NoisFre: Noise-Tolerant Memory Fingerprints from Commodity Devices for Security Functions

Yansong Gao®, Yang Su®, Student Member, IEEE, Surya Nepal®, and Damith C. Ranasinghe®

Abstract—Building hardware security primitives with on-device memory fingerprints is a compelling proposition given the ubiquity of memory in electronic devices, especially for low-end Internet of Things devices for which cryptographic modules are often unavailable. However, the use of fingerprints in security functions is challenged by the small, but unpredictable variations in fingerprint reproductions from the same device due to measurement noise. Our study formulates a novel and pragmatic approach to achieve highly reliable fingerprints from device memories. We investigate the transformation of raw fingerprints into a noise-tolerant space where the generation of fingerprints is intrinsically highly reliable. We derive formal performance bounds to support practitioners to easily adopt our methods for applications. Subsequently, we demonstrate the expressive power of our formalization by using it to investigate the practicability of extracting noise-tolerant fingerprints from commodity devices. Together with extensive simulations, we have employed 119 chips from five different manufacturers for extensive experimental validations. Our results, including an end-to-end implementation demonstration with a low-cost wearable Bluetooth inertial sensor capable of on-demand and runtime key generation, show that key generators with failure rates less than $10^{-6}$ can be efficiently obtained with noise-tolerant fingerprints with a single fingerprint snapshot to support ease-of-enrollment.

Index Terms—Hardware fingerprinting, memory fingerprinting, SRAM, flash, EEPROM, root key, error reconciliation

1 INTRODUCTION

VARIOUS schemes have investigated fingerprinting commercial-off-the-shelf (COTS) devices to build security applications: verifying the provenance of integrated circuits (IC) to guard against counterfeiting by fingerprinting IC packages [1]; identifying unlawful 3D printed products by fingerprinting unique textures resulting from 3D printers [2]; authenticating smartphones by fingerprinting the Photo-Response Non-Uniformity of a camera image sensor [3]; and identifying commodity mobile devices by fingerprinting on-board sensors [4], [5].

Compared with fingerprinting methods for on-board sensors and other components like central processor units (CPUs) [1], [3], [4], [5], [6], [7], [8], [9], [10], [11], fingerprinting embedded memories—including static random access memory (SRAM) [12], [13], [14], dynamic random access memory [15], Flash memory [16], [17], and electrically erasable programmable read-only memory (EEPROM)—pervasively embedded in COTS devices is a highly desirable proposition for provisioning security functions, especially in the absence of cryptographic modules. Fingerprinting embedded memory is attractive because: i) memory cells are intrinsic to computing platforms and available in large volumes to obtain many independent fingerprints or secret keys; ii) memory biometrics provides a physical source of true randomness; iii) it removes the need for a protected non-volatile memory for secrets (root keys can be generated on-demand and “forgotten” after usage); and iv) imparts no extra hardware costs to existing COTS devices such as medical devices, wireless sensors, credit cards, wearable devices, and a plethora of low-end Internet of Thing (IoT) devices, which are projected to grow to 75.44 billion worldwide by 2025 [18].

1.1 The Challenge

Whenever a fingerprint is generated from the same device, the digitized fingerprint should be exact for its use in security

![Fig. 1. Conventional schemes extract raw fingerprints from individual memory cells, such as from the random power-up state of each SRAM cell. However, raw fingerprints $f$ regenerated at different time instances from the same device can mismatch a reference raw fingerprint template $f$ due to native bit errors introduced by noise $e$. The red curves in the fingerprint symbol depict errors resulting from noise.](image-url)
functions. However, fingerprints generated at different time instances are susceptible to unpredictable noise, such as thermal noise, supply voltage fluctuations, and device aging, and, consequently, differ in some bits. First, the positions of flipped raw bits vary. Second, the number of flipped raw bits varies from time to time. Thus, it is challenging to determine reliable raw bits, and existing memory fingerprinting schemes cannot naturally tolerate noise in the raw, noisy fingerprint space. Therefore, it is often infeasible to regenerate a fingerprint identical to a reference template that is securely stored (e.g., in a server) for directly building security functions between it and devices, as shown in Fig. 1.

Until now, using approximate and noisy renditions of biometric fingerprint templates in security functions has been demonstrated to be possible with fuzzy extractor (FE) based methods [19], [20]. The FE employs a generation function to transform “fuzzy” biometrics into private secrets together with helper data used in a subsequent reproduction function to reconcile errors and derive the exact private secret from an approximately close template of the original biometric [21], [22], [23]. Employing an FE on a device leads to two fundamental problems: i) the computation overhead introduced on a device by FE logic is high [24] and ii) the associated helper data can be actively manipulated, in helper data manipulation (HDM) attacks, to weaken or even compromise the security of the derived fingerprint or private secret [25], [26]. A generic countermeasure against HDM attacks remains an open challenge [26].

Hence, there remains a significant leap between the desire for re-purposing ubiquitously available memory for security functions and the practicality of exploiting memory fingerprints for security.

While the notion of exploiting tiny hardware fabrication variations to generate memory fingerprints is not new, we challenge the traditional method of reliable fingerprint provisioning and pose the following research questions (RQs):

RQ1: How can we extract intrinsically reliable fingerprints from device memories?

RQ2: If an approach does exist, is the method pragmatic and usable for fingerprinting memory resources on pervasive commodity computing devices?

1.2 Our NoisFre Concept

Current memory fingerprinting schemes extract a fingerprint bit from each memory cell. Fingerprinting under this scheme is susceptible to noise. To the best of our knowledge, for commodity memories, existing techniques fail to accurately capture the noise-tolerance degree of each raw bit to formally determine those extremely reliable bits for direct key usage without the problematic error reconciliation.

We recognize that device memories are a cost-free and abundant source of entropy. Attributing to the ever-decreasing fabrication costs, the size of memory pervasively embedded within devices has become increasingly large. Hundreds of kilobytes (KiB), even in low-end devices, are common (see the devices we tested in Table 1). Consequently, we envision that the entropy of extracted information may be sacrificed for improved reliability. Therefore, we propose the concept of transforming the raw fingerprint space of high information density into a lower-dimensional space with the attribute of being largely invariant to noise—bit flips—observed in the digitized raw fingerprint space or memory biometrics. We refer to this noise-tolerant memory fingerprinting concept as NoisFre.

We illustrate our concept in Fig. 2. Building upon a raw memory biometric source that is a noisy fingerprint space, we propose extracting new fingerprints $F$ in the deliberately transformed noise-tolerant fingerprint space, which can tolerate a desirable noise bound $\theta$. Here, as long as the noise $e$, induced number of raw fingerprint bit errors is less than $\theta$, the regenerated and transformed raw fingerprints are guaranteed to be projected to its reference counterpart $F$ enrolled at the server. More generally, the regenerated and transformed fingerprint $F$ is insensitive to bit errors (resulting from noise) in the raw fingerprint space. Therefore, it can be directly employed—without error reconciliation—as a root key in a security function, despite the noisy renditions of the raw fingerprints at times $t_1$, $t_2$, and $t_3$.

Significantly, we recognize that the best strategy for fingerprint memory is not always directly from the raw noisy fingerprint space, such as directly treating the power-up state of an SRAM memory of a cell as a fingerprint bit, the foundation for all current memory fingerprinting schemes. We argue for exploiting the freely available, abundant entropy of memories. We do not focus on individual raw fingerprint bits but seek to find an invariant property of a group of raw bits to measure, so we can be less concerned of the complexity about the process generating those bits.

![Fig. 2: Illustrating the use of noise-tolerant memory fingerprints from commodity devices for security functions. We transform the raw fingerprint from an $n$-dimensional noisy fingerprint space to an $m$-dimensional space, we refer to as the noise-tolerant fingerprint space, where $m < n$. In the noise-tolerant space, as long as the noise $e_u$ is less than a bound $\theta$, the regenerated and transformed fingerprint can be correctly projected to the reference transformed fingerprint template $F$ securely enrolled and stored on the server. Red curves in the fingerprint symbol depict errors in the raw fingerprint upon regeneration at time instances $t_1$, $t_2$, and $t_3$. Now, the $F$ obtained can serve as a root of trust or a root key for a security function.](Image)
1.3 Contributions and Results

- We exploit the freely available and abundant entropy from memories to propose a new concept—NoisFre—for highly reliable fingerprinting of commodity device memories. The principle is based on transforming from a noisy raw fingerprint space to a lower dimensional, noise-tolerant fingerprint space capable of reconciling noise inherent across multiple measurements of the same raw fingerprint. To corroborate the proposed NoisFre concept, we have developed two specific transformation methods: i) S-Norm and ii) D-Norm (RQ1).

- We formulate analytical models with the expressive power to support the design of security functions and evaluate the transformation methods. We express i) an upper bound for the unreliability of the transformed $F$ bits with respect to the transform function parameters; and ii) the expected fingerprint extraction efficiency—the number of transformed $F$ bits that can be extracted from a given memory size (RQ2).

- We conduct elaborate and extensive evaluations with a synthetic chip model to obtain the massive number of repeated fingerprint measurements necessary to validate our formalization of unreliability and extraction efficiency. Billions of repeated measurements were simulated using the synthetic chip model with bit-level modeling capable of capturing bit-error behavior in SRAM device memories. Our formal models are confirmed to be worst-case bounds in practice (RQ2).

- We extensively test: i) 110 SRAM memory devices from three different manufacturers to experimentally validate NoisFre performance. We focus on SRAM memory, as it is the most commonly embedded memory, especially for low-cost IoT devices. Further, we employ: ii) seven Flash memories and iii) two EEPROM memories for validating the generalizability of NoisFre (RQ2).

