Security Enhancement for RRAM Computing System through Obfuscating Crossbar Row Connections

Minhui Zou*§, Zhenhua Zhu†, Yi Cai†, Junlong Zhou*, Chengliang Wang‡, and Yu Wang†
*School of CSE, Nanjing University of Science and Technology §E-mail: zouminhui@outlook.com
†Dept. of EE, BNRist, Tsinghua University, ‡School of CS, Chongqing University

***Abstract*—Neural networks (NN) have gained great success in visual object recognition and natural language processing, but this kind of data-intensive applications requires huge data movements between computing units and memory. Emerging resistive random-access memory (RRAM) computing systems have demonstrated great potential in avoiding the huge data movements by performing matrix-vector-multiplications in memory. However, the nonvolatility of the RRAM devices may lead to potential stealing of the NN weights stored in crossbars and the adversary could extract the NN models from the stolen weights. This paper proposes an effective security enhancing method for RRAM computing systems to thwart this sort of piracy attack. We first analyze the theft methods of the NN weights. Then we propose an efficient security enhancing technique based on obfuscating the row connections between positive crossbars and their pairing negative crossbars. Two heuristic techniques are also presented to optimize the hardware overhead of the obfuscation module. Compared with existing NN security work, our method eliminates the additional RRAM writing operations used for encryption/decryption, without shortening the lifetime of RRAM computing systems. The experiment results show that the proposed methods ensure the trial times of brute-force attack are more than $(16!)^{17}$ and the classification accuracy of the incorrectly extracted NN models is less than 20%, with minimal area overhead.**

I. Introduction

Neural Networks (NNs) have achieved tremendous success in visual objects recognition and natural language processing. However, this kind of data-intensive applications requires huge data movements between computing units and memory, which challenges the conventional von Neumann architecture separating computation and memory. Emerging resistive random-access memory (RRAM) computing systems have demonstrated great potential in boosting NN computing energy efficiency. Fig. 1 shows the basic structure of an RRAM computing system, which consists of many process elements (PEs) and each PE is made up of peripheral circuits and an RRAM crossbar. RRAM crossbars have the capability of performing matrix-vector-multiplications (MVMs) in memory with $O(1)$ computation complexity [1], which also improves the MVMs computing energy efficiency by eliminating matrix data movements [2]. The most computing-intensive and time-consuming parts of NNs are convolution (Conv) layers and fully-connected (FC) layers, which could be transformed into MVM operations. Thus, RRAM computing systems can improve the energy efficiency of NN computing.

However, the advantage of RRAM computing systems comes with a security cost, i.e., the NN weights stored in the crossbars won't be lost when the systems are powered off. The adversary can exploit this vulnerability to read them out. By stealing the NN weights, the adversary may extract the well-trained NN models from them, which greatly harms the intellectual property of the NN model designers. What's worse,

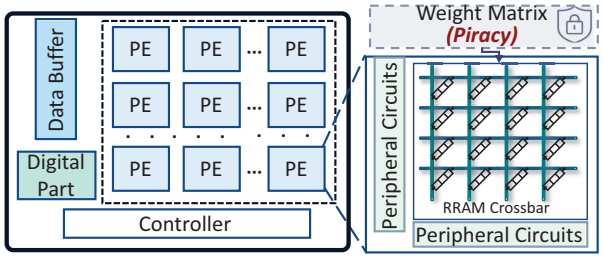

Fig. 1. Basic structure of an RRAM computing system

the malicious usage of the extracted NN models might lead to social crisis. For example, the adversary could insert a trojan in the extracted NN models for facial recognition systems [3]. Hence, protecting the NN weights is of great significance.

A straightforward solution is encrypting the NN weights and decrypting them each time using them. For example, [4] proposed to encrypt the NN models and only decrypt them for authorized users and [5] put forward an approach to only incrementally encrypt the involved data. Nevertheless, those methods inevitably require frequent writing operations to the RRAM devices for encryption. Currently, the RRAM devices only sustain for up to 10^{10} switching cycles [2]. As a result, the NN weight encryption/decryption solutions would shorten the lifetime of the RRAM computing systems. Besides, frequent writing operations to the RRAM devices consume much energy and bring in long latency to the systems [6].

