micromachines

MDPI

Article

STT-DPSA: Digital PUF-Based Secure Authentication Using STT-MRAM for the Internet of Things

Wei-Chen Chien [1], Yu-Chian Chang [2], Yao-Tung Tsou [3,*], Sy-Yen Kuo [2] and Ching-Ray Chang [4]

[1] Graduate Institute of Applied Physics, National Taiwan University, Taipei 106, Taiwan; aoowweenn@gmail.com
[2] Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan; r05943169@ntu.edu.tw (Y.-C.C.); sykuo@cc.ee.ntu.edu.tw (S.-Y.K.)
[3] Department of Communications Engineering, Feng Chia University, Taichung 407, Taiwan
[4] Department of Physics, National Taiwan University, Taipei 106, Taiwan; crchang@phys.ntu.edu.tw
* Correspondence: yttsou@fcu.edu.tw

Received: 3 April 2020; Accepted:13 May 2020; Published: 15 May 2020

Abstract: Physical unclonable function (PUF), a hardware-efficient approach, has drawn a lot of attention in the security research community for exploiting the inevitable manufacturing variability of integrated circuits (IC) as the unique fingerprint of each IC. However, analog PUF is not robust and resistant to environmental conditions. In this paper, we propose a digital PUF-based secure authentication model using the emergent spin-transfer torque magnetic random-access memory (STT-MRAM) PUF (called STT-DPSA for short). STT-DPSA is an original secure identity authentication architecture for Internet of Things (IoT) devices to devise a computationally lightweight authentication architecture which is not susceptible to environmental conditions. Considering hardware security level or cell area, we alternatively build matrix multiplication or stochastic logic operation for our authentication model. To prove the feasibility of our model, the reliability of our PUF is validated via the working windows between temperature interval (−35 °C, 110 °C) and Vdd interval [0.95 V, 1.16 V] and STT-DPSA is implemented with parameters n = 32, $i = o$ = 1024, k = 8, and l = 2 using FPGA design flow. Under this setting of parameters, an attacker needs to take time complexity O(2^{256}) to compromise STT-DPSA. We also evaluate STT-DPSA using Synopsys design compiler with TSMC 0.18 um process.

Keywords: physical unclonable function (PUF); spin-transfer torque magnetic random-access memory (STT-MRAM); identity authentication; hardware security; internet of things

1. Introduction

Presently, the pervasiveness of smart devices with embedded memories is affecting our daily life. It raises great opportunities for large-scale applications such as smart city, home automation, etc. In particular, spin-transfer torque magnetic random-access memory (STT-MRAM), an emergent technical term of embedded memory, is expectantly integrated with the internet of things (IoT). That is, the ability of STT-MRAM to operate at low supply voltages, small bit-cell footprint, non-volatility, and high read/write endurance are suitable and attractive for the IoT applications. The IoT physically connected to the STT-MRAM is a new trend technology/application and a well intersection for the development of communication and micro-electromechanical techniques.

Although there are several advantages for integrating STT-MRAM with IoT devices, the issue of security such as device identity authentication for this kind of devices is still an open challenge and thus has drawn much effort for security research communities. More precisely, an unauthenticated or illegitimate device may share fake information to disturb decisions of Cyber-Physical Systems.

If the vulnerability on IoT devices is not handled properly, then these emerging concepts potentially expose innumerable data into unsafety. Consequently, identity authentication is in the front line of data protection and can prevent from man-in-the-middle attacks. Unlike the resource-rich server, the computation capability and memory space in typical IoT devices are constrained. Traditional symmetric/asymmetric authentication cryptography architectures are not suitable for IoT devices because they would pose a significant overhead in computation on IoT devices.

Through investigating existed authentication architectures for IoT devices, we classified these architectures into two categories: software and hardware-aided architecture. The software architecture aims to exploit the traditional cryptography algorithm but adjust the strength of the algorithm dynamically to fit the operating requirement of an IoT device. The architectures usually suffers from heavy computation of the cryptography algorithm and terribly consumes the usage of memory space. In particular, they are vulnerable to hardware cloning attacks. In contrast, another solution is hardware-aided architecture, e.g., Physical Unclonable Function (PUF)-based architecture [1]. PUF is a new technology exploiting the manufacturing variability of Integrated Circuit (IC) as the unique fingerprint of each IC, in which the input to the PUF is referred as the "*challenge*" and the output is referred as the "*response*". Therefore, a PUF can be modeled as a set of *challenge-response* pairs (CRPs). Briefly, the PUF, a physical entity, can produce output value that is dependent of its physical structure to make it hard to be cloned. With the PUF, researchers have a chance to develop ultra-fast and efficient security primitives against side-channel and other common physical attacks [2].

PUF can be classified into two types [3], namely weak PUF and strong PUF. The weak PUF (*resp*. the strong PUF) means that the number of CRPs is linearly (*resp*. exponentially) in accordance with the number of components whose behavior depends on manufacturing variation. PUF that is a hardware-aided architecture aims to make the computation efficient can reduce the usage of memory space and resist hardware cloning attacks. However, the hardware-aided architecture has a fatal weakness in being susceptible to environmental conditions.

Our main contributions are summarized as follows:

- We propose a digital PUF-based secure authentication model using STT-MRAM PUF as configuration bits for the network of LUTs.
- We use two digital PUFs that have the same function as the basic building block for IoT authentication models.
- We present two versions of authentication model for different requirements on cell area: one is based on matrix multiplication and the other is based on stochastic logic.
- Through a series of attack analyses and hardware resource evaluations, we prove that our method poses lightweight overhead on participating parties and provides high security for IoT devices.
- We implement STT-DPSA on FPGA to prove the feasibility of our model.

The rest of this paper is organized as follows. Section 2 describes related works. Section 3 introduces our system model and attack model. We then describe the details of proposed methods in Sections 4 and 5. We give a system analysis including system complexity and security in Section 6, followed by the experimental results in Section 7. Finally, the conclusion is presented in Section 8.

2. Related Work

To address the essential security problem of identity authentication for low-end IoT devices, it takes much effort in the research of software-based and hardware-based authentication.

Aiming to software-based authentication, the research community in sensor networks strives to devise lightweight and practically feasible but computationally less intensive cryptography algorithms for the resource-limited hardware with lower computation capability, smaller memory size, and limited energy supply. Some studies based on symmetric-key cryptography, such as SPINS [4], TinySec [5], LEAP [6] and SER [7], have taken less computation cost and been proven to be feasible on IoT devices. However, their methods incur a large communication overhead for key distribution. As for public

key cryptography, TinyECC [8] and TinyPBC [9] have been proposed to tackle the expensive key computation cost using key size reduction. Although [4–7,10] have taken effort to enhance the security of IoT devices, they still suffer from a dilemma of computation overhead and security level. The purely software-based cryptography methods suffer from inefficient computations, dramatically consumption of memory space, and have the critical vulnerability of hardware cloning attacks under the inherent resource-constrained environment.

From the perspective of hardware-aided authentication model, PUF is attractive to the research realm recently. Layman et al. [11] proposed a weak PUF, namely Static Random-Access Memory (SRAM) PUF, consisting of symmetric and cross-coupled inverters. Once the power of the SRAM PUF is on, manufacturing variability will be leveraged to push each cell of SRAM PUF toward the logical 1 or 0. Notably, SRAM PUF exploits only one CRP for cryptography key generation rather than device authentication, in which *response* is an unknown initial state caused by manufacturing variability.

Gassend et al. [2] proposed a type of strong PUF, namely Arbiter PUF, to generate several ideal identical delay paths and apply multiplexers to switch between these paths. To create ideal identical delay paths, Arbiter PUF is designed in the layout level. Moreover, the *challenge* to the Arbiter PUF is the control signal of multiplexers. With implementing the Arbiter PUF, two ideal identical paths are fed into an arbiter [12]. When the input of the Arbiter PUF changes, the signal is propagated through two different ideal identical delay paths. Because manufacturing variability of different hardware, the arrival time of these two input signals in an arbiter will be various. As a result, the arbiter can generate a *response* depended on which signal arrives first. This type of PUF can generate several CRPs and be used for lightweight authentication model on IoT devices. The typical deployment of strong PUF for identity authentication model is shown in Figure 1 and described as follows.

The client side equipped with a PUF is manufactured with several CRPs. Moreover, the verifier side has a database of the client's CRPs. When the client needs to be authenticated, the verifier randomly selects one CRP from the database. Then, the verifier sends a selected *challenge* to the client and reserves the corresponding *response* b on his/her hand. Once the client receives the *challenge*, the *challenge* is fed into a PUF for generating a corresponding *response* b'. Subsequently, the client sends the b' to the verifier. Finally, the verifier determines that if $b = b'$, the client is legitimate; otherwise, the client fails to be authenticated.

However, a strong PUF has exponentially growth numbers of CRPs with respect to components whose behaviors are depended on manufacturing variation. This makes the verifier cannot allocate sufficient memory space for entire CRPs of the strong PUF. Additionally, if a CRP is used repeatedly in an authentication model, an attacker can easily crack the authentication model by sending the used *response* to the verifier. As for computationally efficiency, Matched Public PUF (PPUF) [13] was proposed in leveraging the circuit aging model to produce an identical PUF. Nevertheless, it needs high implementation requirement in terms of measurement accuracy and environmental stability.

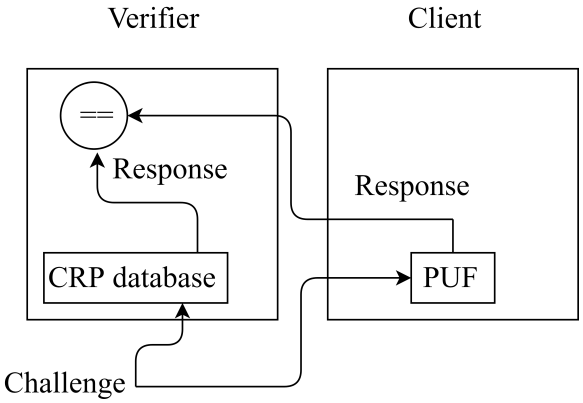


Figure 1. Typical strong PUF-based authentication model.

Xu et al. [14] first proposed digital PUF architecture to resolve the problem of environmental susceptibility in [2,11,13]. They developed Digital Bimodal Function (DBF) by using a randomly generated Boolean function as the large-scale network of lookup tables (LUTs) and exploiting the irreversibility of decomposition of Boolean function to represent the same Boolean function in two forms, namely compact form and complex form. The method in [14] was implemented on FPGA and aimed to exploit the huge gap of computation time and hardware resource between these two forms. However, when configuration bits of LUTs are compromised, the DBF is easily cloned. Therefore, Xu et al. [15] further proposed an advanced model by using Arbiter PUF to generate configuration bits for the network of LUTs. Because the variation of environmental conditions, only stable CRPs can be used to configure the network of LUTs. To provide a practical usage, the Arbiter PUF must take additional effort to obtain stable CRPs under different environment conditions. Some hardware-aided architectures, such as [16–20], also have a fatal weakness in being susceptible to environmental conditions.

In 2015, Xu et al. [21] proposed a digital PUF approach by observing intentional faults that change the function of IC dramatically. Subsequently, Miao et al. [22] proposed a digital PUF approach by using variation of lithography. The digital PUF proposed in [21] was hard to be applied in real-world applications because the intentional faults not only affect the function of IC, but also affect the operating region of transistors. The digital PUF [21] might cause much power consumption and put Complementary Metal-Oxide-Semiconductor (CMOS). gates into an uncertain operating region. In contrast to [21], the digital PUF proposed in [22] was only implemented in simulation stage. The digital PUF in [21] and [22] were difficultly duplicate in both client and verifier sides to drive identity authentication while the digital PUF proposed in [14] can be applied easily. Worse, a typical strong PUF-based authentication model will consume unacceptable memory usages for the verifier.

To solve the drawback of unacceptable memory usages, an emerging method called Public PUF (PPUF) was proposed by [23]. As for the PPUF-based model, compared to the need for the internal secret storage of CRPs, the PPUF model has no secret information. In other words, the authentication information is stored publicly instead of being stored in the verifier side. As a result, this type of PUF does not consume a large amount of memory.

Park et al. [24] proposed a PPUF-based authentication model, namely PUFSec, exploiting the gap of *response* generation time in the PPUF model and PPUF hardware. The PPUF hardware can compute the *response* in a measurably faster time compared to PPUF model. That is, the PPUF-based method exploits a huge computation gap between simulation and execution, and computation gap of challenge between verifier and attacker. However, the computation complexity of the participating parties in PUFSec would growth exponentially with respect to the size of PUF hardware area. Moreover, selecting an appropriate T' which is the threshold time in the PPUF-based model is an open challenge in computationally efficiency for the distinction of an attacker and a legitimate device.

A new low-cost PUF-based temperature sensor for secure remote temperature sensing was proposed by Cao et al. [25]. Cao et al. exploited the approximately linear positive temperature coefficient of CMOS inverter in super-threshold operation to calibrate the running frequency of ring oscillator in a reconfigurable ring oscillator PUF at different temperature. The ring oscillator frequency corresponding to the sensed temperature is fed into a randomizer seeded by the input challenge to select new ring oscillator pairs for comparison to generate a random, unique and physically unclonable digital tag, which is valid for a selected input challenge to a target device at a particular temperature. In 2018, Gu et al. [26] proposed the first IoT secure communication framework for BLE-based networks that guards against device spoofing via fingerprinting-based device authentication. These designs are orthogonal to our STT-MRAM-based PUF.

