IEICE TRANS. FUNDAMENTALS, VOL.E106–A, NO.5 MAY 2023

PAPER

BayesianPUFNet: Training Sample Efficient Modeling Attack for Physically Unclonable Functions

Hiromitsu AWANO[†a)] ***and*** **Makoto IKEDA**[††], ***Members***

SUMMARY This paper proposes a deep neural network named *BayesianPUFNet* that can achieve high prediction accuracy even with few challenge-response pairs (CRPs) available for training. Generally, modeling attacks are a vulnerability that could compromise the authenticity of physically unclonable functions (PUFs); thus, various machine learning methods including deep neural networks have been proposed to assess the vulnerability of PUFs. However, conventional modeling attacks have not considered the cost of CRP collection and analyzed attacks based on the assumption that sufficient CRPs were available for training; therefore, previous studies may have underestimated the vulnerability of PUFs. Herein, we show that the application of Bayesian deep neural networks that incorporate Bayesian statistics can provide accurate response prediction even in situations where sufficient CRPs are not available for learning. Numerical experiments show that the proposed model uses only half the CRP to achieve the same response prediction as that of the conventional methods. Our code is openly available on https://github.com/bayesian-puf-net/bayesian-puf-net.git.

key words: *Bayesian deep learning, modeling attacks, physically unclonable functions*

1. Introduction

In recent years, the evolution of cheap devices such as Raspberry-Pi has pioneered a new paradigm of internet called *Internet of Things (IoT)*, which is expected to create new opportunities for direct integration of the physical world with the cyber world. Because several devices are expected to get involved in the world of open networks, security will remain an important technological aspect in the IoT era. Among security measures, device and/or user authentication is the most fundamental technology because identification is part of almost all communication protocols. Until recently, the best method to implement device authentication has been to store a secret key in non-volatile memory (NVM). However, fabrication of CMOS logic with dedicated NVMs requires additional processes, which increases the device cost. Moreover, the stored secret key is vulnerable to physical attacks, and hence, the device is also required to be designed with an active tamper protection/detection circuit, which further increases the device cost.

Physically unclonable functions (PUFs) have attracted increasing attention as a promising alternative to the secret-key-based device authentication, which utilizes a manufacturing variability of transistors as the secret key [1]. Due to the miniaturization of transistors, variations in the number of doping ions and/or atomic level bumps on the gate electrode result in significant threshold voltage (V_{TH}) variations. The V_{TH} variability is an inherent characteristic of the transistor which cannot be replicated even by the manufacturer of the PUF, making it a unique and unclonable "fingerprint" of each chip.

Typically, a PUF instance takes an n-bit signal called "challenge", and outputs an m-bit signal called the "response." Hence, the PUF can be viewed as a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}^m$. This mapping should be unique and stable despite the variations in the operating conditions. In addition to these requirements, recent progress on machine learning algorithms poses a new security threat on PUFs in the form of "modeling attack tolerance." Because a challenge and the corresponding response are transmitted via an open network, there is a risk of challenge-response pairs (CRPs) being collected by an adversary, who can recover the unique mapping f using a suitable machine learning algorithm by collecting enough CRPs.

Among the numerous PUF implementations on silicon, the *Arbiter PUF* (APUF) is one of the most popular realization [2]. An APUF is composed of multiple delay elements and an arbiter. The n-bit challenge activates a set of delay elements along which a signal propagates. Then, the arbiter digitizes the signal propagation delay to yield a 1-bit response. Although APUFs are superior in their simplicity, design regularity, low hardware overhead, and affinity with VLSI implementations, they are rarely used on their own because of their severe vulnerability to modeling attacks. Hence, recent studies have proposed to combine multiple APUFs as a building block to generate secure APUF compositions: XORPUF [3], Lightweight Secure PUF (LSPUF) [4], Multiplexer PUF (MPUF) [5], and Interpose PUF (IPUF) [6]. Although several studies have reported successful modeling attacks even on these secure APUF compositions, many of them assume that the adversary has access to auxiliary information such as the side channel [7].

Black-box attacks on APUF compositions are known to be difficult even with the help of deep neural networks (DNNs). [8] proposed a two-stage algorithm where denoising autoencoders first extract informative features from the challenge input, and then use logistic regression (LR) to predict the corresponding response signal. The drawback of this method is that feature extraction and classification

Manuscript received May 21, 2022.
Manuscript revised September 26, 2022.
Manuscript publicized October 31, 2022.

[†]The author is with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
[††]The author is with the Graduate School of Engineering, The University of Tokyo, Tokyo, 113-0032 Japan.
a) E-mail: awano@i.kyoto-u.ac.jp

DOI: 10.1587/transfun.2022EAP1061

are conducted separately; hence, the extracted features may not be suitable for response prediction. Further, DNN-based modeling attacks on more challenging APUF compositions such as 5-XOR APUF or LSPUF have not been reported. Recently, an end-to-end method, where both feature extraction and classification are integrated into a single DNN and are jointly trained via a back propagation (BP) algorithm, is applied for modeling attack on a double arbiter PUF (DAPUF), which is a APUF composition [9], [10]. Comprehensive analysis has been done in [11], where successful modeling attacks on four APUF compositions, i.e., XOR APUF, LSPUF, MPUF, and IPUF, have been reported.

Recent intensive research has revealed the modeling attack vulnerabilities of APUF compositions. However, DNNs require a large number of CRPs to achieve high accuracy in training, and thus existing studies may be over-estimating the number of CRPs needed for a successful attack. For example, achieving 98.2% prediction accuracy of 5-XOR APUF responses requires 145k CRPs. Assuming that an edge device makes a single authentication every hour using the 5-XOR APUF, it would take about 16.5 years to collect 145k CRPs, so even if an attacker had intercepted all CRPs transferred during the period, the probability of a successful attack would be low and thus the same PUF instance could continue to be used. However, if the same prediction accuracy could be achieved with half the number of CRPs, the life span in which the same PUF instance could be safely used would be halved, to about eight years. Therefore, in order to accurately estimate the security of PUF, it is necessary to devise an attack method that requires only fewer CRPs.

In this paper, we propose a training-sample efficient DNN-based modeling attack on APUF compositions. When the number of training examples is not enough compared to the number of learnable parameters, the DNN can easily learn a dictionary-like mapping between the challenge and the corresponding response without recovering the hidden mapping f. Hence, the trained DNN will exhibit perfect performance only on the training samples; however, its accuracy of predicting responses corresponding to previously unseen challenges, is expected to be poor. We solve this problem by employing a "Bayesian" neural network (BNN). BNN is a realization of ensemble machine learning where multiple predictors are combined to yield an accurate prediction. Contrary to traditional DNNs, where model parameters are fixed after training, BNN treats model parameters as "random" variables, each of which has a specific statistical distribution. Hence, the inference of BNN is performed by "sampling" parameters from the trained distribution and preparing multiple DNN instances whose outputs are aggregated. By taking the aggregate of multiple predictions, BNNs are known to be prone to overfitting, even in the absence of sufficient training data.

