Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.061069

ARTICLE

Root Security Parameter Generation Mechanism Based on SRAM PUF for Smart Terminals in Power IoT

Xiao Feng[1,2,3,*], **Xiao Liao**[1,3], **Xiaokang Lin**[1,3] **and Yonggui Wang**[1,3]

[1]Information and Communication Research Institute, State Grid Information Telecommunication Group Co., Ltd., Beijing, 102211, China

[2]School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, 100876, China

[3]Smart IoT Business Unit, State Grid Info-Telecom Great Power Science and Technology Co., Ltd., Xiamen, 361008, China

*Corresponding Author: Xiao Feng. Email: wche@bupt.cn

Received: 16 November 2024; Accepted: 07 February 2025; Published: 26 March 2025

ABSTRACT: In the context of the diversity of smart terminals, the unity of the root of trust becomes complicated, which not only affects the efficiency of trust propagation, but also poses a challenge to the security of the whole system. In particular, the solidification of the root of trust in non-volatile memory (NVM) restricts the system's dynamic updating capability, which is an obvious disadvantage in a rapidly changing security environment. To address this issue, this study proposes a novel approach to generate root security parameters using static random access memory (SRAM) physical unclonable functions (PUFs). SRAM PUFs, as a security primitive, show great potential in lightweight security solutions due to their inherent physical properties, low cost and scalability. However, the stability of SRAM PUFs in harsh environments is a key issue. These environmental conditions include extreme temperatures, high humidity, and strong electromagnetic radiation, all of which can affect the performance of SRAM PUFs. In order to ensure the stability of root safety parameters under these conditions, this study proposes an integrated approach that covers not only the acquisition of entropy sources, but also the implementation of algorithms and configuration management. In addition, this study develops a series of reliability-enhancing algorithms, including adaptive parameter selection, data preprocessing, auxiliary data generation, and error correction, which are essential for improving the performance of SRAM PUFs in harsh environments. Based on these techniques, this study establishes six types of secure parameter generation mechanisms, which not only improve the security of the system, but also enhance its adaptability in variable environments. Through a series of experiments, we verify the effectiveness of the proposed method. Under 10 different environmental conditions, our method is able to achieve full recovery of security data with an error rate of less than 25%, which proves the robustness and reliability of our method. These results not only provide strong evidence for the stability of SRAM PUFs in practical applications, but also provide a new direction for future research in the field of smart terminal security.

KEYWORDS: Root security; parameter generation; PUF; smart terminals

1 Introduction

With the advent of the digital economy era, the emerging power grid as a key infrastructure has shifted its intelligent terminal control from network centralisation to covering multiple links such as power generation, power grid, load and energy storage [1–3]. This trend not only significantly increases the attack surface of the power system, but also exacerbates the security risks, making smart terminals a vulnerable point in the security defence of the new power system [4,5]. In this context, ensuring the security of parameters such as keys, random numbers and trust roots has become crucial, as they are the foundation for protecting the security of terminal data.

SRAM Physical Unclonable Function (PUF), an emerging security technology, provides a lightweight security solution for IoT devices due to its inherent randomness and non-replicability. SRAM PUF uses the randomised initial values of SRAM memory cells as a security feature, which are determined during the chip manufacturing process and are unique to each chip. This hardware-based security feature not only provides strong resistance to attack, but also greatly reduces the need for computing and memory resources by eliminating the need for additional key storage and complex cryptographic algorithms.

The diversity and heterogeneity of intelligent power terminals have increased the difficulty of unifying the trust root, weakened trust propagation, and led to the problem of trust root solidification. These issues have long hindered the establishment of trust between platform terminals, limited the dynamic updating of security parameters, and caused significant management overhead in the transmission and application of security parameters [6–8]. SRAM PUFs use small changes in the manufacturing process to create unique 'fingerprints', ensuring that these physical characteristics cannot be replicated or forged. However, the response of SRAM PUFs is affected by environmental factors such as temperature and voltage, which leads to unstable PUF extraction and inconsistent response reading over multiple reads, resulting in reliability issues. For instance, if a cell has a SUP1 (probability of starting as 1) of 0.99, it has a BER of 0.01; similarly, if SUP0 (probability of starting as 0) is 0.99, the BER is 0.01. By averaging over all the cells, we obtain a BER of 0.045 (4.5%). Furthermore, environmental variations and the effects of ageing have been demonstrated to have an impact on the reliability and uniformity of SRAM PUFs. Random variations in the manufacturing process may cause deviations in the power data generated by the SRAM PUF at startup, which not only reduces uniformity, but also prevents the power data from being duplicated due to device mismatches, thus worsening the reliability of the SRAM PUF. Furthermore, the occurrence of ageing effects, such as bias temperature instability (BTI) and hot carrier injection (HCI), has been demonstrated to modify the characteristics of the SRAM cell, thereby affecting the PUF response. This, in turn, has been shown to lead to a long-term degradation of the response, stability, and reliability of the SRAM PUF. Therefore, there is an urgent need to improve the stability and repeatability of SRAM PUF response [9]. Although SRAM PUFs can provide various secure root parameters, there is still a lack of a comprehensive and scalable root security architecture in practical applications. This deficiency has led to the immaturity and limited use of SRAM PUFs in actual products, coupled with a lack of resource and security constraint analysis, unified testing and evaluation standards, and challenges in directly comparing different SRAM PUF implementation schemes, making it difficult for application developers to evaluate the true performance of PUFs.

Regarding the above issues, this paper proposes a root security parameter generation mechanism based on SRAM PUF, which is especially optimized for resource-constrained IoT devices. We have introduced an innovative architecture covering entropy source acquisition, algorithm implementation and configuration management. We have also proposed a series of algorithms to enhance reliability, such as adaptive parameter selection, data pre-processing, auxiliary data generation, and error correction. On this basis, we have established six types of security parameter generation mechanisms. The experimental results confirm that our proposed method achieves complete recovery of secure data under 10 environmental conditions with an error rate of less than 25%, demonstrating strong robustness and reliability. The specific research content of this paper is as follows:

(1) Targeting diversified and differentiated application scenarios for security parameters, a four-layer security parameter system covering entropy source acquisition, algorithm implementation, configuration management, and functional service layers is designed. A hierarchical and modular SRAM-PUF

SPG functional architecture is established to provide reconfigurable underlying security parameter acquisition services for diverse security applications.

(2) Addressing the high reliability and adaptability requirements of security parameters, a series of algorithms including stable feature search, state data preprocessing, auxiliary data generation, fuzzy extraction, key derivation functions, and adaptive error correction coding algorithms are designed. Each algorithm incorporates adaptive environmental parameters and configurable thresholds to enhance the SRAM-PUF SPG's capability to adapt to complex and changing environments.

(3) Targeting the productization and standardization requirements of power terminals, the paper designs SRAM-PUF SPG-related algorithm invocation processes. It proposes collaborative application mechanisms in the stages of initialization, registration authentication, key generation, and update recovery. By coordinating these algorithms, the paper constructs SRAM-PUF SPG services that cover initialization, authentication, key management, update and recovery processes.

2 Related Work

In 2002, Veiga et al. [10] first proposed SRAM PUF for key generation and identity authentication. Guajardo et al. [11] in 2007 constructed PUF functionality using FPGA SRAM for third-party IP protection. Maes et al. [12] introduced the Soft Decision Help Data Algorithm to reduce key generation overhead in SRAM PUFs. Charles et al. [13] in 2014 classified PUFs into strong and weak types, with strong PUFs used for authentication and supporting many challenge-response pairs (CRPs), while weak PUFs support limited CRPs. Ruhrmair et al. [14] reviewed machine learning algorithms for modeling PUF functions and attack requirements. In 2016, Ye et al. [15] demonstrated machine learning attacks against strong PUFs, allowing cloning through monitored CRPs. Subsequent research by Zhang [16], Vijayakumar et al. [17–19], and others focused on enhancing SRAM PUF resilience [20–22].

Two deployment modes have emerged for SRAM PUF [23,24]: one designates specific SRAM regions for secure memory, while the other integrates SRAM storage with PUF functions to minimize area overhead. However, Negative Bias Temperature Instability (NBTI) can cause unreliable startup values in SRAM used for storage. Although error correcting codes (ECC) can mitigate this issue, they increase area overhead, which is disadvantageous for resource-limited IoT devices. Qiu et al. [25] proposed using PUF response preprocessing to reduce ECC area demands. Thus, the reliability issues of SRAM serving dual purposes as memory and PUF highlight the need for preprocessing techniques to improve the performance of dual-purpose SRAM PUF systems.