- To demonstrate the expressive power of our formalization, we investigate the derivation of a root key—the foundation for realizing various security functions. We demonstrate a 128-bit root key with an extremely low key failure rate of less than $10^{-6}$ can be directly obtained by transformed fingerprints to obviate the need for costly noise reconciliation. Significantly, a fingerprint snapshot or single measurement is sufficient for enrolling a key, a process we follow in all our experiments (RQ2).

- As a case study, we implement a NoisFre key generator and a security function on a low-end wearable Bluetooth inertial sensor. We extract a root key directly from native SRAM fingerprints transformed into noise-tolerant $F$ bits for use in a remote attestation primitive. By fundamentally obviating the state-of-the-art method necessary for reconciling noisy key bits, we demonstrate a significant overhead reduction (i.e., 54% compared to reverse FE and 82% compared to FE) and enhanced security. By utilizing the power isolation features, we also demonstrate the realization of run-time and on-demand generation of robust SRAM fingerprints $F$ on this low-end device. A video demo is available at https://youtu.be/O5NWZw-swpw (RQ2).

- We release the 100 chip SRAM memory fingerprint dataset that we collected and open-source code artifacts to facilitate future research at https://github.com/AdelaideAuto-IDLab/NoisFre.

2 Background

We concisely describe the well-known methods of fingerprinting memories. Then we briefly introduce the commonly accepted (reverse) FE to reconstitute a “fuzzy” secret into cryptographic secrets for security functions.

2.1 Fingerprinting Device Memories

We consider the widely used, specifically in low-end devices, SRAM, Flash, and EEPROM memory fingerprinting. As for the SRAM memory, when SRAM is powered up, each cell exhibits a favored power-up state; such an initial state varies from cell to cell and chip to chip. Therefore, each SRAM cell’s power-up state is treated as a fingerprint bit. Fingerprinting SRAM is closely related to the SRAM physical unclonable functions [12], [13].

To extract fingerprints from Flash memory [16], [17], all Flash cells on the same page are first erased to “0”. Then partial programming is applied. As a result of tiny fabrication variations, some cells will remain in state “1” while others flip to “0”. Which cell remains or flips is determined by the random and uncontrollable fabrication process variations. One can treat whether a cell flips as the fingerprint bit—a flip as logic “0”, and otherwise “1”. The partial programming period is pre-determined to ensure balanced “0”/“1” bits in practice. The same procedure is applicable to fingerprint EEPROM.

2.2 Reliable Secrets With Fuzzy Extractors

A widely accepted method to turn noisy hardware fingerprint bits (key material) into usable cryptographic keys is to use an FE [19], [20]. In general, the FE consists of two procedures: i) a secure sketch and ii) an entropy extraction. The secure sketch reconciles errors in the regenerated bits. The entropy extraction (e.g., a cryptographic hash function) compresses the bits into a uniformly distributed cryptographic key with full bit entropy.

The secure sketch construction has a pair of operations, as shown in Fig. 3: i) encoding and ii) decoding. Typically, in the FE setting, the encoding employing an error correction code (ECC) encoder is executed by the server during the fingerprint template enrollment phase to compute helper data $p$ (redundant information). The decoding employing an ECC decoder is performed on the in-field device to recover a reliable fingerprint, $sk$. By recognizing that the encoding function’s computational burden is significantly higher than decoding, it is feasible to place the ECC encoder on the device-side while leaving the computationally complex ECC decoder to the resource-rich server; this method is termed reverse FE (RFE) or reusable FE [19], [27], [28]. More specifically, the encoding function is implemented on the device-side to produce the associated helper
We observe that a transformed bit, $F$, is able to mitigate the impact from multiple raw fingerprint bit errors manifesting as permutations or combinations of an $n$-bit raw fingerprint, $f$. The concept we propose is surprisingly simple but efficient and practical because of the important but inad-vertent reality of large memory volumes intrinsic to devices. From a practical consideration, our critical insight is that memory embedded within modern electronics is large and provides abundant entropy to be exploited without additional costs for security functions. This fact is the foundation for our NoisFre transformation method: trade-off entropy for reliability.

This work proposes two specific NoisFre transformation methods: Single $\ell_1$-Norm (S-Norm) and Differential $\ell_1$-Norm (D-Norm).

3.2 Single $\ell_1$-Norm Transformation (S-Norm)

The $\ell_1$-Norm of a vector is the distance of the vector from an all-zero vector—or the Hamming weight of a vector, as described in Definition 1.

Definition 1 ($\ell_1$-Norm). Let $f$ be a binary vector length $n$ representing a noisy raw fingerprint where $f_j$ is the $j$th bit in $f$; then the $\ell_1$-Norm of $f$ is defined as:

$$\|f\|_1 \triangleq \sum_{j=1}^{n} f_j.$$  

Interestingly, an $\ell_1$-Norm of a vector is permutation invariant. Hence, a new bit $F$ can be obtained by applying an $\ell_1$-Norm over a raw fingerprint vector $f$ as $\|f\|_1$ and as described by the S-Norm below.

Definition 2 (S-Norm). Let the $i$th raw fingerprint bit vector of $n$ bits, where $n$ is an odd integer, be $f_i$. Then S-Norm transform is defined as:

$$F = \begin{cases} 1, & \|f_i\|_1 \geq \left\lceil \frac{n}{2} \right\rceil, \\ 0, & \|f_i\|_1 < \left\lceil \frac{n}{2} \right\rceil. \end{cases}$$

An illustrative example of S-Norm-based transformation is provided in Fig. 5. When $\|f\|_1 > \frac{n}{2}$, where $n$ is an odd integer, the $F$ bit is “1,” and otherwise “0”. This transform has the first desirable property of being insensitive to bit errors manifesting as permutations of a raw fingerprint. For example, despite the raw bit errors at time $t = 1$ for the raw fingerprint from **Block 1** that lead to a permutation in
Fig. 6. Illustrating the role of the noise tolerance parameter $\theta$ in S-Norm-based selection. The plots show the $\ell_1$-Norm distribution of raw noisy fingerprints. It approximates normal distribution. A larger $\theta$ ensures the transformed $F$ bits can tolerate a higher degree of noise.

**Observation 1:** The S-Norm $||f||_1$ yields a representation analogous to the reliability of the new bit $F$.

**Observation 2:** The S-Norm transformed bits are invariant to permutations and combinations of raw bit patterns. Further, $\theta$ provides a desirable lower bound on raw bit errors tolerated by the transform.

**Observation 3:** There is an expected trade-off evidence in Fig. 6. While increasing $\theta$ increases the noise tolerance of the transform, it reduces the number of noise-tolerant fingerprint bits extractable from a given memory.

### 3.3 Differential $\ell_1$-Norm Transformation (D-Norm)

Considering Observation 3 and the distribution in Fig. 6, we recognize that a distance measure capable of presenting a bimodal distribution could provide an intrinsic separation of groups of underlying raw fingerprint bits with the potential to yield higher numbers of noise-tolerant bits. We hypothesize that a differential distance measure may afford such a desirable distribution and propose the D-Norm transform based on a differential distance measure.

**Definition 4 (D-Norm).** Let the lowest and highest $\ell_1$-Norm of $m$ groups (each group is an $n$-bit vector) be $l$ and $h$, respectively, where:

$$h \triangleq \text{arg max}_{f_i \in \{1,...,m\}} (||f_i||_1),$$

$$l \triangleq \text{arg min}_{f_i \in \{1,...,m\}} (||f_i||_1).$$

Now, following the general definition in Section 3.1, the D-Norm transform is defined as:

$$\ell_1^{\text{D-Norm}}(f_i) = \begin{cases} 
0 & \text{if } ||f_i||_1 \leq \frac{m}{2} + \theta \\
\text{sign}(f_i) \left( ||f_i||_1 - \frac{m}{2} - \theta \right) & \text{if } \frac{m}{2} + \theta < ||f_i||_1 < \frac{m}{2} + \theta \\
||f_i||_1 & \text{if } ||f_i||_1 \geq \frac{m}{2} + \theta 
\end{cases}$$

Fig. 6. Illustrating the role of the noise tolerance parameter $\theta$ in S-Norm-based selection.
The raw bit values are measured at two different time instances, \( t = 0 \) and \( t = 1 \), from each block to illustrate the manner in which the D-Norm transform is reliable against raw bit error.

\[
F = \begin{cases} 
1, & h - l \geq 0 \text{ and } |h| < |l| \\
0, & l - h < 0 \text{ and } |h| > |l|.
\end{cases}
\]  

(6)

Here, we denote the spatial index \( i \) (memory address, in practice) of the vector \( \mathbf{f}_j \), chosen for \( h \) based on Equation (4) or \( l \) based on Equation (5) using a square bracket, \( \lbrack h \rbrack \).

The D-Norm-based transformation is illustrated in Fig. 7. In the illustration, the \( \ell_1 \)-Norm of two blocks of \( m = 3 \) groups of \( n = 8 \) bit vectors are evaluated at time \( t = 0 \). In subsequent evaluations of the fingerprint at \( t = 1 \):

- In **Block 1**, we can observe the permutation invariance property, similar to the S-Norm. For example, the highest \( \ell_1 \)-Norm at \( t = 0 \) and \( t = 1 \) is \( h = 5 \) for the third 8-bit vector despite repeated generation of the raw bits not being exact.

- In **Block 2**, we further observe the difference of \( h - l \) is \( 6 - 1 = 5 \) at \( t = 0 \) and shows an extreme case of \( 3 - 3 = 0 \) at \( t = 1 \), where \( F \) bit of “1” remains invariant. Which reflects the combination invariance property.

