1. Introduction

In recent years, mobile devices and edge devices have become ubiquitous, and their interconnection via the internet has become indispensable. As the number of devices owned by individuals increases, security issues become more important. With the frequent transmission of private and personal information through the network, encryption systems are necessary.

Generally, encryption systems or security primitives can be divided into software- or hardware-based systems. Software-based security systems are considered superior to software-based systems. Conventional hardware-based security primitives based on integrated circuits provide device authentication and confidential information protection through encryption elements such as digital signatures. However, they are also susceptible to various attack methods, such as side-channel attacks including power analysis and direct probing. As a result, the demand for physical hardware security primitives that are physically embedded with unique, individual structural properties and high reliability has increased.

Physically unclonable functions (PUFs) have attracted significant attention for their purpose as secret keys. PUFs receive the input and return the output, known as challenge-response pairs (CRPs). Unique CRPs can be generated as they utilize the stochastic characteristics from the inherent randomness of the hardware. Physically unclonable functions (PUFs) insecure as security primitives.

The dissemination of edge devices drives new requirements for security primitives for privacy protection and chip authentication. Memristors are promising entropy sources for realizing hardware-based security primitives due to their intrinsic randomness and stochastic properties. With the adoption of memristors among several technologies that meet essential requirements, the neural network physically unclonable function (NNPUF) is proposed, a novel PUF design that takes advantage of deep learning algorithms. The proposed design integrated with the memristor array can be constructed easily because the system does not depend on write operation accuracy. To contemplate a nondifferentiable module during training, an original concept of loss called PUF loss is devised. Iterations of weight update with the loss function bring about optimal NNPUF performance. It is shown that the design achieves a near-ideal 50% average value for security metrics, including uniformity, diffuseness, and uniqueness. This means that the NNPUF satisfies practical quality standards for security primitives by training with PUF loss. It is also demonstrated that the NNPUF response has an unassailable resistance against deep learning-based modeling attacks, which is verified by the near-50% prediction model accuracy.

The memristor has arisen as a candidate for hardware security applications due to its stochastic behavior in the formation and rupture of conductive filaments. A memristive device changes the internal resistance according to the applied voltage, which shows random and probabilistic switching behavior. Some memristor-based PUFs depend on the switching probability of the memristive device. However, iterative switching with random probability prevents the memristor-based PUFs from succeeding in application due to their drop in reliability.

Another important issue of PUFs is their vulnerability to deep learning attacks. The deep learning model can predict all CRPs with a small number of CRPs, even if the hardware itself cannot be replicated. This weakness against deep learning attacks makes PUFs insecure as security primitives. As some of the existing PUFs, such as arbiter PUFs and bistable ring PUFs, were
simply modeled by linear functions, it was easy to deprive them of their response sets through deep learning attack modeling.

In this work, we propose a novel PUF system called neural network physically unclonable function (NNPUF), which utilizes deep learning algorithms and memristive devices to overcome the existing problems of conventional PUFs (Figure 1a). Our PUF system leverages both the neuromorphic characteristics of the memristor and its randomness. To obtain configurable randomness with PUF metrics (uniqueness, uniformity, diffuseness, and reliability), we suggest a new training algorithm throughout the overall PUF system. This work utilizes deep learning algorithms to attain the security metrics of PUF, and it has significant novelty due to training with a nondifferentiable module. Our encryption system also achieved resistance against deep learning attacks by taking advantage of the intrinsic nonlinearity of the circuit, including the memristor array.

Figure 1. a) Overview of the NNPUF as a secure secret key against an adversary. b) Overall architecture and mechanism of the proposed PUF system showing that input challenge goes through the cell-selection network, the cell selection, and PUF box sequentially to generate output response. c) Iterative update of the cell-selection network by PUF loss. d) The current and the on/off state classification of cells in the PUF box. e) Histogram for the conductance of cells in the PUF box (dashed line: a guide to distinguishing the distribution in two states). f) Conductance map of the memristor array in the PUF box.
The novel PUF architecture, NNPUF, is introduced, and our security primitive, as the result of the training algorithm, is evaluated by security metrics and a deep learning-based prediction model. The main contributions of this work are as follows. 1) The PUF architecture was constructed with deep learning training algorithms and the memristor’s stochastic nature. 2) To achieve PUF security metrics, we contrived the concept of the original loss with the memristor array, a nondifferentiable module. 3) Based on the elementary features of memristors, the building blocks for the PUF system secure randomness and reliability without requiring write operation accuracy. 4) NNPUF’s unassailable resistance against deep learning attacks was retained. 5) Analysis was conducted to determine the conditions required to maintain reliability and improve the resistance of the system.

2. NNPUF

2.1. Architecture and Mechanism

The proposed NNPUF is a multibit-to-multibit PUF that outputs a K-bit-long binary response sequence when the input is a K-bit-long binary challenge sequence. The entire NNPUF system is composed of the Cell-selection Network (left side) and the PUF Box (right side), as shown in Figure 1b. A single bit in a response sequence is gathered in two steps. First, two cell groups, each with M cells, are determined by a challenge sequence. Then, each binary for a response sequence is calculated by biasing the selected cell combinations with \( V_{\text{bias}} \) and comparing the sum of the currents flowing into each group. A detailed description of the architecture and mechanism for NNPUF is as follows.

2.1.1. Challenge

Our PUF system requires a K-bit-long-binary sequence as input, which is the challenge. We denote the set of challenge sequences as \( X \) and one sample of \( X \) as \( x \).