To tackle these problems, this work proposes an effective security enhancement for RRAM computing systems to defense against the NN weights piracy attack. The contributions of this work are summarized below:

- We evaluate the security of existing NN weights mapping method for analog/digital RRAM crossbars. Then two theft metods are analyzed: *theft method 1:* steal weights from single crossbars of each crossbar pair; *theft method 2:* steal weights from both crossbars of each crossbar pair.
- For *theft method 1*, we propose to randomize the bias of the mapping method and apply different biases for each weight, which hides the relationship between each crossbar and its corresponding weight. For *theft method 2*, we design a multiplexer-based obfuscation module, which enhances the storage security of RRAM crossbars through obfuscating the connections between the positive crossbars and their corresponding negative crossbars.
- This work presents two heuristic techniques to reduce the overhead of the obfuscation module. Firstly, we add a layer of demultiplexers in the obfuscation module to reduce the overall number of multiplexers required. Secondly, we explore the sensitivity of each layer/bit to the classification accuracy when the crossbar row con-

978-3-9819263-4-7/DATE20/©2020 EDAA

nections are obfuscated, and protect only the significant layers and significant bits of the NN models. These two techniques reduce the obfuscation module overhead by $33.3\% \sim 99.89\%$.
- At last, this work validates the effectiveness of the proposed protection method on three real-life NNs. The experiment results show with only minimal hardware overhead, the proposed methods ensure the trial times of brute-force attack are more than $(16!)^{17}$ and the classification accuracy of the incorrectly extracted NN models is less than 20%.

II. Preliminaries, threat model, and motivation

A. Preliminaries

The main parts of NNs are FC layers and Conv layers. The computation of FC layers are MVMs, which are described as:

$$y_j = \sum_{i=1}^{m} w_{i,j} \cdot x_i \tag{1}$$

where $x_i (i \in [1, m])$ is the input feature map, $y_j (j \in [1, n])$ is the output activation, and $w_{i,j}$ is the synapse weight. The computation of Conv layers is different but could also be transformed into MVM.

In RRAM computing systems, the inputs are the voltages ($\boldsymbol{V}$) applied to the word-lines (WLs) of the crossbar and the outputs are the accumulated currents ($\boldsymbol{I}$) at the bit-lines (BLs), as shown in Fig.2(b). The input voltages, the conductance of the crossbar cells, and the output currents comply with the Kirchhoff's law, which can be regarded as MVM:

$$I_j = \sum_{i=1}^{m} g_{i,j} \cdot V_i, \tag{2}$$

where $g_{i,j}$ is the conductance of the cell at ith row and jth column of the crossbar.

However, the NN weight w_{ij} could not directly mapped onto g_{ij} because the matrix element w_{ij} could be positive, negative, or zero while the RRAM conductance of the crossbar could only be positive. To solve this, a pair of crossbars are needed to represent the weight matrix, i.e., a positive crossbar and a negative crossbar [1], [7]. As demonstrated in Fig. 2(b), the input voltages are converted into their opposite and then fed to the negative crossbar, then the MVM results come from adding the BL currents of positive and negative crossbars.

As shown in Fig. 2(a), RRAM devices can be categorized as analog RRAM devices [8] and digital RRAM devices [9]. Analog RRAM shows gradual RESET process, which means an RRAM device can be tuned from low resistance state (LRS) to high resistance state (HRS) continuously [2]. So an ideal analog RRAM cell could be tuned into any arbitrary conductance state between LRS and HRS. As to digital RRAM devices, they could only be tuned to limited discrete resistance states, and multiple crossbars are used to represent high precision weights. Thus, the mapping method applied to the analog RRAM crossbars [1], [7] is different from that applied to the digital RRAM crossbars [10], [11].

B. Threat model and motivation

The NN weights are the core intelligent property of NNs. We assume the adversary could turn the the RRAM computing system off and then read the conductance values of the on-chip RRAM cells to recover the weights, and their goal is to steal the weights stored in the RRAM crossbars.

The motivation of this work is to keep the unauthorized users from reading the NN weights correctly from the on-chip RRAM crossbars so that the extracted models do not perform normally. The protection method does not rely on encrypting/decrypting the models, and hence avoid the energy, latency, and lifetime cost caused by RRAM writing operations.

III. Analysis of theft methods

In this section, we evaluate the security of mapping methods for RRAM crossbars, and analyze the theft methods.

A. Mapping method for analog RRAM crossbars

For analog RRAM devices, each NN weight w_{ij} is implemented by a pair of RRAM cells which are connected to opposite input voltages. Assume the maximum and the minimum conductance of RRAM cells are G_{on} and G_{off}, respectively. We denote the conductance of the cells connected with the positive/negative voltage as g_{ij}^{+} / g_{ij}^{-}. Then we have $w_{ij} = g_{ij}^{+} - g_{ij}^{-},\ w_{ij} \in [G_{off} - G_{on}, G_{on} - G_{off}]$.

