Review of security techniques for memristor computing systems

Minhui Zou[1]*, Nan Du[2,3] and Shahar Kvatinsky[1]

[1]Viterbi Faculty of Electrical and Computer Engineering, Technion – Israel Institute of Technology, Haifa, Israel, [2]Institute for Solid State Physics, Friedrich Schiller University Jena, Jena, Germany, [3]Department of Quantum Detection, Leibniz Institute of Photonic Technology (IPHT), Jena, Germany
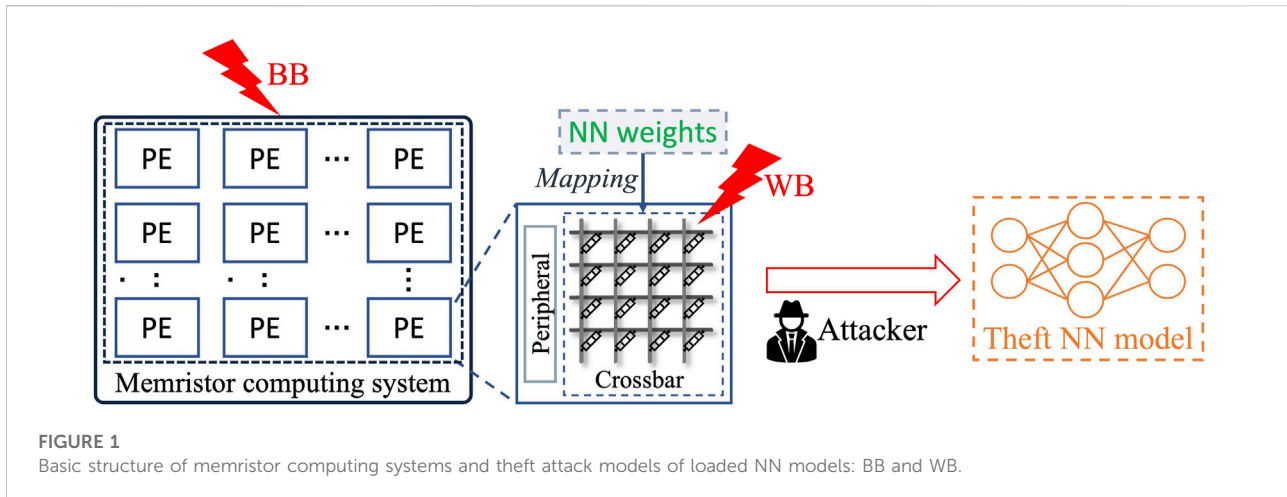
Neural network (NN) algorithms have become the dominant tool in visual object recognition, natural language processing, and robotics. To enhance the computational efficiency of these algorithms, in comparison to the traditional von Neuman computing architectures, researchers have been focusing on memristor computing systems. A major drawback when using memristor computing systems today is that, in the artificial intelligence (AI) era, well-trained NN models are intellectual property and, when loaded in the memristor computing systems, face theft threats, especially when running in edge devices. An adversary may steal the well-trained NN models through advanced attacks such as learning attacks and side-channel analysis. In this paper, we review different security techniques for protecting memristor computing systems. Two threat models are described based on their assumptions regarding the adversary's capabilities: a black-box (BB) model and a white-box (WB) model. We categorize the existing security techniques into five classes in the context of these threat models: thwarting learning attacks (BB), thwarting side-channel attacks (BB), NN model encryption (WB), NN weight transformation (WB), and fingerprint embedding (WB). We also present a cross-comparison of the limitations of the security techniques. This paper could serve as an aid when designing secure memristor computing systems.

KEYWORDS

neural network, memristor computing system, hardware security, theft threat, defense technique, neuromorphic

1 Introduction

Neural network (NN) algorithms have demonstrated great potential in certain areas such as visual object recognition, natural language processing, and robotics (Dong et al., 2021); (Károly et al., 2020). These algorithms involve a large number of vector-matrix multiplications (VMMs) and hence they are both data- and computing-intensive. Conventional computer architectures are designed by following the von Neumann model wherein the computation unit and memory are separate. When running on conventional computer architectures, NN algorithms require huge amounts of matrix data to be moved between the computational unit and memory, which demands a lot of time and energy. To overcome this obstacle, researchers and industry have turned to the emerging computing systems based on memristor devices. These systems could improve

FIGURE 1
Basic structure of memristor computing systems and theft attack models of loaded NN models: BB and WB.

the efficiency of NN algorithms because they process the VMM operations directly in the memory so that data movement between the computing unit and memory is avoided. Moreover, the memristor devices could be structured in the form of a crossbar array. The input signals at the rows (wordlines, WLs) go through the memristor cells to accumulate at the columns (bitlines, BLs), which naturally analogizes VMM operations (Marković et al., 2020). The analogous VMM operations can be finished in a constant number of clock cycles, i.e., their time complexity is $O(1)$ (Li et al., 2015), which further significantly improves the efficiency of the NN algorithms.

As shown in Figure 1, a memristor computing system contains many processing elements (PE), with each PE comprising a memristor crossbar and peripheral circuits. The NN weights are mapped to the memristor crossbars. Thanks to the non-volatile nature of memristor devices, the mapped NN weights will not be lost when the systems are rebooted and thus no remapping is needed. Advanced memristor computing systems exploit the intra- and inter-parallelism of memristor crossbars to boost the energy efficiency of different NN algorithms such as convolutional NN (Wen et al., 2019; Wen et al., 2020), graph NN (Lyu et al., 2022), and spiking NN (Rathi et al., 2022). Various hardware-software co-optimization techniques have been proposed to push memristor computing systems to their energy efficiency limits. For example, Wen et al. (2019), Wen et al. (2020) suggest pruning NN parameters with improved convolution algorithms to reduce the number of required memristors. These advantages of memristor computing systems make them promising for edge devices with constrained computational and energy budgets.