The following summarize our contributions:

- To the best of our knowledge, **this is the first modeling attack applicable in cases when the number of CRPs is small.**
- We demonstrate a successful attack on an APUF, and on a wide variety of APUF compositions, including 5-XORPUF, MPUF, LSPUF, and IPUF for 64-bit challenge sizes.
- Unlike conventional DNN-based modeling attacks, our method requires only a small portion of CRPs, and it does not rely on any auxiliary information such as side-channel information.
- We let the BNN self-learn features, and hence, our method does not require any manual feature engineering for achieving high accuracy, as was done in [8].
- We have made our source code publicity available online.

The structure of this paper is as follows. In Sect. 2, the background and purpose of this study are described. In Sects. 3 and 4, the details of BayesianPUFNet is provided, followed by experimental results using synthetic CRPs provided in [11] for demonstrating the performance of BayesianPUFNet. Then, in Sect. 5, we show the performance of BayesianPUFNet using CRPs of a DAPUF simulated with the SPICE simulator. Finally, concluding remarks are provided in Sect. 6.

2. Preliminaries

2.1 Physically Unclonable Function

PUFs can be categorized into two groups: strong and weak PUFs. The strong PUFs act like mathematical functions that take in a multi-dimensional vector called "challenge" and output a corresponding value called "response." To be useful for device authentication, the response of the strong PUF should be unpredictable; i.e., even if an adversary Eve has a large subset of challenge and response pairs (CRPs), it cannot construct a mathematical model to predict the response of the PUF for unseen challenge input. Further, many strong PUFs provide exponentially large CRP space with the challenge bit width to make it difficult for the adversary to collect CRPs within a realistic time. On the other hand, weak PUFs have very few or only one CRP per PUF instance, which is used as a secret key storage, currently realized using flash or other non-volatile memories. Because weak PUF responses are not intended to be output outside the chip, this paper deals with machine learning attacks on strong PUFs. We describe typical circuit schemes for strong PUFs below.

2.1.1 Arbiter PUF

Figure 1 shows a typical circuit structure of Arbiter PUF (APUF), which is composed of a number of multiplexers (MUXes) and an arbiter. Each MUX allows two input signals to pass without changing the lanes when "0" is given as the challenge bit. Otherwise, inputs at the top and bottom signal lanes are steered to the bottom and top output, respectively, to achieve lane swapping. Even though MUXes are symmetrically designed, signal propagation delays of the

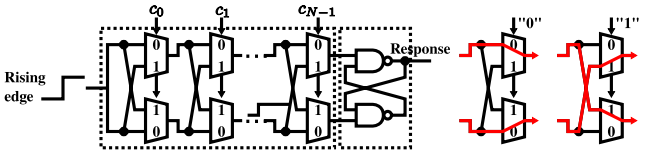

Fig. 1 Circuit structure of APUF.

straight and crossed paths are slightly different due to the V_{TH} variation. Hence, even if the same rising edge signal is given to the input of the MUX-chain, the small differences in signal propagation delay accumulate and result in noticeable differences at the output, which is finally digitized by the arbiter equipped at the end of the MUX-chain.

Because signal propagation delay in digital circuits have a strong linear correlation with variations of underlying physical characteristics such as V_{TH}, the delay difference at the end of the MUX chain can be modeled by a simple linear regression model:

$$\Delta(\boldsymbol{c}) = \boldsymbol{w}^T \boldsymbol{\Psi}, \tag{1}$$

where $\boldsymbol{w} = (w_1, w_2, \cdots, w_N)$ is an N-dimensional parameter vector which reflects the variations of transistors and should not be disclosed. $\boldsymbol{\Phi} = (\Phi_1, \Phi_2, \cdots, \Phi_{N+1})$ is the $(N+1)$-dimensional parity vector whose i-th element is given by:

$$\Phi_i = \begin{cases} 1 & (i = 1) \\ \prod_{j=i}^{N} (1 - 2c_j) & otherwise, \end{cases} \tag{2}$$

where c_j is the j-th bit of the challenge input. Hence, modeling attack of APUFs essentially involves predicting the unknown parameter $\boldsymbol{w}$ given by the CRPs. This problem is known as a linear classification problem, which is quite popular in machine learning (ML) domain for which several efficient algorithms have been proposed, e.g., support vector machine (SVM) or logistic regression (LR). For example, [12] have demonstrated that by training only 1.39×10^{-14}% of CRPs of a 64-stage APUF, responses for unseen challenges can be predicted with an accuracy of 99%. Hence, stand-alone APUF is rarely used in a practical device authentication system.

2.1.2 XOR Arbiter PUF (XOR APUF)

To improve the machine learning attack tolerance of APUF, applying XOR to the output of all x APUFs that take the same challenge input was proposed in [3]. Figure 2 shows an x-XOR APUF whose mathematical model is given by:

$$\mathrm{res}_{\mathrm{XOR}} = \prod_{i=1}^{x} \mathrm{sign}\left(\boldsymbol{w}_i^T \boldsymbol{\Phi}_i\right) = \mathrm{sign}\left(\prod_{i=1}^{x} \boldsymbol{w}_i^T \boldsymbol{\Phi}_i\right). \tag{3}$$

From Eq. (3), we can see that by increasing x, the number of modeling parameters, $\boldsymbol{w}_i^T$, will increase linearly, and hence, exponentially large number of training data and training time will be required while using the traditional machine learning algorithms. Although [12] showed that modeling attack

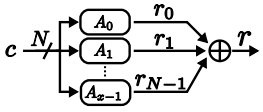

Fig. 2 XOR APUF.

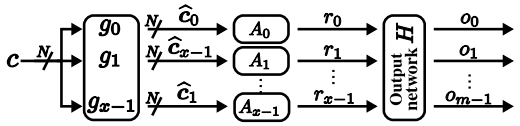

Fig. 3 Circuit structure of LSPUF.

on x-XOR APUF will be infeasible when $x \geq 6$, [13] reported that by carefully re-implementing the LR algorithm, they could break 9-XOR APUF. However, a large training dataset (e.g., 350 million CRPs to successfully break the 9-XOR APUF) and computational resources to memorize such a large training dataset were required. Another approach to break x-XOR APUF is to exploit auxiliary information such as power or current consumption. In this direction, [7] reported that with the help of additional information, up to 16-XOR APUF can be broken. However, the adversary must physically access the target PUF instances, which might be impractical.