The above work did not take into account the environmental challenges faced by SRAM PUFs in practical applications, such as extreme temperatures, high humidity, and strong electromagnetic radiation. These environmental factors severely impact the stability and reliability of SRAM PUFs, resulting in existing research failing to effectively address this bottleneck issue. In response, a methodology has been proposed in the literature [26] to improve the reliability of SRAM PUFs by subjecting them to different operating conditions and aging degradation, which is particularly important for IoT devices as they often operate in changing environments. Furthermore, in the literature [27], SRAM PUFs are enhanced by introducing XOR gates to improve their stability and reliability under different environmental conditions. Additionally, although their work has made progress in protecting IP, there is still a lack of in-depth research on how to achieve high reliability SRAM PUFs on resource-constrained IoT devices.

3 SRAM PUF Security Parameter Generation Model

To develop a security parameter generation mechanism, this paper concentrates on designing the functional architecture for generating security parameters in smart power terminals. It examines hardware technologies and environmental requirements by using integrated smart terminals in distribution substations as a case study. The paper proposes parameter extraction schemes tailored to diverse security needs and evaluates the reliability, stability, and other characteristics of the extracted parameters. Additionally, it provides both qualitative and quantitative assessments of the effectiveness of these parameters in practical applications.

3.1 SRAM PUF Security Parameter Generation Function

Based on the metering and acquisition board MCU, this paper designs the functional framework for SRAM PUF to generate security parameters. According to the properties of SRAM PUF, it generates six categories of security parameters: key derivation seeds, authentication CRPs, true random numbers, trusted measurement keys, root keys, and terminal unique identifiers. These parameters serve as foundational security elements for functions such as terminal identity authentication, trusted computing, key negotiation and authentication, and data encryption and decryption. The specific definitions of system parameters are shown in Table 1.

Table 1: System parameters

Parameters	Value
N	Number of power on cycles.
M	Challenge bits in stable bits
Ω_k	PUF measurement matrices
R	Power-ons
T	Temperature
V	Voltage
I	Current
E	Electromagnetic radiation environmental condition
k	Expansion factor
S	Comprehensive vector
ϕ	Original response sequence
l_1	Sliding window of length
σ	Set the standard deviation

(1) Challenge-Response Mechanism Selection

Considering security strength, space limitations, and the number of CRPs required, smart power terminals aiming for enhanced security need to construct 2^{70}~2^{80} CRPs. If the challenge utilizes only the SRAM unit's addresses, the SRAM PUF would require a space of 128 EB~128 ZB to meet the security requirements of generating a single set of security parameters. Therefore, this paper extends the challenge space using combinations of SRAM unit addresses. Taking SRAM3 with the minimum storage space of 4 KB as an example, the preprocessing stage was first constructed, and SRAM3 was powered on N times, and SRAM3 units were extracted respectively to compare whether there were any changes in N arrays ($N \times 32768$). The unchanged bit was set as stable output bit and the changed bit was set as unstable output bit. According to the typical values obtained by testing various chips and various processes, about 5% of SRAM3 is unstable bit, so the stable bit of SRAM3 is about 31,129 bits. If M bit is selected as a challenge in the above stable bit, C_{31129}^{M} is the number of CRPs of SRAM3.

Therefore, when M is 128, 256 and 512 bits, respectively, the number of CRPs in SRAM3 is approximately 2^{1229}, 2^{2508} and 2^{5017}, exceeding the maximum security level requirement of 2^{90} CRPs. Additionally, it has been determined that the minimum storage requirement to securely accommodate 2^{70}~2^{80} CRPs is 3 KB. Hence, any SRAM in the MCU can meet the requirements for enhanced security.

(2) Parameter Generation Functional Architecture

The security parameter generation of PUF is divided into a four-layer architecture, namely entropy source acquisition layer, algorithm implementation layer, configuration management layer, and functional service layer, as shown in Fig. 1:

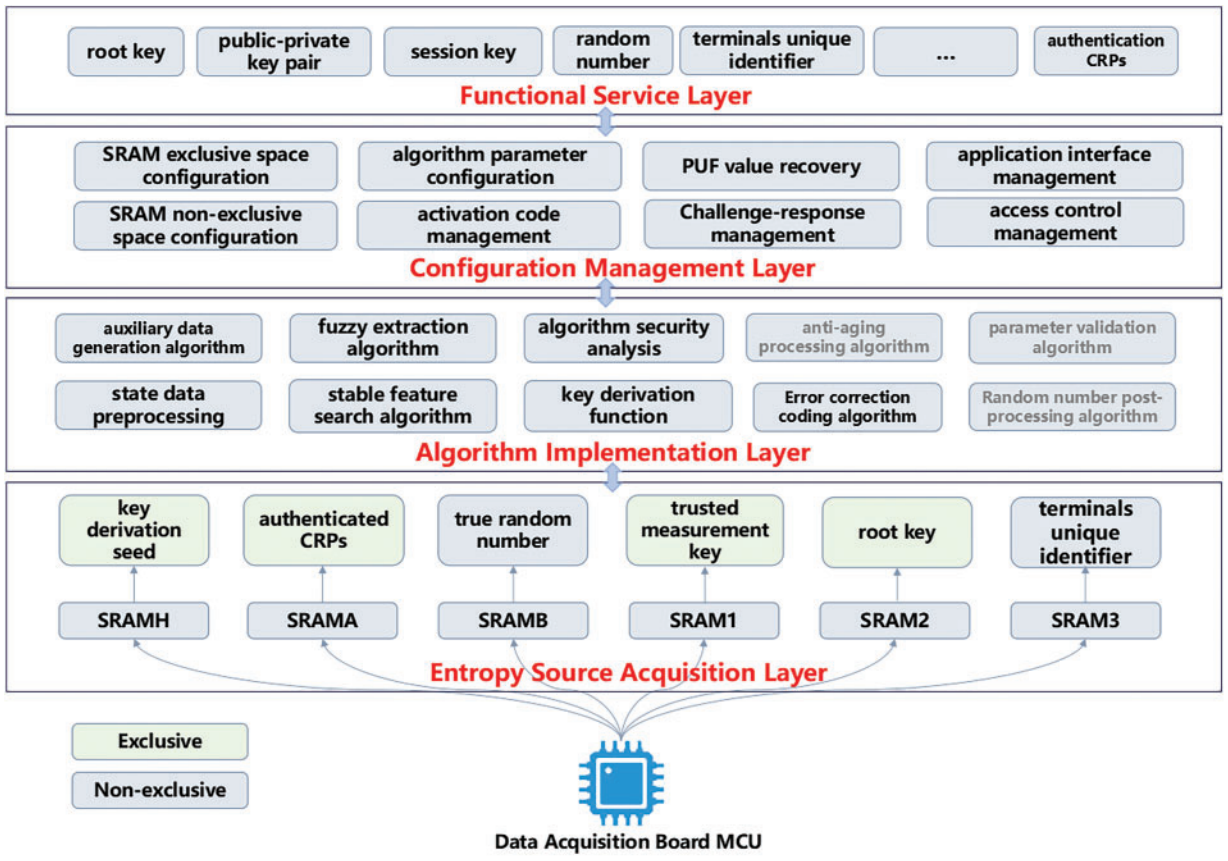

root key
public-private key pair
session key
random number
terminals unique identifier
...
authentication CRPs
Functional Service Layer
SRAM exclusive space configuration
algorithm parameter configuration
PUF value recovery
application interface management
SRAM non-exclusive space configuration
activation code management
Challenge-response management
access control management
Configuration Management Layer
auxiliary data generation algorithm
fuzzy extraction algorithm
algorithm security analysis
anti-aging processing algorithm
parameter validation algorithm
state data preprocessing
stable feature search algorithm
key derivation function
Error correction coding algorithm
Random number post-processing algorithm
Algorithm Implementation Layer
key derivation seed
authenticated CRPs
true random number
trusted measurement key
root key
terminals unique identifier
SRAMH
SRAMA
SRAMB
SRAM1
SRAM2
SRAM3
Entropy Source Acquisition Layer
Exclusive
Non-exclusive
Data Acquisition Board MCU

Figure 1: Root security parameter generation functional architecture based on SRAM PUF

(1) Entropy Source Acquisition Layer

Responsible for extracting stability and randomness information from SRAM as entropy sources for PUF security parameters. Additionally, considering factors such as storage overhead and time costs, an exclusive SRAM PUF approach is used to generate high-security parameters such as root keys, key derivation seeds, trust measurement keys, and authenticated CRPs. A non-exclusive SRAM PUF approach is employed for generating parameters like true random numbers and device unique identifiers, while SRAM allocation is based on parameter usage frequency and length requirements.