2.1.2. Cell-Selection Network

A K-bit-long challenge is provided to the cell-selection network (CSNet), the feed-forward neural network that returns the \( 2 \times M \times K \) output. CSNet functions as a peripheral circuit for a group of memristive crosspoint devices. By training the network (Figure 1c), the optimal performance of the NNPUF is achieved (the detailed training method is described in Section 2.2). The sigmoid function can be used as the activation for the last layer of the network and limits the output in the range of 0–1. CSNet can be constructed with a simple architecture based on fully connected layers, so the compatibility of the network with neuromorphic memory is confirmed (Section 1, Supporting Information).

2.1.3. Cell Selection

Cell selection (or cell-selection array) is the array of indices pointing to devices of the PUF box, and this integer sequence is calculated as the rounded float sequence resulting from multiplying the output of CSNet by the number of total cells in the PUF box, \( N_{\text{cell}} \). Elements in \( y \), one sample of the total cell-selection set \( (Y) \), are divided into two groups: \( A \) and \( B \). Specifically, the array with \( 2MK \) elements is sliced into \( K \) clusters. The \( K_i \) cluster is bisected into two groups \( (A_k, B_k) \) each possessing \( M \) elements, and all elements are the indices referring to corresponding cells. This split of a cell-selection sequence is formulated as

\[
A_k = \{a_{k,m}|a_{k,m} = \gamma_{2M(k-1)+m}, k = 1, 2, \ldots, K, \ m = 1, 2, \ldots, M\} \\
B_k = \{b_{k,m}|b_{k,m} = \gamma_{2M(k-1)+m+1}, k = 1, 2, \ldots, K, \ m = 1, 2, \ldots, M\}
\]

where \( y_i \) is the \( i_{th} \) element of the array \( y \). That is, the \( 2M(k-1) \) + \( m \)th element is \( a_{k,m} \), included in \( A_k \), while the \( 2M(k-1)+M+m \)th element is \( b_{k,m} \), included in \( B_k \). We denote the mapping function from \( X \) to \( Y \) as CSNet: \( X \to Y \).

2.1.4. PUF Box

As mentioned earlier, the PUF box is a basic building block for our security primitive, which is integrated with a set of memristor cells. When a PUF instance is created, the devices in this module are initialized with the conductance set \( G \). All memristive cells in the PUF box follow stochastic behavior, which is a basic characteristic of the memristor. For example, Figure 1d shows the conductance map of an exemplary PUF box, which is composed of a set of memristors with equal proportions of on/off conductance states (Figure 1e). The current levels from 16 \( \times \) 16 cells are randomly polarized, either high or low (Figure 1f). This intrinsic randomness provides an enciphering basis to the cryptographic system. Initializing conductance with stochastic behavior can encapsulate conductance information inside the PUF box and help it avoid being accessed illegally. In addition, controlling the combination of the on/off state affects the reliability of the whole PUF system (see Section 2, Supporting Information, for a more detailed description of memristive crosspoint cells in the PUF box).

2.1.5. Response

Finally, the set of response sequences, \( Z \), is generated by mapping from \( Y \), utilizing the combination of cells with conductance \( G \). We denote this conversion as PUFBox: \( Y \to Z \). For simplicity, it can be expressed as Equation (3) with respect to \( z \), a sample from \( Z \).

\[
z = \text{PUFBox}(y, G) = \text{PUFBox}(\text{CSNet}(x), G)
\]

To obtain a sense of the computing process, \( z_k \), the \( k_{th} \) binary bit of a K-bit-long response sequence is determined by

\[
z_k = \begin{cases} \frac{1}{2}(I_{A_k} > I_{B_k}) & \text{where } I_{A_k} = V_{\text{bias}} \sum_{j \in A_k} G_j, I_{B_k} = V_{\text{bias}} \sum_{j \in B_k} G_j \\ 0(I_{A_k} < I_{B_k}) & \end{cases}
\]

where \( G_j \) is the conductance of the \( j_{th} \) cell in the PUF box. \( I_{A_k} \) and \( I_{B_k} \) are currents by the conductance of cell combinations belonging to groups, \( A_k \) and \( B_k \), respectively.
Hardware architecture for our proposed design mentioned in this section is described in Section 3, Supporting Information.

2.2. Training Algorithm

2.2.1. Approach to Randomness

Our goal is for the entire PUF system to gain functionality as a hardware-intrinsic security primitive. That is, the cell selection extended via CSNet should establish an appropriate response space along with the challenge. Essentially, an appropriate response must be a random sequence for the security to be protected. To achieve this goal, guiding the network to learn the correct ways to select cell combinations is schemed out. However, it is complicated to formulate and impossible to solve the weight update by simple differentiation, as the mapping function PUFBox’s derivative cannot be defined. This elusive and evasive process that converts the challenge into the random secure response could be formulated by substituting the process in which CSNet assigns adequate cell selection to the challenge.

In other words, we defined the problem as CSNet learning how to expand the challenge to proper cell selection, leading to a safe and random response set. We approached the problem by considering the uniformity of the total response and the uniform distribution of the cell selection.

