

Figure 15 shows the average distance to the nearest error as a function of the number of errors in the map. For all cache sizes, the average distance increases with fewer errors. On average, Authenticache performance improves by 1.6% for every new error in the map. This corresponds to a 0.5% decrease in the average distance for each new error.

6.6 Authenticache Lifetime

We expect Authenticache will scale well to different types of devices and applications, including mobile and server-class CPUs. Our study is based primarily on a 4MB LLC that can be found in today's mobile processors such as Apple's A8 [25] and a 32MB LLC such as the one on the Itanium processor we use in our prototype. Table 1 summarizes the available CRP count for different cache sizes and challenge lengths. These numbers represent the daily authentications that are available over a span of 10 years. Even when using the largest challenge size of 512 bits, a 4MB cache provides more than 1K daily authentication transactions. A 32MB cache, on the other hand, supports more than 73K daily authentications

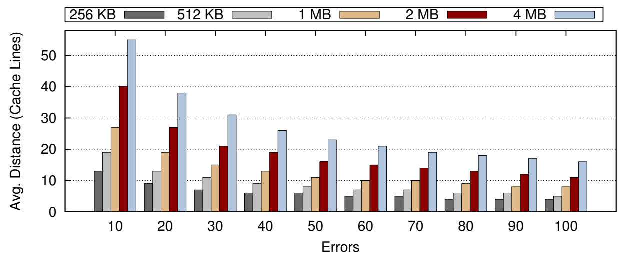


Figure 15: Average Manhattan distance to the nearest error as a function of total number of errors for different cache sizes.

over the same 10-year lifespan. Previous studies [38, 39] suggest that desktop users perform 8-12 authentications per day. While we anticipate this number to be higher for mobile clients, we envision it to be well within the daily numbers shown in Table 1. Furthermore, it is important to highlight that Table 1 only represents the available transactions at a single V_{dd}. Additional CRPs can be made available by using multiple V_{dd}s to generate more error maps.

Challenge Length	Auth. Per Day (4MB LLC)	Auth. Per Day (32MB LLC)
64-bit	9192	588350
128-bit	4596	291175
256-bit	2298	147088
512-bit	1149	73544

Table 1: Daily authentications available for different cache sizes and CRP lengths assuming a 10-year chip lifetime.

6.7 Model Building Attack Case Study

Modeling attacks against a client system generally involve the interception of CRP transactions. These CRPs could then be used to eventually clone the client's PUF [40, 41]. To evaluate the vulnerability of Authenticache to such attacks, we developed a model that progressively establishes dependencies between points in the error map based on observed CRPs. This allowed us to examine the expected amount of model training needed to accurately predict responses. We generated a large number of random and unique CRPs that were presented to the untrained model. For each observed CRP, we tested the accuracy of the prediction, then used it to further refine the model.

Figure 16 shows the percentage of correctly predicted response bits within a 64-bit challenge as a function of CRPs included in the training set. In this experiment, the challenges were confined to a single error map (single V_{dd} setting) to explore the worst case condition. We can see that initially, the model is able to predict approximately 50% of the response bits. This is expected given Authenticache's almost ideal uniformity. The accuracy of the model begins to improve after its training set reaches 40K CRPs. The ability to achieve 70% and 90% prediction rates requires access to 87K and 374K CRPs respectively.

These numbers show that staging a successful model building attack against Authenticache would be difficult. However, to further mitigate exposure to modeling attacks, we propose periodic regeneration of the mapping between physical and logical error locations using the mechanism described in Section 4.5. Regenerating the logical error map after a predefined number of CRPs have been consumed forces an attacker to retrain their model. The frequency at which the error map would be regenerated depends on the amount of noise a system must tolerate. For example, if the system expects 10% noise, then the logical map should be updated before the number of CRPs required to achieve 90% accuracy has been consumed.

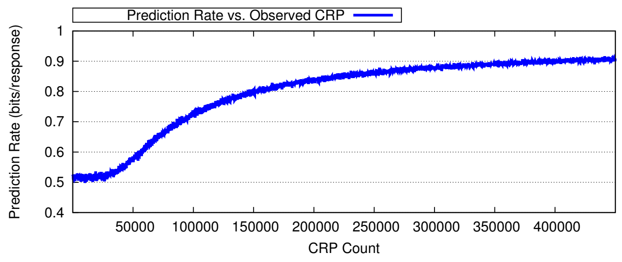


Figure 16: Model prediction accuracy as a function of observed CRPs.