2.1.3 Lightweight Secure PUF (LSPUF)

Along with taking XOR of the output of multiple APUFs, inserting one-way hash functions immediately before the APUF challenge inputs to prevent attackers from controlling the challenges directly was proposed in [4]. A PUF named lightweight secure PUF (LSPUF) is composed of three building blocks — (i) input hashing network, (ii) output hashing network, and (iii) APUFs. Figure 3 illustrates an LSPUF composed of x APUFs. It first converts the challenge input $\boldsymbol{c}$ to internal challenges $\widehat{\boldsymbol{c}}_i$ using input hashing network g_i for $i = 1, 2, \cdots, x$. Then, the $\widehat{\boldsymbol{c}}_i$ are directed to the corresponding APUFs to yield internal response outputs r_i. Finally, $\boldsymbol{r} = (r_1, r_2, \cdots, r_x)$ is passed to the output hashing network to yield an m-bit response output $\boldsymbol{o} = (o_1, o_2, \cdots, o_m)$ according to the formula:

$$o_i = \bigoplus_{i=1}^{m} r_{((i+s+j) \mod x)}, \tag{4}$$

where s is the design parameter that controls the behavior of the output network. Owing to the input and output networks, LSPUF can achieve enhanced ML attack tolerance. However, the response of LSPUF still suffers from the reliability issue of each elemental APUF; hence, x cannot be made large in practice.

2.1.4 Multiplexer PUF

While combining multiple XOR APUFs increases ML attack tolerance, the LSPUF suffers from degraded reliability

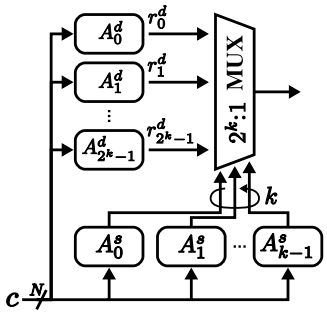

Fig. 4 Circuit structure of (n, k)-MPUF.

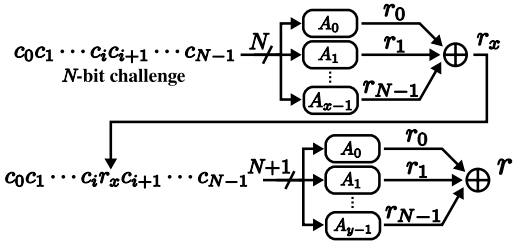

Fig. 5 Circuit structure of IPUF.

and increased circuit footprint. To strike a balance between circuit footprint, ML attack tolerance, and reliability, [5] proposed to combine multiple APUFs with a multiplexer (Multiplexer PUF (MPUF)) [5]. Figure 4 shows an (n, k)-MPUF composed of a 2^k-to-1 multiplexer and $2^k + k$ APUF instances. The outputs of 2^k APUF instances are selected by the remaining k APUF instances connected to the k-bit select signal of the multiplexer. Through extensive analysis, the authors showed that the (64,3)-rMPUF, a variant of the MPUF, has a comparable ML attack tolerance to the 10-XOR APUF, while its reliability is as high as a 4-XOR APUF.

2.1.5 Interpose PUF (IPUF)

Although the above-mentioned APUF compositions place the APUFs in parallel and combine their output by using a combinational circuit, [6] proposed another type of APUF composition, known as Interpose PUF (IPUF), where APUFs are placed in serial; i.e., the first APUF output is fed as a challenge to the succeeding APUF. Figure 5 shows the structure of an (x, y)-IPUF, where a response of an x-XOR PUF (r_x) is "interposed" after the i-th challenge bit (c_i) to form an $N + 1$-bit challenge for the succeeding y-XOR PUF to yield the final response of the IPUF. It has been shown in [6] that an (x, y)-IPUF has a much higher robustness to modeling attacks with comparable hardware footprint and reliability than an $(x + y)$-XOR APUF.

2.1.6 Double Arbiter PUF (DAPUF)

Although the operational principal of the APUF and its variants is simple, implementing them on silicon is difficult; i.e., the paired MUX chains must be designed carefully to have

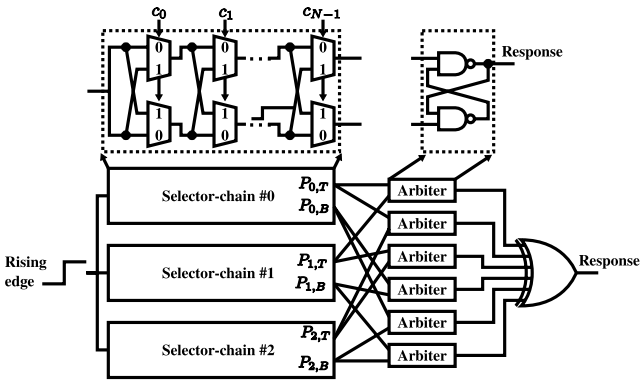

Fig. 6 Circuit structure of DAPUF.

good symmetries so that signal propagation delays along every possible path may be determined by manufacturing variations only. For the ease of implementation, a double arbiter PUF (DAPUF) has been proposed [14], which is composed of arbiters and multiple MUX chains placed in parallel. Let us denote outputs of the top and bottom paths of the i-th MUX chain as $P_{i,T}$ and $P_{i,B}$, respectively, for $i = \{0, 1, \cdots, N-1\}$. Then, the arbiters measure the race between $P_{j,z}$ and $P_{k,z}$ for $j = \{0, 1, \cdots, N-2\}$, $k = \{j+1, j+2, \cdots, N-1\}$ and $z \in \{\mathrm{T}, \mathrm{B}\}$. Finally, the response of the DAPUF is generated by taking XOR of the arbiter outputs. Figure 6 shows the architectural overview of a 3-1 DAPUF.

The basis for the difference between an XORAPUF and DAPUF is the delay lines that are measured by the arbiter: in a 2-XOR PUF, each arbiter measures the delay difference between the top and bottom delay lines of the same MUX chain. In a DAPUF, on the other hand, the top and bottom of the MUX chain are divided into separate groups, and the signal propagation delay within each group is measured. In a DAPUF, the arbiter measures the delay difference only at the top or bottom, which is considered an advantage as it improves the symmetry.

2.2 Modeling Attacks on PUFs

Because an open network is used to establish the communication between PUF instances and an authentication server, an adversary has the chance to collect CRPs with which it can construct a mathematical model, which can predict the response bits for unseen challenge inputs. This type of impersonation attack is called a *modeling attack*, and is illustrated in Fig. 7.

Several attempts at modeling attacks have revealed that PUFs whose security is based on the linear-additive model, such as the APUFs, are vulnerable to modeling attacks based on simple linear classification algorithms such as support vector machine (SVM) or logistic regression method (LR) [12]. To improve the robustness of PUFs to modeling attacks, modern PUF architectures employ non-linearities such as XOR operations.