(2) Algorithm Implementation Layer

Responsible for preprocessing extracted entropy source information through filtering, equalization, quantization, etc. The layer then uses corresponding algorithms to convert this information into random data, key data, and unique identifiers, validating their randomness, uniqueness, and stability.

(3) Configuration Management Layer

Manages PUF configuration information including challenge generation, response storage, and provides mechanisms for PUF reset functionalities such as key erasure and factory reset. Controls access permissions to PUF configurations and interfaces, implementing necessary isolation and authorization mechanisms.

(4) Functional Service Layer

High-level encapsulation of security parameter functionalities and provision of service interfaces to external applications. Based on underlying PUF hardware, algorithms, and configuration management infrastructure, this layer supplies security parameters such as identifiers, keys, random numbers, and trust measurement keys for specific security services including device authentication, key negotiation, and data encryption.

In summary, the algorithm implementation layer is the core of parameter generation functionality design, directly influencing the overall security, reliability, and practicality of the system. Furthermore, the layered design of parameter generation functionality facilitates future standardization and engineering of PUF technology, enhancing modularity and scalability of terminal security functionalities.

3.2 SRAM PUF Security Parameter Generation Algorithms

This paper focuses on proposing design schemes for a series of algorithms within the algorithm implementation layer, including Stable Feature Search Algorithm, State Data Preprocessing Algorithm, Auxiliary Data Generation Algorithm, Fuzzy Extraction Algorithm, Key Derivation Function, Error Correction Code Algorithm, among others. The Random Number Postprocessing Algorithm, being relatively mature, is not discussed here.

(1) Stable Feature Search Algorithm

To select high-stability and high-randomness feature positions from SRAM PUF challenges for generating SRAM PUF responses, it is necessary to perform multiple restarts of the smart power terminal. During the registration phase of SRAM PUF functionality, repeated measurements and statistical analysis of SRAM PUF responses are conducted to assess the stability and randomness indicators of each response position, such as flip probability, Hamming distance, entropy, etc. Subsequently, all positions are sorted based on comprehensive indicators, and positions of higher quality are selected as stable features. This algorithm relies primarily on data collection, statistical computation, and feature selection.

During the data collection phase, assuming the smart power terminal is powered on m times, a set configurable PUF measurement matrices $\Omega_k = \{\mathrm{R}, \mathrm{T}, \mathrm{V}, \cdots, \mathrm{I}, \mathrm{E}\}$ is constructed. Here, R, T, V, I, E represent matrices of responses across m power-ons, temperature, voltage, current, and electromagnetic radiation environmental condition matrices, respectively. The matrix set can be expanded based on an expansion factor k. During the indicator statistics phase, stability is assessed through indicators such as flip probability, Hamming distance, and mutual information. In the feature selection phase, the comprehensive vector S is sorted in descending order, a threshold t is set, and bit positions with values greater than t are considered candidate features. The Stable Feature Search Algorithm (SFSA) is shown as Table 2.

Table 2: Stable feature search algorithm

Algorithm: Stable Feature Search Algorithm

```
Input: Initial response matrix $R_{m\times n}$, with dimensions $n \times m$
Output: Stability Assessment Metric $S(j)$
Progress
    For each bit position $j$ from 1 to $n$,
        1. calculate the flip probability $P[j]$
            $count = 0$
                For each power-on instance $i$ from 1 to $m$:
                if $R[i, j]! = R[i+1, j]$:
                    $count+ = 1$
        $P[j] = count/(m-1)$
        2. calculate the Hamming distance $D(j)$
            $dis = 0$
            For each power-on instance $i$ from 1 to $m$;
                $H(i, l) = sum(R[i, j] \oplus R[l, j]), dis = dis + H[i, l]$
            $D(j) = dis/C(m, 2)$
        3. calculate the entropy
            For matrix $R[j]$, count $n_0$ of 0 and $n_1$ of 1 among $m$ measurements.
            $p_0 = n_0/m, p_1 = n_1/m, H(j) = -p_0 \log_2 p_0 - p_1 \log_2 p_1$
        4. calculate the mutual information $I(j|\Delta)$
            Normalize parameters to obtain $\Delta = \omega_1\overline{T} + \omega_2\overline{V} + \omega_3\overline{I} + \omega_4\overline{E}$
            For each $\Delta$:
                count occurrences $n_2$ over $m$ measurements
                $p_2 = n_2/m, H(\Delta) = -\text{sum}(p_2 \log_2 p_2)$
            count $n_0$ of 0 and $n_1$ of 1 among $m$ measurements.
                $p_{0\Delta} = n_{0\Delta}/m, p_{1\Delta} = n_{1\Delta}/m$
                $H_\Delta(j|\Delta) = -p_{0\Delta} \log_2 p_{0\Delta} - p_{1\Delta} \log_2 p_{1\Delta}$
                $H(j|\Delta) = sum(p_\Delta H_\Delta(j|\Delta)), I(j|\Delta) = H(j) - H(j|\Delta)$
        5. calculate the auto-correlation coefficient $r(j)$
            $add = 0, add\_num = 0, add\_den = 0$
            For $1 \le i \le m$
                $add = add + R(i, j), \mu[j] = add/m$
            For retardation time $t_{time}$: $1 \le i \le m - t_{time}$
                $add\_num = add\_num + (R(i, j) - \mu(j)) \times (R(i + t_{time}, j) - \mu(j))$
                $add\_den = add\_den + (R(i, j) - \mu(j))^2$
$R(j, t_{time}) = add\_num/add\_den$
take the average of $R(j, t_{time})$ over different times $t_{time}$ to get $R(j)$
normalize stability index $S(j)$
    return $S(j)$
End
```

The main advantage of the SFSA proposed in this paper is its comprehensive evaluation method. SFSA effectively identifies feature bits with high stability by comprehensively assessing multiple feature indicators, which enables the algorithm to identify feature bits that perform stably under multiple environmental conditions. In contrast, as shown in Table 3, traditional methods may focus on a single metric, such as flip probability or Hamming distance, which may lead to a lack of stability in complex environments. SFSA improves the accuracy and robustness of feature selection by combining multiple metrics.

Table 3: Stable feature search algorithm

Feature	SFSA	Traditional
Stability assessment	Combining multiple indicators	Single indicator
Environmental adaptation	High, adapted to complex environments	Low, may fail in complex environments
Accuracy	High, reducing false selection of features	Low, may misselect unstable features
Robustness	Strong against environmental changes	Weak and vulnerable to environmental changes

This paper proposes a composite algorithm for evaluating regional metrics of SRAM. By comprehensively assessing multiple characteristic indicators, the algorithm effectively identifies bit positions with high stability. Compared to traditional methods that evaluate individual bit positions, this algorithm considers the correlated influences among SRAM units, thereby enhancing the accuracy and reliability of selection. Additionally, the algorithm introduces dynamic updating and parameter adjustment to adapt to long-term variations in SRAM characteristics, maintaining the stability of PUF responses.

(2) State Data Preprocessing Algorithm

Based on the stable feature bits selected above, the raw response signals collected after SRAM registration inevitably contain various high-frequency noises, such as thermal noise, radio frequency noise, and power supply noise. These noises can cause short-term fluctuations and instability in the security parameters generated, making noise reduction necessary. First, a mean filtering method is used to remove high-frequency noises from the original response of the stable feature bits and extract low-frequency features. Using a sliding window approach, the state data preprocessing algorithm (SDPA) for an original response sequence ϕ of length n is shown as Table 4.

Table 4: SRAM unit filtering output algorithm

Algorithm: SRAM Unit Filtering Output Algorithm

```
Input: Original response sequence $\phi$, with length $n$
Output: Filtered output sequence $\phi'$
Progress:
    choose an appropriate window length $l_1$, where $l_1 = 2n + 1$, $n \in Z$
    For each position $i$ in the original response sequence $\phi$ from 1 to $n$:
        $left = \max(1, i - l_1/2)$, $right = \max(1, i + l_1/2)$, $sum = 0$
        For $j$ from left to right
            $sum += \varphi[j]$
        $\phi'[i] = sum/l_1$
```

(Continued)

Table 4 (continued)

```
    return $\phi'$
End
```

For the current position i, consider all SRAM cells within a sliding window of length l_1 centered around i. Compute their arithmetic mean as the filtered output $\phi^{(1)}$ to smooth out noise using sliding average filtering and extract the primary features of the PUF response. It's important to note that when processing the beginning and end segments of the response sequence, the sliding window may extend beyond the sequence boundaries. For these boundary cases, zero-padding, mirroring, or special handling methods can ensure the integrity and consistency of filtering process.