In both **Block 1** and **Block 2**, the fingerprint bit \( F \) remains robust to the raw fingerprint bit error patterns observed at different measurement times. However, a combination of \( n \) bits with a single change in the number of raw binary “1” bits can lead a D-Norm projection at the proximity of the decision boundary in Equation (6) to cross that boundary. Hence, the resulting \( F \) bits from such raw fingerprints effectivley display low reliability. Therefore, similar to S-Norm, we propose winnowing raw fingerprints based on their \( |h - l| \) projections. We describe the D-Norm-based Selection method in Definition 5 and generalize the approach using a noise tolerance parameter \( \theta \) to bound the combinations of bit patterns the transform needs to tolerate, using the differential \( \ell_1 \)-Norm of the raw fingerprint vectors measured once (i.e., at \( t = 0 \)).

**Definition 5 (D-Norm-Based Selection).** From a block of \( m \) different \( n \)-bit raw noisy fingerprint vectors \( \mathbf{f}_i \) for \( i \in \{1, \ldots, m\} \) extracted at \( t = 0 \), the block is selected for fingerprinting the device using D-Norm if \( h \) and \( l \) as defined in Equations (4) and (5) satisfy:

\[
|h - l| \geq \theta.
\]  

(7)

To understand the significance of the D-Norm-based selection method and the role of the noise tolerance parameter \( \theta \), we employ the Nordic Semiconductor chip fingerprinting technique used in S-Norm. The resulting distribution of monitoring measurements (at time \( t = 0 \)) for two cases of a small and a large \( \theta \) for blocks of \( n \times m \) raw fingerprint bits is shown in Fig. 8. Interestingly, the distribution of \( |h - l| \) approximates a bimodal distribution; each mode represents those vectors mapping to \( F = "1" \) and “0”, respectively. Importantly, the two clear groupings of \( n \times m \) bit blocks based on the D-Norm distance measure results in an intrinsic separation.

Now, consider the blocks of \( h - l \) raw fingerprint bit vectors (green bar) in blocks at the boundary of the selection criteria, in Equation (7), where \( |h - l| = \theta \) for the two cases of a small and a large \( \theta \). These blocks of bits represent those most likely to lead to a bit error in a transformed bit \( F \). When \( \theta \) is small, e.g., \( \theta = 2 \), two bit flips in the raw fingerprint in a subsequent measurement is enough to push \( |h - l| \) across the \( h - l = 0 \) decision boundary, defined in Equation (6), and result in a \( F \) bit flip. In contrast, when \( \theta \) is large, e.g., \( \theta = 4 \), at least five raw fingerprint bit changes are required to flip the \( F \) bit in a subsequent evaluation. Therefore, we can expect the \( F \) bits selected upon a larger \( \theta \) to be more reliable.

The D-Norm method effectively sacrifices more of the available entropy (\( n \times m \) raw bits are transformed into 1-bit \( F \)) than S-Norm. However, the differential distance measure \( h - l \) is bimodal and, thus, D-Norm is expected to yield a significantly higher number of noise-tolerant \( F \) bits.

4 Formalizing Performance Measures

We now formulate and derive analytical models to: i) provide an upper bound for the unreliability of noise-tolerant fingerprint bits and ii) evaluate the expected number of noise-tolerant fingerprint bits that can be extracted from...
each of the transform methods—the extraction efficiency. We summarize the analytical formulations from our detailed derivations differed to Appendix A, available online for interested readers.

4.1 Reliability

We employ the well-known measure of bit error rate (BER) to quantify the reliability of transformed fingerprint F:

\[ \text{BER}_F = \text{FHD}(F, F'), \]

where F and F' are two fingerprint measurements at distinct times from the same physical memory. The function FHD() is the fractional Hamming distance between the (binary) vectors F and F'. Commonly, F is a reference fingerprint template measured at t = 0, and F' is the reevaluation under a potentially different device operating condition, such as temperature, and, therefore, is subject to noise. A lower BER indicates higher tolerance to noise introduced from the raw fingerprints.

**S-Norm Reliability.** The expected BER of noise-tolerant fingerprints for the S-Norm transformation BER_F is formulated in Equation (9); we defer details of the derivation to Appendix A.2, available in the online supplemental material:

\[ \text{BER}_F = \frac{1}{n} \sum_{i=0}^{n-1} \left( 1 - \text{binocdf}(\theta + i, n \theta \text{BER}_F) \right) \times \text{binopdf}(i, n \theta - \theta, \text{BER}_F). \]

Here, binopdf and binocdf are density and cumulative density functions of a binomial distribution, respectively. The BER_F is a function of \( \theta \), n and the BER of the raw noisy fingerprint bits, BER_F. If we select the worst-case BER_F of any memory chip, Equation (9) provides a worst-case (upper-bound) assessment of BER_F.

**D-Norm Reliability.** A D-Norm transform employs a block of m groups—each group with n raw fingerprint bits—to be transformed into a 1-bit F. The BER_F of D-Norm is expressed in Equation (10); we defer the derivation of the formula to Appendix A.3, available in the online supplemental material:

\[ \text{BER}_F = \frac{1}{n} \sum_{i=0}^{n-1} \left( 1 - \text{binocdf}(\theta + i - 1, n + \theta, \text{BER}_F) \right) \times \text{binopdf}(i, n - \theta, \text{BER}_F). \]

We can observe the reliability of transformed bits from the D-Norm to be related to \( \theta \) (the selection criterion), n and BER_F. Again, Equation (10) provides an upper-bound estimation when the worst-case BER_F is assumed. Notably, the BER_F is independent of the number of groups m within the block.

4.2 Extraction Efficiency

We define extraction efficiency \( \eta \) as the number of obtainable transformed bits, \( F \), subject to a given noise-tolerance \( \theta \), from the total number of available memory bits expressed in KiB.

**S-Norm Extraction Efficiency.** The extraction efficiency of S-Norm can be expressed as below; the detailed derivation is deferred to Appendix A.4, available in the online supplemental material:

\[ \eta_{\text{SNorm}} = \frac{1}{n} \sum_{i=0}^{n-1} \left( 1 - \text{binocdf}(\frac{n}{2} + \theta, n, 0.5) \right) \times \text{binocdf}(\frac{n}{2} - \theta - 1, n, 0.5) \times (1024 \times 8). \]

Here, the term \( 1 - \text{binocdf}(\frac{n}{2} + \theta, n, 0.5) \) expresses the case when the \( \ell_1 \)-Norm of an n-bit f is larger than the selection threshold \( \frac{n}{2} + \theta \), assuming that the probability of each bit being “1”/“0” is 50%. While the term \( \text{binocdf}(\frac{n}{2} - \theta - 1, n, 0.5) \) formulates the alternative case when the \( \ell_1 \)-Norm of a n-bit f is less than or equal to \( \frac{n}{2} - \theta - 1 \). Both cases comprise vectors that satisfy the selection criterion in Equation (3). We can see that the overall extraction efficiency should be the sum of the above two cases divided by n—recall that n raw bits transform into a 1-bit f. The 1024 x 8 term expresses the extraction efficiency as bit/KiB—number of selected reliable bits F out of 1 KiB memory.

**D-Norm Extraction Efficiency.** A D-Norm transform obtains a 1-bit F from a block of m, n-bit raw fingerprint vectors. We define the probability that a given block will meet the selection criterion (\( h - l \geq \theta \)) in Equation (7) as \( P_{\text{select}}^{\text{DNorm}} \) (recall that we refer to the lowest \( \ell_1 \)-Norm as l, and the highest \( \ell_1 \)-Norm as h, out of all m groups within a block). The direct derivation of \( P_{\text{select}}^{\text{DNorm}} \) is non-trivial. Instead, we use a different but equivalent problem and defer the details to Appendix A.5, available in the online supplemental material. We formulate the extraction efficiency of D-Norm as:

\[ \eta_{\text{DNorm}} = \frac{1}{n \times m} \times P_{\text{select}}^{\text{DNorm}} \times (1024 \times 8). \]

Here, the term \( \frac{1}{n \times m} \) expresses n x m raw bits producing a single F bit, while the 1024 x 8 constant facilities express the result in terms of bits/KiB of memory. Given the complexity of formulating \( \eta_{\text{DNorm}} \), the fitness of the formalized expression is further validated through running extensive numerical experiments (defined in Section 5), with the results detailed in Fig. 24 in Appendix, available in the online supplemental material.

4.3 Summary

Our formulation of reliability allows a security practitioner to evaluate, for a given transform, suitable transform parameters (e.g., the number of bits n to employ in a raw fingerprint vector f and noise tolerance parameter \( \theta \)) for extracting new fingerprint F. The extracted F will have an expected worst-case error bound given by BER_F. Then, the \( \eta \) yields the total number of such noise-tolerant bits BER_F that can be extracted from a given memory size.

5 EXPERIMENTAL VALIDATIONS

For comprehensively evaluating NoisFre we used 119 commodity chips consisting of three memory types pervasive in
TABLE 1
Memory Datasets

| Manufacturer Model | Abbr | Tech Node | Memory Type | Memory Size | Quantity | Repeat Times | Operating Range (°C) | Enrolling Condition | Worst Condition | Worst BER |
|--------------------|------|-----------|-------------|-------------|----------|--------------|----------------------|---------------------|-----------------|-----------|
| Nordic (ours) nRF52832 | NORDIC | 55 nm | SRAM | 64 KiB | 12 + 88\(^a\) | 100 | -15-80°C | 25°C | 80°C | 6.09% |
| ISSI [29], [30] IS61WV25616BLL | ISSI | 110 nm | SRAM | 256 KiB | 4 | 30 | 25-80°C | 25°C | 80°C | 8.29% |
| IDT [29], [30] IDT71416S | IDT | 130 nm | SRAM | 512 KiB | 6 | 50 | 25-80°C | 25°C | 80°C | 5.42% |
| Winbond [17] W29N02GV Flash | Flash | 46 nm | FLASH | 69,696 Bytes/256 MiB\(^b\) | 7 | 99 | 0-100,000 P/E Cycles | 0th P/E Cycle | after 100,000th P/E Cycle | 16.26% |
| Microchip (ours) 24LC256 | EEPROM | 350 nm | EEPROM | 2 KiB/32 KiB\(^c\) | 2 | 100 | 14-80°C | 14°C | 80°C | 16.37% |

\(^a\)The NORDIC and EEPROM datasets we collected will be released, remaining public datasets are from https://www.trust-hub.org/data.
\(^b\)Notably these public datasets focus on room temperature and high-temperature evaluations. Other operating corners are incomplete.
\(^c\)The tested Flash memory size in the public dataset is 69,696 bytes, while the total memory size is 256 MiB.
\(^d\)Experimental studies demonstrate that the BER\(\text{F}\) of Flash memory is mainly affected by the programming/erase (P/E) cycles, equivalent to wear-out or aging, but negligibly affected by voltage and temperature [17]. The maximum endurance is 100,000 according to the dataset.