7. RELATED WORK

7.1 PUF Designs

The majority of circuit-based PUF designs fall into two categories: delay-based and memory-based. Delay-based PUFs exploit the slow-down of certain paths relative to others as a result of process variation. Such PUFs mostly rely on signal propagation that is controlled through either arbitration or oscillation. In the case of arbitration, a set of external bits are applied to induce a race condition between data and clock signals as they propagate through predefined logic blocks [1, 3, 6, 42]. Oscillation-based techniques, on the other hand, rely on comparing the difference in accumulated oscillations between circuits over a predefined time duration [3, 7].

Memory-based PUFs exploit differences in how memory devices react to process variation. In general, the power-on state of balanced memory devices such as 6T SRAM cells, depends on the amount of stochastic noise present in the system. However, because of process variation, memory cells favor certain power-on states over others (e.g. "1" vs. "0"). Memory-based PUFs leverage this observation by constructing large blocks of SRAM that can be addressed randomly [8, 9]. In addition to comparing power-on states, some designs compare memory sensitivities to write failures [10, 11]. Other techniques leverage different storage elements such as latches instead of traditional SRAM cells [43].

Virtually all prior work we are aware of on silicon PUFs requires dedicated circuitry to be added to the processor or system, incurring design and deployment costs. Authenticache leverages existing on-chip error correction logic in processor caches and the error signature of each chip to implement system authentication with virtually no additional hardware support.

7.2 Side-Channel Attacks

Several studies have discussed the vulnerability of processor caches to side-channel attacks [28, 29, 44, 45, 46, 47, 48, 49, 50]. Based on our design, we anticipate techniques such as electromagnetic emanation and power fluctuation analysis to present likely vectors of attack against Authenticache. For instance, an attacker could utilize the aforementioned approaches to correlate signatures with ECC related activity in an attempt to reverse engineer the error map. To mitigate this risk, firmware can be configured to interleave authentication related accesses to the cache with random transactions.

Prime and probe is another technique commonly applied against caches [28, 29, 30]. Although Demme et al. [44] suggest that large caches have a low vulnerability factor, firmware could serve as a protection layer against such attacks, irrespective of the cache size. For example, the authentication handler could be marked to utilize uncacheable regions as a way of preventing malicious user-processes from scanning the firmware's working set after cores are resumed back to the OS.

7.3 Other PUF Applications

Cryptographic key generation represents another application for PUFs. This type of application is generally reserved for PUFs that possess a limited number of CRPs. Unlike traditional key generators, PUFs have the advantage of obviating the need for secure non-volatile memory and expensive tamper sensing packages as a result of their inherently random behavior. However, PUFs suffer from the susceptibility to environmental noise conditions which make it difficult to accurately reproduce previously generated keys. To address this issue, error correction techniques can be employed to enable precise reconstruction of noiseless keys [1, 2, 3, 7, 51, 52, 53]. Similarly, the output of such PUFs can be used as seeds into pre-existing key-generation algorithms.

8. CONCLUSION

The rapid growth in mobile and wearable technology is driving the need for cost effective designs that can autonomously safeguard digital content. In this study, we propose Authenticache, a novel PUF-based authentication mechanism that leverages correctable errors in caches as fingerprints. This approach demonstrates that a silicon-based PUF without dedicated hardware is not only feasible, but very robust and highly resilient to environmental and measurement noise. We realize a proof-of-concept to show that the system is practical and inexpensive to deploy in production.

We evaluate the proposed solution using both hardware and extensive simulations. Our results demonstrate Authenticache's ability to withstand up to 142% of noise while maintaining a misidentification rate that is below 1 ppm. We characterize the design across a variety of configurations including different cache capacities, correctable error profiles, and CRP sizes. Finally, we present a secure mechanism that mitigates Authenticache's exposure to model building attacks, enabling the design to sustain thousands of daily authentications.

Acknowledgements

The authors would like to thank the anonymous reviewers for their feedback and revision suggestions. We also extend special thanks to Edward Suh, Yinqian Zhang, and members of The Ohio State Computer Architecture Research Lab for their valuable discussions and insights on this work. Finally, we thank HP for providing infrastructure assistance to make this research possible, and the National Science Foundation for financial support under grant CCF-1253933.

9. REFERENCES

[1] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, pp. 1126–1141, August 2014.

[2] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Design Automation Conference (DAC)*, pp. 9–14, 2007.

[3] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pp. 3–37, Springer Berlin Heidelberg, 2010.