First, the uniformity of the response is one of the metrics used to evaluate the encryption performance of PUFs. To be a random and unpredictable cryptographic key, response sequences should not be biased to either 0 or 1. The uniformity of a cell selection is intended to be uniformly selected so that PUFs escape from cryptographic instability. If only the cells located in the middle indices among several cells in the PUF box are extremely selected, the cells located outside will lose the opportunity to engage in determining the response. Implicitly, concentration on a few cells is likely to attenuate randomness built on the initial stochastics of the memristive conductance. When the most frequently selected cell suffers from sudden damage to retention, it will likely threaten response reliability and built-in security. The measurement of response reliability is dealt with in Section 3.2.

2.2.2. Desirable Cell Selection

Based on the aforementioned approach, we assumed that the performance of the PUF would depend on whether CSNet learned to improve the uniformity of the response and the uniform distribution of the cell selection. This motivated us to design an objective function that helped insecure cell selection move toward secure cell selection, which brings about better response quality.

In this section, we mathematically organize how to approach the right cell selection, deriving a preferable PUF performance.

Desirable cell selection, an adjusted version of cell selection, was devised to improve poor cell selection. It contributes to enabling PUFBox to adjust weight updates and improve the performance of the system. To achieve this, we developed a weighted learning function, which contributed to deriving desirable cell selection.

Desirable cell selection, an adjusted version of cell selection, was devised to improve poor cell selection. It contributes to enabling CSNet to shift undesirable cell selection, which improves the uniformity of the total response set and forms the nonuniform distribution of the total cell-selection set.

The set of desirable cell-selection sequences, \( \mathcal{Y} \), is acquired from the mapping function Desirable: \( Y \to \mathcal{Y} \). Given that \( \mathcal{Y} \) is constituted in reference to the response set \( Z \) mapped from the cell-selection set \( Y \), a sample from \( \mathcal{Y} \), \( \mathcal{y} \), is expressed as

\[
\mathcal{y} = \text{Desirable}(y, z) = y + \Delta y
\]

where \( \Delta y \) is the difference between a cell-selection sequence and a desirable cell-selection sequence. \( \Delta y \) is of the same size as \( y \) and derived from the product of Uniformity Degree \( \mu \), and Cell Diffusion \( \gamma \). As shown in Equation (7)

\[
\Delta y = \text{Uniformity Degree}(y, z) \cdot \text{Cell Diffusion}(y, z)
\]

2.2.3. Uniformity Degree

The desirable cell selection is intended for the uniformity of the response derived from the current cell selection to get closer to its ideal mean value of 0.5. To take into account this intention, Uniformity Degree (UD) is a scalar value that incentivizes \( \Delta y \), and the formula for calculating it is as follows

\[
\text{Uniformity Degree} = \frac{\alpha}{1 + e^{-\beta \cdot \mu}} + \beta \quad \text{where } \mu = 0.5 - \langle UF \rangle
\]

where \( \langle UF \rangle \) is the average uniformity of the total response set \( Z \) and \( \mu \) indicates how close \( \langle UF \rangle \) is to its ideal value of 0.5. Showing a form similar to that of the logistic function, UD has \( \alpha, \beta, \gamma, \delta \) as constants and \( \mu \) as a variable. Applied to \( \Delta y \), UD assesses the uniformity quality of the total response set \( Z \) and regulates the magnitude of rearrangement in cell selection.

From the perspective of model training, the UD functions as an additional learning rate scheduler to stabilize learning. The uniformity quality is monitored, and the amount of update to the model is adjusted adaptively. More experimental results of training for the change of UD are provided in Section 3, Supporting Information.

2.2.4. Cell Diffusion

When selections are duplicated andlocked intensely, they should be diffused to engage uniformly in constructing the response space. Cell Diffusion (CD) is introduced for cells to be chosen evenly. CD reflects the slope of the total selection distribution and the selection difference of each cell from the least selection, and it leads CSNet learning in the desired direction.

Later, the process of obtaining CD from an example of a response sequence and the total selection distribution is described, as shown in Figure 2. 1) A response array (101000) has more 0 s than 1 s (Response in Figure 2a). 2) Therefore, to adjust the number of 0 s in a response sequence, a single
The bit, which is 0, is randomly picked up. In the figure, the fifth bit in the sequence is chosen (a red box in Figure 2a). 3) Indices of engaged cells required to decide the chosen bit are found in a cell-selection sequence. In the figure, the \( p \)th cell and \( q \)th cell engage in the chosen bit (Cell Selection in Figure 2a). 4) The total selection number of engaged cells is counted corresponding to the indices we found in the previous step (the upper distribution of Figure 2b). Using the total selection number, the selection difference from that of the lowest selected cell is calculated. Here, to ignore all bits without the chosen bit, they are replaced by zeros to minimize the effect on other bits. \( \Delta_j \) represents the selection difference of the \( j \)th cell (Selection Difference in Figure 2a).

5) The signs of the derivative of the total selection distribution at the indices of engaged cells are checked. In the figure, a positive slope is shown (\( +1 \)) at the \( p \)th cell and a negative slope (\( -1 \)) at the \( q \)th cell (Slope of Selection in Figure 2a).