COTS devices, especially in low-end IoT devices and extensive simulation-based experiments with billions of bit generations to overcome the practical hurdle of demonstrating extremely low BER and key failure rates with physical chips. In the following:

- We validate our analytical models for reliability and extraction efficiency.
- We assess the performance of the noise-tolerant fingerprints by evaluating the uniqueness and uniformity of \(\text{F}\).

5.1 Evaluation Approaches

We consider three evaluation approaches described below.

Predictions (Analytical Model). In this evaluation, we use the analytical models formalized in Section 4 to predict extraction efficiency and the BER\(\text{F}\) of the transformed fingerprints.

Simulations (Synthetic Chip Model). To evaluate the reliability of the transformed fingerprint, a massive number of repeated measurements and the management of the data for analysis are required. For example, if we want to validate whether a 128-bit NoisFre enabled key can achieve a failure rate of \(10^{-6}\) as done in Section 6.2, the BER\(\text{F}\) needs to be no more than \(7.81 \times 10^{-9}\). To test this with physical measurements, approximately \(10^8\) repeated measurements are required from the same chip instance. Such a measurement process would take more than nine years and generates roughly 6 Terabyte (TB) of data—this is merely for one 64-KiB chip. Such a massive testing regime is impractical, as detailed in Measurement (Physical chips). Instead, we employed a synthetic memory chip model (detailed in Appendix A.1, available in the online supplemental material). The model follows the physical unclonable function (PUF) response model summarized in [31] and assumes each bit to have a binomial probability of a bit flip across repeated measurements based on employing a worst-case BER measured from a physical SRAM chip (see Table 1) as the binomial probability parameter value for \(p\). Using the synthetic chip model, for instance, 100 million (\(10^8\)) times of simulations can be completed in approximately 53 hours or 2.2 days using a laptop equipped with quad-core Intel Core i7-10510U CPU and 16 Gigabyte (GiB) RAM. The synthetic chip, models bit errors and when applied with the worst-case BER is sufficient for evaluating reliability and extraction efficiency. Therefore, we employ the data from the simulated measurements to determine \(\eta\) and the BER\(\text{F}\).

Measurements (Physical chips). Performing massive testing on physical chips is impractical. For example, obtaining 100 repeated measurements from an nRF52832 physical chip (in the NORDIC dataset) takes four minutes and 45 seconds, and this generates 6.25 Megabyte (MiB) of data (the SRAM memory size of the single chip is 64 KiB). Then, we can estimate that 100 million (\(10^8\)) repeated measurements for a single physical chip under a single operating corner will take 3,298.6 days (or nine years) and generate 5.96 TB of data. Therefore, we confirm extraction efficiency and the transformed fingerprints' BER validated using the synthetic chip model with a limited number of repeated physical chip measurements. However, we dedicate the physical chip measurements across a large batch of 100 chips to evaluate the quality of the transformed bits because properties such as fingerprint uniformity and uniqueness are affected by fabrication variations not incorporated in the synthetic chip model used for simulations. The datasets we used are described in Physical chips–Fingerprint Datasets below.

Physical chips–Fingerprint Datasets. Specifications of: i) three SRAM; ii) one Flash memory; and iii) one EEPROM datasets are summarized in Table 1 and described in detail in Appendix C, available in the online supplemental material. Each dataset is obtained from chips from a different manufacturer. Further, the datasets describe multiple repeated measurements of raw fingerprint bits under each operating condition—see the operating range in Table 1.

We use the NORDIC dataset for extensive validations, considering the fact that it is collected with the broadest
operating range and highest number of repeated measurements (100 repeated measurements). In addition, we use the remaining datasets to corroborate the generality of our approaches. When we evaluate the BER of transformed bits, BER_U, we report the average from repeated evaluations. Notably, the enrolled reference template is only based on the first single measurement.

5.2 Validating Extraction Efficiency and Bit Error Rate

We employ simulations with the synthetic chip model to conduct the necessary massive number of repeated measurements to assess the reliability of transformed fingerprint F. To generate the results, for each parameter combination (i.e., n, θ in S-Norm and m, n, and θ in D-Norm) of a NoisFre transform in Fig. 9 for S-Norm and Fig. 10 for D-Norm, we simulated one million repeated measurements using a synthetic chip with a memory capacity of up to 16 MiB.

5.2.1 Using Simulations

S-Norm. The evaluation results from S-Norm are detailed in Fig. 9 under various n and θ settings. Based on Fig. 9, we can confirm that the formalization of BER_U in Equation (9) provides a conservative estimation of the selected F bits. The results for extraction efficiency are in good agreement with Equation (11) used to predict the number of F bits that can be expected from a given chip.

D-Norm. The validation results of D-Norm are shown in Fig. 10. As expected, the BER_U plotted in Fig. 10 reduces substantially as the θ is increased. Again, we can confirm that the formalized BER_U in Equation (10) is a conservative estimate because it is always shown to be higher than the synthetic chip model results. Further, the extraction efficiency derived in Equation (12) provides an accurate prediction of the number of bits of F that can be expected from a given chip under various D-Norm settings (n, m, and θ).

5.3 Evaluating Uniformity and Uniqueness

In addition to the two crucial performance measures we formulated, reliability and extraction efficiency, we further consider measures that evaluate other qualities of the transformed bits in terms of uniqueness and uniformity (see [32] for a definition of these measures). In the following, our evaluations are based on the measurements obtained from the 100 physical chips in the augmented NORDIC dataset.

Uniqueness Evaluation. Essentially, uniqueness measures how different the fingerprints are between devices. However, the formal definition of uniqueness based on fractional Hamming distance [32] cannot be directly applied for F fingerprints because the transformed bits are generated from one million raw bits.

2. One hundred measurements are due to the impracticality of conducting the necessary number of repeated measurements with physical chips, as detailed in Section 5.1.

3. Fractional Hamming distance (FHD) is a distance measure between two vectors of equal length, defined as the number of positions in the two vectors with different values, normalized by the vector length.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
different physical memory blocks from chip to chip, and the number of such bits obtained could also vary from chip to chip. To account for this, we propose evaluating the uniqueness of S-Norm and D-Norm transformed fingerprints based on the following approach:

1) Given a set of transformation parameters (such as $n$, $m$, and $u$ for D-Norm), we extract all the $f$ bits for each of the $N$ (i.e., 100) devices.

2) Given a pair of devices out of $\binom{N}{2}$, we identify the device which produces the $F$ string with the smaller number of $f$ bits within this pair, and truncate the longer bit string to the same length. Then, we calculate the fractional inter-chip Hamming distance for this pair.

3) We repeat the process in Step 2) for all the $\binom{N}{2}$ pairs to obtain the uniqueness measurement over the 100-chip dataset.

The uniqueness of raw fingerprints, S-Norm fingerprints, and D-Norm fingerprints is illustrated in Fig. 12. The mean uniqueness of the raw fingerprints is 0.48, with a standard derivation of 0.037. For S-Norm, the mean uniqueness achieves the ideal value of 0.50 under all tested settings, and the largest standard derivation is 0.037 under the setting of $(n = 15$ and $\theta = 7$) and $(n = 47$ and $\theta = 9$). The mean uniqueness of the D-Norm fingerprints also exhibits the ideal value of 0.50, except for settings $(n = 32$, $m = 128$, $\theta = 16$), but the mean uniqueness of 0.49 is still nearly the ideal value. In general, as the number of extracted fingerprints in a tested sample decreases, we also observe an increase in standard deviation; this is expected because of the resulting small sample size for statistical analysis.

Uniformity (Bias) Evaluation. Uniformity measures the balance between zeros and ones in a fingerprint vector. The uniformity distribution of raw fingerprints, S-Norm fingerprints, and D-Norm fingerprints is illustrated in Fig. 13. The uniformity of the raw fingerprint is very close to the ideal value of 0.5, with a very small standard deviation. The uniformity of S-Norm and D-Norm methods—across the various parameter settings—is close to the ideal value, albeit with a slight bias toward “1”. Notably, such slight biases are acceptable for key derivation and can be simply compensated by using a few more fingerprint bits when deriving a key [33].

6 DERIVING CRYPTOGRAPHIC KEYS FOR SECURITY FUNCTIONS

We demonstrate the expressive power of our formalization by investigating the derivation of root keys from commodity

---

Fig. 11. S-Norm and D-Norm validation on the datasets including three types of SRAM memories from three different manufacturers (NORDIC, ISSI, and IDT), one type of Flash memory, and one type of EEPROM memory with S-Norm setting $(n = 63)$ and D-Norm settings $(n = 64$, $m = 4)$. Note the number of repeated measurements is finite (see Table 1) and inadequate to demonstrate any errors when the expected BER$_T$ is considerably less than $1/\text{(number of repeated measurements)}$.