Based on the above analysis, there exist two main mapping methods for analog RRAM devices [1], [7]. The mapping method of [1] is shown in Eq. 3, where $G_{mid} = \frac{G_{off}+G_{on}}{2}$. All the RRAM cells are initialized to the bias G_{mid} and then tuned accordingly.

$$g_{ij}^{\pm} = G_{mid} \pm \frac{1}{2} \cdot w_{ij} \tag{3}$$

The mapping method of [7] is shown in Eq. 4. Similarly, all the RRAM cells are initialized to the bias G_{off} and then tuned accordingly.

$$\begin{aligned} when\ w_{ij} \geqslant 0, &\begin{cases} g_{ij}^{+} = G_{off} + w_{ij} \\ g_{ij}^{-} = G_{off} \end{cases} \\ when\ w_{ij} < 0, &\begin{cases} g_{ij}^{+} = G_{off} \\ g_{ij}^{-} = G_{off} - w_{ij} \end{cases} \end{aligned} \tag{4}$$

For these two mapping methods, there exist two theft methods:

1) *Theft method 1: access to only one single crossbar of each positive/negative crossbar pair.* In both the methods of [1] and [7], the RRAM conductance varies linearly with the scaled weights. The disadvantage of them is that the adversary could easily infer w_{ij} from either g_{ij}^{+} or g_{ij}^{-}. Take the mapping method of [1] for example, $w = 2 \cdot (g^{+} - G_{mid})$ or $w = 2 \cdot (G_{mid} - g^{-})$.
2) *Theft method 2: access to both the two crossbars of each positive/negative crossbar pair.* With access to both of g^{+} and g^{-}, the adversary could easily infer the corresponding w by simple subtractions.

B. Mapping method for digital RRAM devices

Different from analog RRAM devices, the precision of digital RRAM devices is limited, so multiple digital RRAM devices are required to represent the high precision weight, as shown in Fig. 2 (b) [10], [11]. For example, 1-bit digital RRAM devices can only be tuned into HRS or LRS. Then

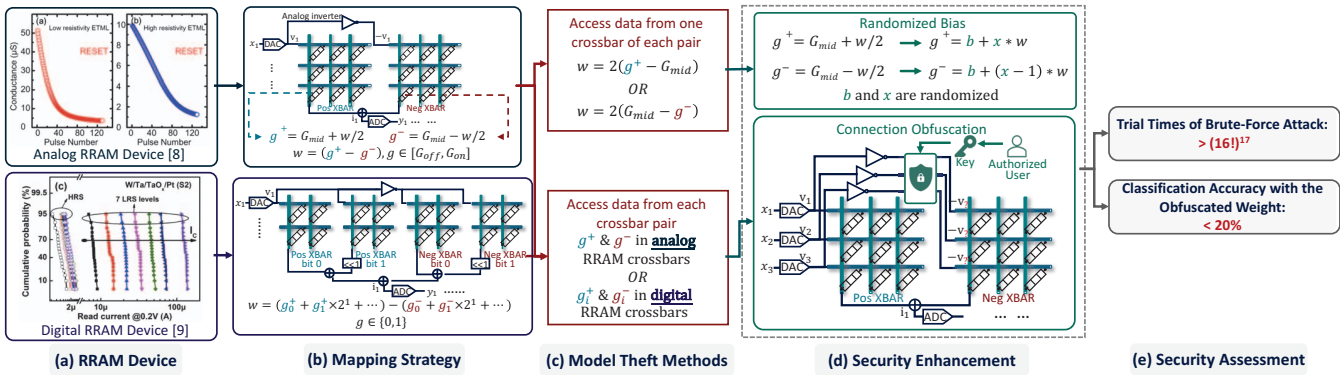

Fig. 2. Overview of the proposed methods: (a) the characteristics of analog RRAM and digital RRAM; (b) the different mapping methods for analog RRAM and digital RRAM; (c) two model theft methods; (d) the security enhancement to counter the theft methods; (e) the security assessment after applying the security enhancement.

TABLE I
THE MAPPING METHOD FOR 1-BIT DIGITAL RRAM DEVICES

w_{ij}	g_{ij}^+	g_{ij}^-	w_{ij}	g_{ij}^+	g_{ij}^-
-1	HRS	LRS	0	LRS	LRS
1	LRS	HRS	0	HRS	HRS

a multi-bit weight matrix is divided into multiple 1-bit sub-matrices and each sub-matrix is implemented by a pair of 1-bit digital RRAM crossbars. The mapping results for 1-bit digital RRAM are shown in Table I. As we can see, there is no linear relationship between w_{ij} and g_{ij}^+ or g_{ij}^-, which means the adversary can not steal weight values directly by only accessing one single crossbar of each pair. For instance, assume g_{ij}^+ is in HRS, w_{ij} could be -1 or 0, with the possibility of 50% and 50%, respectively. In general, assume the precision of the original weight matrix is p bits, the matrix has m rows and n columns, and the RRAM is b-bit. Then the adversary has to try $2^{m*n*b*\lceil \frac{p-1}{b} \rceil}$ times to recover the whole matrix. Let's say p, b, m, n are 8, 2, 128, and 128, respectively. The adversary has to try 2^{131072} times, which is impossible to break for current computing power.