Recently, inspired by the success of artificial neural networks (ANNs) in image classification, researchers are con-

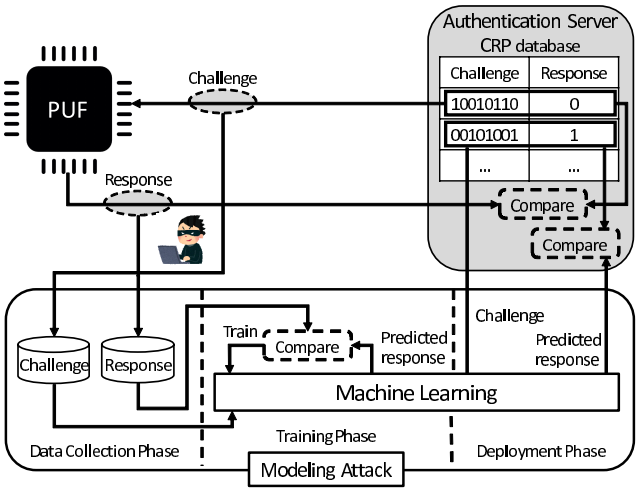

Fig. 7 Typical situation of modeling attack for PUF-based authentication scheme.

sidering the application of ANNs for modeling attacks on PUFs. [8] were the first to apply ANNs to extract intermediate feature vectors to enhance modeling accuracy. The authors reported that ANNs combined with LR to successfully predict 68% of the CRPs of a 3-1 DAPUF having a 32-bit challenge input. Subsequently, we proposed PUFNet [10], which can be trained end-to-end, and showed that it can achieve a prediction accuracy of over 80% even for 3-1 DAPUFs, which are said to be difficult to attack using existing methods. An exhaustive experiment using simulated CRPs of several PUF architectures was discussed in [11], which demonstrated that despite extensive efforts of PUF designers to improve machine learning attack tolerance, DNNs successfully broke all considered APUF variants. However, previous work overlooked the problem that training DNNs requires a large number of CRPs to achieve acceptable classification accuracies. For example, [11] used 145k CRPs for a successful modeling attack on a 64-bit 5-XOR APUF, which may have overestimated the safety of PUF. In this paper, we propose a BayesianPUFNet, a Bayesian extension of PUFNet [10], and show that it can achieve high prediction accuracy with less training data. This result contributes to a strict estimate of the safety of PUF.

2.3 Bayesian Neural Network

Traditional ANNs have millions of trainable parameters; hence, huge training samples are required to avoid overfitting. To solve this issue, the "ensemble" algorithm has been developed, where predictions from multiple neural networks are aggregated to create the final prediction [15], [16]. BNNs, an extension of ANNs, are capable of constructing ensemble models with limited memory overhead. In BNNs, every parameter is treated as a "random variable" that is associated with a probability distribution. During BNN inference, parameters are first sampled from the associated probability distribution, and then a standard forward propagation is conducted with the sampled parameter. By

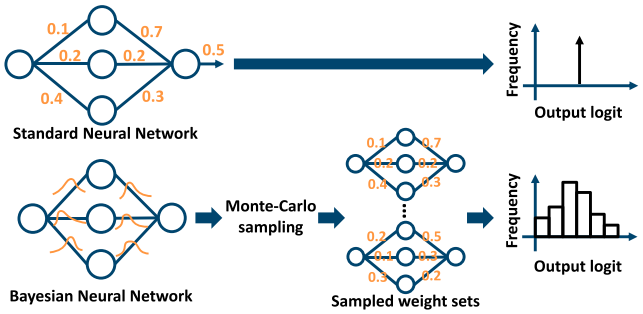

Fig. 8 Comparison of a standard DNN (upper part) and BNN (lower part).

repeating this procedure, i.e., parameter sampling and forward propagation, multiple predictions with slightly different model parameters are obtained. The final prediction is chosen to be the one that has the highest frequency among these predictions. By "smoothing out" predictions, BNNs successfully combat over fitting issue.

Figure 8 shows the fundamental difference between standard DNNs and BNNs. In the standard DNNs, only a single set of weights involves for inference and hence a single output is obtained. On the other hand, in BNNs, the probability distributions of the weights are adjusted through training; in BNN inference, the weight realizations are sampled and the histogram of the output is obtained through multiple iterations of inference. Therefore, BNNs can make predictions considering all possible parameters and are not prone to overfitting. Note that DNNs and BNNs cover the same class of functions. Hence, given enough training data, both DNN and BNN predict the same output.

The training objective of a BNN is to find the posterior distribution over the weights, $\mathbf{W} = (\boldsymbol{w}^1, \boldsymbol{w}^2, \cdots, \boldsymbol{w}^L)$, of an L-layered network for the training data $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_M)$, and the target label, $\boldsymbol{Y} = (y_1, y_2, \cdots, y_M)$. Here, $\boldsymbol{w}^l$ is the $N_l \times N_{l-1}$ dimensional weight matrix of l-th layer having N_{l-1} input and N_l output nodes, $\boldsymbol{x}_i$ is a D-dimensional input vector, and M is the number of training samples. By using the Bayes' theorem, the posterior distribution can be represented by a combination of the likelihood and prior distributions:

$$P(\boldsymbol{W}|\boldsymbol{X}, \boldsymbol{Y}). \tag{5}$$

In practical applications, the posterior distribution is not tractable, and hence, the *variational inference* technique has been developed to approximate it, i.e., $P(\boldsymbol{W}|\boldsymbol{X}, \boldsymbol{Y}) \approx q(\boldsymbol{W}|\boldsymbol{\theta})$, where $q(\cdot|\cdot)$ is a variational posterior distribution and $\boldsymbol{\theta}$ is the variational parameter vector. The training objective is to minimize the Kullback-Leibler (KL) divergence between q and the true posterior P:

$$KL\left(q(\boldsymbol{W}|\boldsymbol{\theta})||p(\boldsymbol{W}|\boldsymbol{X}, \boldsymbol{Y})\right). \tag{6}$$

We use Monte Carlo (MC) dropout to approximate the inference of the BNN. Dropout is a well-known technique in the DNN community, which refers to dropout neurons in a neural network. More specifically, at each training stage,

neurons are either dropped out with a probability of $1-p$ or retained with a probability of p. Hence, the (i,j)-element of $\boldsymbol{w}^l$ can be obtained from

$$w_{ij}^l = b_{ij}^l \cdot \mu_{ij}^l, \tag{7}$$

where μ_{ij}^l is a trainable parameter and b_{ij}^l is a random sample drawn from a Bernoulli distribution, i.e., b_{ij}^l takes "1" with probability p_{ij}^l and "0" with probability $1-p_{ij}^l$. The dropout probabilities, p_{ij}^l, can be optimized. However, we fixed them at 0.5, which is a typical choice [17]. With dropout, a neural network has less memory capacity, which helps preventing over-fitting. Unlike in conventional neural networks, in BNNs, dropout is applied not only during training but also during inference, which results in stochastic predictions; i.e., the network makes slightly different prediction at each inference for the same input. Finally, we take majority voting among the stochastic outputs to improve classification accuracy.