Fig. 12. Uniqueness evaluation using physical nRF52832 chips. The plots summarize the mean ($\mu$) and standard deviation ($\sigma$) across a subset of S-Norm parameters, $S(n, \theta)$, and D-Norm parameters, $D(n, m, \theta)$ applied to our large dataset of 100 chips.
memory chips facilitated by our analytical models. The dynamic and direct generation of cryptographic keys from memory fingerprint transformations into noise-tolerant bits is a basis for building security functions because: i) memory biometrics is a true source of randomness and ii) it removes the need for a protected non-volatile memory—keys can be generated on-demand and “forgotten” after usage.

In the following sections, we elaborate on a method for employing the new fingerprint F obtained from the NoisFre transformation to realize a cryptographic key generator (Section 6.1) and evaluate the practical realization of such a key generator (Section 6.2); we defer the security analysis of the key generation process to Section 8.3.

6.1 A Method for Realizing a NoisFre Key Generator

A typical memory fingerprint-based key generation method involves two steps: i) a one-time secure key enrollment on the server-side and ii) on-demand secure key regeneration on the device-side [22], [24], [34], [35]. Positions of transformed bits f should be provisioned during the key enrollment phase and provided during the key regeneration phase. We refer to these positions using a mask. Recall that we have referred to those raw bits that produce a 1-bit f as a block. For the S-Norm, one block has n raw bits, while one block has n x m raw bits in the D-Norm; for both methods, n raw bits form one l1-Norm. In the discussion that follows, we consider key generation under two practical settings:

- Devices with write once read many (WORM) memory for storage of the mask defined to select the memory regions to be used in the NoisFre transform prior to deployment.
- Devices without WORM memory where the mask has to be transmitted, for example, through a wireless communication channel.

6.1.1 On-Server Secure Key Enrollment

First, we describe the one-time secure key enrollment process, depicted in Fig. 14. This process is performed in a secure environment by the server.

Protocol. The one-off on-server secure key enrollment protocol with NoisFre is as follows:

1) Fingerprint memory is a memory region from which the raw device memory fingerprint f is extracted.

2) The raw fingerprint f is processed by the server. The NoisFreTransformSelection process determines a noise-tolerant fingerprint vector F and the corresponding mask based on the parameters n, m, and θ determined by a security practitioner. Notably, a practitioner can employ the analytical expressions derived in Section 4 to determine the appropriate parameter values.

3) Both F and the mask are stored in the server’s secure database (DB, indexed by, for example, the device identification number [id], although not explicitly shown here for simplicity).

4) Optionally, the mask can be stored inside the device’s WORM memory.

6.1.2 Dynamic On-Device Secure Key Generation

Now, we consider the realization of on-device secure key generation with a device memory fingerprint biometric. We illustrate the key generation method in Fig. 15.

Protocol. The dynamic on-device secure key generation protocol with NoisFre is as follows:

1) If the device implements WORM memory to store the mask, as in Fig. 15a, the server fetches device-specific information from the DB, such as the enrolled F.

2) If the device does not implement WORM memory, the server fetches device-specific information from the DB, such as the enrolled F and mask, as in Fig. 15b. The mask is transferred from the server to the device over a (non-secure) wireless communication channel. To ensure the integrity of the mask, a message authentication code (MAC) tag is computed by the server as tag ← MACF(mask) and appended to the mask.

3) The device dynamically generates a new noisy raw fingerprint f′ from the fingerprint memory.

4) The device computes F ← NoisFre.Transform(f′, mask), where NoisFre.Transform() is a function defined by, for example, the D-Norm transform in Equation (6).

5) If the device does not implement a WORM memory, then the mask is sent by the server, as shown in Fig. 15b; the device computes tag′ ← MACF(mask). To check the integrity of the mask, the tag is compared to the tag supplied by the server. If the two
values match, output the NoisFre fingerprint F; otherwise, output ⊥.

6) Now both the server and the device share the same highly reliable F to be used as a shared secret in a security function.

6.2 Evaluations
We begin our systematic evaluation of cryptographic key generation with the following question and employ the formal models and the physical chip measurements for our evaluations.

What is the reliability of a k-bit NoisFre fingerprint F?

Transformed fingerprint F can be directly utilized as a cryptographic key because they are invariant to a desirably high number of noise-induced bit error patterns—these F bits exhibit a high noise tolerance. The overall failure rate $P_{\text{fail}}$ of a k-bit noise-tolerant key F can be expressed as:

$$P_{\text{fail}} = 1 - (1 - \text{BER}_F)^k.$$  \hfill (13)

Recall that the formalized BER$_F$ in Section 4.1 is conservative. Therefore, the $P_{\text{fail}}$ in Equation (13) will also yield a conservative estimation. We expect a key failure rate in practice to be lower than our prediction here. This hypothesis is validated with an extensive simulation-based on a large simulated chip with up to one billion repeated noise-tolerant key bit extraction, as illustrated in Fig. 16.

Next, considering a practitioner’s desire for a $P_{\text{fail}} < 10^{-6}$ performance target for typical industrial applications,

4. Notably, there is nothing fundamentally preventing us from aiming for a lower key failure rate. We can see from Fig. 16 that a larger $\theta$ will achieve a lower failure probability.

Fig. 16. Validation of Equation (13). The simulated chips are based on the worst-case BER$_F$ = 6.09% from the NORDIC chip set. The parameters selected are $n = 32$, $m = 16$ and varied D-norm parameter $\theta$ from 1, ..., 17. We conducted 10 million re-evaluations of a 128-bit noise-tolerant fingerprint for each value of $\theta = 1, ..., 16$ and one billion evaluations for $\theta = 17$. Our results corroborates Equations (10) and (13) as an upper bound on the failure rate of a NoisFre fingerprint employed as a cryptographic key. An even lower $P_{\text{fail}}$ is achievable if a larger $\theta$ is used. We halted our investigation at $\theta = 17$ as it answers the question we investigated.

as highlighted in [36] and recent studies [26], [34], [37], [38], [39], [40], we investigate the following question.

What is the most efficient transformation method presenting the highest extraction efficiency while ensuring sufficient reliability for F to be direct use as a 128-bit cryptographic key with a failure rate lower than $10^{-6}$ under worst-case raw fingerprint BER?

We employ NORDIC SRAM-based synthetic data to facilitate the massive number of evaluations necessary to address the question. The evaluation process is described below:

1) We determine the BER$_F$ corresponding to a 128-bit key with a failure rate of $10^{-6}$ using Equation (13). The resulting BER$_F$ is $7.81 \times 10^{-9}$.

2) For each $n \in \{1, 2, 3, \ldots, 256\}$, evaluate the minimum $\theta$ for the required BER$_F$ (Equation (9) for S-Norm and Equation (10) for D-Norm) to ensure BER$_F < 7.81 \times 10^{-9}$. In these equations, we employ the mean of the worst-case BER$_F$ = 6.09% of NORDIC dataset to compute an upper bound for BER$_F$.

3) For S-Norm, the extraction efficiency $\eta$ is calculated with Equation (11) using the $n$ and $\theta$ determined in the previous step.

4) D-Norm requires us to further determine the $m$ value that can provide the highest $\eta$. As observed in Fig. 24 in the Appendix, available in the online supplemental material, $\eta$ changes smoothly with respect to $m$. To reduce the search-time overhead, we applied a grid-based search technique: i) evenly select $j$ sample points from the entire domain of $m \in \{1, 256\}$; ii) calculate the $\eta$ for each $m = 1, 2, 3, \ldots, j$; iii) find the $m$ values corresponding to the highest and the second highest $\eta$; iv) refine the search domain to be between the two points found in step iii); and v) repeat from i) to iv) to locate the $m$ that gives the highest $\eta$.

The results from our investigation are depicted in Fig. 17; here, we plot the occurrences of extraction efficiency as a function of $\eta$ from all the combinations of S-Norm parameters ($n$ and $\theta$) and D-Norm parameters ($n$, $\theta$ and $m$). We can conclude that the D-Norm always affords significantly higher extraction efficiencies conditioned on the 128-bit
Fig. 17. Extraction efficiency comparison between D-Norm and S-Norm. For S-Norm and D-Norm extractions, we have evaluated 16,384 and 8,388,480 parameters combinations, respectively. The $\theta$ of D-Norm may take any value in $[1, n]$, while in S-Norm, the $\theta$ is restricted within $[1, n/2]$. Meanwhile, D-Norm extraction employs the additional parameter, $m$. Therefore, the possible combinations of parameters for D-Norm are magnitudes larger than that of S-Norm.

Given: i) different sizes of memories embedded within various COTS electronics and ii) BER$_F$ characteristics of noisy fingerprints from different memory technologies:

\begin{align*}
\text{What is the lowest key failure rate } P_{\text{fail}} \text{ achievable for a 128-bit key } F \text{ from each memory technology and manufacturer considered in our study?}
\end{align*}

3) We use BER$_F$ substituted into Equation (13) to determine the best $P_{\text{fail}}$ of the selected and transformed $F$ with at least 128 bits.

Results are summarized in Table 2. Taking the expected BER$_F$ across the smallest SRAM dataset, the lowest $P_{\text{fail}}$ expected from a chip with SRAM capacity of 64 KiB is in the magnitude of $10^{-5}$. Notably, $P_{\text{fail}}$ reported in Table 2 is conservatively estimated from formulations. In practice, $P_{\text{fail}}$ is expected to be much better. Importantly, with more abundant and freely available on-chip SRAM, represented in the IDT dataset, a remarkably low key failure rate of $5.29 \times 10^{-9}$ is achievable.

As expected, the higher worst-case BER$_F$ of the EEPROM and Flash datasets implies that the techniques in NoisFre are not able to select a 128-bit $F$ with a satisfactory $P_{\text{fail}}$. However, the Flash memory tested benefits from a high memory capacity (256 MiB compared to 32 KiB for EEPROM) and we can achieve orders-of-magnitude better $P_{\text{fail}}$ than EEPROM.