Thus, theft method 1 does not work for digital RRAM devices. However, theft method 2 still work for them. With access to both positive and negative crossbars of corresponding weight matrix, the matrix could be easily recovered.

IV. SECURITY ENHANCEMENT

In this section, we propose the countermeasures against the two theft methods mentioned in Section III.

A. Countermeasure against theft method 1: explore the space of the bias and apply different biases to each matrix element

To counter the theft method 1, we first show that the bias of the mapping method of [1] or the mapping method of [7] has a vast value space. Then by applying different biases to each matrix element, theft method 1 would fail to recover the weight matrix from accessing a single crossbar of the positive/negative crossbar pair.

1) Explore the space of the bias: Let's assume

$$G_{on} = \eta G_{off}, \tag{5}$$

where η ($\eta > 1000$ [2]) is the G_{on}/G_{off} ratio.

Theoretically, the ideal analog RRAM cell can be tuned into any arbitrary conductance state between G_{off} and G_{on}. Then the weight mapping methods for analog RRAM devices can be described as:

$$\begin{aligned} when\ w_{ij} \geqslant 0, &\begin{cases} g_{ij}^+ = b_1 + x_1 \cdot w_{ij} \\ g_{ij}^- = b_1 + (x_1 - 1) \cdot w_{ij} \end{cases} \\ when\ w_{ij} < 0, &\begin{cases} g_{ij}^+ = b_2 + x_2 \cdot w_{ij} \\ g_{ij}^- = b_2 + (x_2 - 1) \cdot w_{ij} \end{cases} \end{aligned} \tag{6}$$

where the bias $b_1 \in [G_{off}, G_{on}]$ for $w \geqslant 0$ and the bias $b_2 \in [G_{off}, G_{on}]$ for $w < 0$.

For ensuring g_{ij}^+ and g_{ij}^- to be continuous within their range, g_{ij}^+ and g_{ij}^- must satisfy $\lim_{w_{ij} \to 0^-} g_{ij}^+ = g_{ij}^+|_{w_{ij}=0}$ and $\lim_{w_{ij} \to 0^-} g_{ij}^- = g_{ij}^-|_{w_{ij}=0}$. Thus we get $b_1 = b_2$. Assume $b_1 = \lambda G_{off}$, where λ ($\lambda \in [1, \eta]$) is the bias scale. Then, according to Eq. (6), we observe that g_{ij}^+ and g_{ij}^- monotonically increase or decrease with w_{ij}. Thus the cell's maximum and minimum conductance could only be reached when w_{ij} is in one of its two ends. That is

$$g_{ij}^{\pm}|_{w_{ij}=\pm(G_{on}-G_{off})} \in [G_{off}, G_{on}], \tag{7}$$

Together with Eq. (5)(6)(7), we get

$$x_1 = \frac{\eta - \lambda}{\eta - 1},\ x_2 = \frac{\lambda - 1}{\eta - 1}. \tag{8}$$

Thus, Eq. (6) can be rewritten as

$$\begin{aligned} when\ w_{ij} \geqslant 0, &\begin{cases} g_{ij}^+ = \lambda G_{off} + \frac{\eta-\lambda}{\eta-1} \cdot w_{ij} \\ g_{ij}^- = \lambda G_{off} - \frac{\lambda-1}{\eta-1} \cdot w_{ij} \end{cases} \\ when\ w_{ij} < 0, &\begin{cases} g_{ij}^+ = \lambda G_{off} + \frac{\lambda-1}{\eta-1} \cdot w_{ij} \\ g_{ij}^- = \lambda G_{off} - \frac{\eta-\lambda}{\eta-1} \cdot w_{ij} \end{cases} \end{aligned} . \tag{9}$$

For analog RRAM cells that could be tuned into the state of any conductance value between G_{on} and G_{off}, the bias scale λ can be any random value between 1 and η. Though the precision of the peripheral circuits limits the precision of the bias b_1 (or b_2), we argue the number of the possible values of λ is considerably large.

2) Apply randomized biases to weight matrix: Based on the abundant number of bias b_1 (or b_2) found above, we propose to randomize the bias chosen for each element in the weight matrix. Denote the number of the bias choices as N_b and the number of crossbar cells as N_c. Then the time complexity of inferring the stored weights from a single analog crossbar is $O({N_b}^{N_c})$. Assuming that $N_b = 1000$ and $N_c = 256$, then the trial times of recovering the weight matrix from one single analog RRAM crossbar through brute-force is 1000^{256}.