In this paper, we propose a digital PUF-based authentication model using STT-MRAM PUF for IoT devices. To address the problems of exponential initialization time and dramatically memory consumption caused in [14,15], we use two identical digital PUF for our authentication model. Moreover, we provide the configuration bits of LUTs using STT-MRAM PUF. Because the hardware area

of STT-MRAM PUF is significantly lower than others PUF such as Arbiter PUF and SRAM PUF [27], we use STT-MRAM PUF to configure the network of LUTs. The advantages of our digital PUF-based authentication model include efficient computation, efficient memory usages, and unclonable hardware using stable CRPs of STT-MRAM PUF regardless of operational and environmental conditions. The computation overhead of participating parties in our protocol is lightweight while our model still exploits the huge gap between execution and simulation, so that an attacker needs to take much effort in terms of execution time and computational complexity for compromising our architecture.

To the best of our knowledge, we first propose a method using two digital PUFs with the cell of STT-MRAM PUF, which have the same function as the basic building block for the IoT authentication architecture. Table 1 presents comparisons of our model with previous studies in terms of the computational complexity, susceptibility, initialization time, and the resource complexity. Our method outperforms the well-known methods in terms of the computational and resource complexity but not susceptible to environmental conditions.

Table 1. Comparisons.

Methods	Computationa Complexity (Verifier Side)	Computational Complexity (Client Side)	Susceptibility	Initialization Time	Resource Complexity
PUFSec [24]	Exponential	Exponential	Yes	Linear	Exponential
Matched PPUF [13]	Linear	Linear	Yes	No	Constant
DBF [14]	Polynomial	Linear	No	Exponential	Exponential
DBF with Arbiter PUF [15]	Polynomial	Linear	Exponential	Yes	Exponential
STT-DPSA (our method)	Linear	Linear	No	Linear	Polynomial

3. System and Attack Models

In this section, we describe the system model and attack model for our proposed architecture.

3.1. System Model

Our system model is shown in Figure 2. The PUF circuit used in STT-DPSA is a digital model and a type of strong PUF. STT-DPSA has numerous CRPs with the ideal performance of statistic metrics, i.e., inter-Hamming distance being equal to 0 and uniformity of *response* being equal to 0.5 [28]. For easy representation, the design of PUF in a digital model for STT-DPSA and the design of STT-MRAM PUF are represented as $\mathcal{P}^D$ and $\mathcal{P}^S$, respectively.

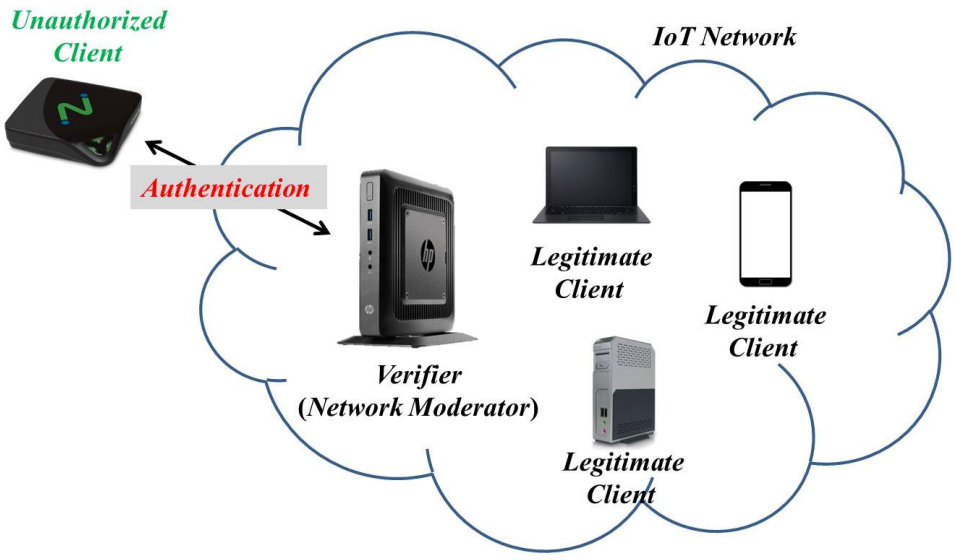


Figure 2. System model.

In addition, we assume that there are two roles in our model, namely verifier and client. The verifier, a moderator in an IoT network, is a trusted part for authenticating clients via $\mathcal{P}^D$. The client, an IoT device, intents to join a dominated network for communicating with other legitimate members.

3.2. Attack Model

We assume that attackers can eavesdrop the information transmitted between the verifier and client, but attackers cannot retrieve any information stored in both of them. In other words, attackers can perform man-in-the-middle attacks through recording the information exchanged between the verifier and client, and try to impersonate the legitimate client in the next round of authentication based on these records.

4. Design of a Strong Digital Physical Unclonable Function (PUF) Based on Spin-Transfer Torque Magnetic Random-Access Memory (STT-MRAM) PUF

The structure of $\mathcal{P}^D$ is shown in Figure 3, which composes of several LUTs that are configured by $\mathcal{P}^S$ to mitigate environmental susceptibility.

A cell of STT-MRAM and $\mathcal{P}^S$ used in our method are shown in Figures 4 and 5, in which the MTJ is a component consisting of two ferromagnetic layers separated by a thin insulator and is implemented by two states using simple resistor model: anti-parallel (AP) and parallel (P). The resistance of the MTJ in the P state is comparably lower than the resistance of the MTJ in the AP state. In theory, if the two MTJs have the same state and all their physical parameters are the same, then they should have the same resistance value. In fact, they have manufacturing variations led to the inevitable difference between the two ideal identical resistance values. As a result, the $\mathcal{P}^S$ exploits the resistance difference caused by the manufacturing variability of the MTJs in the same state. The dual in-line package of our MTJs is shown in Figure 6. Through wire bonding, two adjacent MTJs are selected to use.

$\mathcal{P}^S$ uses two cross-coupled inverters to amplify the difference between the resistances of the MTJs in the same state. Figure 5 exhibits the design of $\mathcal{P}^S$, in which output signal BIT is a *response* of the $\mathcal{P}^S$. The generation process of *response* for the $\mathcal{P}^S$ is as follows.

In the beginning, MTJ A and MTJ B are configured to the same state. Active-low signal RCLK is set to high voltage while the two NMOS transistors controlled by RCLK is 'on' and the top PMOS transistor connected to Vdd is 'off'. Moreover, BIT and BITB are short to the digital ground. After switching RCLK to low voltage, the top PMOS transistor is 'on' and the two NMOS transistors controlled by RCLK become 'off'. BIT and BITB are pre-charged and activate the other transistors. Since the two cross-coupled inverters are activated, BIT and BITB are amplified to opposite digital levels in the regeneration behavior by the resistance difference between MTJ A and MTJ B. At the end, the output signal BIT is a *response* of the $\mathcal{P}^S$ and BITB is the inverse of BIT.

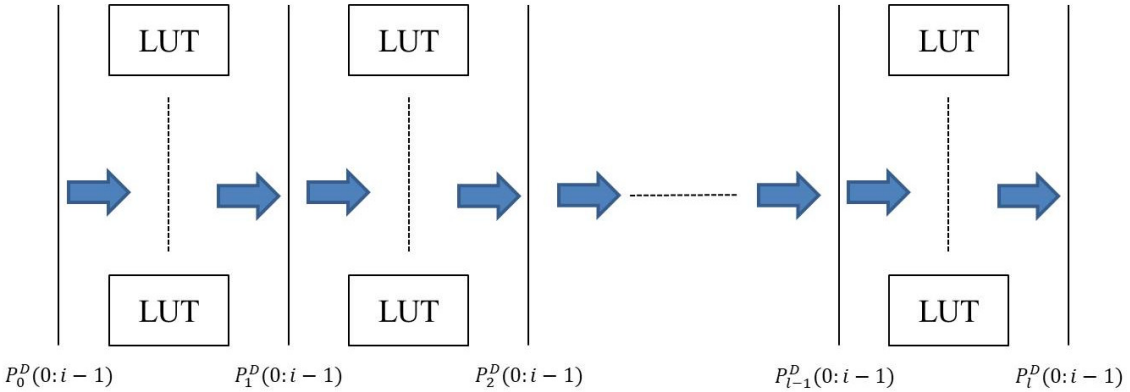


Figure 3. Structure of $\mathcal{P}^D$.

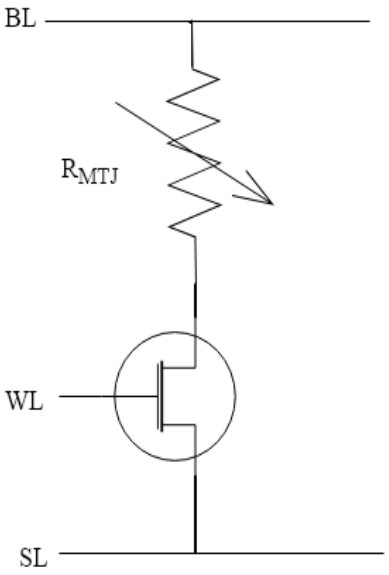


Figure 4. A cell of STT-MRAM.

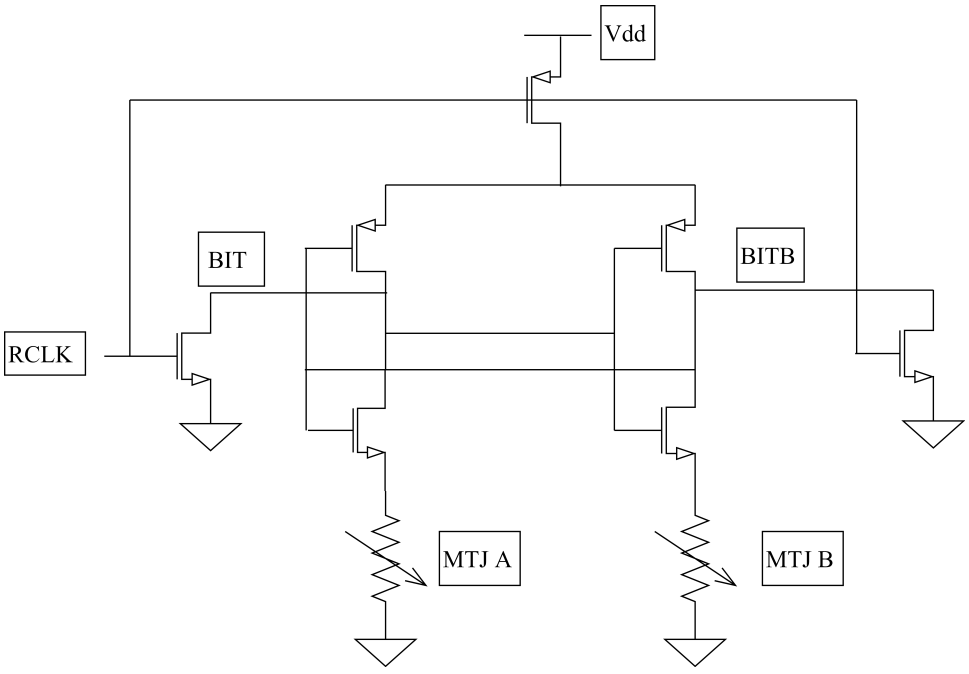


Figure 5. Design of $\mathcal{P}^S$.

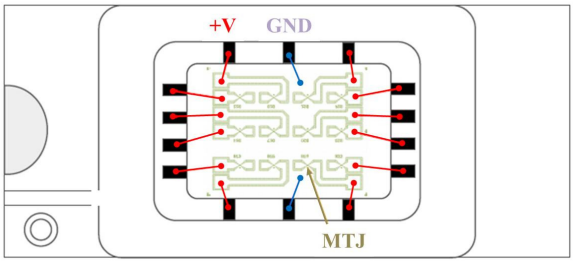


Figure 6. Dual in-line package of our MTJs.

Notably, the $\mathcal{P}^S$ is used as a key of LUTs. That is, we exploit the $\mathcal{P}^S$ to generate configuration bits of LUTs. Therefore, we just use the *response* of a $\mathcal{P}^S$ once. The validation of unpredictability for a $\mathcal{P}^S$

under different environmental conditions is demonstrated in Section 7. With the composition of our $\mathcal{P}^S$ and LUTs, we can model the $\mathcal{P}^D$ in our authentication architecture.

We decompose $\mathcal{P}^D$ into multiple stages, in which one stage consists of several LUTs. Let the number of primary input be i and the number of primary output be o. In our design, we let $i = o$; let the number of stages be l; let the number of LUTs be equal to i in each stage of the $\mathcal{P}^D$. As shown in Figure 3, $\mathcal{P}_0^D(0 : i-1)$ is a bus consisting i primary inputs and is the input for the first stage of the LUT network when the output for the first stage is $\mathcal{P}_1^D(0 : i-1)$. Each stage of $\mathcal{P}^D$ follows the same rule. Finally, $\mathcal{P}_l^D(0 : i-1)$ will be the bus that consists of i primary outputs. Each LUT is programmed using verilog code in the structure of 4-inputs LUT, as exhibited in the following verilog code of module LUT (in, value), in which the signal in (3:0) is the input of LUT module consists of 4 bits while the signal value is the output bit of a LUT module.