In summary, for SRAM—the most prevalent memory type in IoT devices—a 128-bit key with a key failure rate less than $10^{-6}$ can be efficiently obtained given an adequate SRAM memory capacity. However, for memory types exhibiting severely high BER$_F$, for example, EEPROM and Flash, the method itself is insufficient to gain a satisfactory $P_{\text{fail}}$. Although, NoisFre does significantly reduce the key failure rate given the higher capacity of Flash memory for selecting bits. Notably, with such high BER$_F$ memory characteristics, even the state-of-the-art, efficient method of RFE-based key generators are unlikely to deliver a computationally tractable solution on resource limited devices. We discuss this limitation further in Section 8.4.

7 SECURITY FUNCTION IMPLEMENTATION FOR COMPARISON

Here, we describe a case study implementing a NoisFre-based key generator followed by performance and implementation overhead comparisons against the lightweight, state-of-the-art (R)FE-based method.

7.1 An Overview

The entities, a Verifier and a Prover, involved in this case study are illustrated in Fig. 18a. The Verifier consists of a server and a wireless network gateway (smartphone). The Prover refers to a wireless sensor node (Bluetooth sensor).

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
Dataset (Type) & worst case BER$_F$ (mean) & Mem. size & $n$ & $m$ & $\theta$ & $P_{\text{fail}}$ Equation (13) \\
\hline
NORDIC (SRAM) & 6.09% & 64 KiB & 29 & 65 & 13 & $4.04 \times 10^{-5}$ \\
ISSI (SRAM) & 8.29% & 256 KiB & 50 & 128 & 19 & $3.56 \times 10^{-5}$ \\
IDT (SRAM) & 5.42% & 512 KiB & 83 & 128 & 25 & $5.29 \times 10^{-9}$ \\
Winbond (Flash) & 16.26% & 256 MiB$^1$ & 120 & 128 & 41 & $2.52 \times 10^{-4}$ \\
Microchip (EEPROM) & 16.37% & 32 KiB$^1$ & 14 & 61 & 9 & $4.01 \times 10^{-1}$ \\
\hline
\end{tabular}
\caption{The Lowest Key Failure Rate $P_{\text{fail}}$ Achievable for Obtaining a 128-bit Key $F$ for Each Investigated Memory Dataset Using D-Norm}
\end{table}

$^1$Recall that the tested size of Flash and EEPROM memory are 69 KiB and 2 KiB. When calculating the number of selected noise-tolerant bits, the memory sizes are scaled up by assuming the entire 256 MiB Flash memory and 32 KiB EEPROM memory are available for fingerprinting. Here Mem. size is the abbreviation for Memory size. Worst case BER$_F$ is the mean of the value calculated across the chips in a given dataset. Notably, as described in Section 6.2 and illustrated in Fig. 16, Equation (13) provides a conservative upper bound, the actual key failure rates will be much lower in practice.
In this setup, the server functions as a coordinator, holds the enrolled Prover’s information in the database, and issues commands to instruct the Prover to perform remote attestation. The gateway bridges the communication between the server and the Prover. The traffic between the server and the gateway is assumed to be secure by applying standard security protection mechanisms. The Prover, communicating wirelessly, is deployed in an (insecure) environment. Details of the corresponding attestation protocol are provided in Fig. 19. Our case study aims to:

- Implement a lightweight remote attestation routine suitable for a Prover with a constrained resource by following [41].
- Experimentally demonstrate that SRAM fingerprints can be accessed on-demand and at run-time by exploiting the low-cost micro controller unit (MCU)’s memory power control features—SRAM regions are arranged in blocks can be individually powered on or off.

**Remote Attestation Mechanism.** An overview of the remote attestation mechanism based on a NoisFre key generator is illustrated in: i) Fig. 19, where we assume the Prover has no WORM memory available for storing a mask and that it has to be transmitted over the wireless communication channel (the worst-case setting in terms of implementation overhead) and ii) Fig. 20, where we assume the Prover has available WORM memory. We assume the Prover has already undergone the enrollment phase we described in Section 6.1.1. The enrollment is conducted by the Verifier in the current setting.

A remote attestation can be requested anytime. First, the Verifier scans for visible Provers by sending a “hello” message. Once there is a Prover in the horizon responding with its unique identifier id, the Verifier fetches the Prover’s information (e.g., F and mask) from the secure database DB by using the id as an index. Second, if the received id matches one of that stored in the Verifier’s DB, the Verifier instructs the Prover to perform attestation—by sending the mask and MAC tag for Provers with no WORM memory, as in Fig. 19. In this context, the Prover performs a power cycling of memory banks solely corresponding to the fingerprint zone and dynamically generates F, following the steps described in Section 6.1.2. After confirming a ready acknowledgment from the Prover, the Verifier randomly generates a challenge (a nonce) and sends it to the Prover along with the address addr and the length of the target application program (App) code bin. The Prover’s response resp is generated using MAC computed with the noise-tolerant fingerprint F. The Verifier compares the received response resp with a locally calculated reference response resp’. The remote attestation is accepted if resp and resp’ match and rejected otherwise.

If the Prover device implements WORM memory for storing a mask, the protocol can be simplified as shown in Fig. 20. (a) System overview. (b) Experiment setup: Verifier consists of a laptop as the cloud server and a smartphone as the gateway; device is a commercial widely used nRF52832 Bluetooth-LE sensor. See the demo video for more details https://youtu.be/O5NWZw-swpp.

![Fig. 18.](image)

![Fig. 19.](image)

![Fig. 20.](image)
Fig. 20; in our end-to-end demo implementation, we consider this simpler case, and describe the implementation details in Fig. 26 in Appendix B, available in the online supplemental material.

### 7.2 Overhead Comparisons

#### Implementation Details

We provide an overview of the system implemented in Fig. 18b and defer details to Appendix B, available in the online supplemental material. Further, we refer the reader to our open-source code release\(^6\) for detailed descriptions of our implementation, including dynamic and run-time key generation from SRAM fingerprints. A video demonstration of the end-to-end implementation is available at https://youtu.be/O5NWZw-swppw.

We implemented a D-Norm-based key generator on an nRF52832 chip with the smallest on-chip SRAM capacity and BER\(_d\) of 4.93% tested under \(-15\) to \(80^\circ\)C operating range. We used \(n=32, m=48, \theta = 13\) for D-Norm parameters determined by Equations (10), (12), and (13) to be able to extract a 128-bit NoisFre key capable of a key failure rate below \(9.15 \times 10^{-6}\).

For comparisons, we implemented the (R)FE-based key generators summarized in Table 3 to achieve a key failure rate to closely match \(10^{-6}\). As discussed in Section 2.2, in an FE, the device executes the computationally-heavy decoding function, while in an RFE, the device executes the more lightweight encoding function. In our end-to-end demonstration, to achieve a comparable failure rate to that of the D-Norm-based NoisFre key generator, the (R)FE implementation needs 13 parallel blocks of \((n = 63, k = 10, t = 13)\) BCH code\(^7\) to provide a similar key failure rate.

**Implementation Overhead:** The implementation overhead evaluates the usage of two system resources: random-access memory (for run-time data) and clock cycles (for code executions). Overall, in terms of obtaining a 128-bit reliable key with a key failure rate of \(9.15 \times 10^{-6}\), the implementation of the D-Norm-based NoisFre method with parameters \((n = 32, m = 48, \theta = 13)\) takes 51,044 clock cycles.\(^8\) If the mask is provided by the server and transmitted over a wireless channel, an additional 45,622 clock cycles are required for mask integrity checks.

In contrast, the FE-based and the lightweight state-of-the-art RFE-based method introduces significantly higher overheads to achieve a 128-bit reliable key with a slightly inferior key failure rate of \(2.45 \times 10^{-5}\). Specifically, as evaluated and shown in Fig. 21, the on-device FE decoding and RFE encoding functions consume 285,311 and 109,850 clock cycles, respectively. Both methods need an additional 60,755 clock cycles for helper data integrity checks. In comparison with the state-of-the-art FE and RFE, for meeting a comparable key failure rate, NoisFre reduces clock overhead by 72% and 43%, respectively, if the mask or helper data is transmitted over the wireless channel requiring helper data integrity checks. However, if the mask or helper data for all of the method are stored on a device’s WORM memory, clock cycles required in comparison to NoisFre reduces by 82% (compared to FE) and 54% (compared to RFE).

It is worth emphasizing that we have compared NoisFre with an RFE capable of deriving a key with a failure rate of \(P_{\text{fail}} \approx 10^{-6}\). However, as we show in Table 2, if an IDT chip is used in the implementation, we can obtain a key with a significantly lower key failure rate by exploiting the free, abundant memory; now, \(P_{\text{fail}}\) can be \(\approx 5 \times 10^{-9}\). Attempting to achieve such a small \(P_{\text{fail}}\) using an (R)FE will lead to significantly higher overheads. The (R)FE-based key provisioning method introduces increasing execution overheads if a lower key failure rate is desired as illustrated in Table 3. For example, if an IDT SRAM chip is used as the fingerprinting barometric source instead of the NORDIC chips’ internal SRAM, a 128-bit key with failure rate of \(1.69 \times 10^{-9}\) requires 188,487 (3.69 times larger) clock cycles with the RFE-based method or 967,599 (18.95 times larger) clock cycles with the FE-based method, compared with 51,044 clock cycles for our NoisFre-based method. Hence, in contrast to (R)FE

---

### Table 3: Implementation Overhead of (R)FE Employing BCH Codes

| Fingerprint source (BER\(_d\)) | BCH(n,k,t) Block number | Key failure rate | Key size (bits) | Helper data size (bits) | Fuzzy Extractor decoding | Reverse Fuzzy Extractor encoding | Helper data integrity check |
|---------------------------------|-------------------------|-----------------|-----------------|------------------------|--------------------------|-------------------------------|-----------------------------|
| NORDIC (4.93%) (63,10,13) | 13 | 2.45 \(\times\) \(10^{-5}\) | 130 | 689 | 285,311 | 109,850 | 60,755 |
| IDT (5.42%) (127,15,27) | 9 | 1.69 \(\times\) \(10^{-9}\) | 135 | 1008 | 967,599 | 188,487 | 84,013 |

---

\(^{6}\) See https://github.com/AdelaideAuto-IDLab/NoisFre

\(^{7}\) BCH code is a class of cyclic error-correcting codes, named after its inventors Bose, Chaudhuri, and Hocquenghem, constructed using polynomials over Galois field.