Due to potential variations in response distributions between different SRAM PUFs and the same SRAM PUF under different environmental conditions, an adaptive quantization mechanism is needed to adjust quantization thresholds dynamically, enhancing robustness. After powering the SRAM m times, gather samples of the candidate $\phi^{(1)}$ into set $\Phi = (\phi_1^{(1)}, \phi_2^{(1)}, \cdots, \phi_{\mathrm{m}}^{(1)})$, with a sample mean μ denoted as $(\phi_1^{(1)} + \phi_2^{(1)} + \cdots + \phi_m^{(1)})/n$. Based on this, set the standard deviation σ as $\sqrt{\left(\left(\phi_1^{(1)} - \mu\right)^2 + \left(\phi_2^{(1)} - \mu\right)^2 + \cdots + \left(\phi_m^{(1)} - \mu\right)^2\right) / (m-1)}$, and define upper and lower thresholds $t_{up} = \mu + k_1\sigma$ and $t_{\mathrm{down}} = \mu - k_2\sigma$, where k_1 and k_2 are adjustable parameters. For each value $\phi_i^{(1)}$ in $\phi^{(1)}$:

- ➢ Quantize as 1 if $\phi_i^{(1)} > t_{up}$,
- ➢ Quantize as 0 if $\phi_i^{(1)} < t_{down}$,

Otherwise, treat as an indeterminate state, discard, or mark as a random bit.

Through iterative algorithms, obtain the final bit sequence $\phi^{(2)}$ after adaptive quantization.

Ideally, the response of an SRAM PUF should exhibit a uniform distribution of 0 and 1 s. This uniformity maximizes randomness, making the response resistant to statistical attacks and cloning attempts. However, in practice, responses may exhibit imbalance where the probabilities of 0 and 1 differ. This imbalance reduces the randomness of the PUF response, making it susceptible to statistical attacks and predictions. Therefore, it is necessary to perform balancing adjustments to equalize the proportions of 0 and 1 in the PUF response as closely as possible. To achieve this, calculate the counts of 0 and 1 s in $\phi^{(2)}$ as z_1 and z_2. If $|z_1 - z_2| > k_3$ and k_3 is a configurable threshold, perform the following balancing process:

- ➢ If $z_1 > z_2$, randomly select $u(u < |z_1 - z_2|)$ 0 and flip them to 1.
- ➢ If $z_1 < z_2$, randomly select $u(u < |p_1 - p_2|)$ 1 and flip them to 0.

Through iterative algorithms, obtain the final balanced bit sequence $\phi^{(3)}$. Due to the presence of repetitive patterns in SRAM PUF responses, such as consecutive sequences of 0 or 1, or fixed bit combinations, the randomness of the response is diminished, potentially becoming a security vulnerability. Therefore, it is essential to detect and remove these repetitive patterns to enhance the security of SRAM PUFs as random and key sources.

This paper proposes a state data preprocessing algorithm termed "noise filtering-adaptive quantization-balance correction-repetitive pattern removal". The four sub-algorithms mentioned can be organically combined to comprehensively address various quality issues in PUF state data, such as noise interference, response imbalance, and repetitive patterns. Compared to traditional single preprocessing methods, this algorithm offers a more comprehensive and systematic approach to improving the quality of PUF state data. It provides a flexible, scalable, and hardware-friendly solution for state data preprocessing.

(3) Auxiliary Data Generation Algorithm

The Auxiliary Data Generation Algorithm (ADGA) iteratively applies the aforementioned state data preprocessing algorithm to generate the response sequence $\Phi^{(4)} = (\phi_1^{(4)}, \phi_2^{(4)}, \cdots, \phi_\mathrm{m}^{(4)})$. The median response $\phi_{mid}^{(4)}$ is selected as the reference response, which serves as the baseline for subsequent error correction and auxiliary data generation.

(4) Adaptive Error Correcting Coding Algorithm

Traditional error correction schemes such as BCH codes and Reed-Solomon codes exhibit good error correction capabilities but still have room for improvement in terms of implementation complexity and resource overhead. In response to specific requirements and constraints of SRAM PUFs, innovatively constructing efficient and lightweight error correcting coding algorithms is a crucial research direction. This paper combines polynomial error correction codes with permutation mapping principles and proposes the Adaptive Error Correcting Coding Algorithm (AECCA).

Polynomial codes, due to their simple algebraic structure, facilitate hardware implementation and optimization, thereby reducing hardware resource overheads. The introduction of permutation mapping mechanisms disrupts the positional correlation of PUF responses, enhancing the efficiency of error-correcting codes and reducing the required redundancy. An adaptive coding strategy is employed that dynamically adjusts coding parameters based on the actual quality of PUF responses, avoiding excessive or insufficient encoding, and balancing error correction capability and coding overhead. Iterative decoding algorithms effectively exploit the soft information of PUF responses to enhance decoding success rates.

(5) Fuzzy Extraction Algorithm

Fuzzy extraction algorithms enhance the fault tolerance and robustness of PUFs but are often limited to academic prototypes. These prototypes usually focus on improving one or a few metrics, such as security or reliability, without achieving a balanced optimization across multiple design dimensions. Consequently, they often face practical challenges in real-world applications, failing to meet stringent deployment requirements. Therefore, this paper proposes a comprehensive fuzzy extraction algorithm (FEA) that aims to balance and optimize multiple dimensions effectively.

The fuzzy extractor algorithm in this paper adopts adaptive error correction coding, dynamically adjusting coding strategies based on the noise characteristics of PUF responses to enhance coding response stability. Introducing permutation increases randomness and unpredictability of coding responses, thereby enhancing key security. A random matrix M generates auxiliary data AD, safeguarding AD confidentiality and integrity through randomization and hashing algorithms. Matrix projection extracts keys from permutation-encoded responses, simplifying key management. Adaptive threshold quantization adjusts quantization thresholds dynamically based on projection result distributions, improving key extraction robustness. In auxiliary data reconstruction, integrity verification ensures consistency and reliability. Key verification assesses key reconstruction success, enhancing system fault tolerance.

(6) Key Derivation Algorithm

The output of the fuzzy extractor serves as the initial key. Despite its stability and randomness, issues such as inadequate uniformity in initial key distribution, varying application requirements, and risks of key reuse attacks arise from direct initial key application, leading to security and compatibility concerns. Therefore, a key derivation function is introduced post-fuzzy extractor to transform the initial key into a final key tailored to diverse application needs. This Key Derivation Algorithm (KDA) incorporates mechanisms like multifactor integration, high-security hash functions, flexible parameter selection, and hardware optimization to achieve secure key derivation, outlined as follows:

As shown in Table 5, in the key derivation algorithm, $x[0]$ is used in the initialisation step to combine the input parameters into a single input value for the subsequent key generation process. This ensures that the key generation process takes into account all relevant security factors such as application context and user-specific identifiers. Depending on the type of key, different key generation strategies are used. For session keys and symmetric keys, we use a hash-based iterative process, which helps to increase the randomness and complexity of the key. For asymmetric keys, we use Elliptic Curve Cryptography, which is a widely accepted security method that provides strong security. Our approach considers not only key generation, but also key security and application scenarios. By using the salt value and the number of iterations, we can resist rainbow table attacks and brute force attacks. In addition, by using parallel computing, we can improve the efficiency of key generation.

Table 5: Key derivation algorithm

Algorithm: Key Derivation Algorithm

```
Input: key $key$ of length $l_7$ generated by the fuzzy extractor, application identifier $ID$ for the smart power terminal APP_ID, length $l_8$, session identifier SID, target key length $l_9$, salt $Salt$, iterations $iter$, parallelism degree $p$, Key type $type$ (0-session key, 1-asymmetric key, 2-symmetric key)
Output: target key
Progress:
1. Initialization, combine the key key, the salt value Salt, the application identifier APP_ID or the session identifier SID into an initial input $x[0]$ depending on the key type.
    if $type == 0$ or $type == 2$:
        $x[0] = key || Salt || ID || \text{int\_}u32(n)$
    else
        $x[0] = key || Salt || SID || \text{int\_}u32(n)$
2. key generation
    if $type == 0$ or $type == 2$:
      for $i = 1$ to $iter$:
         divide $x[i-1]$ into $p$ substrings: $x[i-1] = x[i-1,1] || \cdots || x[i-1,p]$
            for $j = 1$ to $p$:
               $y[i,j] = SM3(x[i-1,j])$
            $x[i] = y[i,1] || \cdots || y[i,p]$
         output the first $l_9$ bits of $x[iter]$ as the session key
  else
        Select parameters $a, b, p$, where $p > 2^{191}$, to generate elliptic curve $y^2 = x^3 + ax + b \bmod p$
        randomly select base point $G(x_g, y_g)$ and the order $n$ of base point.
        calculate $d = key \bmod(2^{l_9} - 1)$ and $P = (xP, yP) = [d]G$;
        output the key pair $(d, P)$, $d$ is private key and $P = (x, y)$ is the public key.
End
```

The key derivation algorithm is relatively simple, with its security relying mainly on the security of the SM2 and SM3 algorithms. In addition, this algorithm incorporates multiple factors such as Salt, APP tags, and utilizes adjustable parameters including iteration count, key length, Salt length, and parallelism. These parameters can be appropriately balanced according to the security requirements of the business. Adopting multi-round hash iteration significantly enhances the security strength of derived keys, while leveraging parallel computing to fully utilize multi-core MCU processing power can markedly improve the efficiency and throughput of KDA execution.