Notably, the configuration bits in the always statement of the verilog code must be provided in an unpredictable manner. The response signal from BIT of a $\mathcal{P}^S$ is used to configure the LUT. For example, 16 cells of $\mathcal{P}^S$ are used to configure a 4-input LUT, in which the response signals (i.e., BITs) are 1110000100000001. Next, we describe our authentication model based on the design of $\mathcal{P}^D$ using $\mathcal{P}^S$ as configuration bits.

5. Authentication Model

In this section, we present two authentication models. One is based on matrix multiplication and the other is based on stochastic logic, in which both of them exploit $\mathcal{P}^D$ as a common building block.

5.1. Authentication Model Based on Matrix Multiplication

To develop an efficient and a feasible PUF-based authentication model in terms of computational time and memory space regardless of operational and environmental conditions, we do not directly exchange CRPs generated by $\mathcal{P}^D$ between the client and verifier for identity authentication. Instead, we propose a strong digital PUF-based authentication model through exchanging the CRPs of matrix A that will be updated in each authentication round using the *response* of $\mathcal{P}^D$ between the verifier and client. Our authentication model based on matrix multiplication is shown in Figure 7.

Before authentication, we must deploy the matrix A and $\mathcal{P}^D$ to the verifier and client initially. Using the deployed matrix A and $\mathcal{P}^D$, behaviors of the verifier and client are as follows.

Step 1: The verifier (*resp*. the client) generates a random number $\mathcal{N}_v$ (*resp*. $\mathcal{N}_c$) using pseudo-random number generator (PRNG) [12] and exchanges the random number with the client (*resp*. the verifier).

Step 2: Both of the verifier and client form a string C by concatenating $\mathcal{N}_c$ with $\mathcal{N}_v$.

Step 3: Both of the verifier and client use the string C as a *challenge* of his/her $\mathcal{P}^D$ and obtain a corresponding *response* (called R for short) in a matrix form.

Step 4: Both of the verifier and client compute the matrix multiplication AR and replace matrix A with AR. If the resulted matrix A is a zero matrix, then we must re-initialize the matrix A and go back to step 1. That is because a zero matrix used in step 5 will lead to a result of zero vector that can be easily predicted by an attacker.

Step 5: The verifier selects a column vector x generated by PRNG as a *challenge* and sends x to the client. Subsequently, the verifier (*resp*. the client) computes Ax as b (*resp*. b'). Next, the client sends b' to the verifier. If $b = b'$, then the verifier authorizes the client as a legitimate device. Otherwise, the client fails to be authorized and viewed as an illegitimate one. We show behaviors of the verifier and client in Algorithms 1 and 2, respectively.

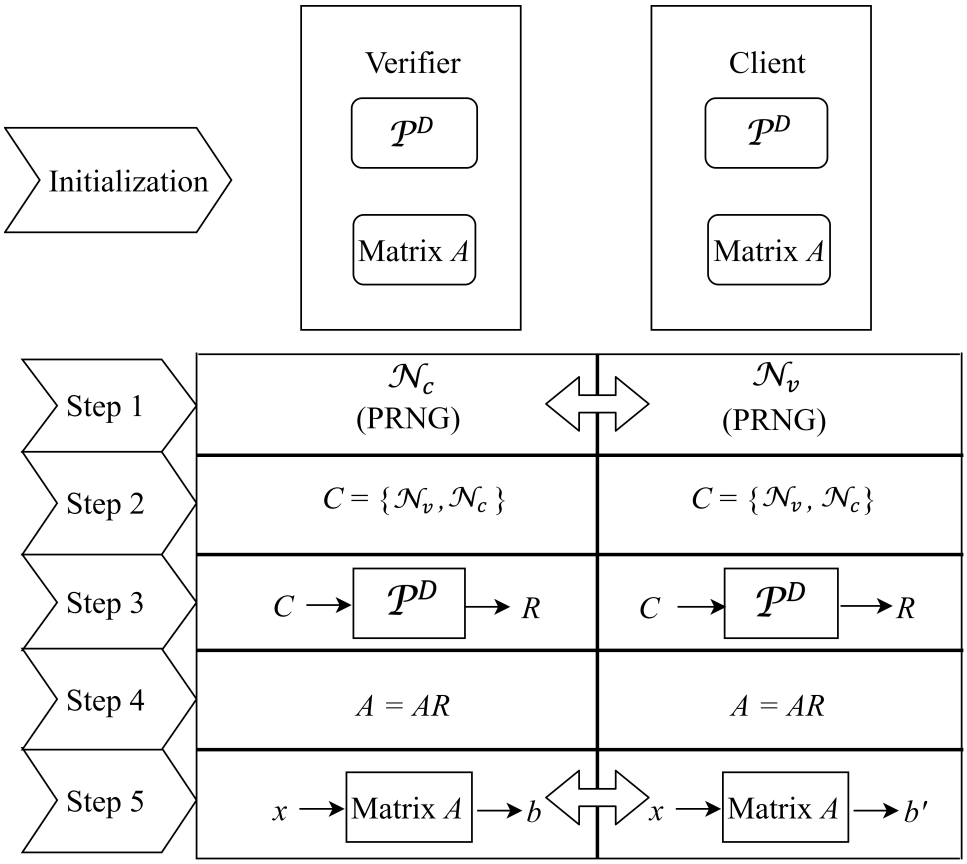


Figure 7. Steps of authentication model.

Algorithm 1: The verifier Side

Input: $\mathcal{P}^D$, A, N_c, and b'

Output: Whether the authentication process succeeds

`// Flow chart of authentication model based on matrix multiplication for the verifier side is as shown in the left side of Figure 8.`

Generate a random number $\mathcal{N}_v$ and send it to the client;
Receive a random number $\mathcal{N}_c$ from the client and concatenate it with the $\mathcal{N}_v$ to form a string C;
Generate the *response* of $\mathcal{P}^D$ R: $R = \mathcal{P}^D(C)$;
Compute $A = AR$;
if *A is a zero matrix* **then**
    Re-initialize the matrix A;
    Go to step 1;
else
    Randomly select a vector x as a *challenge* and sends it to client;
end
Compute $Ax = b$;
Receive the response b' from the client;
Compare b and b'. If they are the same, then the authentication process succeeds; otherwise, the authentication process fails;

Algorithm 2: The client Side

Input: $\mathcal{P}^D$, A, N_v, and x
Output: b'

```
// Flow chart of authentication model based on matrix multiplication for the
   client side is as shown in the right side of Figure 8.
```

Generate $\mathcal{N}_c$ and send it to the verifier;
Receive $\mathcal{N}_v$ from the verifier and concatenate it with the $\mathcal{N}_c$ to form a string C;
Generate the *response* of $\mathcal{P}^D$ R: $R = \mathcal{P}^D(C)$;
Compute $A = AR$;
if *A is a zero matrix* **then**
    Re-initialize matrix A;
    Go to step 1;
else
    receive the *challenge* vector x from the verifier;
end
Compute $Ax = b'$ and send it to the verifier;

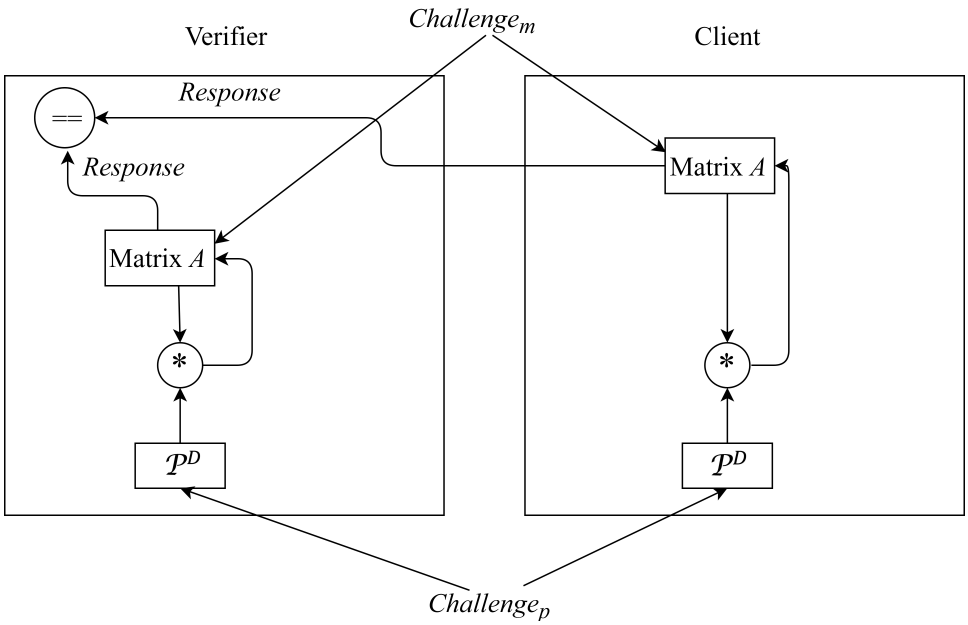


Figure 8. Flow chart of authentication model based on matrix multiplication.

Notably, we exchange the vector x and the vector b between the verifier and client. These two vectors are the information regarding to the matrix A. More precisely, the vector x is an input of A. After operating a matrix multiplication, the vector b becomes an output of A. Therefore, we call the exchanged vectors as CRPs of A. For easy presentation, the CRPs of A is known as CRP^A. Because we refresh the A at each round of authentication using the *response* of $\mathcal{P}^D$, it is difficult to predict the next round of CRP^A based on the used one. Therefore, $\mathcal{P}^D$ can satisfy the requirement of designing a strong PUF-based authentication model [3].

5.2. Authentication Model Based on Stochastic Logic

Before introducing the authentication model based on stochastic logic, we present the preliminary knowledge of stochastic logic used to construct our authentication model. Unlike the traditional binary encoding used to treat data in binary number, stochastic logic is applied to encode the data in the form of a bitstream using probability model [29,30], in which the probability for a bitstream with r 1's and

$N - r$ 0's denotes r/N. With operating stochastic logic, addition and multiplication are candidates that can be processed using only one logic component.

Using the stochastic adder, the two inputs of adder are bitstreams x_b and y_b, and the output is the bitstream z_b while a random input s generates 0 or 1 with fair probability. The vectors x and y are encoded as the probability of the specific bit when the bitstreams x_b and y_b are equal to 1. In one clock cycle, the probability of the specific bit when $z_b = 1$ is $\frac{1}{2} \times x + \frac{1}{2} \times y$. Consequently, if the length of bitstreams x_b and y_b is N, then the expected number of bit 1 in z_b is $\frac{x+y}{2}$. This is a scaled adder of two values implemented by one multiplexer. Using a multiplier with two inputs implemented by one AND gate, the two inputs of AND gate are bitstreams x_b and y_b, and the outputs are the bitstreams z_b. The vectors x and y are encoded as the probability of the specific bit in bitstream x_b being 1. Assumed that the encoding bitstreams x_b and y_b are independent. In one clock cycle, the probability of the specific bit when $z_b = 1$ is $x \times y$. Consequently, if the length of bitstreams x_b and y_b is N, then the expected number of bit 1 in z_b is $x \times y$. Generally, the cell area for designing stochastic logic is significantly small. Therefore, we synthesize stochastic logic adder and multiplier to reduce the cell area of our model. Subsequently, we will present our authentication model based on stochastic logic.

The authentication model based on stochastic logic is shown in Figure 9, in which $\mathcal{P}^D$ is used as a building block. Unlike authentication model based on matrix multiplication, stochastic logic version does not need to initialize the matrix A in both verifier and client. Using the deployed $\mathcal{P}^D$, behaviors of the verifier and client are as follows.

Step 1: The verifier (*resp*. the client) generates a random number $\mathcal{N}_v$ (*resp*. $\mathcal{N}_c$) using PRNG and exchanges the random number with the client (*resp*. the verifier).

Step 2: The verifier and client form a string C_1 by concatenating $\mathcal{N}_c$ with $\mathcal{N}_v$.

Step 3: The verifier and client use the string C_1 as a *challenge* of his/her $\mathcal{P}^D$ and obtain a corresponding *response* (called R_1 for short).

Step 4: The verifier and client repeat steps $1 - 3$ and obtain a corresponding *response* (called R_2 for short).

Step 5: The verifier and client treat R_1 and R_2 as bitstreams and take these two bitstreams as inputs to a stochastic logic adder for obtaining a corresponding calculation result $value$ and $value^{'}$.

Step 6: The client sends $value^{'}$ to the verifier. Finally, if $value = value^{'}$, then the verifier authorizes the client as a legitimate device. Otherwise, the client fails to be authorized and viewed as an illegitimate one. We show the behaviors of the verifier and client in Algorithms 3 and 4, respectively.