\(^{8}\) This was tested with nRF52832 SoC, via J-link EDU V10.1 debugger, with nRF5 SDK Ver. 15.3.0, Keil uvision 5.25.2.0 and ARM CC compiler Ver. 5.06 Update 6. Optimization setting = -O3.

---

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
methods, the on-device computational overhead of the proposed NoisFre key generator remains constant, regardless of the desired key reliability and only depends on the size of the key to be derived.

8 DISCUSSION

8.1 Generality of NoisFre

Although our work focused predominantly on SRAM, considering its ubiquity in low-end IoT devices and the simplistic nature of fingerprint extraction, the NoisFre fingerprinting methods presented are applicable for other memories, including Flash and EEPROM memories validated in our study. In principle, it can be applied to other hardware fingerprinting methods [42], [43], given an abundant raw digital fingerprint bit space.

8.2 Provisioning Fingerprints at Run-Time

Flash and EEPROM memory fingerprints can be accessed during run-time. However, for SRAM fingerprinting, the most common method is to utilize its initialization pattern at power-up as a fingerprint, although there are other means [44]; for example using data retention voltage [44] or intentionally putting SRAM cells under a meta-stable state. These methods usually require customized peripheral circuitry, which tends to be unavailable in COTS devices. Thus, SRAM fingerprinting generally requires power cycling to read the start-up values. As a matter of fact, some low-end microcontrollers allow direct control over the powering of individual SRAM banks [45] (e.g., the low-end nRF52832 studied in this work). Consequently, by leveraging such a feature, SRAM fingerprint-based root keys can be requested during run-time.

8.3 Security Analysis

We have looked at the problem of achieving a pragmatic, on-device key derivation method using noisy memory fingerprints. NoisFre fundamentally obviates the need for computationally intensive on-device ECC logic for the task. It is thus immune to HDM attacks [25], [26] that strategically tamper the helper data associated with the ECC to weaken or compromise the key extracted using the state-of-the-art (R)FE methods. The vulnerability is induced by the usage of ECCs (see Section 2.2). Various ECCs are examined and shown to be vulnerable to HDM attacks [26]. A generic countermeasure against HDM attacks appears to be an open challenge. The NoisFre scheme has sought to remove the necessity for helper data associated with key generation in an RFE and, thus, avoid the HDM attacks that exploit helper data. In the following, we consider the security of our proposed key derivation method in the context of prior methods based on the state-of-the-art (R)FE methods.

8.3.1 Threat Model

Memory fingerprint-based key provisioning studies rarely explicitly define a threat model [33], [39], [46] and operate under the assumption that the key material (i.e., memory fingerprint) cannot be directly accessed. However, studies focusing on incorporating key derivation methods to provide a security function, such as authentication or remote attestation [47], [48], [49], [50], follow a threat model. Therefore, we follow the threat model reasoned therein, along with the assumption that the key material cannot be directly accessed.

Specifically, we consider that an adversary cannot access the raw fingerprint and temporary data stored in RAM or internal chip registers during key derivation. The attacker can tamper with public information used to assist the key derivation. Notably, in prior work, such information would be the ECC associated helper data in a (R)FE-based reliable key derivation method [26]—in our key derivation approach, we assume the mask is public information. The mask is sent to a device over a communication channel together with a method for assessing the integrity of the mask or is stored in WORM.

8.3.2 Mask Manipulation Attack

In use cases where the mask is sent to a device over a communication channel, it is possible for an attacker to manipulate the mask. Therefore, we consider mask manipulation attacks.

In the context of a NoisFre-based key generator, a MAC tag is produced over the mask using the derived F to ensure the integrity of the mask—more specifically, tag = MACF(mask), with F being the reliable secret key, as illustrated in Fig. 15b. The MAC tag and MAC can be publicly stored off-chip and/or stored on-chip. Subsequently, the MAC tag can be regenerated to validate the integrity of a mask stored on-device or transmitted to the device prior to the use of the key derived on-demand, as illustrated in Fig. 15b. Now, the probability of making a modification without being detected is $\frac{1}{k}$ with $k$ the length of the derived key. It will be $\frac{1}{2^k}$ for a typical 128-bit key.

Although we adopted a simple mechanism in this study to ensure mask integrity, other mechanisms have been proposed to ensure the integrity of helper data in the context of state-of-the-art (R)FE methods [25]. Thus, we can also employ these existing methods to ensure the integrity of the mask for NoisFre key derivation methods.

8.3.3 Brute-force Attack

For completeness, we also assess the attack complexity of a brute-force attack on a NoisFre-based key derivation method. The attacker may utilize a brute-force attack to determine the derived key. However, this is extremely challenging when the key is appropriately sized. For a brute-force attack, the probability of finding the correct derived key is $\frac{1}{2^k}$, which is computationally infeasible given a reliable key with a typical length of $k = 128$ bits.

8.3.4 Aging Attack

The data stored in a SRAM cell can gradually affect its start-up state. This is called data-dependent aging [51]. Given that the key derivation is based on a physical primitive, we also consider aging attacks that may attempt to exploit the small changes in behavior of memory cells that occur as a result of aging the underlying electronic components.

In use cases where a write access protected (e.g., using a memory protection unit [MPU]) memory cannot be
allocated for generating fingerprints and where the memory space used for fingerprints must be shared with user application code, an attacker may utilize malicious code on the device to continuously write specific memory patterns to the SRAM used for device fingerprinting. Such an attempt can accelerate aging and can potentially degrade the reliability of a NoisFre key generation method.

In use cases where a dedicated memory cannot be allocated for generating fingerprints, several simple mitigation strategies already exist. First, the aging effect is data dependent. The user can employ an anti-aging method, such as writing reverse data patterns to mitigate the aging effect validated as an efficient approach to counter aging [51]. Second, the SRAM unreliability induced by aging, even over six years, is small—only 2% [51]. Hence, a simple anti-aging method for NoisFre is to allow the server to intentionally assume a higher worst-case BER during the enrollment phase to count for or tolerate the aging effect by trading off a slight increase in SRAM volume required to retain the same NoisFre key reliability. If the available memory volume is constrained, a further low-cost anti-aging measure is for the server to adopt the trial-and-error method reported in [52] to recover the least reliable transformed $f$ bits, because the server can ascertain the bit-specific reliability of each $f$ bit. Notably, in this approach, all the computation overhead is offloaded to the server without imparting any overhead to the device.

8.4 Limitations and Future Work
Our study is not without limitations. As shown in Fig. 17, the highest extraction efficiency (i.e., the number of fingerprint bits with a BER$_F < 7.81 \times 10^{-9}$ that can be extracted from a unit-sized memory block) that NoisFre can achieve is 0.62 bits per KiB. Hence, extracting a usable (e.g., 128-bit) secure key from a highly resource-limited device with a mere 2 KiB memory space (i.e., the SRAM size of the passively powered computational radio frequency identification (CRFID) device studied in [35]) with NoisFre is not immediately possible.

The investigation of potential methods for improving the performance, in particular enhancing the BER$_F$ and/or extraction efficiency $\eta$, is left out of scope for our current study focused on developing NoisFre, formalizations, and extensive evaluations. As a potential direction for future work, it will be interesting to consider approaches, for example, to extract more bits from a given memory. Although our formulation for reliability is applicable for such a method, the analytical formulation of extraction efficiency for such new methods will likely require considerable effort to develop. Importantly, the complexity of the task will provide an interesting direction for future work. Therefore, we leave the investigation of potential means for improving the NoisFre performance, in particular enhancing the BER$_F$ and/or extraction efficiency $\eta$, as potential directions for future research.

9 RELATED WORK
Besides memories, various on-board sensors, such as cameras, accelerometers, gyroscopes, magnetometers, and other components, such as CPU magnetic radiations, are utilized to provide fingerprints [3], [4], [5], [6], [7], [8], [9], [10], [11]. Other recent works also explore commodity scanners to fingerprint 3D objects to track them [2] and exploit the package variations as fingerprints for anti-counterfeiting [1]. However, to obtain hardware fingerprints, those fingerprint extractions are relatively complicated in comparison with memory, especially SRAM, enabled fingerprints.

Notably, hardware fingerprinting is closely related to the notion of PUFs [34], [53], [54], [55], [56]. Commodity memory fingerprinting, such as SRAM PUF and Flash PUF, is not new. However, mounting them on low-end IoT devices to derive a usable key for security functions relies on post-error correction to reconcile bits errors, which is cumbersome in terms of both overhead and security in practice. Our simple yet efficient NoisFre memory-fingerprinting approach addresses this gap.