6) CD is calculated by an element-wise multiplication between two arrays obtained from steps 4 and 5 using Equation (9) (Cell Diffusion in Figure 2a). To diffuse in the opposite direction of the slope, or produce an even distribution, as shown in the bottom distribution of Figure 2b, the additional minus sign comes into the equation.

\[
\text{Cell Diffusion} = -1 \ast \text{Slope of Selection} \ast \text{Selection Difference}
\tag{9}
\]

### 2.2.5. PUF Loss

We defined PUF loss, \( \mathcal{L}_{\text{PUF}} \), as the objective function of CSNet to control the uniformity of the response and the uniform distribution of cell selection. To fix the cell selection and bring it closer to desirable cell selection, \( \mathcal{L}_{\text{PUF}} \) is simply defined as

\[
\mathcal{L}_{\text{PUF}} = \mathbb{E}_{y \sim \text{cell--selection}} \|y - \hat{y}\|_2 \\
= \mathbb{E}_{y \sim \text{cell--selection}} \|y - \text{Desirable}(y, z)\|_2
\tag{10}
\]

Desirable cell selection functions as a pseudo-label to guide a sound learning process. In preliminary experiments, when the L2 norm is replaced by the L1 norm, or an absolute difference between a cell-selection sequence \( y \) and a desirable cell-selection sequence \( \hat{y} \), worse performance is observed. With respect to the challenge set \( X \) and for the mapping function CSNet: \( X \rightarrow Y \), we express the objective function, \( \mathcal{L}_{\text{PUF}} \) as

\[
\mathcal{L}_{\text{PUF}}(\text{CSNet}) = \mathbb{E}_{x \sim \text{challenge}} \| \text{CSNet}(x) - \text{Desirable} \text{(CSNet}(x), \text{PUFBox}(\text{CSNet}(x), G))\|_2
\tag{11}
\]

Finally, \( \mathcal{L}_{\text{PUF}} \) can be differentiable to update the CSNet weight, and we aim to solve

\[
\text{CSNet}^* = \arg \min_{\text{CSNet}} \mathcal{L}_{\text{PUF}}(\text{CSNet})
\tag{12}
\]

During iterations of update, the loss function helps CSNet search for the optimal point at the cell selection and response space. By monitoring PUF loss, we can interpret it as how well the response set gets randomness and how much the NNPUF performance is improved.

### 3. Evaluation

We ran experiments on PUF instances that offered a 16-bit response sequence for a 16-bit challenge sequence for fast training and experiments. NNPUF is trained following the summarized pseudocode in Algorithm 1. The size of the cell group, \( M \), is set up with 4, and therefore CSNet offers 128 outputs for 16 inputs. The PUF box, the memristor array constructed with 256 TaO\( x \) devices, is modeled by parameters and equations from Choi et al.\[31\] Exploiting HSPICE, conductance, and the corresponding current are extracted from the device in the PUF box by applying a read voltage of 0.2 V. The parameters for UD are adopted, as shown in Figure S5b, Supporting Information, Section 4. To tune the weight of CSNet, stochastic
Algorithm 1. Training NNPUF with \( \mathcal{L}_{\text{PUF}} \).

Input: Input Challenge: \( X \),
- Initial Weight of Cell-selection Network: \( W \),
- Conductance Set of the Memristor Array in PUF Box: \( G \),
- Learning Rate: \( \alpha \), Decay Rate: \( d \)

1: for \( \text{iter} = 1, \ldots, n \) do
2. \( Y_{\text{iter}} \leftarrow \text{CSNetw}(X) \) \hspace{1cm} //by Equation (1–2)
3. \( Z_{\text{iter}} \leftarrow \text{PUFBox}(Y_{\text{iter}}, G) \) \hspace{1cm} //by Equation (3–4)
4. \( \Delta Y_{\text{iter}} \leftarrow \text{Uniformity Degree} \left( Z_{\text{iter}} \right) \ast \text{Cell Diffusion} \left( Y_{\text{iter}}, Z_{\text{iter}} \right) \) \hspace{1cm} //by Equation (7–9)
5. \( \hat{Y}_{\text{iter}} \leftarrow Y_{\text{iter}} + \Delta Y_{\text{iter}} \) \hspace{1cm} //by Equation (6)
6. \( \mathcal{L}_{\text{PUF}} \leftarrow \| Y_{\text{iter}} - \hat{Y}_{\text{iter}} \|_2 \) \hspace{1cm} //by Equation (10–11)
7. \( W \leftarrow \text{SGD} \left( V_{W}, \mathcal{L}_{\text{PUF}}, \alpha, d \right) \) //update CSNet weight to solve Equation (12)

end for

gradient descent (SGD) is adopted as an optimizer with \( 10^{-3} \) for the learning rate decaying at a fixed rate of \( 5 \times 10^{-5} \).

3.1. Analysis of Training

CSNet targets two states until the training ends. The first target is uniformity in the distribution of selected cells. The second is that the uniformity of the total response approaches 0.5. In this section, we analyze how the training with PUF loss proceeds and the result of learning, the uniformity of the NNPUF response, and the uniform distribution of the cell selection.

PUF loss imposes a strict condition on CSNet to select cells uniformly while training, as shown in Figure 3a,b. Figure 3a shows that the training narrows the gap between the selection numbers of the most selected cell and the least selected cell. More specifically, the diffusion process of selection distribution is shown in Figure 3b. The first distribution map (far left) in Figure 3b displays the initial distribution of the total cell selection. Before training starts, the selections are normally distributed or intensely concentrated in the middle of the indices if the last layer of the CSNet is the sigmoid function.