Algorithm 3: Verifier Side

Input: $\mathcal{P}^D$, N_c, $value$, and $value'$

Output: Whether the authentication process succeeds

```
// Flow chart of the authentication model based on stochastic logic for the
   verifier side is as shown in the left side of Figure 10.
```

1 Generate a random number $\mathcal{N}_v$ and send it to the client;
2 Receive a random number $\mathcal{N}_c$ from the client and concatenate it with the $\mathcal{N}_v$ to form a string C_1;
3 Generate the *response* of $\mathcal{P}^D$ R_1: $R_1 = \mathcal{P}^D(C_1)$;
4 Repeat steps 1–3 and obtain $R_2 = \mathcal{P}^D(C_2)$;
5 Treat R_1 and R_2 as bitstreams and compute $value = (R_1 + R_2)/2$ using stochastic logic adder;
6 Receive the $value'$ from the client;
7 Compare $value$ with $value'$. If $value = value'$, then the authentication process succeeds; otherwise, the authentication process fails;

Algorithm 4: Client Side

Input: P^D, and N_v
Output: $value'$

```
// Flow chart of the authentication model based on stochastic logic for the
   client side is as shown in the right side of Figure 10.
```

1 Generate a random number $\mathcal{N}_c$ and send it to the verifier;
2 Receive a random number $\mathcal{N}_v$ from the verifier and concatenate it with the $\mathcal{N}_c$ to form a string C_1;
3 Generate the *response* of $\mathcal{P}^D$ R_1: $R_1 = \mathcal{P}^D(C_1)$;
4 Repeat steps 1–3 and obtain $R_2 = \mathcal{P}^D(C_2)$;
5 Treat R_1 and R_2 as bitstreams and compute $value' = (R_1 + R_2)/2$ using stochastic logic adder;
6 Send the $value'$ to the verifier;

Notably, in the step 5 of authentication model, the selected input of stochastic logic multiplexer is provided by the same random source deployed in the verifier and client. With this assumption, *value* is equal to $value'$ if they are transmitted over the air without being destroyed by an attacker. In the next section, we will analyze our proposed model in system complexity and security.

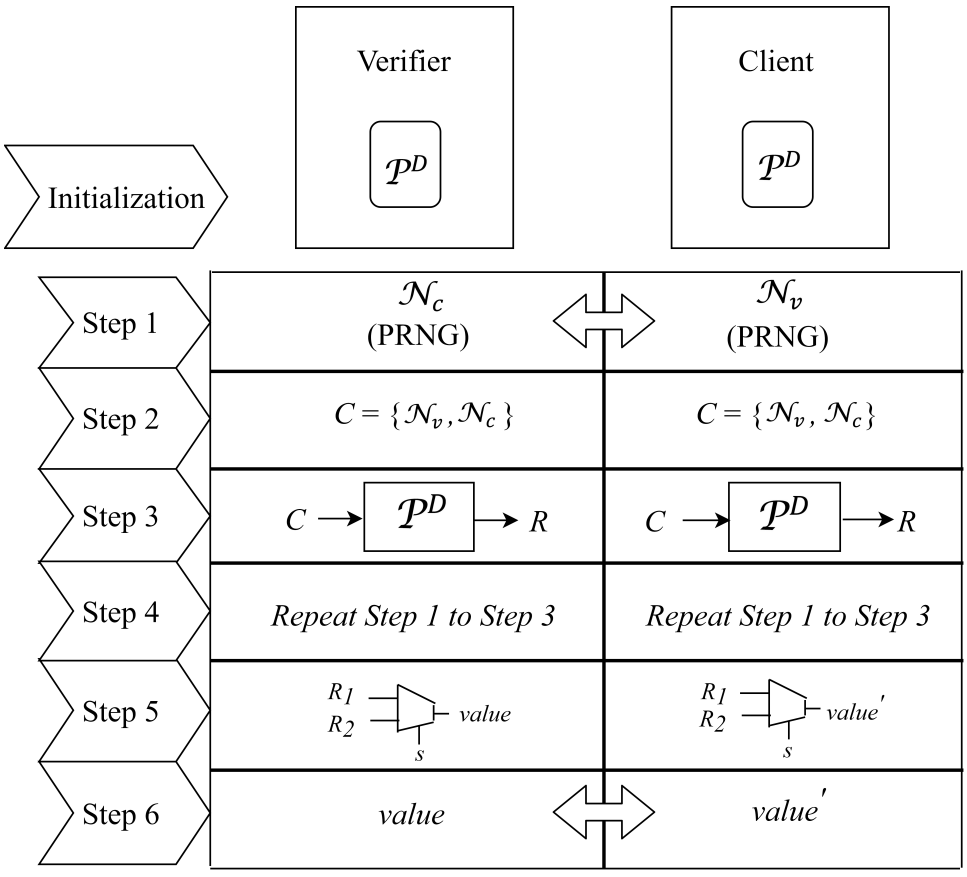


Figure 9. Steps of stochastic logic-based authentication model.

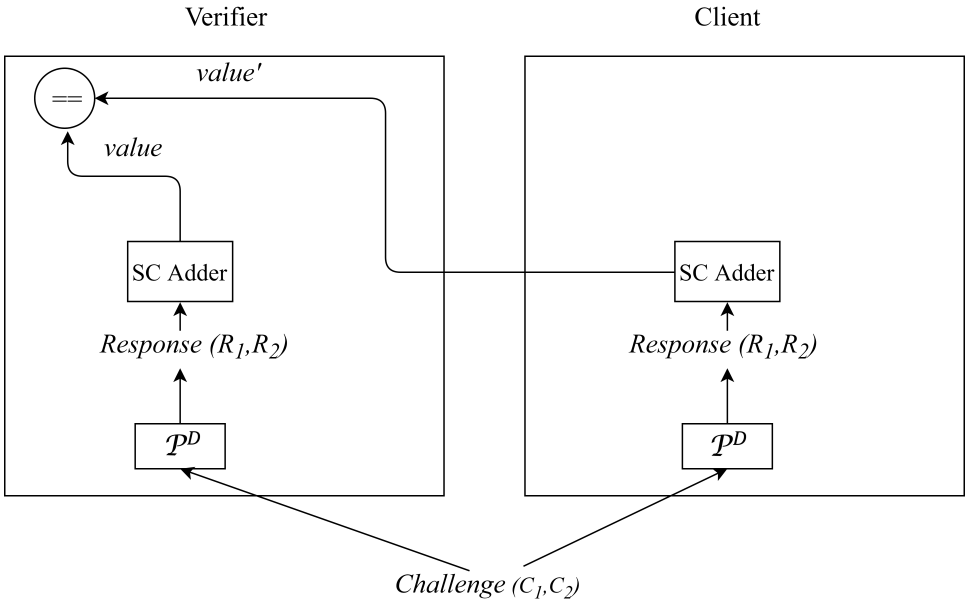


Figure 10. Flow chart of the authentication model based on stochastic logic (SC).

6. System Analysis

Because our proposed model is implemented in digital circuit, two major issues need to be discussed. First, the cell area of digital circuit affects the cost of manufacturing. Second, the delay of critical path affects the upper bound of operating frequency. Therefore, we analyze the cell area and the delay of critical path of proposed model based on ASIC cell-based design flow and prove that our method can resist against man-in-the-middle attacks in the following sections.

6.1. System Complexity

The dimension of the matrix A is $n \times n$, in which each element in A is represented by k bits. Let the length of input bits of $\mathcal{P}^D$ be i and let the length of output bits of $\mathcal{P}^D$ be o, where $i = o$. Moreover, a 4-input LUT is used as a basic block for $\mathcal{P}^D$. We also let the length of $\mathcal{P}^D$ be l. Subsequently, we will discuss the effects of parameters n, k, and l on the cell area and the delay of critical path for our authentication model.

In the first three steps of matrix multiplication process (as shown in Figure 7), $\mathcal{N}_v$ and $\mathcal{N}_c$ are generated using a PRNG and exchanged between the client and verifier to produce a $challenge_p$ C. Next, C is used as an input for $\mathcal{P}^D$ to produce a $(n \times n)$-matrix *response* R which will be multiplied by the matrix A. Finally, the client and verifier execute matrix multiplications to do identity authentication. For estimating the cell area of the proposed model, we evaluate the effect of matrix multiplications and the execution of $\mathcal{P}^D$. Because the cell area of matrix multiplication for two $n \times n$ matrices is significantly larger than that of matrix multiplication for one $n \times n$ matrix by one $n \times 1$ matrix, we only evaluate the cell area of matrix multiplication for two $n \times n$ matrices. In addition, the cell area of PRNG is smaller than the cell area of matrix multiplication. As for the delay of critical path, it is mainly incurred by matrix multiplications and the execution of $\mathcal{P}^D$. The delay of matrix multiplication for two $n \times n$ matrices is higher than the matrix multiplication for one $n \times n$ matrix by one $n \times 1$ matrix. In terms of PRNG, LFSR is implemented as a PRNG. When the clock edge is triggered, the PRNG is ready to be used. For simplicity of analysis, we put the execution of $\mathcal{P}^D$ and the matrix multiplication of two $n \times n$ matrices in the same clock cycle. Operations of other parts in proposed model are put in other clock cycles. By this approach, we make sure that the execution of $\mathcal{P}^D$ and the matrix multiplication of two $n \times n$ matrices are the critical paths.

As shown in Figure 9, the first three steps are as the same as matrix multiplication. The step 4 is just the repeat of steps 1–3. To compute $value$ and $value'$, we only need one multiplexer to compute.

Consequently, the cell area and the delay of critical path for one multiplexer are significantly lower than that for matrix multiplication. We assume that the delay of critical path and cell area of stochastic logic version for authentication model mainly come from $\mathcal{P}^D$. In the following, we analyze the cell area and the delay of critical path in the verifier and client for matrix multiplication version and stochastic logic version.

Theorem 1. *When authentication model is based on matrix multiplication, the cell area at the verifier/client side is formulated as:*

$$\mathcal{A}_c = i \cdot l \cdot LUT_c + n^3 \cdot k \cdot A_{am} + R, \tag{1}$$

where i, l, LUT_c, n, k, A_{am}, and R denote the number of bits input to $\mathcal{P}^D$, the length of $\mathcal{P}^D$, the average cell area of a 4-input LUT, the dimension of the matrix A, the bits of each element in the matrix A, the average cell area of 1-bit adder and multiplier, and the others, respectively.

Proof. The two main factors affecting the cell area at the verifier/client side are the execution of $\mathcal{P}^D$ and matrix multiplication, in which the number of LUTs in the $\mathcal{P}^D$ is $i \cdot l$. As a result, the cell area of $\mathcal{P}^D$ is $i \cdot l \cdot LUT_c$, where LUT_c is the average cell area of a 4-input LUT. For the matrix multiplication, each element of new matrix generated by the multiplication of two $n \times n$ matrices is the result of inner product of two $n \times 1$ vectors, where inner product operation of two $n \times 1$ vectors needs n adders and n multipliers. A new matrix generated by two $n \times n$ matrices has n^2 elements. Therefore, multiplication of two $n \times n$ matrices needs $n \cdot n \cdot n$ adders and multipliers. As for the parameter k, the number of bits used to encode one element of A has linear effect on the cell area of the adder and multiplier. Taking operation of addition of two 3-bit values as an example, addition of two 3-bit values will need 15 gates in a typical case. When we change the number of bits to encode one element of A, it has linear effect on the cell area of an adder. The operation of a multiplication is the same as that of an adder. As a result, the cell area of matrix multiplication is $n^3 \cdot k \cdot A_{am}$, where A_{am} is the average cell area of 1-bit adder and 1-bit multiplier. Finally, we represent the area of the remaining part of proposed model (i.e., PRNG) as R. Therefore, the cell area at the verifier/client side is $i \cdot l \cdot LUT_c + n^3 \cdot k \cdot A_{am} + R$. □

Theorem 2. *When authentication model is based on matrix multiplication, the delay of critical path at the verifier/client side is formulated as:*

$$\mathcal{D}_{cp} = l \cdot LUT_d + n \cdot k \cdot A_{DD}, \tag{2}$$

where n, k, A_{DD}, l, and LUT_d denote the dimension of the matrix A, the bits of each element in the matrix A, the average delay of a 1-bit adder, the length of $\mathcal{P}^D$, and the average delay of a 4-input LUT, respectively.

Proof. The $\mathcal{D}_{cp}$ for the verifier/client side is incurred by the execution of $\mathcal{P}^D$ and the matrix multiplication AR. From the primary input of $\mathcal{P}^D$ to the output of $\mathcal{P}^D$, as shown in Figure 3, the parallel network of LUTs is with l levels of 4-input LUTs. Therefore, $\mathcal{D}_{cp}$ is proportional to the length of $\mathcal{P}^D$ and represents as $l \times LUT_d$. Additionally, to analyze the execution of matrix multiplication, multiplication of two $n \times n$ matrices will generate a new matrix with n^2 elements, where each element is the result of inner product of two $n \times 1$ vectors. Moreover, $\mathcal{D}_{cp}$ is affected by the delay of executing an adder while processing inner product for these two matrices to result in n^2 elements. These resulted elements can be computed completely within the delay time of executing n adders. As for the parameter k, it has the linearly effect on the delay of matrix multiplication. Taking an operation for adding two 3-bit values, the computation time for the addition of these two values is in 3-adder delay. When we change the number of bits to encode one element of the matrix A, the number of bits used to encode one element of the matrix A has linearly effect on the delay of matrix multiplication. That is,

the delay of matrix multiplication can be formulated as $n \times k \times A_{DD}$. Therefore, the delay of critical path at the verifier/client side is $l \cdot LUT_d + n \cdot k \cdot A_{DD}$. □

Theorem 3. *When authentication model is based on stochastic logic, the cell area at the verifier/client side is formulated as:*

$$\mathcal{A}_c = i \cdot l \cdot LUT_c + R, \tag{3}$$

where i, l, LUT_c, and R denote the number of bits input to $\mathcal{P}^D$, the length of $\mathcal{P}^D$, the average cell area of a 4-input LUT, and the others, respectively.