The idea of dropout can be viewed as a variant of ensemble learning, which is a meta learning algorithm to combine diverse models to achieve better modeling performance. Although combining several models leads to better performance, model size increases linearly. Hence, a careful selection of the number of models to be combined is necessary to strike a balance between model size and performance. On the other hand, BNN provides good efficient because it utilizes "randomness" on inference; i.e., the dropout at inference time introduces "divergence" at each inference.

Generally, a loss function tailored for BNN inference must be determined. However, [17] showed that the minimization of cross entropy loss function is mathematically equivalent to minimize the KL divergence. Hence, inserting dropout layers between each linear layer and introducing dropout not only during training but also during inference is enough.

3. Bayesian Deep Neural Network for Modeling Attack

The following summarizes the essence of the proposed BayesianPUFNet architecture.

Preprocessing: As in a previous study [11], the parity of the challenge input was calculated using Eq. (2) and used as the input to the neural network.

Activation function: Traditionally, the sigmoidal function defined as

$$\sigma(x) = \frac{1}{1+\exp(-x)} \tag{8}$$

was commonly used. However, since its derivative,

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1-\sigma(x)), \tag{9}$$

is always smaller than "1," gradients of neural networks found using backpropagation may vanish when the network has a large number of layers. To overcome this problem, BayesianPUFNet employs Rectified linear unit (ReLU) as the activation function defined as:

$$\mathrm{ReLU}(x) = \begin{cases} x & x > 0 \\ 0 & x \le 0 \end{cases} \tag{10}$$

the derivative of which is given by:

$$\frac{\partial}{\partial x}\mathrm{ReLU}(x) = \begin{cases} 1 & x > 0 \\ 0 & x \le 0 \end{cases}, \tag{11}$$

which helps the network learn sparse representations while preventing gradient vanishing.

Weight initializer: The synaptic weights of the BayesianPUFNet are initialized according to the Xavier initialization method [18], where an initial weight is sampled from a normal distribution with the mean and standard deviations set to be zero and $\sqrt{2/(N_{\text{in}}+N_{\text{out}})}$, respectively. Here, N_{in} and N_{out} are the number of synaptic connections to and from the neuron, respectively. This initialization helps the gradients to propagate through the network with a reasonable dynamic range. We found that weight initialization significantly affects the BayesianPUFNet performance.

Activation normalization: We also found that batch normalization (BN) can greatly increase the performance of the BayesianPUFNet. It prevents the saturation of non-linear activation functions to stabilize network optimization.

Loss function: As used in [11], we use cross entropy as the loss function. Hence, our training objective is to minimize

$$L(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left\{y_i \log(\widehat{y_i}) + (1-y_i)\log(1-\widehat{y_i})\right\}, \tag{12}$$

where y_i and $\widehat{y_i}$ are the target label and network output, respectively. N is the number of training samples.

MC Dropout applied only for first layer: During the training of the BayesianPUFNet, a few synaptic weights exist in the first layer are randomly fixed to zero, which limits the flexibility of the model, and thus acts as regularizer to avoid overfitting. Note again that we also apply dropout during inference to yield slightly different model predictions for ensembling. Although MC dropout is applied to all layers in general BNN, in the proposed method, MC dropout is applied only to the first layer. The reason for this is that applying MC dropout to all layers slows down the learning convergence. In addition, as we will verify in numerical experiments, even if MC dropout is applied only to the first layer, we can still enjoy the advantage of BNNs, i.e., high prediction accuracy can be obtained even with small training data.

4. Preliminary Experiment Using Synthetic CRPs

4.1 Experimental Setup

To compare BayesianPUFNet with conventional DNN-based

Table 1 Hyper parameters.

Hidden layer activation	ReLU	
Optimizer	Adam	
Learning rate	10^{-3}	
Loss function	BCE	
Initializer	Glorot Normal	
# of neurons	APUF	(5,1)
	XOR APUF	(30,28,1)
	LS-PUF	(100,100,100,100,1)
	MPUF	(30,30,15,1)
	IPUF	(50,50,50,1)

ML attack, we first conducted a numerical experiment using publicly available synthetic CRPs [11]. In the dataset, five APUF variants (APUF, XOR APUF, LSPUF, MPUF, and IPUF) are synthetically generated using Matlab simulation models using the approach adopted in [5], [12]. Here, the stage delay of each APUF is assumed to be identically distributed and follow Gaussian distribution with mean μ=0.1 and standard deviation σ=1. For simulating imperfect reliability induced by noise such as random telegraph noise [19], an additive random noise following the normal distribution with mean μ=0 and standard deviation σ=0.01 is superimposed to the stage delay at each response generation. In order to accurately compare with existing studies, the number of neurons and other network architectures were kept the same as in [11]. The hyper parameters are listed in Table 1.

For each PUF, the generated CRPs are split into two parts, i.e., the training set consisting of at most 80% of the total CRPs, and the validation set consisting of the remaining 20%. The conventional DNN model and the proposed BayesianPUFNet are implemented using TensorFlow framework. Note that, BayesianPUFNet applies MC dropout not only during the training phase but also during the inference phase. The dropout rate is set to be 0.5 throughout our experiment, which indicates that only half of the neurons were activated during network inference. As shown in Sect. 3, the parity of the challenge vector is used as the network input. The initial synaptic weights are sampled randomly according to the Xavier initialization method. To see the impact of randomness on the final model performance, every experiment is repeated 10 times and 95% confidence intervals of the prediction accuracy are provided.

4.2 Experimental Results

4.2.1 Performance of BayesianPUFNet over Conventional BNN

Before reporting machine learning attack results on APUF variants, we firstly evaluated the performance of BayesianPUFNet over a conventional BNN. Figure 9 shows the modeling accuracy of a 64-bit 5-XOR APUF for BayesianPUFNet, a conventional BNN, and non-Bayesian DNN-based method [10] as a function of the number of training epochs. Error bars in the figure indicate 95% confidence intervals. Note again that the difference between BayesianPUFNet and a conventional BNN is that the layers to which

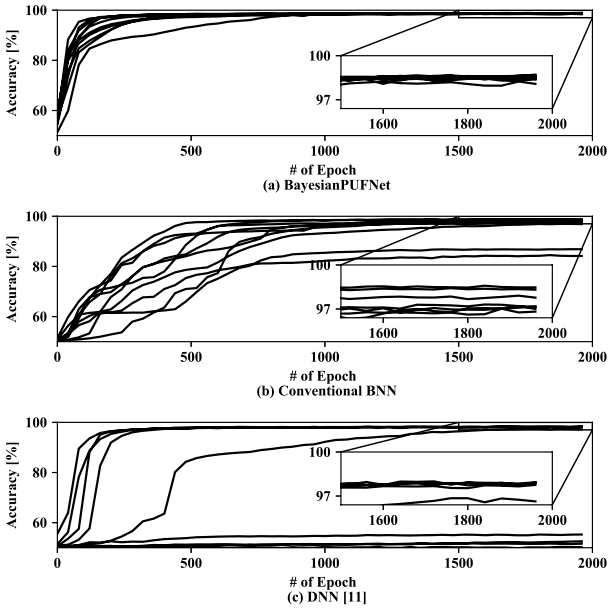

Fig. 9 Comparison of (a) BayesianPUFNet, (b) conventional BNN, and (c) DNN [11].