To spread out selections of cells, PUF loss imposes large penalties on the concentrated selections of cells. The degree of penalty imposed depends on the learning rate and the hyperparameter of training. If the learning rate is too small, the diffusion of cell selection proceeds slowly due to a small penalty. If the learning rate is too large, the cells are biased toward both ends with a severe penalty. Therefore, the appropriate value should be configured for the learning rate. With a suitable learning rate value, the cell selection suffers from an early irregular diffusion phase as soon as the training begins (the second distribution map in Figure 3b). Subsequently, whole selections consistently diffuse as long as the CSNet trains, and then all cells participate almost equally in determining the response (the remaining maps in Figure 3b). Supplementary Section 4 illustrates in more detail how the learning rate affects NNPUF learning.

As shown in Figure 3c, PUF loss decreases gradually as training progresses after exponential decay (inset of Figure 3c) at an early irregular diffusion phase. The response difference (\( \Delta z \)) for a response sequence, of which the length is 16, decreases similar to the shape of the loss (Figure 3d). When the training begins, PUF loss forces the network to change response sequences heavily.

Figure 3. a) Maximum and minimum selection among all cells in the PUF box throughout training. b) Distribution maps showing changes in the cell selection. c–e) Training curves: (c) the logarithmic value for PUF loss (\( \mathcal{L}_{\text{PUF}} \)), (d) the response difference, and (e) the distance of uniformity from 0.5 (ideal value for the uniformity). The loss, uniformity, and difference decrease consistently as training progresses. After cells are adequately diffused, training enters stability.
so that the cell selection diffuses to seamlessness, and about half of the response set is reversed. Throughout the training, the reversion rate falls progressively. Figure 3e shows the uniformity change during iterations. Due to the comparability of the graphs in Figure 3c–e, it can be said that the loss, the response difference, and the uniformity tightly correlate with each other.

The overall declines indicate that PUF loss performs weight updates for CSNet to explore the challenge and cell selection space for two main objectives that the network desires to accomplish for the related response set. Each objective is given a different weight depending on the context of training. For example, in the initial stage of learning, selections do not spread out as much, so the loss is mainly minimized by diffusing the selections of cells. We found that learning enters a stable phase when the uniformity of the total response oscillates around an ideal 0.5 value after selections of cells are distributed appropriately.

While solving the optimization problem, the loss undergoes noisy curves (Figure 3c). The adoption of the SGD optimizer can be a factor for noisy reduction. SGD updates the network frequently with high variance, and this causes considerable fluctuation on the loss. However, by applying additive learning rate scheduling with slow decaying, the CSNet converges to the minimum point for the optimization problem, avoiding oscillation around local minimas.\footnote{32,33} As PUF loss is designed considering a nondifferentiable function, it inevitably relies on some accidents. This involves some noise on graphs, but it is minor enough to ignore.

3.2. Characterization of PUF Security Metrics

To numerically evaluate the response attained from an NNPUF instance trained with PUF loss, we measured uniformity (UF), diffuseness (DF), uniqueness (UQ), and reliability (bit error ratio; BER). These parameters are widely used metrics for evaluating the performance of PUFs. UF measures the balance between 0 and 1. DF measures the difference in two response sequences corresponding to two different challenge sequences. For the 16-bit-long binary response set of the observed NNPUF, the UF and DF achieved 50.00 ± 12.49\% and 50.00 ± 12.50\%, respectively (Figure 4a,b). The two metrics are remarkably close to an ideal mean and an ideal standard deviation for the random binary sequence (≃N(0.5, 0.125)). We also found that NNPUF achieved near-ideal UF and DF performance with different lengths of the challenge (e.g., 8, 32, 64, 128) by training with PUF loss. UQ measures the discrepancy in two response sequences generated from different PUF instances for the same challenge sequence. In the experiment, we evaluated the UQ by training 10 NNPUF instances. The average UQ of 10 NNPUF instances for the 16-bit-long response sets was 49.98 ± 12.49\%, as shown in Figure 4c. Even though the conductance set in the PUF box is the same, a different training result, or a set of CSNet weights, generates a unique response set. As the three PUF security metrics are near ideal, the response sequences of NNPUF are almost uncorrelated random binary sequences.

The bit error ratio (BER), the fourth metric, measures the discrepancy between two response sequences generated at different times for the same challenge sequence from one PUF instance, and it is calculated using the intratrial Hamming distance. Put simply, BER, or reliability, assesses robustness against innate fluctuation in PUF, such as measurement error or device conductance variation. To investigate proper configuration deriving the performance of NNPUF, reliability according to different settings is evaluated. In the PUF box, as an entropy source from the set of memristive cells, some physical features evidently affect the reliability of NNPUF. The sneak path, the unexpected path for the current, is one of those features that need to be reduced. It substantially degrades the efficiency of read operation in the memristor-based crossbar array. The number of neighboring cells and interconnected wire resistances in the crossbar-structured circuit are attributed to increasing sneak path leakage (Figure 5a). The effect of the sneak path on the reliability of the system is shown in Figure 5b. To regulate the number of neighboring cells, the devices are clustered as a subarray. A smaller subarray size with fewer adjacent devices reduces not only the interference currents of the sneak path but also the BER of the whole PUF system. The lower wire resistance decreases BER as well. If relatively high conductive memristors are used, such as the TaO\textsubscript{2} device we have chosen, the effect of wire resistance must be minimized. Considering the sneak path, constituent cells in the PUF box must be isolated from each other. This can be implemented by gating the memristors on the crossbar array using transistors or selectors.\footnote{34}

We identified that the characteristics of each memristive cell control the reliability of the PUF system. First, the on/off ratio of