3.3 SRAM PUF Security Parameter Generation Process

SRAM PUF generates various security parameters essential for cryptographic algorithms and security protocols, categorized into six types: key parameters (root key, asymmetric key, symmetric key, session key), identity parameters (device fingerprint, chip ID), and random numbers (random seed, CPRS).

The generation process starts with initial CRPs derived from SRAM's physical characteristics. Stable feature search algorithms sort stable and unstable output bits. Unstable bits provide entropy for random seeds, which are refined by a random number post-processing algorithm. Stable bits are split, with some forming CRPs for chip IDs and authentication, while others undergo state data preprocessing and adaptive error correction to generate root keys. Data from multiple SRAM startups and adaptive error correction outputs are processed by fuzzy extraction and key derivation algorithms to produce asymmetric, session, and symmetric keys. This robust framework ensures the secure generation of vital cryptographic parameters.

Each algorithm within SRAM PUF is divided into four stages: initialization, registration authentication, key generation, and update recovery. Each stage involves different algorithms and closely interacts with the configuration management layer, as depicted in Fig. 2.

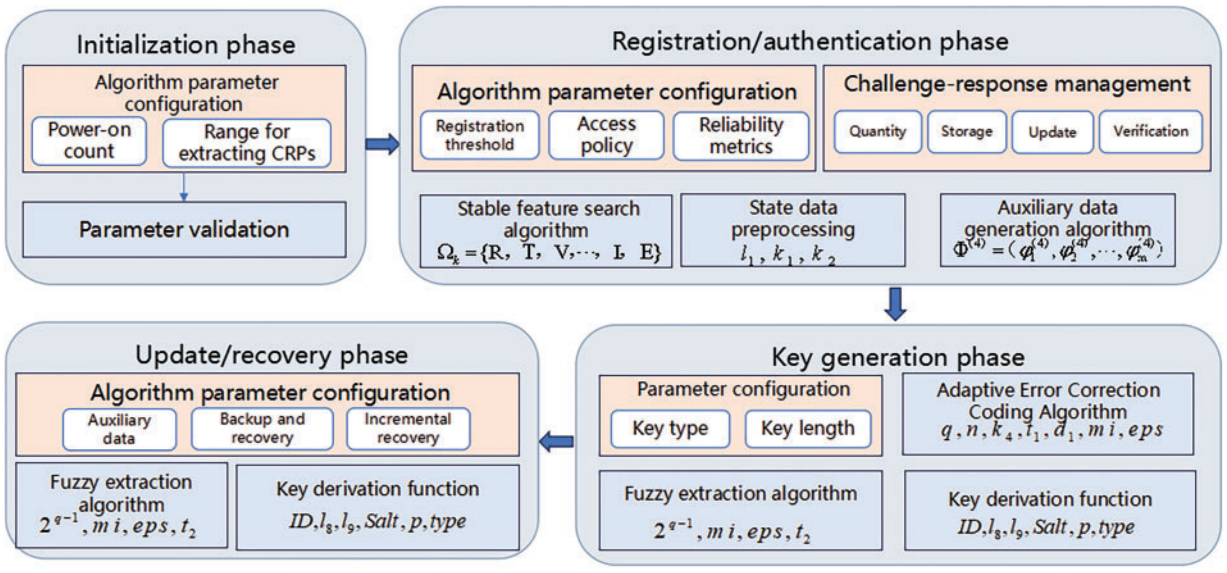

Initialization phase
Algorithm parameter configuration
Power-on count
Range for extracting CRPs
Parameter validation
Registration/authentication phase
Algorithm parameter configuration
Registration threshold
Access policy
Reliability metrics
Challenge-response management
Quantity
Storage
Update
Verification
Stable feature search algorithm $\Omega_k=\{R, T, V, \cdots, I, E\}$
State data preprocessing l_1, k_1, k_2
Auxiliary data generation algorithm $\Phi^{(4)}=(\varphi_1^{(4)}, \varphi_2^{(4)}, \cdots, \varphi_m^{(4)})$
Update/recovery phase
Algorithm parameter configuration
Auxiliary data
Backup and recovery
Incremental recovery
Fuzzy extraction algorithm $2^{q-1}, mi, eps, t_2$
Key derivation function $ID, l_8, l_9, Salt, p, type$
Key generation phase
Parameter configuration
Key type
Key length
Adaptive Error Correction Coding Algorithm $q, n, k_4, t_1, d_1, mi, eps$
Fuzzy extraction algorithm $2^{q-1}, mi, eps, t_2$
Key derivation function $ID, l_8, l_9, Salt, p, type$

Figure 2: Phase division of SRAM-PUF SPG

During initialization, the service parameter configuration module provides initial parameters and validates SRAM PUF initialization parameters. For registration authentication, it supplies challenge-response management parameters, and algorithms for stable feature search, data preprocessing, and auxiliary data generation create high-quality CRPs and error correction data. In key generation, algorithm parameters and challenge-response management modules offer parameters for adaptive error correction coding and fuzzy extraction, producing various keys. During update recovery, these modules use algorithms to recover critical parameters from auxiliary data. The configuration management layer oversees algorithm parameters, data storage, and operations throughout SRAM PUF's lifecycle.

4 Experimental Results and Analysis

To validate the effectiveness of the security parameter generation mechanism, this paper first constructs a framework for integrating security parameters and sets up a testing environment using typical electric power terminals to verify the functionality, performance, and security of the SRAM PUF security parameter generation function.

4.1 SRAM PUF Security Parameter Generation Performance Verification

Based on the algorithm invocation process, this section presents the integrated architecture of multiple SRAM PUF instances. Adopting a modular and parameterized design approach, clear interfaces and configuration parameters are defined. The interface layer abstracts and encapsulates the details of underlying algorithm implementations, and reusable IP libraries are developed for different PUF types and application scenarios. These libraries provide unified application interfaces, enabling secure applications and developers to access, configure, and operate PUFs flexibly, supporting customization and extension.