MC dropout is applied, i.e., BayesianPUFNet applies MC dropout only to the first layer, while the conventional BNN applied it to all layers. Note also that the same experiment is repeated 10 times to see the impact of random initialization of parameters. DNN-based method failed to exceed 60% prediction accuracy in 5 out of 10 trials. On the other hand, the two methods using Bayesian neural networks achieved a prediction accuracy of over 80% in all trials. Comparing (a) and (b), it can be seen that the proposed method achieves higher prediction accuracy with a smaller number of epochs. This may be due to the fact that the proposed method replaces only the first layer with a Bayesian neural network, which improves the learning ability without sacrificing convergence speed.

We further measured the computation time required per epoch using a computer equipped with an AMD EPYC 7702 processor, 128 GB of memory, and an NVIDIA 3090Ti GPU. The results showed that BayesianPUFNet takes 0.682 seconds per epoch of learning, whereas conventional BNN takes 0.985 seconds and DNN takes 0.491 seconds. Accordingly, BayesianPUFNet requires 1.39 times more computation time per epoch than the DNN. However, considering that the BayesianPUFNet achieved a prediction accuracy of over 90% for all 10 trials, while the prediction accuracy of the existing method was below 60% for 5 of the 10 trials, we consider the increase in computation time per epoch to be acceptable.

To investigate the impact of network structure on prediction accuracy, we trained three networks with different sizes of hidden layers and calculated the prediction accuracies. The results are shown in Fig. 10(a). The black line in the figure shows the prediction accuracy of the network consisting of 30 and 28 neurons in the hidden layer and 1 neuron in the output layer as a function of CRPs used for training.

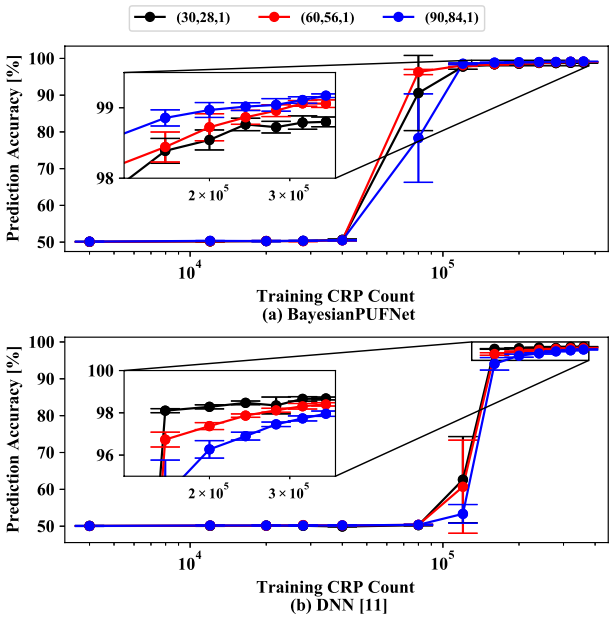

Fig. 10 Impact of neuron count on prediction accuracy.

Similarly, the red and blue lines show the prediction accuracy when the size of the hidden layer is doubled and tripled, respectively. Similar experiments were also conducted for non-Bayesian DNN, whose results are shown in Fig. 10(b). The figure shows that the inference accuracy varies slightly depending on the network structure, but generally shows the same trend. In addition, for both BayesianPUFNet and DNN, the smaller the hidden layer size, the more quickly the prediction accuracy rises. On the other hand, comparing the prediction accuracy when there are enough CRPs for training, we can also see that the prediction accuracy for DNN is higher for networks with smaller hidden layers, whereas for BayesianPUFNet, it is higher for networks with larger hidden layers. This is attributed to the fact that the DNN is slightly over-trained due to insufficient training data relative to the network complexity.

4.2.2 Arbiter PUF

Although APUFs are known to be vulnerable to machine learning attack, we first modeled a 64-bit APUF to demonstrate the completeness of BayesianPUFNet. Fig. 11 compares the modeling accuracies using conventional DNN-based method [11], LR method [12], and the BayesianPUFNet for different number of training CRPs. We randomly choose a fraction of the training data to train both the BayesianPUFNet and conventional DNN model to see the performance of BayesianPUFNet for small training data. It can be clearly seen that the BayesianPUFNet achieves a higher prediction accuracy compared to the conventional DNN especially when the number of training samples is small. Because of the linearity of APUF, LR, which has a simpler model, achieves higher prediction accuracy than DNN. Also, given a sufficient number of CRPs for training,

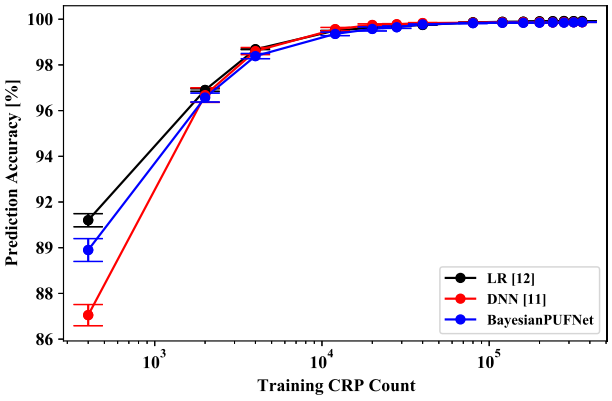

Fig. 11 Modeling accuracy result for 64-bit APUF.

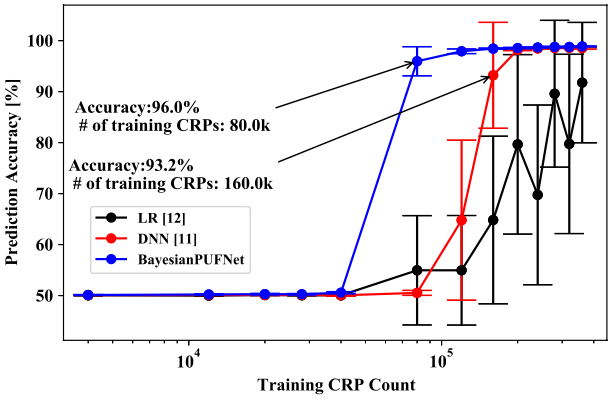

Fig. 12 Modeling accuracy result for 64-bit XOR APUF.

both DNN and BNN converge to the same prediction accuracy. This suggests that DNN and BNN cover the same problem class.