Thus we claim that by randomizing the bias for each weight, theft method 1 is well thwarted.

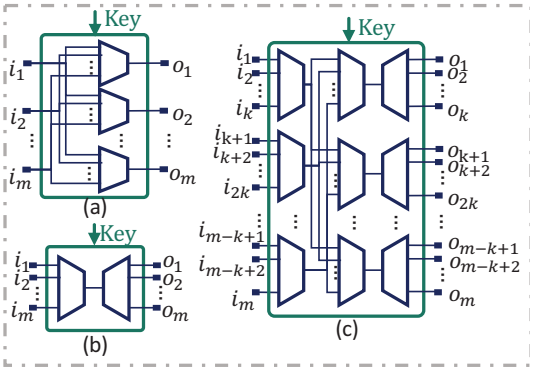

Fig. 3. Different implementation of the obfuscation module: (a) each time m inputs and m output are connected; (b) each time only 1 input and 1 output are connected; (c) combines (a) and (b) so that each time x inputs and x outputs are connected.

B. Countermeasure against theft method 2: hide the row connections of each positive/negative crossbar pair

By accessing both of the positive and negative crossbars and taking subtraction operations, it is quite trivial for the adversary to infer the stored NN weights. In this part, we propose to hide the connections between the positive and negative crossbars, so that the adversary can not infer the NN weights even with access to both of the pair crossbars.

For the sake of hiding crossbar row connections, we design a row connection obfuscation module and insert it between positive and negative crossbars, which is demonstrated in Fig. 2(d). As shown in Fig. 3(a), the obfuscation module is based on multiplexers, which has been applied in digital circuits [12], [13]. Because the inputs and outputs of the inserted obfuscation module are analog signals, we use analog switches, which are made up of pairs of MOSFET transistors, instead of digital multiplexers to make Fig. 3(a) applicable in our case. The obfuscation module hides the connections of crossbar rows. Unless knowing the specific connection relationship (key), directly use the stored weight value for computing will greatly harm the NN accuracy.

For an $m \times m$ obfuscation module, there are $m!$ possible combinations of connections between its inputs and outputs. Take $m = 64$ as an example, $m!$ is over 2^{295}, which is very hard to break for brute-force attack.

However, for the implementation of the the obfuscation module in Fig. 3(a), it costs m m : 1 multiplexers [13]. Applying this kind of obfuscation module to each crossbar pair in RRAM computing systems will bring non-negligible extra area overhead. In order to settle this problem, we put forward two techniques to reduce the overhead of the multiplexer-based obfuscating module:

1) Optimization technique 1: reducing multiplexers' number through adding a layer of demultiplexers: To reduce the area costs of the the obfuscation module, i.e., Fig. 3(a), we could use an m : 1 multiplexer and a 1 : m demultiplexer instead, which are shown in Fig. 3(b). However, this solution results that for each cycle only one row of the positive crossbar and its corresponding row of the negative crossbar are involved in computing, resulting in unacceptable latency costs.

TABLE II
LATENCY/AREA OVERHEAD COMPARISON OF FIG. 3

Implementation	Fig. 3(a)	Fig. 3(b)	Fig. 3(c)
Normalized latency	1×	256×	16×
Normalized area	1×	0.0078×	0.0110×

In order to mitigate the area/latency overhead in Fig. 3(a)/(b), we combine these two solutions together, which is shown in Fig. 3(c). Assume $m = xk, (x, k \in Z)$. The combination solution uses $2x$ k : 1 multiplexers and x 1 : k demultiplexers. The security of the m : m combination obfuscation module is $x! \cdot (k!)^x$. Each time x rows of the positive crossbar and the negative crossbar are calculated. Besides, in RRAM computing systems, only part of WLs are activated each time due to the current limitation of crossbar BLs [14]. Therefore, if x is equal to the number of enabled WLs in each cycle, the combination solution does not bring in any additional latency overhead.

For example, Table II shows the area/latency overhead comparison of these three implementations of a 256 : 256 obfuscation module. Assume each cycle 16 WLs are turned on and $x = 16$. As we can see, compared with Fig. 3(a), Fig. 3(c) brings in 16 times latency overhead. However, in this case where 16 WLs are enabled, the additional latency overhead brought in by Fig. 3(c) is the same with that brought in by Fig. 3(a). And the area overhead of Fig. 3(c) is only 1.10% of that of Fig. 3(a).