Proof. The main factor affecting the cell area in the verifier/client side are the execution of $\mathcal{P}^D$, in which the number of LUTs in $\mathcal{P}^D$ is $i \cdot l$. As a result, the cell area of $\mathcal{P}^D$ is $i \cdot l \cdot LUT_c$, where LUT_c is the average cell area of a 4-input LUT. Finally, we represent the cell area of the remaining part (i.e., PRNG) as R. Therefore, the cell area in the verifier/client side is $i \cdot l \cdot LUT_c + R$. Obviously, the cell area in using stochastic logic for authentication model is smaller than the cell area in using matrix multiplication for authentication model. □

Theorem 4. *When authentication model is based on stochastic logic, the delay of critical path at the verifier/client side is formulated as:*

$$\mathcal{D}_{cp} = l \cdot LUT_d, \tag{4}$$

where l and LUT_d denote the length of $\mathcal{P}^D$ and the average delay of a 4-input LUT, respectively.

Proof. $\mathcal{D}_{cp}$ for the verifier/client side is incurred by the execution of $\mathcal{P}^D$. From the primary input of $\mathcal{P}^D$ to the output of $\mathcal{P}^D$, as shown in Figure 3, the parallel network of LUTs is with l levels of a 4-input LUT. Because $\mathcal{D}_{cp}$ is proportional to the length of $\mathcal{P}^D$ and represents as $l \times LUT_d$, the delay of critical path at the verifier/client side is $l \cdot LUT_d$. Obviously, the delay of critical path in using stochastic logic is smaller than that using matrix multiplication for authentication model. □

In summary, our proposed model in matrix multiplication and stochastic logic poses linearly growth on delay of critical path and poses polynomial growth on cell area when changing our system parameters. More importantly, our model is not affected by environmental condition because it is implemented using digital circuit.

6.2. System Security

In this section, we prove that our system can resist against man-in-the-middle attacks, in which an attacker may try to break our system by eavesdropping information exchanged between the verifier and client. The attacker would drive the following attacks: (1) $\mathcal{P}^D$ modeling attacks, (2) modeling attack to the matrix A of matrix multiplication authentication model, (3) brute-force attacks to $\mathcal{P}^D$, (4) brute-force attacks to the matrix A of matrix multiplication authentication model, (5) brute-force attacks to the *response* of the matrix A of matrix multiplication authentication model, (6) *value* modeling attacks to stochastic logic authentication model, and (7) resistance against machine-learning attacks.

6.2.1. $\mathcal{P}^D$ Modeling Attacks

The $\mathcal{P}^D$ modeling attack is a machine-learning type of attacks [31], which aims to predict the behaviors of $\mathcal{P}^D$ by observing some CRPs of $\mathcal{P}^D$. In fact, we do not reveal any parameter of CRP_D in our authentication model. As a result, our authentication model is resilient against $\mathcal{P}^D$ modeling attacks.

6.2.2. Modeling Attack to the Matrix A of Matrix Multiplication Authentication Model

An attacker may intent to eavesdrop the information transmitted over the air and try to model the matrix A to compute the privacy parameter b.

Lemma 1. *If the same matrix A exists in different authentication rounds, then the matrix A can be easily obtained by solving the linear equation with the known CRPs.*

Proof. Let the dimension of the matrix A be $n \times n$. In our authentication model, we resolve the same linear system $Ax = b$ between the verifier and client, where the dimensions of x and b are $n \times 1$. During the process of authentication, the model will reveal a pair (x, b). For example, if a legitimate client successfully gets the authentication from the verifier three times, then an attacker can resolve the matrix A by using the pairs (x_1, b_1), (x_2, b_2), and (x_3, b_3). The attacker can form a new linear system $A'x = b$, where the dimension of A' is $(3 * n) \times n^2$ and the dimensions of x and b are $n^2 \times 1$. Expanding this concept, if the client successfully gets the authentication from the verifier j times, then the attacker can form a new linear system $A'x = b$, where the dimension of A' is $(j * n) \times n^2$ and the dimension of x and b are $n^2 \times 1$. In this linear system, it satisfies: $rank(A') \leq n^2 < (j * n)$. If we randomly select *challenges* in the model with increasing authentication rounds, then the rank of A' constructed by an attacker will increase. Finally, the rank of A' will be the upper-bound value n^2. The solutions of this linear system include the general solution of $A'x = 0$ and a particular solution of $A'x = b$. If the matrix A' is full column rank, then the number of solution will be zero or one solution. Moreover, we know that there must be a solution that is exactly the matrix A in the linear system. The attacker can rule out the case of no solution and solve the linear system. As a result, the attacker can solve the matrix A easily after several rounds of authentication. □

In the authentication model based on matrix multiplication, we refresh the matrix A in every rounds of authentication. For an attacker, he/she needs to obtain A at one specific time by observing one CRP^A. However, to obtain the matrix A by observing one CRP^A results in a linear system of infinitely many solutions.

Lemma 2. *If we refresh the matrix A for every authentication rounds, then A will be infinitely many solutions for the linear system constructed by an attacker.*

Proof. Because we refresh the matrix A for every rounds of authentication, the linear system constructed by the attacker actually results different matrices in every rounds of authentication. Moreover, these matrices are mutually independent, so the attacker can only resolve the matrix A at ending of one specific round by constructing the linear system using one CRP. Suppose that the number of elements in the matrix $\hat{A}$ is $n \times n^2$. Holding $rank(\hat{A}) \leq n < n^2$, the number of elements in null space of the matrix $\hat{A}$ is $n^2 - rank(\hat{A}) > 0$. The solutions of linear system are the general solution of $\hat{A}x = 0$ and one particular solution of $\hat{A}x = \hat{b}$. This linear system must have one solution that is exactly the matrix A, thus, the attacker can rule out the scenario of no solution. Consequently, the linear system $\hat{A}x = \hat{b}$ constructed by the attacker has infinitely many solutions. □

Briefly, it is hard to model the matrix A or $\mathcal{P}^D$ using the recorded information. However, an attacker may try to use brute-force attacks against the matrix A or the $\mathcal{P}^D$. Therefore, we analyze brute-force attacks to the $\mathcal{P}^D$ and the matrix A in the following sections.

6.2.3. Brute-Force Attacks to $\mathcal{P}^D$

As proven in Section 6.2.1, an attacker cannot model the $\mathcal{P}^D$ effectively using the recorded information. However, the attacker can blindly guess the configuration bits of the $\mathcal{P}^D$. In other words, the attacker can try to duplicate the $\mathcal{P}^D$ using brute-force methods. Given $l \cdot i$ LUTs in the $\mathcal{P}^D$, we need to use $16 \cdot l \cdot i$ bits to model the $\mathcal{P}^D$ in the initialize phase of proposed model. As a result, the

attacker will take exponentially time complexity $O(2^{16 \cdot l \cdot i})$ to compromise the $\mathcal{P}^D$. In contrast, $\mathcal{D}_{cp}$ of our model only costs linear time complexity when the $\mathcal{P}^D$ and the matrix A are known in system.

We can easily create huge computational gap between our proposed model and the knowledge of an attacker using brute-force attacks to model $\mathcal{P}^D$. Even if the attacker can duplicate the $\mathcal{P}^D$, he/she still needs to compromise the matrix A to break our model. Therefore, we analyze the system security when an attacker exploits brute-force attacks to compromise the matrix A in the next section.

6.2.4. Brute-Force Attacks to the Matrix A of Matrix Multiplication Authentication Model

With performing man-in-the-meddle attacks, an attacker needs to find one solution among infinitely many solutions of a linear system to model the matrix A. However, the attacker can only blindly guess elements in the matrix A. In other words, the attacker can only try to duplicate the matrix A using brute-force methods.

Let the number of elements in the matrix A be $n \times n$ and each element in the matrix A be k bits. The computational complexity to model the matrix A is exponential complexity $O(2^{k \cdot n^2})$. In contrast, the complexity of $\mathcal{D}_{cp}$ is linear with $n \cdot k \cdot A_{DD} + l \cdot LUT_d$.

After discussing the brute-force attacks against $\mathcal{P}^D$ and the matrix A, an attacker can also focus on breaching the *response* of the matrix A. Consequently, we analyze brute-force attacks to the *response* of the matrix A in the next section.

6.2.5. Brute-Force Attacks to the *Response* of the Matrix A of Matrix Multiplication Authentication Model

Let the number of elements for the *response* of A be $n \times 1$ and each element in a vector be k bits. To blindly retrieve the correct *response*, an attacker needs to take exponentially time complexity $O(2^{k \cdot n})$.

Subsequently, we analyze the $value$ modeling attacks to stochastic logic authentication model.

6.2.6. The $Value$ Modeling Attacks to Stochastic Logic Authentication Model

With the assumption of the strong PUF described in Section 3, the two *responses* generated in the model are mutually independent. Moreover, within one *response*, all individual bits are mutually independent. As a result, an attacker can model the $value$ calculated in step 5 of stochastic logic authentication model as a random variable, which is the transformation of i independent binomial trials with probability of success equals to 0.5, where the transformation is the sum of these independent binomial trials. After transformation, $value$ is a random variable with binomial distribution $(i, 0.5)$. The probability mass function (PMF) of binomial distribution is $\binom{i}{k} \cdot 0.5^k \cdot 0.5^{i-k}$, where k is the number of success, namely the $value$. The mean of $value$ is $0.5i$ and for binomial distribution, the $P(value = 0.5i)$ has the largest probability compared to other possible points. For an attacker, he/she will always guess the $value$ with largest probability. In other words, the attacker should always guess the $value$ to be $0.5i$. For instance, given $i = 2^{10}$, $P(value = 2^9) = \binom{2^{10}}{2^9} \cdot 0.5^{2^9} \cdot 0.5^{2^9} \approx 0.0249$. Obviously, it is relatively easy for an attacker to break the authentication system with probability 0.025. When the size of $\mathcal{P}^D$ arises, $P(value = 0.5i)$ will decrease. When i is large, it is hard to compute the PMF of binomial distribution. However, we can approach to $value$ by using normal distribution, in which the mean is $0.5i$ and variance is $0.25i$. Given $i = 10^8$, we can approach to $value$ using a random variable with $N(\frac{10^8}{2}, \frac{10^8}{4})$. Using continuity correction, $P(value = 0.5i) = P(value = \frac{10^8}{2}) \approx P(49999999.5 < X < 50000000.5)$, where X is a random variable with $N(\frac{10^8}{2}, \frac{10^8}{4})$. After standardization, $P(value = 0.5i) = P(value = \frac{10^8}{2}) \approx P(-0.0001 < Z < 0.0001) = 0.00008$, where Z is a random variable with $N(0, 1)$.

As the aforementioned descriptions, the cell area of matrix multiplication authentication model entails polynomial growth with respect to the size of $\mathcal{P}^D$. In contrast, when i increases, authentication model based on stochastic logic poses slightly increasing on the size of cell area.

6.2.7. Resistance against Machine-Learning Attacks

The resistance against machine-learning attacks is a characteristic of a PUF representing the difficulty to predict its CRPs using machine-learning techniques [16]. Our approach STT-DPSA used the strong STT-MRAM PUF, which means that the number of CRPs is exponentially in accordance with the number of components whose behavior depends on manufacturing variation of MTJs. The variation of MTJs has been proven by [32,33], in which response cannot be predicted better than random guesses; that is, inter-Hamming distance being equal to 0 and uniformity of response being equal to 0.5. As a result, STT-DPSA can resist against machine-learning attacks.

7. Evaluation

To prove the uniqueness of $\mathcal{P}^S$, we demonstrated the simulation results of $\mathcal{P}^S$ under different temperatures. In addition, we demonstrated synthetic results of delay of critical path and cell area with different parameters of STT-DPSA based on ASIC cell-based design flow.

The standard cell library used to synthesize the verilog code of proposed model is NanGate FreePDK45 open cell library [34]. Each authentication round in our model STT-DPSA takes 5 clock cycles. In addition, we implemented LFSR [12] as a PRNG in STT-DPSA.

7.1. Unpredictability Validation of $\mathcal{P}^S$

Our STT-MRAM PUF $\mathcal{P}^S$ circuit was implemented in SPICE model [35] via Monte Carlo simulation with three-dimensional variation of the MTJ shape. The simulation results are shown in Figure 11. The upper sub-figure is the waveform of signal RCLK and the others are waveforms of signal BIT under temperatures in unit of Celsius: 0, 50, and 80 from top to down. Initially, both MTJ A and MTJ B are in the *P* state. We set signal RCLK to Vdd so that BIT is 0. Then we switch signal RCLK to 0, followed by BIT will be pre-charged to the undetermined level. Finally, BIT changes to 0/1 with the corresponding MTJ resistance difference between MTJ A and MTJ B. In Figure 11, the simulation results of signal BIT diverging at different time slots demonstrate that our $\mathcal{P}^S$ is unpredictable under different temperatures.