![Figure 4](image-url)  
**Figure 4.** The measurement of security metrics for NNPUF configured with $K=16$, $M=4$, and $N_{cell}=256$. a) UF of the response. A normalized Hamming weight or a Hamming distance to a zero sequence is used to get UF. b) DF of the response. A Hamming distance between two response sequences is used to get DF. c) UQ of the response, calculated by the inter-PUF Hamming distance.
the memristor, the ratio of current in the high-resistance state (HRS), and the low-resistance state (LRS) must be sufficiently high to maintain the robustness of NNPUF CRPs. By manipulating the modeling equation parameters of the TaOₓ memristor, we simulated it with varying on/off ratios. Potentiation and depression of manipulated devices are shown in Figure 5c; analog current on/off ratios are about 2, 4, 8, 16, 32, and 64. The on/off ratio effect on the response BER is noticeable, as shown in Figure 5d. As the current on/off ratio is higher, the result is more robust and even protected from excessive error. An improved on/off ratio contributes to enhancing readout margins, which serves to enlarge the gap between two current sums when calculating the response.

The second characteristic is the conductance distribution of the memristor by stochasticity. Stochastic behavior of memristor, which is related to the intrinsic randomness of the NNPUF design, is also a factor in the reliability. We simulated with five probabilities to program the memristor: p = 0%, 25%, 50%, 75%, and 100% (see Figure S2, Supporting Information Section 2). BER results from the PUF box constructed with those programming probabilities are shown in Figure 5e. The batch initialized with a set of memristors that have 50% switching probability (# On state/# Off state = 128:128) holds the most robust response set. The reason is similar to that of the on/off ratio. The conductance distribution with half on and half off widens the gap between the two current sums. To reduce BER, the bias condition that retains such switching probability should be obtained for the memristor composing the PUF box.

An uneven distribution of cell-selection might lead to high BER, or low reliability. For this reason, we designed a simulation in which cell concentration and fluctuation degree affect reliability. We instantiated and trained distinct NNPUFs. Each NNPUF instance shows a different cell-selection distribution. The selection number of the most selected cells varies between about 7000 and 20,000. As we wanted to examine the effect of selection concentration, we controlled other settings to be as similar as possible. For example, the index of the most selected cell coincides, and the selection number of the lowest selected cell is not zero. As more selection is focused on one cell, the BER becomes higher (Figure 5f). In addition, the conductance fluctuation definitely damages the reliability of the PUF. How much influence the heterogeneity in the distribution has and the degree of standard error varies from distribution to distribution, but the trend the graph illustrates is enough to say that uneven selection...
distribution might lead a PUF to cryptographic instability. On the contrary, NNPUF accomplished the uniform distribution, as shown in Figure 3a,b, and it is away from the instability.

3.3. Unpredictability

The total response of NNPUF has a tremendous number of potential cell combinations, so it cannot be predicted from cell combinations (Section 5, Supporting Information). In addition to mathematical proof, it is necessary to check whether an adversary can model all CRPs by acquiring some of them. If an adversary can observe the regularity, an entire response set can be guessed using only a partial set of the response. Figure 6a shows not only that each bit gets the same ratio between one and zero, but also that the challenge and response do not coincide on each bit. Moreover, we modeled a kind of side-channel attack by analyzing the power consumption distribution based on the cell combination that determines the response. The powers used to compute for 0 and 1 have exactly indistinguishable distributions. There is no connection between power consumption and response value, as shown in Figure 6b. To certify, we further tested the NIST and calculated the autocorrelation factor (ACF). The NIST test and the ACF calculation are commonly used verification methods to evaluate the randomness of a sequence. The randomness of the NNPUF’s response set is supported by passing two examinations successfully (see Section 5, Supporting Information).

Supposing that there is a high correlation between the challenge and response set, an adversary can speculate from a partial set of pairs. Despite the foregoing saying that CRPs of NNPUF have no clear and distinguishable relationship, we modeled another type of deep learning attack to validate the prospect of invisible correlation. We constructed the DNN prediction model by tuning some hyperparameters and manipulating components from the architecture proposed by Awano et al. [13] There are two sets of CRPs: nonrandom sequences and NNPUF response sequences. The former set, which has over 10 000 sequences, is prepared with two conditions: the uniformity of the sequence is poor (UF < 0.25, 0.75 < UF), or the step of random walk is too distant from zero (hits ±6 or 0)

Figure 6. a) Distribution of 1, 0, and discrepancy between CRPs. Blue represents the ratio of 1 in each response bit index, and red represents the ratio of 0 in each response bit index. Both values show that they are close to 50%. Green represents the difference between CRPs. b) Distribution of power consumption for the response. The similarity between the power consumption (P = CV²) of zeros and ones, which are calculated from the sum of conductance combinations and read voltage (=0.2 V). c) Prediction accuracy by deep learning modeling attack. In the case of a nonrandom response set, the deep learning model’s prediction accuracy increases as the ratio of the training set increases. However, the response of NNPUF shows a prediction close to 50%. All test experiments were executed with 200 test sets. d) Response prediction accuracy from the cell selection and challenge according to uniformity and diffuseness. e) Response prediction accuracy from the cell selection with varying M and Ncell. We trained CSNets with different M and Ncell. Each NNPUF instance generated response sets sharing a similar quality (UF & DF ≈ 50.0%). The selections were also diffused almost uniformly. The accuracy is near 50% as M and Ncell increase.
for the nonrandom sequence increases to about 90%, the prediction accuracy for the NNPUF response remains constant near 0.5. As the ideal accuracy is 0.5, the randomness of the NNPUF response was demonstrated. Further demonstrations using other DL-based modeling attacks such as long short-term memory\cite{10} and generative adversarial network\cite{11} are described in Section 6, Supporting Information.