2) Optimization technique 2: protecting only partial NN layers/weight bits: To further reduce the area overhead of the obfuscation module while keeping the security, we explore the sensitivity of each layer to the NN classification accuracy when the crossbar row connections are obfuscated. In our experiment, we obfuscate the crossbar row connections of one layer, and test the classification accuracy of the incorrectly extracted NN model, i.e., without recovering the weights from obfuscation. Then the low accuracy ensure the security of the obfuscation method. The results are shown in Fig. 4, we notice that obfuscating each single layer has different impacts on NN classification accuracy. We call the layer which causes low accuracy after obfuscation as a significant layer, and denote the most significant layer as MSL. As we can see, the layers close to the model inputs are generally more significant, because the error caused by the protection method will propaganda through the rest layers. For example, when we obfuscate crossbar connections of the first layer, the accuracy of all the incorrectly extracted NNs is $< 45\%$. In the Section V, we will show that by only obfuscating minimal number of significant layers, the NN models are still well protected.

What is more, for digital RRAM devices, multiple crossbar pairs are utilized for representing a high-precision value. According to [15], the MSB of a weight is more important than the LSB of it. Therefore, we also study the impact of bit obfuscation on accuracy in digital RRAM computing systems, which is shown in Fig. 5. From the results we know that only obfuscating the corresponding crossbar pairs of the significant bits can still ensure the security.

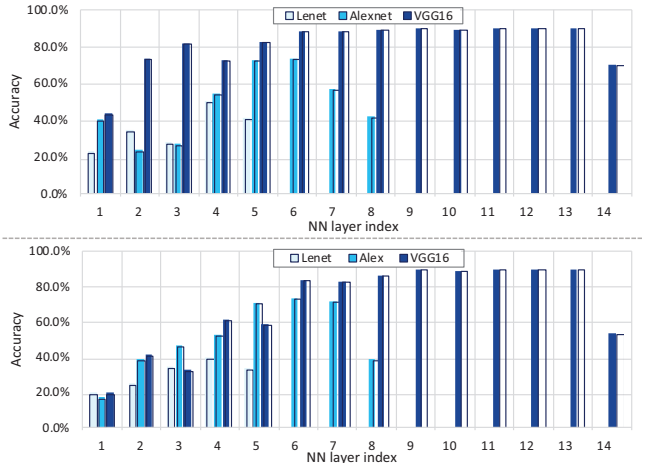

Fig. 4. Classification accuracy of extracted NN models with obfuscating only one single layer for ***Up:*** analog RRAM; ***Down:*** digital RRAM. Lower accuracy means the layer has more significant impact on the NN model

C. Overall workflow of the security-enhanced RRAM computing systems

For an analog RRAM computing system, besides the obfuscation module, a hardware random number generator (HRNG) and a tamper-proof memory (TPM) are also embedded in the system. The HRNG is to generate the random bias scale λ for the mapping method and the TPM is to store the key of the obfuscation module. Provided with a clean analog RRAM computing system, the authorized user first load his/her NN weights to the on-chip buffer and his/her obfuscation key to the TPM. The conductance g^+ and g^- of each pair of cells are determined with the λ generated by the HRNG. Then the rows of the positive crossbar are tuned according to the corresponding matrix while the rows of its pairing negative crossbar are permuted according to the obfuscation key from the TPM and then are tuned. After setup, each time the analog RRAM computing system is used for inference, the obfuscation module needs to be configured according to key from the TPM. As to the digital RRAM computing system, it does not need the HRNG. The obfuscation module of the digital RRAM computing system is the same with that of the analog RRAM computing system.

V. Experiments

We implement the proposed methods on three NN models: LeNet [16], AlexNet [17], and VGG16 [18]. These models are modified and well trained on Cifar10 dataset with 8-bit weights [19].For analog RRAM devices, we assume the G_{on}/G_{off} ratio η is 1000. For digital RRAM devices, we assume they could only be tuned to HRS or LRS (1-bit). And the area of RRAM devices we used for simulation refers to [20]. The crossbar size is 256×256, and we assume 16 WLs can be enabled simultaneously. The size of the obfuscation module for a pair of protected crossbars is $256:256$, which consists of 32 $16:1$ multipliers and 16 $1:16$ demultipliers. The obfuscation modules are simulated based on $45nm$ technology.

A. Security assessment of the full obfuscation method

Firstly, we evaluate the security of the full obfuscation method, i.e., all crossbar pairs are protected. As shown in Table III, the classification accuracy of the incorrectly extracted NN models that are fully obfuscated are all less than 12% for analog and digital RRAM computing systems. The results mean that without the correct keys of the obfuscation modules, the extracted NN models by the adversary are useless.