4.2.3 XOR Arbiter PUF

Figure 12 shows the modeling accuracy of a 64-bit 5-XOR APUF. We again confirm that the BayesianPUFNet requires much fewer training data for successful modeling attack. Specifically, the BayesianPUFNet achieved prediction accuracy of 96% when 80×10^3 CRPs were used for training. In our experiment, the conventional DNN could only achieve 93.2% prediction accuracy even when the number of training samples was doubled.

4.2.4 Lightweight Secure PUF

Because an LSPUF outputs multiple response bits, we modeled individual output bits as in [11]. The circular shift parameter s was set to 0. Figure 13 summarizes the modeling accuracy of the first response bit of a 6-5 LS-PUF. Again, we can see that the BayesianPUFNet achieved higher prediction accuracy even when the training dataset is small.

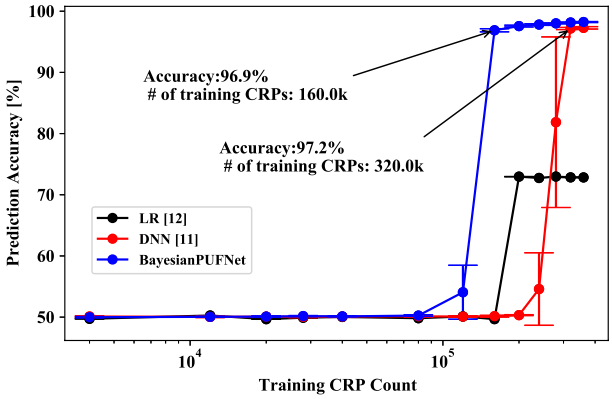

Fig. 13 Modeling accuracy result for 64-bit LS-PUF.

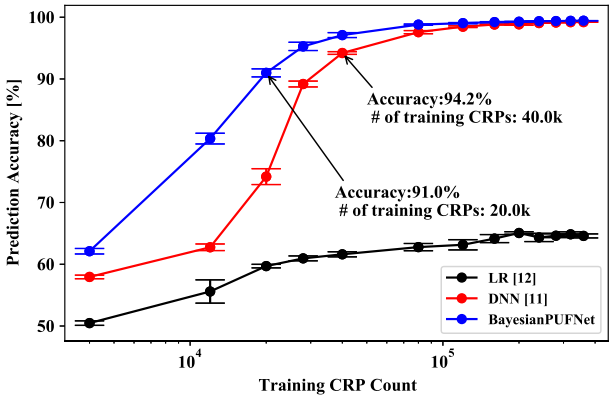

Fig. 14 Modeling accuracy result for 64-bit MPUF.

4.2.5 Multiplexer PUF

Figure 14 shows the results of a modeling attack on a 64-bit MPUF. Comparing Figs. 12 and 14, we notice that an MPUF can be modeled with relatively fewer number of CRPs than XOR APUFs. Specifically, to achieve over 90% accuracy, the BayesianPUFNet required 20k CRPs whereas the conventional method required 40k CRPs.

4.2.6 Interpose PUF

Modeling attack results on a 64-bit IPUF are summarized in Fig. 15. We confirmed that the BayesianPUFNet required fewer training data than the conventional DNN. Specifically, the BayesianPUFNet required only 80k CRPs to achieve over 90% prediction accuracy whereas the conventional DNN required more than 120k CRPs to achieve comparable prediction accuracy.

4.3 Discussion

It can be seen that the LR model is effective for attacks on highly linear PUFs such as APUF, but not for PUFs with increased non-linearity such as LSPUF, MPUF, and IPUF. Furthermore, to investigate the vulnerability of XORPUF,

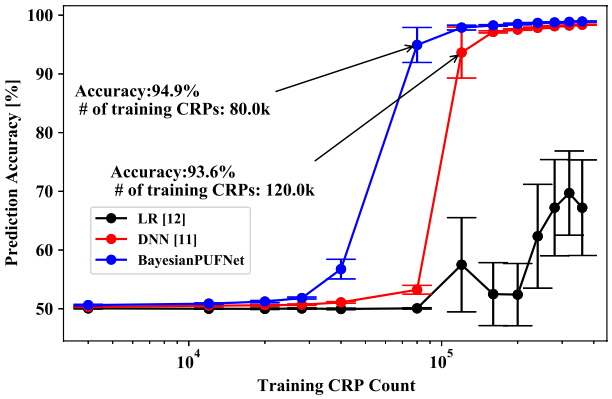

Fig. 15 Modeling accuracy result for 64-bit IPUF.

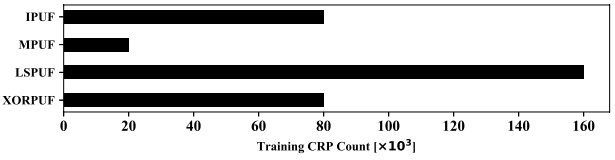

Fig. 16

LSPUF, MPUF, and IPUF to machine learning attacks, we calculated the number of CRPs required to achieve a prediction accuracy of over 90% using the proposed method for each of them, which are summarized in Fig. 16. The results indicate that LSPUF is the most resistant to machine learning attacks in our experiments.

5. Experiment Using CRPs Obtained via Transistor-level Circuit Simulation

To evaluate machine learning attack tolerance on a more realistic situation, we designed a 3-to-1 DAPUF using a 65 nm Predictive Technology Model (PTM) [20] and simulated the challenge-to-response behavior by using a SPICE simulator.

5.1 Experimental Setup

To reproduce the impact of manufacturing variability, V_{TH} of each transistor is sampled from a normal distribution. The Pelgrom coefficient is set to be 4 mV·μm according to the silicon measurement on 65 nm technology [21]. From the transistor-level simulation, the responses corresponding to 160k randomly generated challenges are obtained. Among the 240k CRPs obtained, 216k CRPs are randomly selected and used to train the BayesianPUFNet. Because the total number of CRPs are 2^{32} (about 4 billion), 100k CRPs correspond to only 5.59×10^{-3}% of them. Using the remaining 24k CRPs, the performance of the trained BayesianPUFNet for predicting responses to unseen challenge inputs is validated.