[4] "Verayo Simply Secure." http://www.verayo.com.

[5] "Intrinsic ID." http://www.intrinsic-id.com.

[6] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *IEEE Symposium on VLSI Circuits*, pp. 176–179, 2004.

[7] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *ACM Conference on Computer and Communications Security (CCS)*, pp. 148–160, 2002.

[8] 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 Workshop*, 2007.

[9] D. E. Holcomb, W. P. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," *Proceedings of the Conference on RFID Security*, 2007.

[10] A. R. Krishna, S. Narasimhan, X. Wang, and S. Bhunia, "Mecca: A robust low-overhead PUF using embedded memory array," *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*, pp. 407–420, 2011.

[11] Y. Zheng, M. S. Hashemian, and S. Bhunia, "RESP: A robust physical unclonable function retrofitted into embedded SRAM array," in *Design Automation Conference (DAC)*, pp. 1–9, 2013.

[12] A. Bacha and R. Teodorescu, "Using ECC feedback to guide voltage speculation in low-voltage processors," in *International Symposium on Microarchitecture (MICRO)*, pp. 297–307, December 2014.

[13] A. Bacha and R. Teodorescu, "Dynamic reduction of voltage margins by leveraging on-chip ECC in Itanium II processors," in *International Symposium on Computer Architecture (ISCA)*, pp. 297–307, June 2013.

[14] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *Embedded Systems Design with FPGAs*, pp. 245–267, Springer, 2013.

[15] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 94–99, 2010.

[16] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs," in *International Conference on Reconfigurable Computing and FPGAs (RECONFIG)*, pp. 298–303, IEEE Computer Society, 2010.

[17] Y. Su, J. Holleman, and B. P. Otis, "A digital 1.6 pj/bit chip identification circuit using process variations," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 69–77, January 2008.

[18] R. Maes, *Physically Unclonable Functions: Constructions, Properties and Applications*. PhD thesis, Katholieke Universiteit Leuven, 2012.

[19] A. Agarwal, B. Paul, S. Mukhopadhyay, and K. Roy, “Process variation in embedded memories: failure analysis and variation aware architecture,” *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 1804–1814, September 2005.

[20] B. Calhoun and A. Chandrakasan, “A 256-kb 65-nm sub-threshold SRAM design for ultra-low-voltage operation,” *IEEE Journal of Solid-State Circuits*, vol. 42, no. 3, pp. 680–688, 2007.

[21] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, “Statistical design and optimization of SRAM cell for yield enhancement,” in *International Conference on Computer-aided Design (ICCAD)*, pp. 10–13, 2004.

[22] Y. Pan, J. Kong, S. Ozdemir, G. Memik, and S. W. Chung, “Selective wordline voltage boosting for caches to manage yield under process variations,” in *Design Automation Conference (DAC)*, pp. 57–62, 2009.

[23] B. Zhai, D. Blaauw, D. Sylvester, and S. Hanson, “A Sub-200mV 6T SRAM in 0.13μm CMOS,” in *International Solid-State Circuits Conference (ISSCC)*, pp. 332–606, February 2007.

[24] R. Riedlinger, R. Bhatia, L. Biro, B. Bowhill, E. Fetzer, P. Gronowski, and T. Grutkowski, “A 32nm 3.1 billion transistor 12-wide-issue Itanium processor for mission-critical servers,” in *International Solid-State Circuits Conference (ISSCC)*, pp. 84–86, February 2011.

[25] J. Ho, B. Chester, C. Heinonen, and R. Smith, “The iPhone 6 review: A8’s CPU: What comes after Cyclone?.” AnandTech, September 2014. http://www.anandtech.com.

[26] “Intel Itanium architecture software developer’s manual, 2010, revision 2.3.”

[27] “Intel Itanium processor family system abstraction layer specification, 2008, revision 3.3.”

[28] F. Liu and R. B. Lee, “Random fill cache architecture,” in *International Symposium on Microarchitecture (MICRO)*, pp. 203–215, December 2014.

[29] D. A. Osvik, A. Shamir, and E. Tromer, “Cache attacks and countermeasures: The case of AES,” in *The Cryptographers’ Track at the RSA Conference on Topics in Cryptology (CT-RSA)*, pp. 1–20, 2006.

[30] C. Percival, “Cache missing for fun and profit,” in *The Technical BSD Conference (BSDCan)*, May 2005.

[31] Y. Kim, L. K. John, S. Pant, S. Manne, M. Schulte, W. L. Bircher, and M. S. S. Govindan, “AUDIT: Stress testing the automatic way,” in *International Symposium on Microarchitecture (MICRO)*, pp. 212–223, December 2012.