TABLE III
Classification accuracy of correctly/incorrectly extracted NN models that are fully obfuscated

	Correctly extracted	Incorrectly extracted (analog RRAM)	Incorrectly extracted (digital RRAM)
LeNet	65.13%	11.82%	9.04%
AlexNet	73.57%	9.81%	10.32%
VGG16	90.07%	9.61%	11.36%

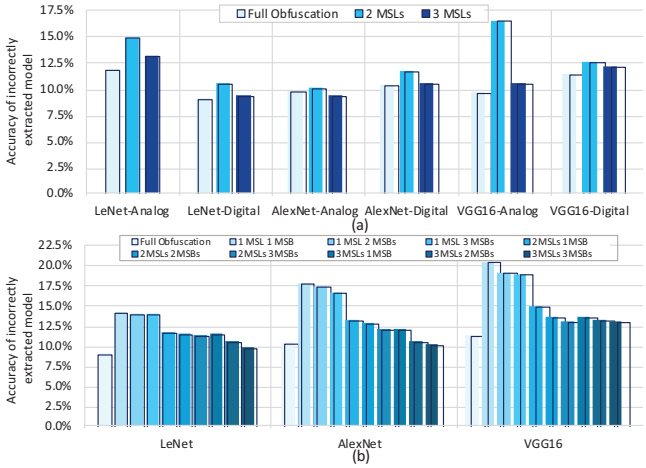

Fig. 5. Classification accuracy of incorrectly extracted NN models with obfuscating only ***(a):*** significant layers (MSLs); ***(b):*** significant bits (MSBs) of significant layers (MSLs)

B. Security assessment of the partial layer obfuscation method

In this part, we will show that by only obfuscating minimal number of NN layers, the weights of the three NN models are still well protected.

1) Only obfuscating the significant layers: We only obfuscate two or three MSLs of these NN models and the results are shown in Fig. 5(a). As can be seen, with obfuscating only two MSLs, the classification accuracy of all the incorrectly extracted NN models would decrease to $< 17\%$.

2) Only protecting the significant bits of the significant layers: Additionally, for 1-bit digital RRAM, we further reduce the overhead by only protecting the significant bits of the significant layers as demonstrated in Section IV-B. Based on previous experiments, we obfuscate the one, two, and three MSBs of MSLs. The results are shown in Fig. 5(b). As we can see, by obfuscating only one MSB of the two MSLs, the classification accuracy of all the incorrectly extracted NN models are $< 15\%$.

C. Security and hardware overhead

For the effectiveness of the proposed method, we set three obfuscation object thresholds (α): 14%, 17%, and 20%. For example, $\alpha = 14\%$ means the classification accuracy of the incorrectly extracted NN models that are obfuscated is less than 14%. As to security, obfuscating only one pair of crossbars would provide security of $(16!)^{17}$, which is secure

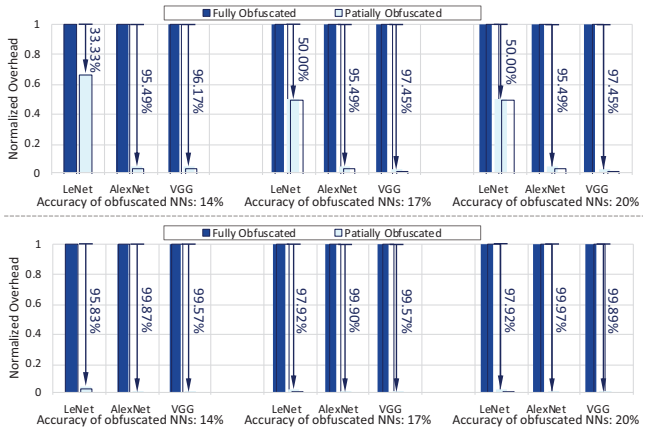

Fig. 6. Hardware area overhead reduced by obfuscating only partial NN layers and weight bits under three obfuscation object thresholds compared with fully obfuscating all layers and all bits. ***Up:*** analog RRAM; ***Down:*** digital RRAM

TABLE IV
AREA OVERHEAD OF THE PROPOSED METHODS COMPARED WITH THE AREA OF RRAM CROSSBARS UNDER THREE OBFUSCATION OBJECT THRESHOLDS (α)

		$\alpha = 14\%$	$\alpha = 17\%$	$\alpha = 20\%$
LeNet	analog RRAM	1.6220%	1.2166%	1.2166%
	digital RRAM	0.1014%	0.0507%	0.0507%
AlexNet	analog RRAM	0.1098%	0.1098%	0.1098%
	digital RRAM	0.0032%	0.0024%	0.0008%
VGG16	analog RRAM	0.0932%	0.0622%	0.0622%
	digital RRAM	0.0104%	0.0104%	0.0026%

enough. The weights of all the three NNs are mapped to more than one crossbar pairs, so the security of them is guaranteed.

Fig. 6 shows the ratio of area overhead reduced by partial obfuscation under different obfuscation object thresholds compared with full obfuscation. As can be seen, the overhead reduced by $33.33\% \sim 97.45\%$ and $95.83\% \sim 99.97\%$ for analog and digital RRAM computing system, respectively.