5.2 Experimental Results

Figure 17 summarizes the experimental results. The

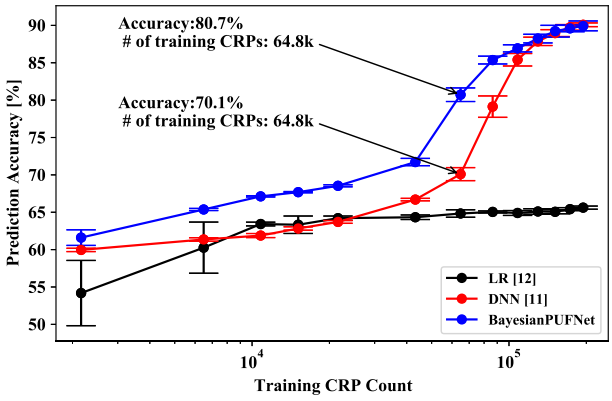

Fig. 17 Modeling accuracy result for 32-bit DA-PUF.

BayesianPUFNet achieved over 80% prediction accuracy with only 64.8 k CRPs of the training data, while the existing method achieve only about 70% prediction accuracy with the same number of training data. This confirms that the BayesianPUFNet can achieve higher prediction accuracy with less data than existing methods, even for realistic CRP data sets.

6. Conclusion

By introducing Bayesian neural networks, BayesianPUFNet succeeded in achieving high prediction accuracy with a small number of CRPs. In an experiment using synthetic CRPs simulating XOR-APUF, BayesianPUFNet achieved over 90% prediction accuracy using approximately 80k CRPs for training, whereas the existing methods required twice as many training samples to achieve equivalent prediction accuracy. To further examine the performance of BayesianPUFNet under realistic conditions, we simulated a DAPUF using SPICE to obtain CRPs and trained BayesianPUFNet and the existing method. The results showed that BayesianPUFNet achieved over 80% prediction accuracy with 64.8k CRPs, while the existing method required an additional 21.6k CRPs to achieve the same accuracy. Our findings highlight a new vulnerability in PUF-based device authentication systems.

References

[1] U. Rührmair and D.E. Holcomb, "PUFs at a glance," Design, Automation and Test in Europe, pp.1–6, March 2014.
[2] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, "Design and implementation of PUF-based "Unclonable" RFID ICs for anti-counterfeiting and security applications," Int. Conf. on RFID, pp.58–64, April 2008.
[3] G.E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," Design Automation Conf., pp.9–14, 2007.
[4] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," Int. Conf. on Comput.-Aided Design, pp.670–673, 2008.
[5] D.P. Sahoo, D. Mukhopadhyay, R.S. Chakraborty, and P.H. Nguyen, "A multiplexer-based arbiter PUF composition with enhanced reliability and security," IEEE Trans. Comput., vol.67, no.3, pp.403–417, 2018.
[6] P.H. Nguyen, D.P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. van Dijk, "The interpose PUF: Secure PUF design against state-of-the-art machine learning attacks," IACR Transactions on Cryptographic Hardware and Embedded Systems, vol.2019, no.4, pp.243–290, Aug. 2019.
[7] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson, "Efficient power and timing side channels for physical unclonable functions," Int. Workshop on Cryptographic Hardware and Embedded Systems, pp.476–492, 2014.
[8] R. Yashiro, T. Machida, M. Iwamoto, and K. Sakiyama, "Deep-learning-based security evaluation on authentication systems using arbiter PUF and its variants," Int. Workshop on Security, pp.267–285, 2016.
[9] M. Khalafalla and C. Gebotys, "PUFs deep attacks: Enhanced modeling attacks using deep learning techniques to break the security of double arbiter PUFs," Design, Automation and Test in Europe, pp.204–209, 2019.
[10] H. Awano, T. Iizuka, and M. Ikeda, "PUFNet: A deep neural network based modeling attack for physically unclonable function," Int. Symp. on Circuits and Syst., pp.1–4, 2019.
[11] P. Santikellur, A. Bhattacharyay, and R.S. Chakraborty, "Deep learning based model building attacks on arbiter PUF compositions," Cryptology ePrint Archive, Report 2019/566, 2019. https://eprint.iacr.org/2019/566
[12] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," Comput. and Commun. Security, pp.237–249, 2010.
[13] J. Tobisch and G.T. Becker, "On the scaling of machine learning attacks on pufs with application to noise bifurcation," Radio Frequency Identification, S. Mangard and P. Schaumont, eds., pp.17–31, 2015.
[14] T. Machida, D. Yamamoto, M. Iwamoto, and K. Sakiyama, "Implementation of double arbiter PUF and its performance evaluation on FPGA," Asia and South Pacific Design Automation Conf., pp.6–7, Jan 2015.
[15] T.G. Dietterich, "Ensemble methods in machine learning," Int. Workshop on Multiple Classifier Syst., pp.1–15, 2000.
[16] Z. Ghahramani and M. Beal, "Propagation algorithms for variational bayesian learning," Neural Information Processing Syst., T. Leen, T. Dietterich, and V. Tresp, eds., pp.507–513, MIT Press, 2001.
[17] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," International Conference on International Conference on Machine Learning, pp.1050–1059, 2016.
[18] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," Int. Conf. on Artificial Intelligence and Statist., Proc. of Mach. Learning Res., vol.9, pp.249–256, May 2010.
[19] K. Hung, P. Ko, C. Hu, and Y. Cheng, "Random telegraph noise of deep-submicrometer MOSFETs," IEEE Electron Device Lett., vol.11, no.2, pp.90–92, 1990.
[20] Nanoscale Integration and Modeling (NIMO) Group, "Predictive Technology Model (PTM)," http://ptm.asu.edu/
[21] S. Saxena, C. Hess, H. Karbasi, A. Rossoni, S. Tonello, P. McNamara, S. Lucherini, S. Minehane, C. Dolainsky, and M. Quarantelli, "Variation in transistor performance and leakage in nanometer-scale technologies," IEEE Trans. Electron Devices, vol.55, no.1, pp.131–144, Jan 2008.

Hiromitsu Awano received his B.E. degree in Informatics and M.Sc. and Ph.D. degrees in Communications and Computer Engineering from Kyoto University in 2010, 2012, and 2016, respectively. He was with Hitachi, Ltd., Tokyo, Japan in 2016. In 2017, he joined the VLSI Design and Education Center, The University of Tokyo, Japan, where he is an assistant professor. His research interests include CAD for VLSI design and hardware accelerator for machine learning. He was a research fellow of japan society for the promotion of science and a member of IEEE, IEICE, and IPSJ.

Makoto Ikeda received the B.S., M.S., and Ph.D. degrees in electronic engineering from the University of Tokyo, Tokyo, Japan, in 1991, 1993, and 1996, respectively. He joined the Electronic Engineering Department, University of Tokyo, as a Faculty Member in 1996, and he is currently a full Professor with the department of Electrical Engineering and Information Systems, at the University of Tokyo. At the same time he has been involving the activities of VDEC (VLSI Design and Education Center, the University of Tokyo), to promote VLSI design educations and researches in Japanese academia. His interests include the hardware security, including cryptographic engine design, asynchronous system design and smart image sensor designs. He is a member of IEEE, IEICE Japan, IPSJ and ACM.