The NNPUF decides the response through challenge and cell selection, which is the intermediate procedure of producing the output sequence. Thus, the response prediction from the cell selection needed, and it is expected to be better than that of the challenge. We organized four different NNPUF instances to collect response sets and corresponding cell-selection sets. Each has a response set with a different mean value for both UF and DF. Figure 6d shows, as we expected, the accuracy from cell selection is slightly higher than that from the response. The difference between the two accuracy values might depend on how the CRPs are arranged. However, we repeatedly found that the prediction from the cell selection was marginally easier. This is probably because the relationship between the cell selection and response was larger than that between the challenge and response. Figure 6d also shows that uniformity and diffuseness of response set affect the prediction score. When the metrics are far from their ideal values, the response set can be easily predicted. Considering that the CSNet learns how to assign cells to get optimal performance of PUF such as uniformity through the update of weights, it can be said that training with PUF loss makes NNPUF more resilient to DNN-based modeling attacks. Moreover, the variables that have impacts on the reliability, as shown in Figure 5, should be controlled to maintain the resilient randomness that deceives predictors.

The cell selection, positioned more closely to the response than the challenge, should have a stronger safety feature, or be a more haphazard sequence. The degree of the cell-selection randomness is bound up with the size of the cell group (M) and the number of cells in the memristor array of the PUF box (N_{cell}). The cell group size is involved with the number of cells that participate in calculating one bit of a response sequence, so if it becomes larger, the relationship between the cell selection and response gets more nonlinear. A larger number of cells in the PUF box offer a wide range of options for a single element in a cell-selection sequence to select along to a challenge sequence. As shown in Figure 6e, increased M and N_{cell} make the prediction problem harder. The prediction model performed well when M and N_{cell} were small, but when they were large, or the number of cases varied, the model did not work correctly. However, when variables are larger than certain values, it seems that the level of unpredictability is almost the same. CSNet with M equal to four deceives the prediction model to a nearly identical level as that with M equal to eight, which implies that M is enough to be around four if CSNet is designed considering prediction. An NNPUF instance with a small N_{cell} can be considered an instance with a large N_{cell} and a nonuniform distribution of selections. In other words, if few cells are selected for producing response sequences, net N_{cell} decreases, which leads to NNPUF becoming predictable. Thus, diffusion of cells can be seen as an additional factor of NNPUF unpredictability.

4. Conclusion

In this article, we suggest a PUF system made up of cryptographic hardware with a memristor array called NNPUF. Our proposed hardware can be easily constructed with various types of memristive devices which show stochastics. We also solved an optimization problem to improve NNPUF performance and satisfy security functions by introducing PUF loss, a new loss function regarding a nondifferentiable module. The response set generated through this specialized training algorithm showed excellent performance in UF, DF, and UQ based on the intrinsic nature of the memristor array. The new secure response can be reconfigured via simple retraining of CSNet weights. Furthermore, the resistance to deep learning attacks, despite simple memristor array configurations, is highly competitive compared with other existing PUFs with vulnerability. We evaluated various settings for the NNPUF to maintain the reliability and unpredictability of its response. We believe that the proposed design has great potential for further research; for example, NNPUF can respond to more complicated input images, such as fingerprints. By substituting convolutional neural networks for CSNet, the image signal as a challenge of NNPUF is scaled down and reproduces a cell-selection sequence. A more complex structure with neuromorphic devices is expected to improve NNPUF to be more invulnerable against adversaries, considering that the feed-forward network is replaceable by other neuromorphic memory as well as the memristor (see Section 1, Supporting Information).

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

The authors acknowledge support from the National Research Foundation of Korea (grant Nos. 2020R1C1C1007464, 2019M3F3A1A02072336, and 2019M3F3A1A01074452). This work was funded by Nanomedical Devices Development Project of NNFC (CSM2103M001). The HSPICE (SYNOPSIS) was supported by the IC Design Education Center (IDEC), Korea.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

Data available on request from the authors.