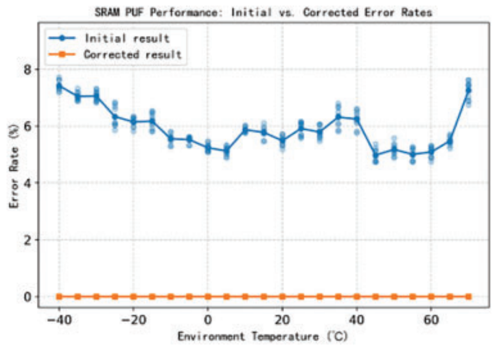
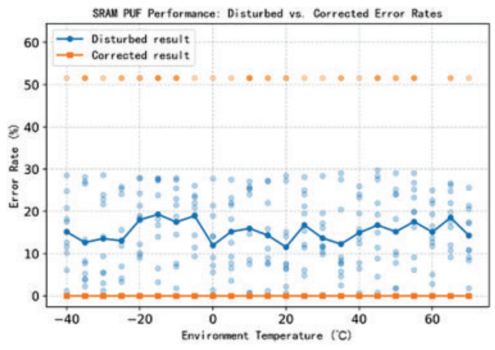
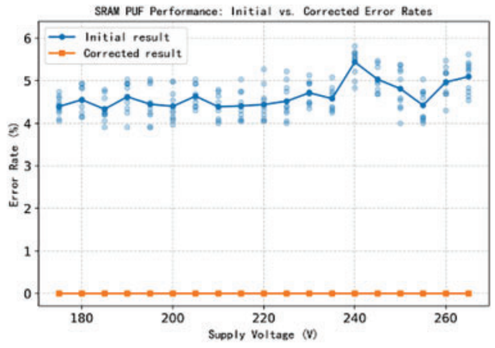
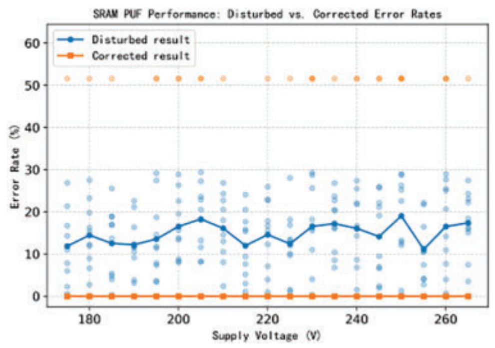
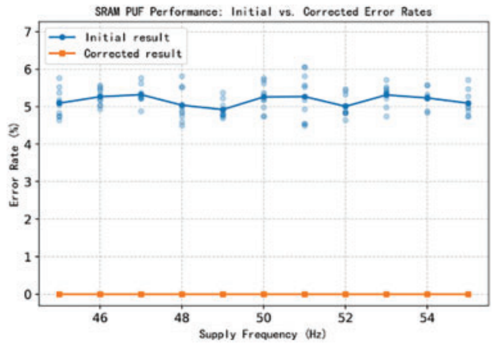
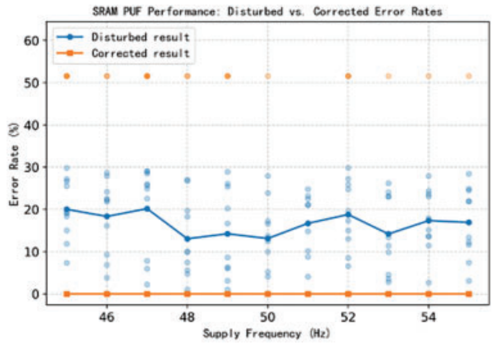
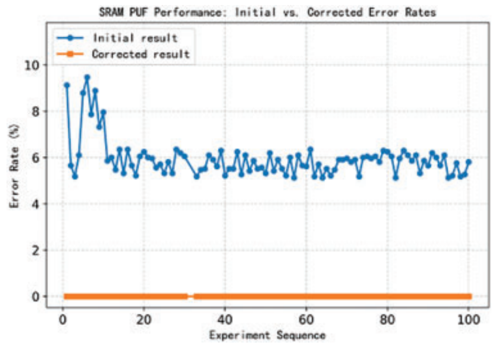
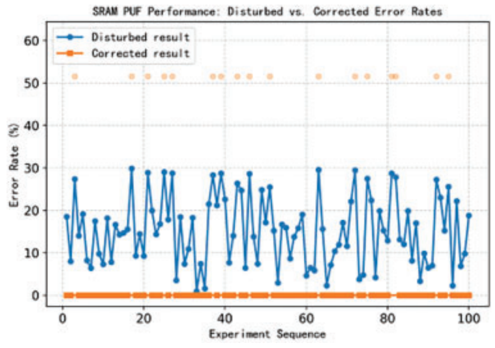
The Huada HC32F4A0 chip is selected as the MCU for the data acquisition board [28], the chip is Cortex-M4 architecture, integrated FPU, MPU, DSP supporting SIMD instructions, the highest operating frequency of 240 MHz, up to 300 DMIPS or 825 Coremarks computing performance, support for a wide range of voltages (1.8–3.6 V), the device grade for industrial-grade (operating temperature of −40°C~105°C), the chip contains 2 MB of Flash and 516 KB SRAM, 6 independent clock sources, 16 high-performance analog peripherals, multiple timers, 142 GPIO, up to 32 communication interfaces. Flash and 516 KB of SRAM, 6 independent clock sources, 16 high-performance analog peripherals, multiple timers, 142 GPIOs, and up to 32 communication interfaces. Security parameter functions, algorithms, and processes are developed on it, followed by functional testing, performance testing, and algorithm security analysis. Due to space constraints, the testing environment is built using the data acquisition board of a feeder intelligent integrated terminal, using root key generation as an example for functional and performance testing.

(1) Reliability

Running the SRAM PUF program and executing operational instructions, the robustness and reliability of PUF can be quantified through bit error rate, which is the number of different bits between multiple responses divided by the code length.

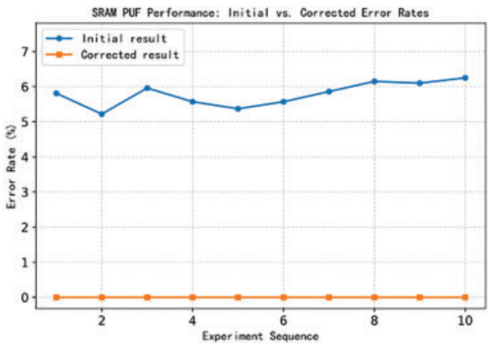
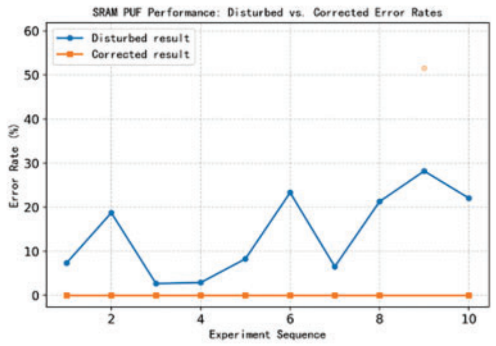
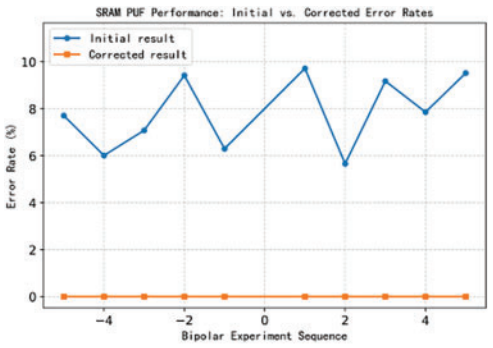
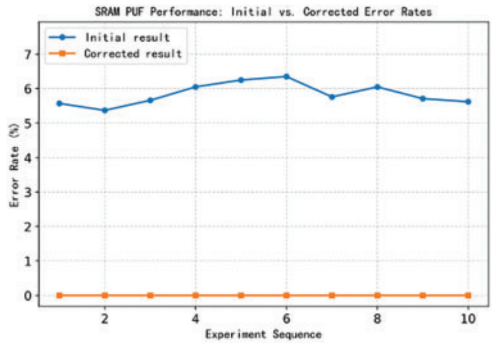
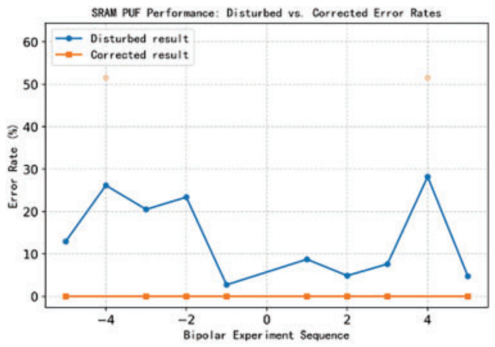
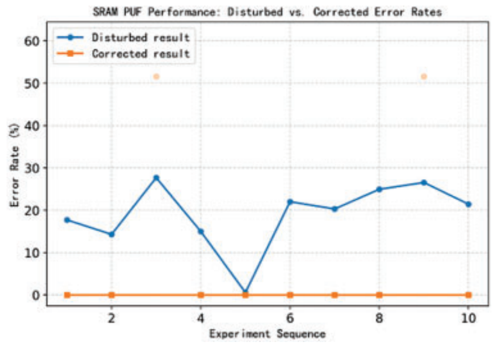
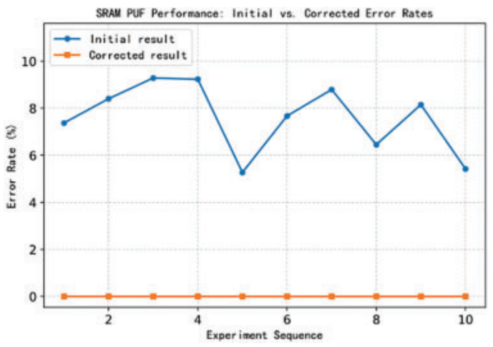
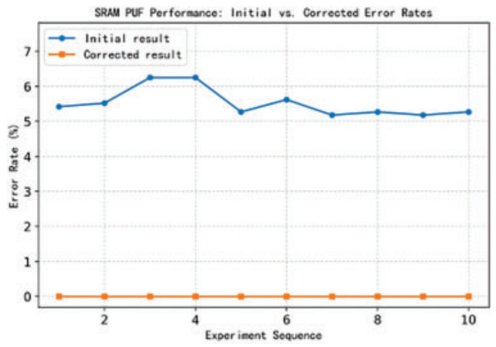
High and low temperature equipment, relay protection devices, power failure simulators, EMC tests, voltage surge simulators, and other equipment are used to test initial bit error rates and algorithm correction rates for 2048-bit security parameters under ten physical indicators such as temperature, power supply environment, frequency, voltage sag, and short-term interruption. During algorithm execution, disturbances are randomly added to 2048 bits to calculate disturbance rate and bit error rate after algorithm correction. Specific test results are shown as Table 6.