We exploit the idea of a differential measurement in the formulation of the D-Norm method based on the base distance ($\ell_1$-Norm) to improve the extraction efficiency (number of $f$ bits with a desired noise tolerance) from a given memory. Interestingly, in PUF studies, formulating methods to exploit a differential gap or comparison has been utilized by extrinsic PUFs—implemented with additional hardware—such as ring oscillator PUF (RO-PUF) [55], [56], [57], [58] and arbiter PUF (APUF) [59] to obtain responses with improved reliability. The concept has subsequently been applied in [55], [56] to enhance reliability and address aging of electronic components in RO-PUFs facilitated by the ease with which RO frequency differences are already measured and can be directly used. In our intrinsic memory studies, we exploit a base distance, $\ell_1$-Norm, to generate a differential measurement to build the D-Norm transform for memory PUFs intrinsic to COTS devices. As discussed in Section 3.3, we recognized that the differential formulation can yield significantly more reliable bits compared to S-Norm employing (simply, the $\ell_1$-Norm base distance). Thus, in our work, we combine these two mathematical concepts (base distance with a differential measure) together to extract more noise-tolerant bits from a memory PUF—a method that can be used with intrinsic memories widely exist in COTS devices.

10 CONCLUSION
By exploiting ubiquitously embedded memory within commodity computing devices, the proposed NoisFre approach constructively extracts transformed memory fingerprints that were embodied with a high tolerance to noise affecting the generation of fingerprints. With a simple, single, one-off fingerprint enrollment measurement, NoisFre is able to judiciously identify highly reliable transformed fingerprints serving as hardware root key or root of trust to directly support various security functions for a wide range of COTS electronic devices. Besides formalization of two specific S-Norm and D-Norm fingerprint transformation methods and extensive empirical validations on SRAM, Flash, and EEPROM memories using 119 physical chips in total, we have conducted a case study with an end-to-end implementation of a remote attestation security service employing NoisFre fingerprints to significantly reduce the overhead in comparison with the state-of-the-art RFE method for...
constructing reliable fingerprints for a key generator. We also demonstrate how SRAM fingerprints can be generated at run-time by utilizing individual memory-bank power control features on MCUs. Overall, NoisFre is a simple but practical method, especially for existing low-end commodity electronic devices.

References

[1] S. N. Dhanuskodi, X. Li, and D. Holcomb, “COUNTERFOIL: Verifying provenance of integrated circuits using intrinsic package fingerprints of photolithography cameras,” in Proc. USENIX Secur. Symp., 2020, pp. 1255–1272.

[2] Z. Li, A. S. Rathore, C. Song, S. Wei, Y. Wang, and W. Xu, “PrinTracker: Fingerprinting 3D printers using commodity cameras,” in Proc. ACM Conf. Comput. Commun. Secur., 2018, pp. 1306–1323.

[3] Z. Ba, S. Piao, X. Fu, D. Koutsounikolas, A. Mohaisen, and K. Ren, “Combing: Combining power authentication with built-in-camera,” in Proc. Netw. Distrib. Syst. Secur. Symp., 2018, pp. 1–15.

[4] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, “Mobile device identification via sensor fingerprinting,” 2014, arXiv: 1408.1476.

[5] J. Zhang, A. R. Beresford, and I. Sheren, “SensorID: Sensor calibration fingerprinting for smartphones,” in Proc. IEEE Symp. Secur. Privacy, 2019, pp. 350–370.

[6] Y. Son, J. Noh, J. Choi, and Y. Kim, “Gyrosfinger: Fingerprinting drones for location tracking based on the outputs of MEMS gyroscopes,” ACM Trans. Privacy Secur., vol. 21, no. 2, pp. 1–25, 2018.

[7] Y. Kim and Y. Lee, “Cam PUF: Physically unclonable function based on CMOS image sensor fixed pattern noise,” in Proc. ACM Ann. Des. Automat. Conf., 2018, pp. 1–6.

[8] O. Willers, C. G. Sebald, and H. Seidel, “MEMS gyroscopes as physical unclonable functions,” in Proc. ACM Conf. Comput. Commun. Secur., 2016, pp. 591–602.

[9] Z. Ba, Z. Qin, X. Fu, and K. Ren, “CIM: Camera in motion for smartphone authentication,” IEEE Trans. Inf. Forensics Secur., vol. 14, no. 11, pp. 2987–3002, Nov. 2019.

[10] E. Quiring, M. Kirchner, and K. Rieck, “On the security and applicability of fragile camera fingerprints,” in Proc. Euro. Symp. Res. Comput. Secur., 2019, pp. 1–18.

[11] Y. Cheng, X. Ji, J. Zhang, W. Xu, and Y.-C. Chen, “DeMiCPU: Device fingerprinting with magnetic signals radiated by CPU,” in Proc. ACM Conf. Comput. Commun. Secur., 2019, pp. 1149–1170.

[12] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, “FPGA intrinsic PUFs and their use for IP protection,” in Proc. Cryptographic Hardware Embedded Syst., 2007, pp. 63–80.

[13] H. Daniele, W. P. Burleson, and K. Fu, “Power-up SRAM state as an identifying fingerprint and source of true random numbers,” IEEE Trans. Comput., vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[14] A. Van Herreweghe, “Lightweight PUF-based key and random number generation,” Ph.D. dissertation, Fac. Eng. Sci., Arenberg Doctoral School, Leuven, Belgium, 2015.

[15] A. Schaller et al., “Decay-based DRAM PUFs in commodity devices,” IEEE Trans. Dependable Secure Comput., vol. 16, no. 3, pp. 462–475, May/Jun. 2019.

[16] Y. Wang, W.-K. Yu, S. Wu, G. Malysa, G. E. Suh, and E. C. Kan, “Flash memory for ubiquitous hardware security functions: True random number generation and device fingerprints,” in Proc. IEEE Symp. Secur. Privacy, 2012, pp. 33–47.

[17] Z. Guo, X. Xu, M. M. Tehranipoor, and D. Forte, “FFD: A framework for fake flash detection,” in Proc. ACM Ann. Des. Automat. Conf., 2017, pp. 1–6.

[18] Statista, “Internet of things (IoT) connected devices installed base worldwide from 2015 to 2025,” 2016. Accessed: May 07, 2020. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[19] X. Boyen, “Reusable cryptographic fuzzy extractors,” in Proc. Cryptography Hardware Embedded Secur., 2004, pp. 82–91.

[20] Y. Dodis, L. Reyzin, and A. Smith, “Fuzzy extractors: How to generate strong keys from biometrics and other noisy data,” in Proc. Springer Adv. Cryptol.-Eurocrypt, 2004, pp. 523–540.

[21] C. Bosch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, “Efficient helper data key extractor on FPGAs,” in Proc. Cryptography Hardware Embedded Syst., 2008, pp. 181–197.
[45] N. Semiconductor, “nRF52832 product specification v1.4,” Oct. 2017. Accessed: Nov. 30, 2022. [Online]. Available: https://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.4.pdf
[46] L. Kusters and F. M. Willems, “Secret-key capacity regions for multiple enrollments with an SRAM-PUF,” IEEE Trans. Inf. Forensics Secur., vol. 14, no. 9, pp. 2276–2287, Sep. 2019.
[47] A. Aysu, E. Gulcan, D. Moriyama, P. Schaumont, and M. Yung, “End-to-end design of a PUF-based privacy preserving authentication protocol,” in Proc. Springer Int. Workshop Cryptography Hardware Embedded Syst., 2015, pp. 556–576.
[48] W. Feng, Y. Qin, S. Zhao, Z. Liu, X. Chu, and D. Feng, “Secure code updates for smart embedded devices based on PUFs,” in Proc. Int. Conf. Cryptol. Netw. Secur., 2017, pp. 325–346.
[49] F. Kohnh, S. Kaftan, and V. Kostakos, “PUFs in the era of AI: A survey and a look forward,” in Proc. IEEE Int. Symp. Dependable Secure Comput., 2018, pp. 333–348, Third Quarter 2016.
[50] T. Rahman, D. Forte, J. Fahrny, and M. Tehranipoor, “ARO-PUF: An aging-resistant ring oscillator PUF design,” in Proc. Conf. Des. Automat. Test Eur., 2014, pp. 1–6.
[51] G. E. Suh and S. Devadas, “Physical unclonable functions for device authentication and secret key generation,” in Proc. ACM Conf. Des. Automat. Conf., 2007, pp. 9–14.
[52] R. Hesselbarth, F. Wilde, C. Gu, and N. Hanley, “Large scale RO PUF analysis over slice type, evaluation time and temperature on 28nm Xilinx FPGAs,” in Proc. IEEE Int. Symp. Hardware Oriented Secure Trust. 2018, pp. 126–133.
[53] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, “Silicon physical random functions,” in Proc. ACM Conf. Comput. Commun. Secur., 2002, pp. 148–160.

Yansong Gao received the PhD degree from the University of Adelaide, Australia, in 2017. He was a postdoc research fellow with Data61, CSIRO, Australia. He is now with the Nanjing University of Science and Technology, China, as an associate professor with the School of Computer Science and Engineering. His current research interests include hardware security, system security, and AI security and privacy.

Yang Su (Student Member, IEEE) received the BEng degree with first class Honours in electrical and electronic engineering from the University of Adelaide, Australia, in 2015. He is currently working toward the PhD degree with the University of Adelaide. He worked as a research associate with the University of Adelaide from 2015-2016. His research interests include hardware security, physical cryptography, embedded systems and system security.

Surya Nepal received the PhD degree from RMIT University, Australia. He joined CSIRO in 2000. He is currently a principal research scientist with Data61, CSIRO. He is the leader of the Distributed System Security Group, Data61, CSIRO. His main research interests include the development and implementation of technologies in the area of distributed systems including Web services, cloud computing, IoT, and Big Data, with a specific focus on security, privacy, and trust. He is currently one of the associate editors of the IEEE Transactions on Services Computing and IEEE Transactions on Dependable and Secure Computing.

Damith C. Ranasinghe received the PhD degree in electrical and electronic engineering from the University of Adelaide, Australia. In the past, he was a visiting scholar with the Massachusetts Institute of Technology, Cambridge, MA, and a post-doctoral research fellow with the University of Cambridge, Cambridge, U.K. Currently, he is an associate professor with the University of Adelaide. His research interests include embedded systems, system security, autonomous systems, and deep learning.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.