As exhibited in Figure 11, *response* bits of $\mathcal{P}^S$ can be generated within around 30 nanoseconds (ns). Using these *response* bits, the construction of $\mathcal{P}^D$ can be completed by executing ASIC design flow within limited amount of time. Therefore, with the designs of $\mathcal{P}^D$ and $\mathcal{P}^S$, we can realize the initialization phase of STT-DPSA with constant time complexity rather than the traditional PUF-based models that need unrealistic amount of time to generate CRPs of a strong PUF.

7.2. Reliability Validation of $\mathcal{P}^S$

To demonstrate the reliability of $\mathcal{P}^S$, we validated whether our $\mathcal{P}^S$ was unchangeable under variations of temperature and Vdd. In Figure 12, the results revealed that the working windows between temperature interval (-35 °C, 110 °C) and Vdd interval (0.95 V, 1.16 V) for $\mathcal{P}^S$ are reliable.

7.3. Effect of the Parameter l

We varied the length l of $\mathcal{P}^D$ when fixed the length $i = 64$ of input bits to the $\mathcal{P}^D$ and let k be equal to 8. Figures 13 and 14 demonstrated the delay of critical path and the cell area, respectively, in which l posed linear growth on the delay of critical path and the cell area. Obviously, the delay of critical path and the cell area are about 8 ns and 2.17 um*um, respectively, for both of practical and theoretical evaluations at $l = 4$.

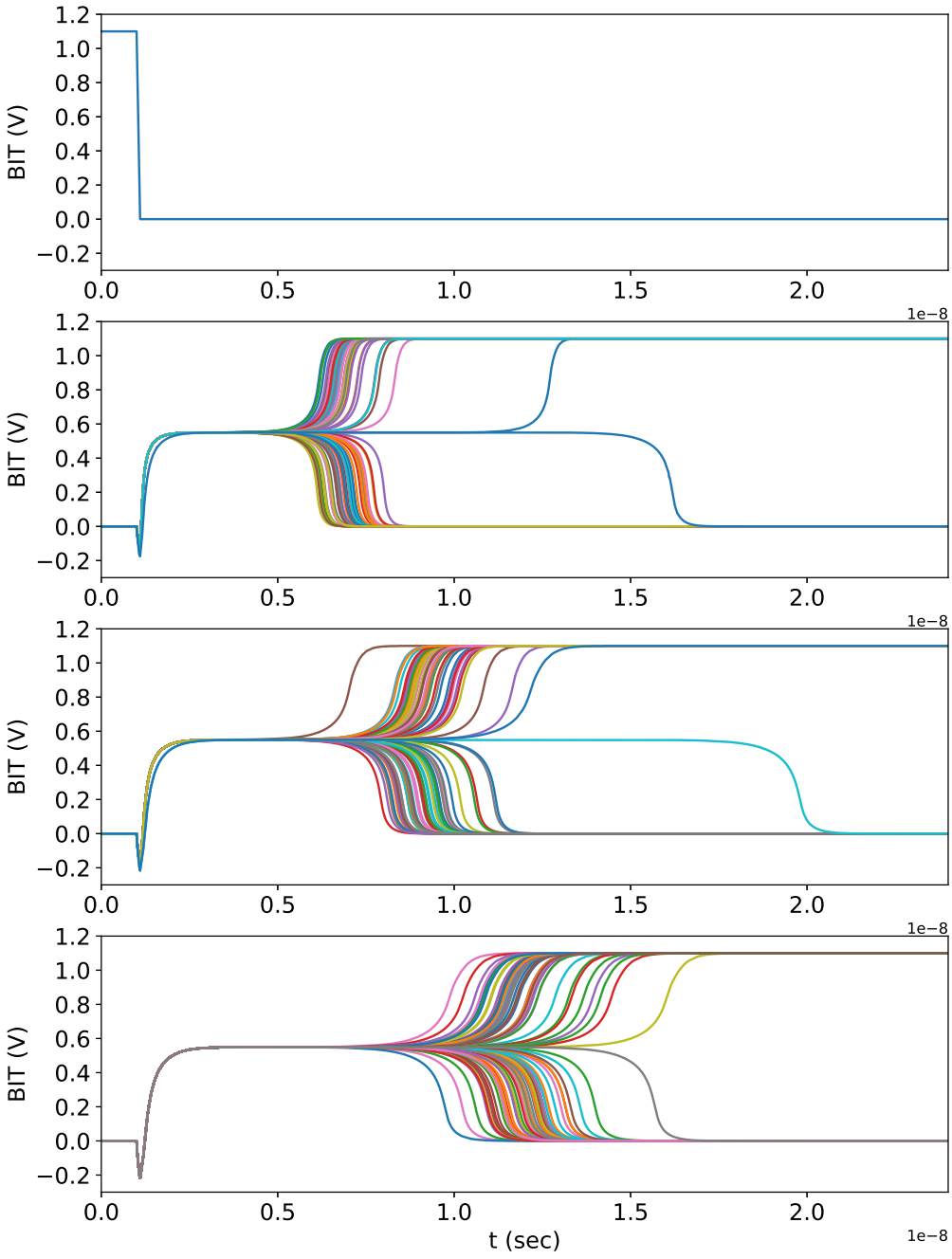
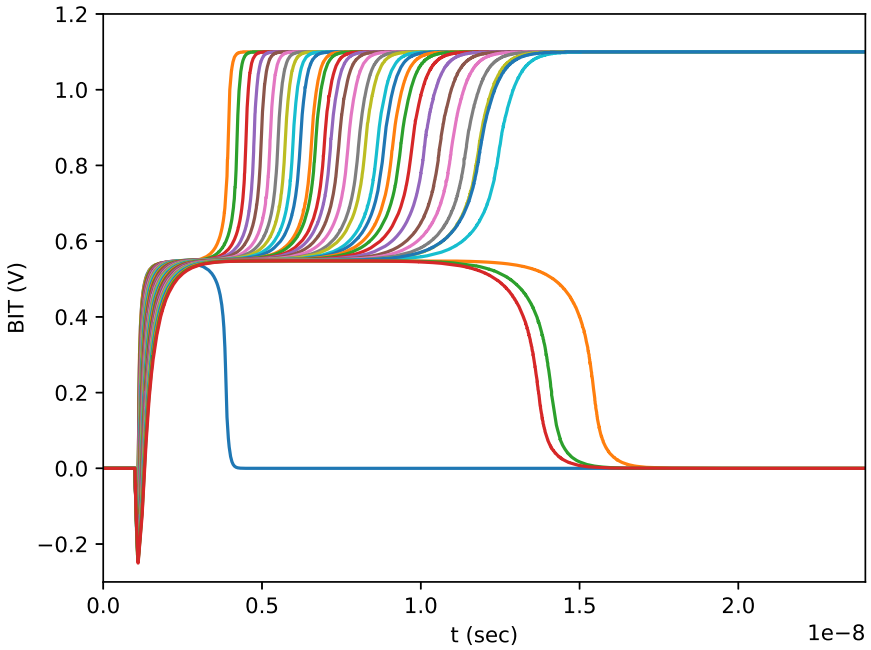
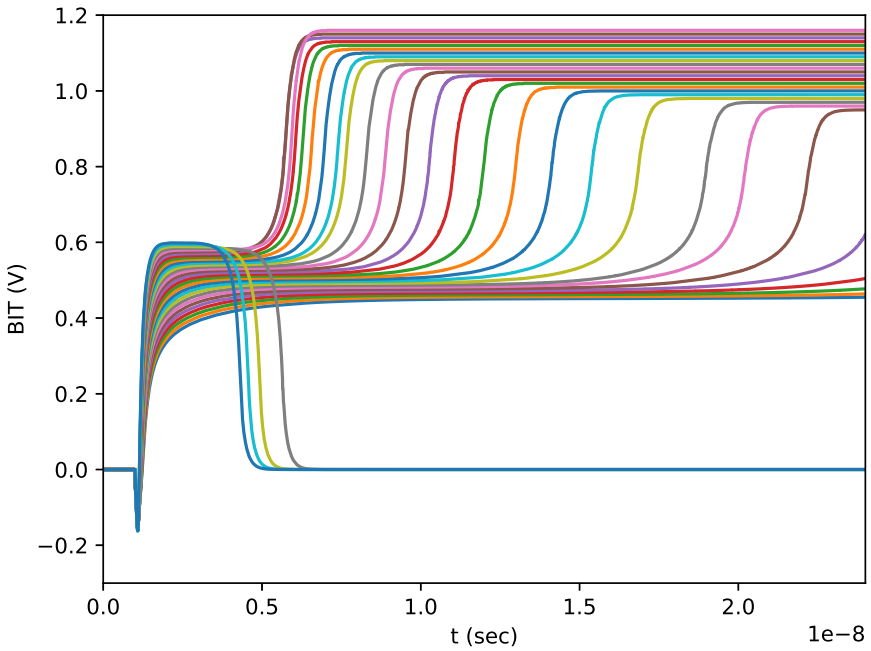


Figure 11. The uniqueness validation of $\mathcal{P}^S$.



(**a**) Temperatures range from −40 °C to 125 °C, the expected result is BIT = 1 (Vdd = 1.1 V), while it fails at −40 °C (blue line), 115 °C (orange line), 120 °C (green line), and 125 °C (red line).



(**b**) Vdds range from 0.9V to 1.2V at 25 °C, the expected result is BIT = 1 (Vdd), while it fails when Vdd > 1.16 V (four lines BIT = 0), and it cannot reach BIT = 1 on time when Vdd < 0.95 V because RCLK pulse ends at 25 ns.

Figure 12. Reliability validation of $\mathcal{P}^S$ under two MTJs with a bit resistance-area product (RA) difference, MTJ A: 5 Ω-μm^2 and MTJ B: 5.2 Ω-μm^2.

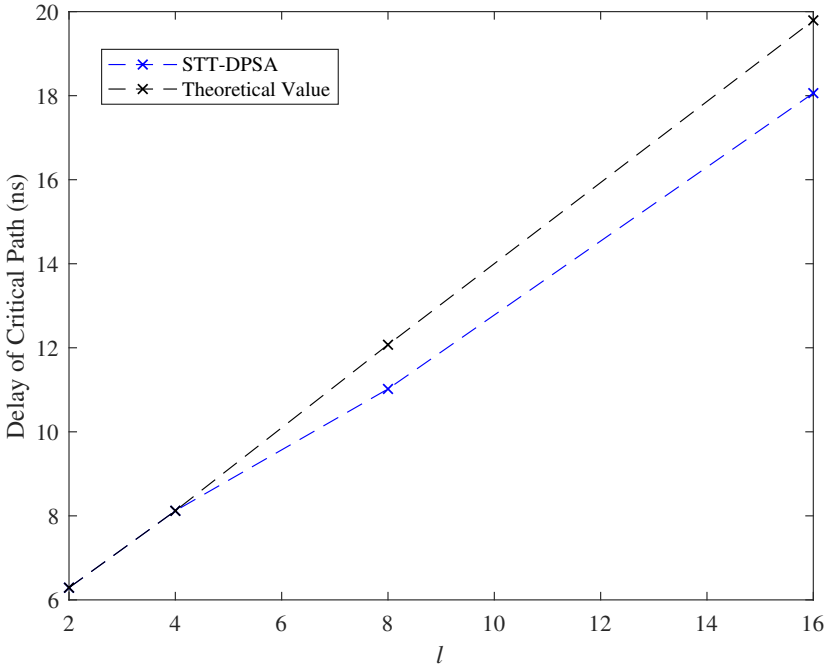


Figure 13. Effect of the parameter l with respect to the delay of critical path.

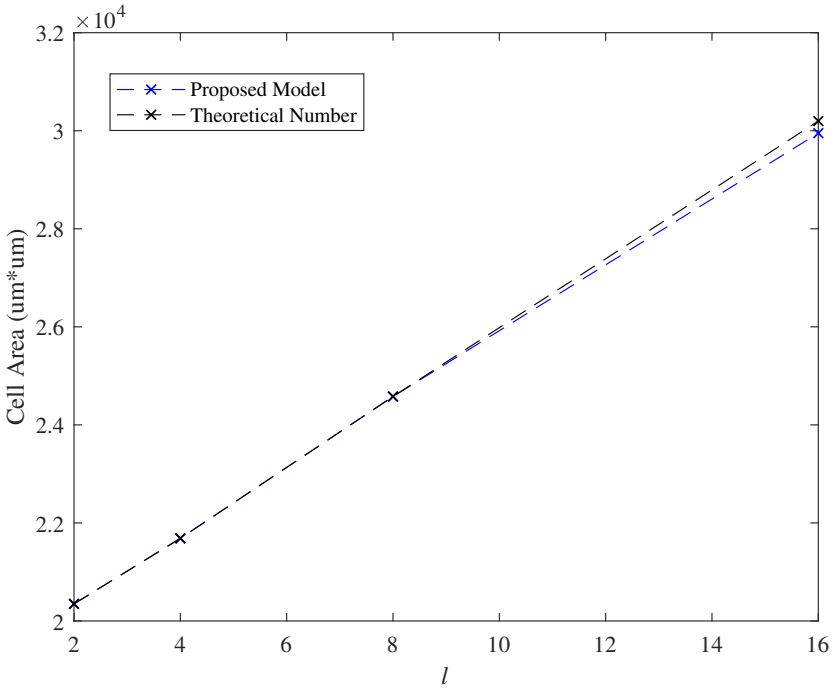


Figure 14. Effect of the parameter l with respect to the cell area.