Table 6: Experimental results

Indicators	Initial result correction	Disturbance result correction
Environment temperature		
Mains input		
Frequency change		
Voltage sag and momentary interruption		

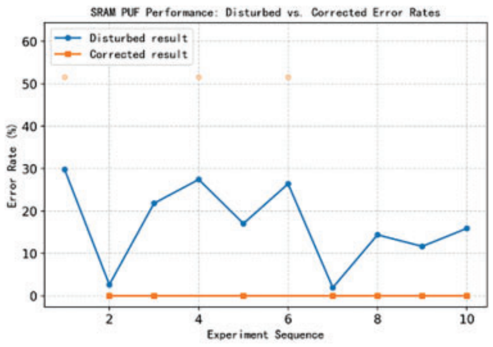
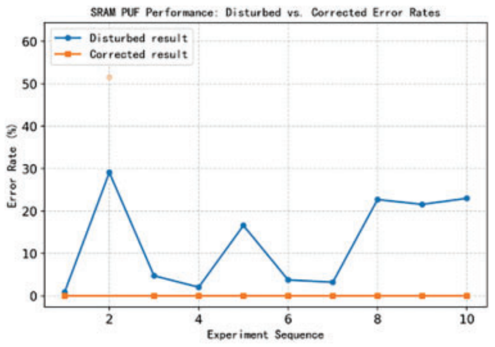
(Continued)

Table 6 (continued)

Indicators	Initial result correction	Disturbance result correction
Power frequency magnetic field immunity		
Surge voltage	SRAM PUF initial average error rate 5.18%; SRAM PUF error rate after correction 0	SRAM PUF disturbance average error rate 24.9%; SRAM PUF error rate after correction 0
Surge (impact) immunity		
Electrostatic discharge immunity test		
Short-duration overcurrent		

(Continued)

Table 6 (continued)

Indicators	Initial result correction	Disturbance result correction
Short-duration overvoltage		

(2) Timeliness

As shown in Fig. 3, Fifty tests were conducted on a laptop, and the average time taken to obtain the root key was 0.4 s. When transplanted to the data acquisition board for another fifty tests, due to the startup time of other peripherals on the board, the average time to obtain the root key was 2.33 s.

4.2 SRAM PUF Security Parameter Generation Security Verification

Using the root key as an example, 500 sets of tests were conducted on four terminals. The expected Hamming distance between chips is shown in Figs. 4 and 5, which satisfies the requirement of being within the specified range, with a standard deviation less than 0.5.

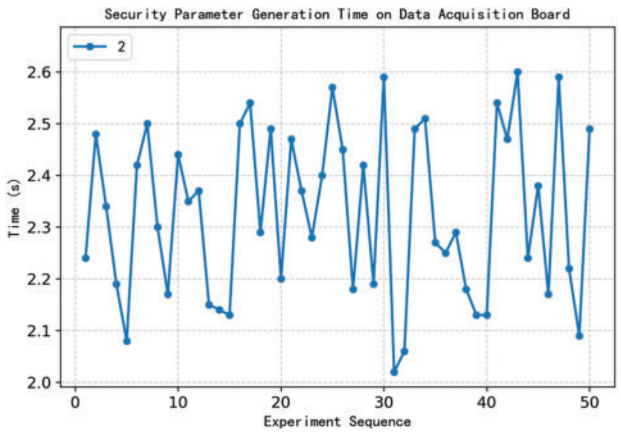


Figure 3: SRAM-PUF SPGstartup time

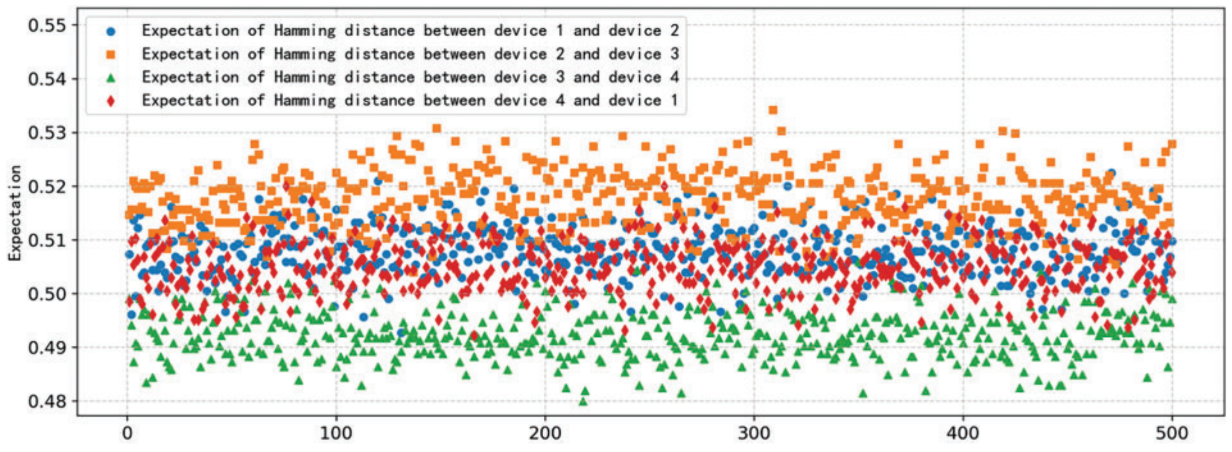


Figure 4: Expected value of inter-block Hamming distance in 500 tests

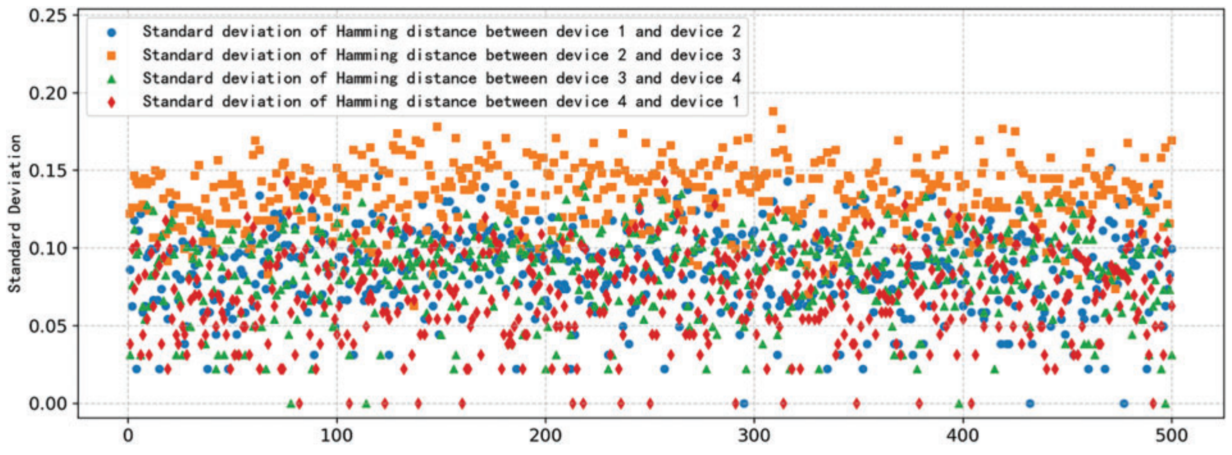


Figure 5: Standard deviation of inter-block Hamming distance in 500 tests-12pt

5 Conclusion

This article proposes a security parameter generation mechanism based on SRAM PUF. Firstly, to meet the enhanced security needs of smart power terminals, it introduces a method for selecting a challenge-response mechanism and designs the architecture of the security parameter generation function. It also establishes a security parameter generation framework that covers layers such as the entropy source acquisition layer, algorithm implementation layer, configuration management layer, and functional service layer. Secondly, it introduces a series of algorithms for the algorithm implementation layer, including algorithms for searching stable characteristics, preprocessing state data, generating auxiliary data, fuzzy extraction, and key derivation functions. Then, it designs the call flow of the security parameter generation function and the framework for the functional implementation of each stage, and builds a multi-instance PUF functional integration mechanism. Experiments have proven that results possess excellent reliability, stability, and security and can effectively support the construction of a terminal trusted root system, laying a theoretical foundation for subsequent industrial promotion and safe applications. In terms of the application and promotion of the method, the system is particularly suitable for the context of the orderly opening of the power grid and the dramatic increase in the demand for collaborative operation, in which cross-domain collaboration scenarios of different business terminals of electric power, such as the collaboration of the distribution automation system and the terminals belonging to the power consumption information collection system, the security level of the interacting parties is different and the security mechanisms of some of them are heterogeneous, so it is necessary to construct the identity trust transfer mechanism across the heterogeneous domains and to continuously assess the trustworthiness of the other party. The system generates device unique identifiers and authentication CRPs and public keys through the SRAM PUF function to achieve identity registration and authentication, which effectively improves the authentication efficiency of terminals interacting with power intelligent terminals in the same region and across domains, and reduces the authentication overhead.