Keywords

deep learning, hardware security, memristors, physically unclonable functionss
[1] G. E. Suh, S. Devadas, in 2007 44th ACM/IEEE Design Automation Conf., IEEE, San Diego, CA, USA 2007, pp. 9–14.
[2] O. Körmerling, M. G. Kuhn, in Proc. USENIX Workshop on Smartcard Technology, USENIX Association, Berkeley, CA 1999, p. 2.
[3] G. E. Suh, C. W. O’Donnell, I. Sachdev, S. Devadas, in 32nd Int. Symp. on Computer Architecture, IEEE Computer Society, Madison, WI, USA 2005, pp. 25–36.
[4] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, S. Devadas, IEEE Trans. Very Large Scale Integr. Syst. 2005, 13, 1200.
[5] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, S. Devadas, IEEE Trans. Inf. Forensics Secur. 2013, 8, 1876.
[6] G. Hospodar, R. Maes, I. Verbauwhede, in 2012 IEEE Int. Workshop on Information Forensics and Security, IEEE, Piscataway, NJ 2012, pp. 37–42.
[7] X. Xu, U. Rührmair, D. E. Holcomb, W. Burleson, in Radio Frequency Identification (Eds: S. Mangard, P. Schaumont), Springer International Publishing, Cham 2015, pp. 3–16.
[8] S. Bhunia, M. Tehranipoor, Hardware Security: A Hands-on Learning Approach, Morgan Kaufmann, Amsterdam, Netherland 2018.
[9] C. Helmeier, C. Boit, D. Nedospasov, J.-P. Seifert, in 2013 IEEE Int. Symp. on Hardware-Oriented Security and Trust, 2013, pp. 1–6.
[10] U. Rührmair, H. Busch, S. Katzenbeisser, in Towards Hardware-Intrinsic Security - Foundations and Practice (Eds: A. R. Sadeghi, D. Naccache), Springer Berlin Heidelberg, Berlin, Heidelberg 2010, pp. 79–96.
[11] M. Roel, Physically Unclonable Functions : Constructions, Properties and Applications, Katholieke Universiteit Leuven, Leuven, Belgium 2012.
[12] Y. Gao, S. F. Al-Sarawi, D. Abbott, Nat. Electron. 2020, 3, 81.
[13] Y. Pang, B. Gao, B. Lin, H. Qian, H. Wu, Adv. Electron. Mater. 2019, 5, 1800872.
[14] J. Rajendran, G. Rose, R. Karri, M. Potkonjak, Nano-PUF: A Memristor-Based Security Primitive, IEEE, Amherst, MA, USA 2012.
[15] G. S. Rose, N. McDonald, L.-K. Yan, B. Wysocki, in 2013 IEEE/ACM Int. Conf. Computer-Aided Design, IEEE, San Jose, CA, USA 2013, pp. 830–833.
[16] P. Koebel, U. Kocabas, A.-R. Sadeghi, in 2013 Design, Automation and Test in Europe Conf. Exhibition, IEEE, Grenoble, France 2013, pp. 428–431.
[17] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, D. Abbott, Sci. Rep. 2015, 5, 12785.
[18] S. Gaba, P. Knag, Z. Zhang, W. Lu, in 2014 IEEE Int. Symp. Circuits Systems, IEEE, Melbourne, VIC, Australia 2014, pp. 2592–2595.
[19] R. Liu, H. Wu, Y. Pang, H. Qian, S. Yu, in 2016 IEEE Int. Symp. on Hardware Oriented Security and Trust, IEEE, McLean, VA, USA 2016, pp. 13–18.
[20] R. Zhang, H. Jiang, Z. R. Wang, P. Lin, Y. Zhuo, D. Holcomb, D. H. Zhang, J. J. Yang, Q. Xia, Nanoscale 2018, 10, 2721.
[21] A. Mazady, M. T. Rahman, D. Forte, M. Anwar, IEEEJ. Emerg. Sel. Top. Circuits Syst. 2015, 5, 222.
[22] W. Che, J. Plusquelliel, S. Bhunia, in 2014 IEEE/ACM Int. Conf. Computer-Aided Design, IEEE, San Jose, CA, USA 2014, pp. 148–153.
[23] G. S. Rose, C. A. Meade, in 2015 52nd ACM/EDAC/IEEE Design Automation Conf., IEEE, Amherst, MA, USA 2015, pp. 1–6.
[24] J. Kim, T. Ahmed, H. Nili, J. Yang, D. S. Jeong, P. Beckett, S. Sriman, D. C. Ranasinghe, O. Kavehei, IEEE Trans. Inf. Forensics Secur. 2018, 13, 437.
[25] H. Nili, G. C. Adam, B. Hoskins, M. Prezioso, J. Kim, M. R. Mahmoodi, F. M. Bayat, O. Kavehei, D. B. Strukov, Nat. Electron. 2018, 1, 197.
[26] J. Rajendran, R. Karri, J. B. Wendt, M. Potkonjak, N. McDonald, G. S. Rose, B. Wysocki, Proc. IEEE 2015, 103, 829.
[27] M. Uddin, M. B. Majumder, G. S. Rose, IEEE Trans. Nanotechnol. 2017, 16, 396.
[28] X. Xu, W. Burleson, in Design, Automation & Test in Europe Conf. Exhibition, Springer, Cham, Manhattan, New York, USA 2014, pp. 1–6.
[29] U. Chatterjee, R. S. Chakraborty, J. Mathew, D. K. Pradhan, in 2016 29th Int. Conf. VLSI Design, 2016 15th Int. Conf. Embedded Systems, IEEE, Kolkata, India 2016, pp. 535–540.
[30] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair, The Bistable Ring PUF: A New Architecture for Strong Physical Unclonable Functions, IEEE, San Diego CA, USA 2011.
[31] S. Choi, P. Sheridan, W. D. Lu, Sci. Rep. 2015, 5, 10492.
[32] R. Kleinberg, Y. Li, Y. Yuan, CoRR 2018, abs/1802.0.
[33] K. You, M. Long, M. I. Jordan, J. Wang, CoRR 2019, abs/1908.0.
[34] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain, K. N. Salama, Microelectronics J. 2012, 44, 176.
[35] H. Awano, T. Iizuka, M. Ikeda, in 2019 IEEE Int. Symp. Circuits Systems, IEEE, Sapporo, Hokkaido, Japan 2019, pp. 1–4.
[36] A. Graves, CoRR 2013, abs/1308.0.
[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Adv. Neural Inf. Process. Syst. 2014, 27.