7.4. Effect of the Parameter n

We exhibited the effect of the parameter n on the delay of critical path and the cell area as $l = 2$ and $k = 8$ in Figures 15 and 16. Because we suppose that the analysis presented in Theorem 2 is the naivest hardware architecture, different hardware architectures in practice would affect the evaluated results. In our hardware platform, the evaluated delay of critical path and cell area are lower than the theoretical estimation. Moreover, Figures 15 and 16 indicated that our model posed linearly growth

overhead on delay of critical path and posed polynomially growth overhead on cell area while varying the parameter n.

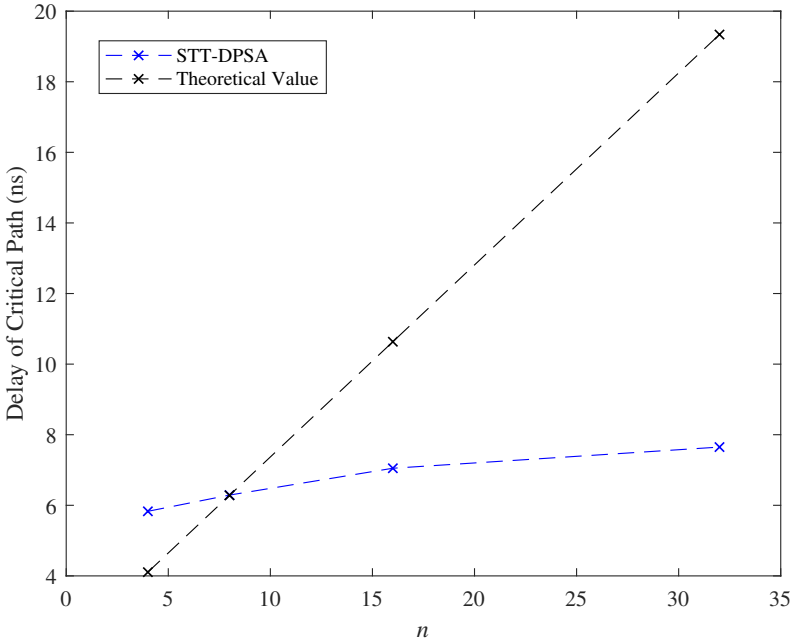


Figure 15. Effect of the parameter n with respect to the delay of critical path.

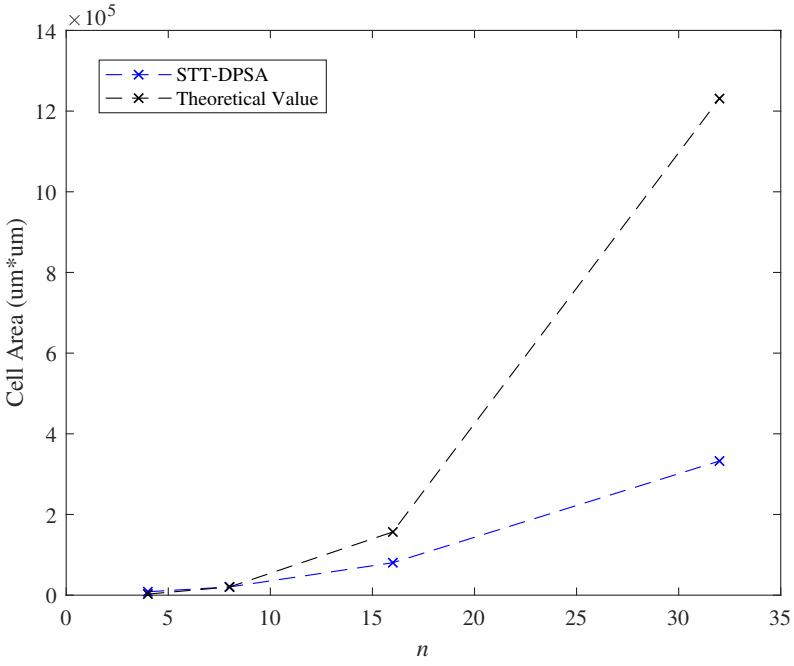


Figure 16. Effect of the parameter n with respect to the cell area.

7.5. Effect of the Parameter k

In this section, effects of the parameter k on the delay of critical path and the cell area as $l = 2$ are shown in Figures 17 and 18. Observing from Figures 17 and 18, the parameter k posed linearly growth on the delay of critical path and the cell area. At $k = 16$, the evaluated delay of critical path and cell area are about 8.5 ns and 3.5 um*um, respectively.

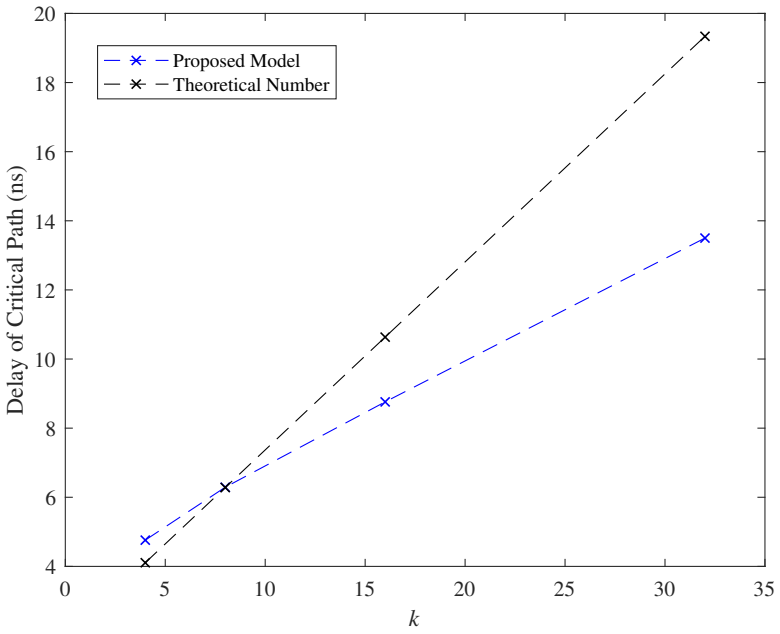


Figure 17. Effect of the parameter k with respect to the delay of critical path.

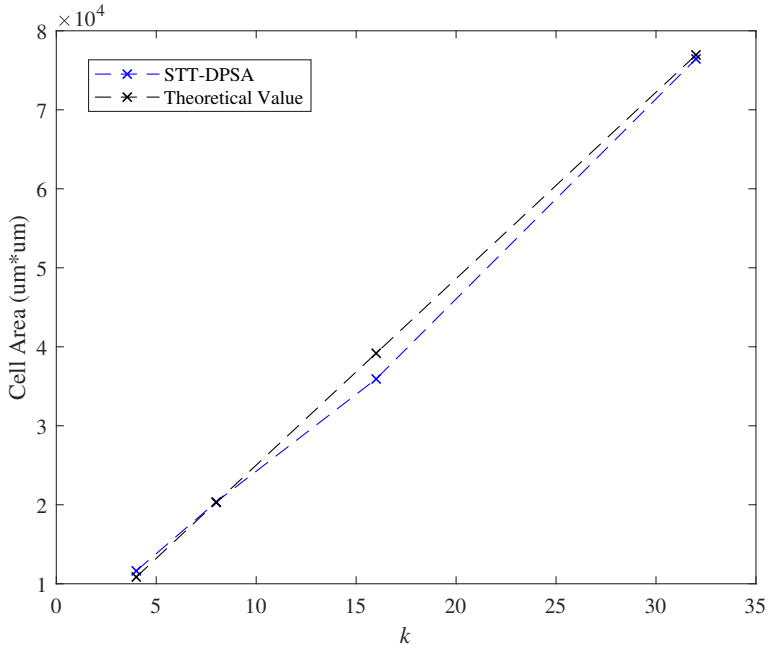


Figure 18. Effect of the parameter k with respect to the cell area.

7.6. Execution Time of Authentication

We evaluated the execution time of authenticating a client in our proposed model compared to a brute-force method launched by an attacker while varying the parameters k and n. The evaluated results are shown in Figures 19 and 20. As previous discussions, the most effective attack against our proposed model is to blindly guess the *response* of the matrix A. Consequently, the security of proposed authentication model is $O(2^{k \cdot n})$. As a result, we can create giant time execution gap between the brute-force method and our model. For instance, the attacker will take about computational complexity $O(2^{256})$ to compromise STT-DPSA under the settings of $n = 32$, $k = 8$, $i = o = 1024$. This security level is equivalent to the AES-256 cryptography system.

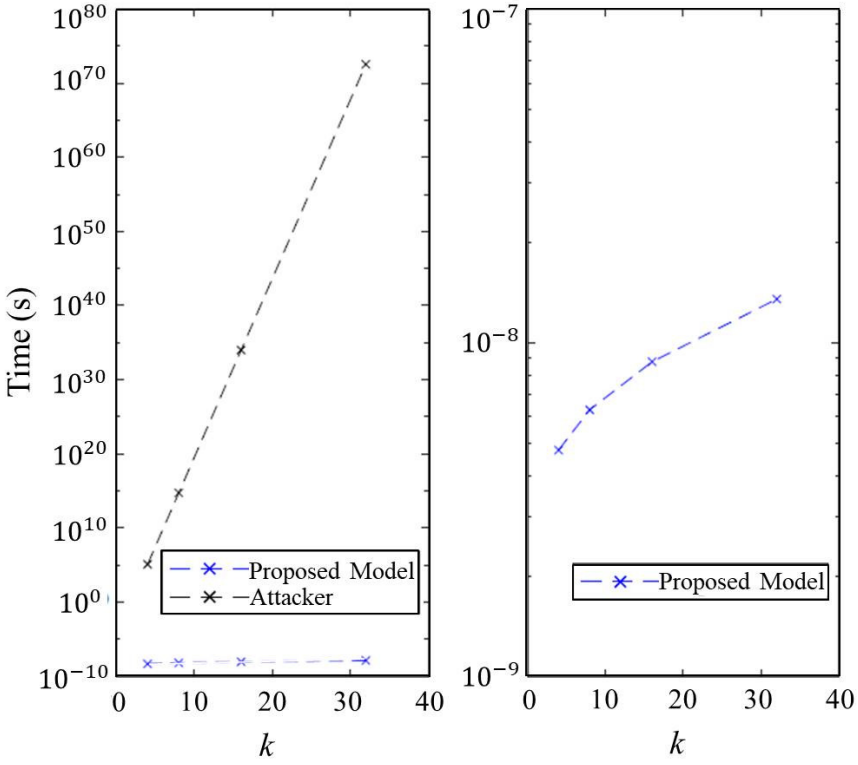


Figure 19. Execution time of an attacker and STT-DPSA to do authentication under $n = 8$ and $l = 2$ while varying the parameter k.

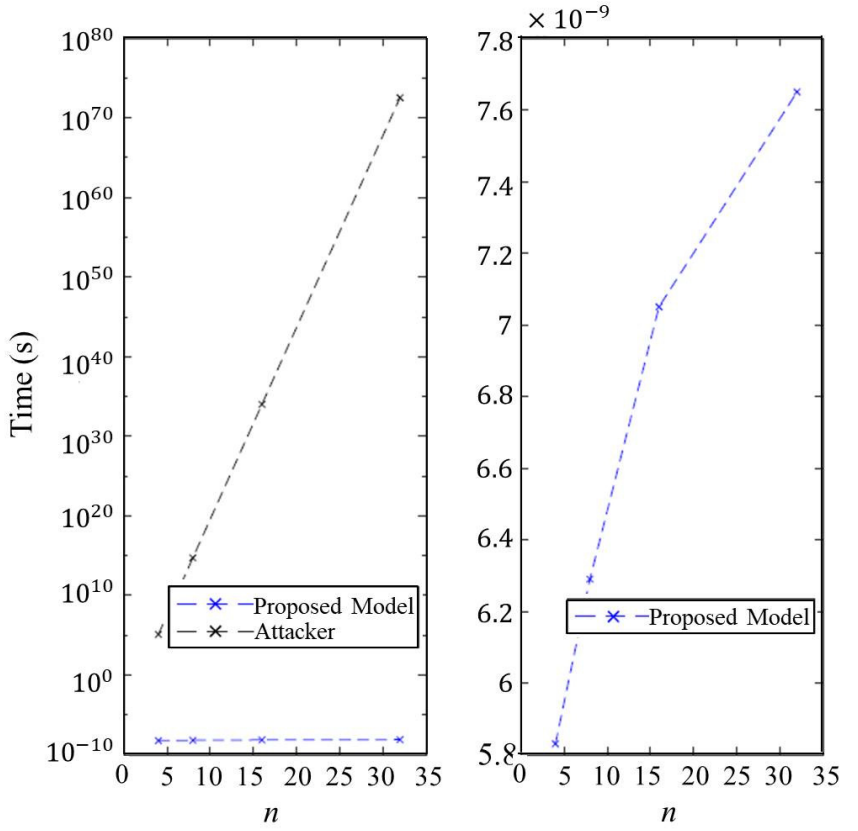


Figure 20. Execution time of an attacker and STT-DPSA to do authentication under $k = 8$ and $l = 2$ while varying the parameter n.