Table IV shows the hardware area overhead of the obfuscation modules compared with that of the RRAM crossbars in the RRAM computing systems under different obfuscation object thresholds. For example, to ensure the classification accuracy of the obfuscated LeNet model less than 14%, the area overhead is only 1.6220% of the area of analog RRAM crossbars mapped with LeNet weights. And for AlexNet and VGG, the area overhead of obfuscation module is $< 0.1\%$, which comes from two facts: 1. the crossbars need to be obfuscated account for a small part of the entire computing system, e.g., only $< 3.5\%$ crossbars need to be protected in VGG and AlexNet; 2. the area of obfuscation module is quite small compared with crossbars, i.e., $< 1\%$ of the crossbar pair's area. Therefore, our proposed method can effectively protect the NN models in RRAM crossbars with minimal area overhead.

VI. CONCLUSION

The RRAM computing systems are vulnerable to piracy attack targeting at stealing the NN weights stored in the RRAM crossbars. To thwart the attack, this paper proposed an effective countermeasure based on obfuscating the crossbar row connections between the positive crossbars and their pairing negative crossbars. Besides, we put forwards two techniques to reduce the hardware overhead of the obfuscation module. The experiment results show that the proposed methods ensure the trial times of brute-force attack are more than $(16!)^{17}$ and the classification accuracy of the incorrectly extracted NN models is less than 20%, with only minimal area overhead.

VII. ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China (No. 61832007, 61622403, 61672115, 61621091, 61532017, 61802185), Beijing National Research Center for Information Science and Technology (BNRist), Beijing Innovation Center for Future Chips, and Natural Science Foundation of Jiangsu Province (No. BK20190447, BK20180470).

REFERENCES

[1] B. Li *et al.*, "RRAM-Based Analog Approximate Computing," *IEEE TCAD*, vol. 34, pp. 1905–1917, 12 2015.
[2] C. Wang *et al.*, "Cross-point Resistive Memory: Nonideal Properties and Solutions," *ACM Transactions on Design Automation of Electronic Systems*, vol. 24, pp. 1–37, 6 2019.
[3] M. Zou *et al.*, "PoTrojan: powerful neural-level trojan designs in deep learning models," *arXiv preprint arXiv:1802.03043*, pp. 1–7, 2 2018.
[4] W. Li *et al.*, "P3M: A PIM-based Neural Network Model Protection Scheme for Deep Learning Accelerator," in *ASPDAC '19*, pp. 633–638, ACM Press, 2019.
[5] S. Chhabray and Y. Solihin, "I-NVMM: A secure non-volatile main memory system with incremental encryption," *ISCA' 11*, pp. 177–188, 2011.
[6] P. Yao *et al.*, "Face classification using electronic synapses," *Nature Communications*, vol. 8, 2017.
[7] Z. He *et al.*, "Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping," in *DAC '19*, pp. 1–6, ACM Press, 2019.
[8] W. Wu *et al.*, "A methodology to improve linearity of analog RRAM for neuromorphic computing," *VLSIT'18*, vol. 2018-June, pp. 103–104, 2018.
[9] A. Prakash *et al.*, "Demonstration of low power 3-bit multilevel cell characteristics in a TaOx-Based RRAM by stack engineering," *IEEE Electron Device Letters*, vol. 36, no. 1, pp. 32–34, 2015.
[10] Y. Cai *et al.*, "Low Bit-width Convolutional Neural Network on RRAM," *TCAD*, pp. 1–1, 2019.
[11] Z. Zhu *et al.*, "A Configurable Multi-Precision CNN Computing Framework Based on Single Bit RRAM," in *DAC '19*, pp. 1–6, ACM Press, 2019.
[12] Z. Guo *et al.*, "Investigation of obfuscation-based anti-reverse engineering for printed circuit boards," in *DAC '15*, pp. 1–6, ACM Press, 2015.
[13] S. Zamanzadeh and A. Jahanian, "Automatic netlist scrambling methodology in ASIC design flow to hinder the reverse engineering," in *IEEE/IFIP VLSI-SoC*, pp. 52–53, 2013.
[14] W. H. Chen *et al.*, "A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," *ISSCC'18*, vol. 61, pp. 494–496, 2018.
[15] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv preprintarXiv:1510.00149*, 2015.
[16] Y. Lecun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[17] A. Krizhevsky *et al.*, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.
[18] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, pp. 1–14, 9 2014.
[19] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," *Science Department, University of Toronto, Tech.*, pp. 1–60, 2009.
[20] W. Wu *et al.*, "Suppress variations of analog resistive memory for neuromorphic computing by localizing V_o formation," *Journal of Applied Physics*, vol. 124, p. 152108, 10 2018.