[32] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, and J. B. Carter, “Active management of timing guardband to save energy in POWER7,” in *International Symposium on Microarchitecture (MICRO)*, pp. 1–11, December 2011.

[33] R. Bertran, A. Buyuktosunoglu, P. Bose, T. Slegel, G. Salem, S. Carey, R. Rizzolo, and T. Strach, “Voltage noise in multi-core processors: Emperical characterization and optimization opportunities,” in *International Symposium on Microarchitecture (MICRO)*, pp. 368–380, December 2014.

[34] M. S. Gupta, J. L. Oatley, R. Joseph, G.-Y. Wei, and D. M. Brooks, “Understanding voltage variations in chip multiprocessors using a distributed power-delivery network,” in *Design Automation and Test in Europe (DATE)*, pp. 624–629, 2007.

[35] D. Herrell and B. Beker, “Modeling of power distribution systems for high-performance microprocessors,” *IEEE Transactions on Advanced Packaging*, vol. 22, no. 3, pp. 240–248, 1999.

[36] R. Joseph, D. Brooks, and M. Martonosi, “Control techniques to eliminate voltage emergencies in high performance processors,” in *International Symposium on High Performance Computer Architecture (HPCA)*, pp. 79–90, February 2003.

[37] M. Shevgoor, J.-S. Kim, N. Chatterjee, R. Balasubramonian, A. Davis, and A. N. Udipi, “Quantifying the relationship between the power delivery network and architectural policies in a 3D-stacked memory device,” in *International Symposium on Microarchitecture (MICRO)*, pp. 198–209, 2013.

[38] E. Hayashi and J. Hong, “A diary study of password usage in daily life,” in *SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 2627–2630, 2011.

[39] D. Florencio and C. Herley, “A large-scale study of web password habits,” in *International Conference on World Wide Web (WWW)*, pp. 657–666, 2007.

[40] U. Rührmair and J. Sölter, “PUF modeling attacks: An introduction and overview,” in *Design Automation and Test in Europe (DATE)*, pp. 348:1–348:6, 2014.

[41] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, “Modeling attacks on physical unclonable functions,” in *ACM Conference on Computer and Communications Security (CCS)*, pp. 237–249, 2010.

[42] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, “Extracting secret keys from integrated circuits,” *IEEE Transactions on Very Large Scale Integration Systems*, vol. 13, pp. 1200–1205, Oct. 2005.

[43] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, “The butterfly PUF protecting IP on every FPGA,” *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008.

[44] J. Demme, R. Martin, A. Waksman, and S. Sethumadhavan, “Side-channel vulnerability factor: A metric for measuring information leakage,” in *International Symposium on Computer Architecture (ISCA)*, pp. 106–117, 2012.

[45] R. Callan, A. Zajić, and M. Prvulovic, “A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events,” in *International Symposium on Microarchitecture (MICRO)*, pp. 242–254, 2014.

[46] D. Gullasch, E. Bangerter, and S. Krenn, “Cache games – bringing access-based cache attacks on AES to practice,” in *IEEE Symposium on Security and Privacy (SP)*, pp. 490–505, 2011.

[47] D. J. Bernstein, “Cache-timing attacks on AES,” tech. rep., The University of Illinois at Chicago, 2005.

[48] D. Page, “Theoretical use of cache memory as a cryptanalytic side-channel.” Cryptology ePrint Archive, Report 2002/169, 2002. http://eprint.iacr.org/.

[49] J. Bonneau and I. Mironov, “Cache-collision timing attacks against AES,” in *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 201–215, 2006.

[50] O. Aciçmez, W. Schindler, and c. K. Koç, “Cache based remote timing attack on the AES,” in *Cryptographers’ Track at the RSA Conference on Topics in Cryptology (CT-RSA)*, pp. 271–286, 2006.

[51] M.-D. M. Yu and S. Devadas, “Secure and robust error correction for physical unclonable functions,” *IEEE Design and Test of Computers*, vol. 27, pp. 48–65, Jan. 2010.

[52] Z. S. Paral and S. Devadas, “Reliable and efficient PUF-based key generation using pattern matching,” in *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 128–133, 2011.

[53] M. Bhargava and K. Mai, “An efficient reliable PUF-based cryptographic key generator in 65nm CMOS,” in *Design Automation and Test in Europe (DATE)*, pp. 1–6, 2014.