7.7. Evaluation of Stochastic Logic Authentication Model

In this section, we set the size of $\mathcal{P}^D$ to $i = o = 64$ and indicated the parameters n, k, and l to 8, 8, and 2, respectively. Under this assumption, we used theoretical equation derived in Section 1 to estimate the cell area and the delay of critical path for matrix multiplication and stochastic logic authentication models. Table 2 demonstrated the comparison results. Considering hardware security level or cell area, we alternatively build matrix multiplication or stochastic logic operation for our authentication model. In other words, if we would like to reserve high security for authentication, matrix multiplication model can be used while sacrificing area overhead and delay of critical path; otherwise, stochastic logic model can be used to improve performance of cell area and delay of critical path for STT-DPSA.

Table 2. Comparison results between stochastic logic version and matrix multiplication version.

	Cell Area (μm*μm)	Delay of Critical Path (ns)
Matrix Multiplication Version	20,349	31.45
Stochastic Logic Version	4953	17.25

7.8. Comparisons

STT-DPSA was compared with DBF [14], DBF plus Arbiter [15], PUFSec [24], and Matched PUF [13] in driving device authentication, as shown in Table 3. In our evaluations, STT-DPSA and Matched PUF used pure hardware approach while DBF, DBF plus Arbiter, and PUFSec used hardware-software co-design methodology. For fairly comparison, we set the security level at $O(2^{64})$ for each method and compared them in terms of the cell area, execution time to finish one authentication round, and memory usages. Notably, our evaluation was under the ASIC cell-based design flow using NanGate FreePDK45 open cell library [34].

STT-DPSA outperforms DBF, DBF plus Arbiter and PUFSec in terms of execution time and memory usage while sacrificing cell area. STT-DPSA only spent 31.45 ns for performing one round of authentication while DBF, DBF plus Arbiter and PUFSec cost 0.03, 0.03 and 5.15×10^{-10} in terms of seconds, respectively. For the memory usage, DBF and DBF plus Arbiter and PUFSec all cost unrealistic memory usage while STT-DPSA did not consume memory usages due to its pure hardware implementation. Comparing STT-DPSA with Matched PUF, Matched PUF outperforms STT-DPSA in terms of cell area, execution time but it is extremely susceptible to environment condition.

Table 3. Comparison results.

Method	Cell Area (μm*μm)	Execution Time (s)	Memory Usage (bits)
STT-DPSA	20,349	31.45×10^{-9}	0
DBF [14]	835	0.03	2^{64}
DBF with Arbiter PUF [15]	930	0.03	2^{64}
PUFSec [24]	2245	5.15×10^{-10}	2^{56}
Matched PUF [13]	5376	5×10^{-9}	0

7.9. Implementation of STT-DPSA

Finally, STT-DPSA was implemented with parameters $n = 32$, $i = o = 1024$, $k = 8$, and $l = 2$ using FPGA design flow. The FPGA device that we used is EP4CE115F29C7, which belongs to Cyclone IV E FPGA family of Intel. Under this setting of parameters, an attacker needs to take time complexity $O(2^{256})$ to compromise STT-DPSA. We also evaluated STT-DPSA using Synopsys design compiler with TSMC 0.18 um process. Table 4 is the statistics of STT-DPSA implemented on the FPGA device

and Table 5 is the specification of the TSMC 0.18 um process implementation for STT-DPSA. They demonstrated the feasibility of STT-DPSA.

Table 4. Statistics of the FPGA implementation.

Parameter Settings	$n = 32$, $i = o = 1024$, $k = 8$, and $l = 2$
FPGA Device	Intel Cyclone IV E EP4CE115F29C7
Total Logic Elements	68108/114480 (59%)
Total Registers	3080
Clock Frequency	10^7 Hz

Table 5. Specification of the TSMC 0.18 um process implementation for STT-DPSA.

Supply Voltage (V)	**Cell Area (um^2)**	**Power @ 50 MHz (mW)**
1.8	20,349	1.2278

8. Conclusions

We presented a novel authentication model based on a strong digital PUF for IoT devices. In our implementation, we generated a strong digital PUF using STT-MRAM PUF to configure LUTs in the strong digital PUF. By deploying the strong digital PUF configured by STT-MRAM PUF on the client and verifier, the execution time of doing one authentication round and the cell area is efficient. At the same time, our proposed authentication model can resilient against a series of man-in-the-middle attacks and brute-force attacks. The security of STT-DPSA was analyzed in the theoretical estimation. Moreover, we validated the cell area, the delay of critical path, and the execution time of one authentication round in experiments. Finally, we realized our proposed model using FPGA design flow to prove the feasibility of STT-DPSA.

Author Contributions: Conceptualization, W.-C.C., Y.-C.C., and Y.-T.T.; methodology, W.-C.C., Y.-C.C., and Y.-T.T.; validation, W.-C.C. and Y.-C.C.; formal analysis, Y.-C.C. and Y.-T.T.; investigation, W.-C.C., Y.-C.C., and Y.-T.T.; writing—original draft preparation, W.-C.C. and Y.-C.C.; writing—review and editing, Y.-T.T., S.-Y.K., and C.-R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Science and Technology, Taiwan, under Grant No. MOST 107-2221-E-035-020-MY3 and No. MOST 106-2622-8-002-011-TA.

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

References

1. Suh, G.E.; Devadas, S. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In Proceedings of the 44th ACM/IEEE Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 9–14.
2. Gassend, B.; Clarke, D.; van Dijk, M.; Devadas, S. Silicon Physical Random Functions. In Proceedings of the 9th ACM Conference Computer and Communication Security, Washington, DC, USA, 18–22 November 2002; pp. 148–160.
3. Herder, C.; Yu, M.; Koushanfar, F.; Devadas, S. Physical Unclonable Functions and Applications: A Tutorial. *Proc. IEEE* **2014**, *102*, 1126–1141. [CrossRef]
4. Perrig, A.; Szewczyk, R.; Tygar, J.D.; Wen, V.; Culler, D.E. SPINS: Security Protocols for Sensor Networks. *Wirel. Netw.* **2002**, *8*, 521–534. [CrossRef]
5. Karlof, C.; Sastry, N.; Wagner, D. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In Proceedings of the ACM 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, MD, USA, 3–5 November 2004; pp. 162–175.
6. Zhu, S.; Setia, S.; Jajodia, S. LEAP+: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. *ACM Trans. Sens. Netw.* **2006**, *2*, 500–528. [CrossRef]

7. Tsou, Y.-T.; Lu, C.-S.; Kuo, S.-Y. SER: Secure and Efficient Retrieval for Anonymous Range Query in Wireless Sensor Networks. *Comput. Commun.* **2017**, *108*, 1–16. [CrossRef]
8. Liu, A.; Ning, P. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks, St. Louis, MO, USA, 22–24 April 2008; pp. 245–256.
9. Oliveira, L.B.; Aranha, D.F.; Gouvêa, C.P.; Scott, M.; Câmara, D.F.; López, J.; Dahab, R. TinyPBC: Pairings for Authenticated Identitybased Non-Interactive Key Distribution in Sensor Networks. *Comput. Commun.* **2011**, *34*, 485–493. [CrossRef]
10. Li, S.; Xu, H.; Li, M.; Zhou, X. A Group-Based NTRU-Like Public-Key Cryptosystem for IoT. *IEEE Access* **2019**, *7*, 75732–75740.
11. Layman, P.; Chaudhry, S.; Norman, J.; Thomson, J. Electronic Fingerprinting of Semiconductor Integrated Circuits. U.S. Patent 6 738 294, 30 September 2002.
12. Weste, N.; Harris, D. *Integrated Circuit Design*, 4th ed.; Pearson Education: London, UK, 2011; pp. 377–424.
13. Meguerdichian, S.; Potkonjak, M. Matched Public PUF: Ultra Low Energy Security Platform. In Proceedings of the IEEE/ACM International Symposium on Low-Power Electronics and Design, Fukuoka, Japan, 1–3 August 2011; pp. 45–50.
14. Xu, T.; Wendt, J.; Potkonjak, M. Digital Bimodal Function: An Ultra-Low Energy Security Primitive. In Proceedings of the IEEE Symposium on Low Power Electronics and Design, Beijing, China, 4–6 September 2013; pp. 292–296.
15. Xu, T.; Potkonjak, M. Robust and Flexible FPGA-based Digital PUF. In Proceedings of the IEEE International Conference on Field Programmable Logic and Applications, Munich, Germany, 2–4 September 2014; pp. 1–6.
16. Tanaka, Y.; Bian, S.; Hiromoto, M.; Sato, T. Coin Flipping PUF: A Novel PUF With Improved Resistance Against Machine Learning Attacks. *IEEE Trans. Circuits Syst. II Express Br.* **2018**, *65*, 602–606. [CrossRef]
17. Yu, W.; Wen, Y.; Köse, S.; Chen, J. Exploiting Multi-Phase On-Chip Voltage Regulators as Strong PUF Primitives for Securing IoT. *J. Electron. Test. Theory Appl. (Springer)* **2018**, *34*, 587–598. [CrossRef]
18. Rahman, M.; Hosey, A.; Guo, Z.; Carroll, J.; Forte, D.; Tehranipoor, M. Systematic correlation and cell neighborhood analysis of SRAM PUF for robust and unique key generation. *J. Hardw. Syst. Secur.* **2017**, *1*, 137–155. [CrossRef]
19. Yu, W.; Chen, J. Masked AES PUF: a new PUF against hybrid SCA/MLAs. *IET Electron. Lett.* **2018**, *54*, 618–620. [CrossRef]
20. Govindaraj, R.; Ghosh, S.; Katkoori, S. Design, Analysis and Application of Embedded Resistive RAM based Strong Arbiter PUF. *IEEE Trans. Dependable Secure Comput.* **2018**. [CrossRef]
21. Xu, T.; Potkonjak, M. Digital PUF using Intentional Faults. In Proceedings of the IEEE International Symposium on Quality Electronic Design, Santa Clara, CA, USA, 2–4 March 2015; pp. 448–451.
22. Miao, J.; Li, M.; Roy, S.; Ma, Y.; Yu, B. SD-PUF: Spliced Digital Physical Unclonable Function. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 927–940. [CrossRef]
23. Beckmann, N.; Potkonjak, M. Hardware-Based Public-Key Cryptography with Public Physically Unclonable Functions. In *International Workshop on Information Hiding*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 206–220.
24. Park, S.; Lim, S.; Jeong, D.; Lee, J.; Yang, J.; Lee, H. PUFSec: Device Fingerprint-based Security Architecture for Internet of Things. In Proceedings of the IEEE Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017; pp. 1–9.
25. Cao, Y.; Guo, Y.; Liu, B.; Ge, W.; Zhu, M.; Chang, C. A Fully Digital Physical Unclonable Function Based Temperature Sensor for Secure Remote Sensing. In Proceedings of the 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, China, 30 July–2 August 2018; pp. 1–8.
26. Gu, T.; Mohapatra, P. BF-IoT: Securing the IoT Networks via Fingerprinting-Based Device Authentication. In Proceedings of the IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Chengdu, China, 9–12 October 2018; pp. 254–262.
27. Das, J.; Scott, K.; Bhanja, S. MRAM PUF: Using Geometric and Resistive Variations in MRAM Cells. *ACM J. Emerg. Technol. Comput. Syst. (JETC)* **2016**, *13*, 1–15. [CrossRef]
28. Maiti, A.; Gunreddy, V.; Schaumont, P. A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. In *Embedded Systems Design with FPGAs*; Springer: New York, NY, USA, 2012; pp. 245–267.

29. Alaghi, A.; Qian, W.; Hayes, J.P. The Promise and Challenge of Stochastic Computing. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 1515–1531. [CrossRef]
30. Alaghi, A.; Li, C.; Hayes, J.P. Stochastic Circuits for Real-Time Image-Processing Applications. In Proceedings of the 50th ACM/IEEE Design Automation Conference, Austin, TX, USA, 29 May–7 June 2013; pp. 1–6.
31. Rührmair, U.; Sehnke, F.; Sölter, J.; Dror, G.; Devadas, S.; Schmidhuber, J. Modeling Attacks on Physical Unclonable Functions. In Proceedings of the ACM Conference on Computer and Communications Security, Chicago IL, USA, 4–8 October 2010; pp. 237–249.
32. Tsou, Y.-T.; Zhen, H.; Kuo, S.-Y.; Chang, C.-R.; Fukushima, A.; Rong, B.-D. SPARR: Spintronics-based Private Aggregatable Randomized Response for Crowdsourced Data Collection and Analysis. *Comput. Commun.* **2020**, *152*, 8–18. [CrossRef]
33. Fukushima, A.; Seki, T.; Yakushiji, K.; Kubota, H.; Imamura, H.; Yuasa, S.; Ando, K. Spindice: A Scalable Truly Random Number Generator Based on Spintronics. *J. Appl. Phys. Express* **2014**, *7*, 083001. [CrossRef]
34. NanGate FreePDK45 Open Cell Library. Available online: http://projects.si2.org/openeda.si2.org/projects/nangatelib (accessed on 10 February 2020).
35. UMN MTJ SPICE Model. Available online: http://mtj.umn.edu (accessed on 10 February 2020).