Acknowledgement: Not applicable.

Funding Statement: This paper is supported by National key Research and Development Program "Security Protection Technology for Critical Information Infrastructure of Distribution Network" (2022YFB3105100).

Author Contributions: Xiao Feng and Xiao Liao conceived and designed the experiments, performed the data analysis, and wrote the manuscript. Xiaokang Lin contributed to the experimental design and performed the experiments. Xiao Feng and Yonggui Wang contributed to the data analysis tools and revised the manuscript critically for important intellectual content. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets generated and/or analyzed during the current study are not publicly available due to privacy concerns.

Ethics Approval: This study complies with the ethical standards of the institution and the National Research Council.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Xu X, Liu X, Xu Z, Dai F, Zhang X, Qi L. Trust-oriented IoT service placement for smart cities in edge computing. IEEE Internet Things J. 2020;7(5):4084–91. doi:10.1109/JIOT.2019.2959124.
2. Guo S, Dai Y, Xu S, Qiu X, Qi F. Trusted cloud-edge network resource management: DRL-driven service function chain orchestration for IoT. IEEE Internet Things J. 2020;7(7):6010–22. doi:10.1109/JIOT.2019.2951593.
3. Kuang Z, Ma Z, Li Z, Deng X. Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing. J Syst Archit. 2021;118(99):102167. doi:10.1016/j.sysarc.2021.102167.
4. Josilo S, Dan G. Wireless and computing resource allocation for selfish computation offloading in edge computing. In: Proceedings of IEEE INFOCOM Conference on Computer Communications; 2019 Apr 29–May 2; Paris, France. p. 2467–75.
5. National Institute of Standards and Technology (NIST). Guidelines for Smart Grid Cybersecurity. NIST Interagency Report 7628 Revision 1. Gaithersburg, MD: National Institute of Standards and Technology. 2014 [cited 2025 Jan 10]. Available from: https://nvlpubs.nist.gov/nistpubs/ir/2014/nist.ir.7628r1.pdf.
6. Chu Z, Lakshminarayana S, Chaudhuri B, Teng F. Mitigating load-altering attacks against power grids using cyber-resilient economic dispatch. IEEE Trans Smart Grid. 2023;14(4):3164–75. doi:10.1109/TSG.2022.3231563.
7. Ivest RL, Adleman L, Dertouzos ML. On data banks and privacy homomorphisms. [cited 2025 Jan 10]. Available from: https://luca-giuzzi.unibs.it/corsi/Support/papers-cryptography/RAD78.pdf.
8. Devi P, Bharti MR. Physical layer security for IoT over Nakagami-m and mixed Rayleigh-Nakagami-m fading channels. Wirel Netw. 2023;29(8):3479–91. doi:10.1007/s11276-023-03422-5.
9. Sezgin A, Boyacı A. AID4I: an intrusion detection framework for industrial Internet of Things using automated machine learning. Comput Mater Contin. 2023;76(2):2121–43. doi:10.32604/cmc.2023.040287.

10. Veiga R, Both CB, Medeiros I, Rosário D, Cerqueira E. A federated learning approach for authentication and user identification based on behavioral biometrics. In: Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2023); 2023 May 22–26; Brasília, Brazil.
11. Guajardo J, Kumar SS, Schrijen GJ, Tuyls P. FPGA intrinsic PUFs and their use for IP protection. In: Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop; 2007 Sep 10–13; Vienna, Austria. p. 63–80.
12. Maes R, van Der Leest V, van Der Sluis E, Willems F, Verbauwhede. Secure key generation from biased PUFs. In: 2015 IEEE International Symposium on Hardware-Oriented Security and Trust; 2015 Sep 13–16; Saint-Malo, France. p. 517–34.
13. Herder C, Yu MD, Koushanfar F, Devadas S. Physical unclonable functions and applications: a tutorial. Proc IEEE. 2014;102(8):1126–41. doi:10.1109/JPROC.2014.2320516.
14. Ruhrmair U, Solter J. PUF modeling attacks: an introduction and overview. In: 2014 Design, Automation & Test in Europe Conference & Exhibition; 2014 Mar 24–28; Dresden, Germany. p. 1–6.
15. Ye J, Hu Y, Li X. RPUF: physical unclonable function with randomized challenge to resist modeling attack. In: Proceedings of the 2016 IEEE Asian Hardware-Oriented Security and Trust; 2016 Dec 19–20; Yilan, Taiwan.
16. Zhang J, Wang Z, Verma N. A machine-learning classifier implemented in a standard 6T SRAM array. In: 2016 IEEE Symposium on VLSI Circuits; 2016 Jun 15–17; Honolulu, HI, USA.
17. Vijayakumar A, Patil VC, Prado CB, Kundu S. Machine learning resistant strong PUF: possible or a pipe dream. In: 2016 IEEE International Symposium on Hardware Oriented; 2016 May 3–5; McLean, VA, USA.
18. Jeloka S, Yang K, Orshansky M, Sylvester D, Blaauw D. A sequence dependent challenge-response PUF using 28nm SRAM 6T bit cell. In: 2017 Symposium on VLSI Circuits; 2017 Jun 5–8; Kyoto, Japan.
19. Che W, Martinez-Ramon M, Saqib F, Plusquellic J. Delay model and machine learning exploration of a hardware-embedded delay PUF. In: 2018 IEEE International Symposium on Hardware Oriented; 2018 Apr 30–May 4; Washington, DC, USA.
20. Kumar S, Niamat M. Machine learning based modeling attacks on a configurable PUF. In: NAECON 2018-IEEE National Aerospace and Electronics; 2018 Jul 23–26; Dayton, OH, USA.
21. Wisiol N, Becker G, Margraf M, Soroceanu TAA, Tobisch J, Zengin B. Breaking the lightweight secure PUF: understanding the relation of input transformations and machine learning resistance. IACR Cryptol Eprint Arch Report 2019/862. 2019 [cited 2025 Jan 10]. Available from: https://eprint.iacr.org/2019/862.
22. Rai VK, Tripathy S, Mathew J. 2SPUF: machine learning attack resistant SRAM PUF. In: 2020 Third ISEA Conference on Security and Privacy; 2020 Feb 27–Mar 1; Guwahati, India.
23. Chen J, Lange T, Andjelkovic M, Simevski A, Lu L, Krstic M. Solar particle event and single event upset prediction from SRAM-based monitor and supervised machine learning. IEEE Trans Emerg Top Comput. 2022;10(2):564–80. doi:10.1109/TETC.2022.3147376.
24. Suragani R, Nazarenko E, Anagnostopoulos NA, Mexis N, Kavun EB. Identification and classification of corrupted PUF responses via machine learning. In: 2022 IEEE International Symposium on Hardware Oriented; 2022 Jun 27–30; McLean, VA, USA.
25. Qiu H, Ma H, Zhang Z, Gao Y, Zheng Y, Fu A, et al. RBNN: memory-efficient reconfigurable deep binary neural network with IP protection for Internet of Things. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2023;42(4):1185–98. doi:10.1109/TCAD.2022.3197499.
26. Mispan MS, Jidin AZ, Sarkawi H, Nasir HM. Performance evaluation of SRAM-PUF based on 7-nm, 10-nm and 14-nm FinFET technology nodes. Int J Nanoelectron Mater. 2023;14(4):345–56.
27. Alheyasat A, Torrens G, Bota SA, Alorda B. Estimation during design phases of suitable SRAM cells for PUF applications using separatrix and mismatch metrics. Electronics. 2021;10(12):1479. doi:10.3390/electronics10121479.
28. Huada Semiconductor Co., Ltd. Huada Semiconductor's Subsidiary Xiaohua Semiconductor Awarded "Best MCU Chip of 2023". [cited 2025 Jan 10]. Available from: https://www.hdsc.com.cn/108.html.