Training an NN model is a computationally heavy task that demands a great deal of energy and financial resources. It may take days or even months (Strubell et al., 2019) and cost up to more than one million US dollars (Sharir et al., 2020) to output a

well-trained NN model from scratch. Furthermore, large NN models involve a huge amount of training data. For example, GPT-3 was trained on 570 GB of filtered data Brown et al. (2020). With NN models becoming larger and deeper, NN model training is becoming less advantageous for those who cannot afford the high training cost or the large training datasets. Additionally, if the training datasets are proprietary, there is always the risk that the trained NN models may leak confidential information contained in the datasets (Rajasekharan et al., 2021). Nowadays, well-trained NN models are deemed to be intellectual property and protecting them from being theft is imperative.

NN models, loaded in memristor computing systems, as shown in Figure 1, may be vulnerable to theft threats such as learning attacks (Tramèr et al., 2016) and side-channel attacks (Hua et al., 2018). Even worse, while the memristor devices' non-volatility might be a appealing feature, it facilitates data theft attacks. In the scenario of using memristor devices as main memory, data theft attacks have been widely considered as real threats (Young et al., 2015; Awad et al., 2016; Awad et al., 2019; Zuo et al., 2019). In memristor computing systems, the data persistence of the memristor devices may also expose the NN weights stored on the memristor crossbars to an adversary. Besides, Huang et al. (2020) claimed it is also possible to able to read the stored NN weights from the systems through micro-probing the peripheral circuits of memristor crossbars. All these threats represent serious security challenges to memristor computing systems.

In this paper, we review different security techniques for protecting NN models for memristor computing systems. Previous works either focused on NN security techniques from the viewpoint of software (Oseni et al., 2021) or security threats facing the memristor computing system hardware (Hu et al., 2022). This review, in contrast, focuses on memristor computing hardware defense security techniques. The structure and major contributions of this paper are summarized as follows:
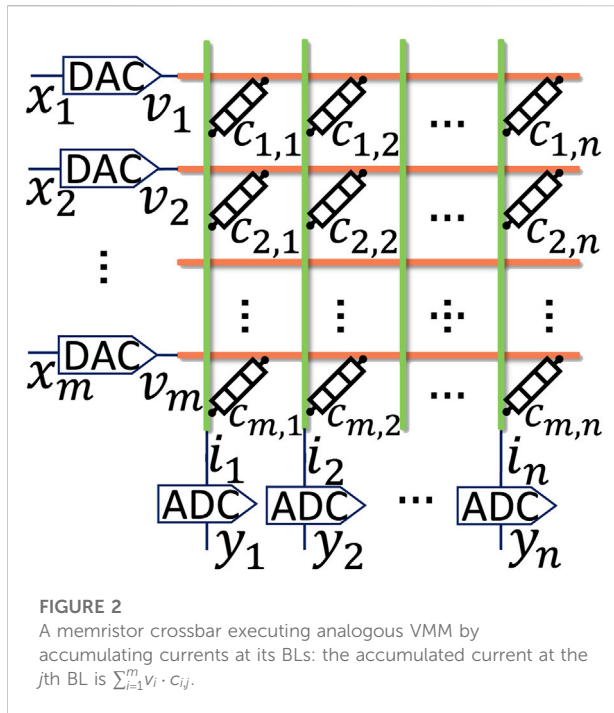
FIGURE 2
A memristor crossbar executing analogous VMM by accumulating currents at its BLs: the accumulated current at the jth BL is $\sum_{i=1}^{m} v_i \cdot c_{i,j}$.

- We categorize the existing security techniques into five classes defined by the black-box (BB) and white-box (WB) threat models.
- We present a brief overview of the existing security techniques under the above categorization.
- We present a cross-comparison of the limitations of the various security techniques.
- We discuss the challenges of the existing countermeasures and suggest future research directions in this field.

2 Background

2.1 Preliminaries

The most computationally heavy and time-consuming parts of NN algorithms are their convolution (Conv) layers and fully-connected (FC) layers. The main work of FC layers can be implemented directly with VMMs, described as:

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

where $x_i (i \in [1, m])$ is the input feature map, $y_j (j \in [1, n])$ is the output, and $w_{i,j}$ is the synapse weight. The main work of Conv layers is different but could also be transformed to be implemented with VMMs. x_i and y_j, respectively, are still the input feature map and the output while each column of the weight matrix is a vector transformed from a filter kernel. To

simplify our discussion, we assume the weights of the Conv layers are already transformed into matrices. Thus, the weights of both the FC layers and the Conv layers are in the form of matrices.

In memristor computing systems, as shown in Figure 2, the input feature maps are transformed into voltages (**v**) by using digital-to-analog converters (DACs) that are applied to the WLs of the memristor crossbars. The BLs of the memristor crossbars output the accumulated currents (**i**). The output currents are then transformed by using analog-to-digital converters (ADCs). The analogous VMMs performed by a memristor crossbar are described as:

$$i_j = \sum_{i=1}^{m} v_i \cdot c_{i,j}, \tag{2}$$

where $c_{i,j}$ is the conductance of the cell at the crossbar's ith row and jth column. Memristor devices can be categorized as either analog (Wu et al., 2018) or digital (Prakash et al., 2014). Analog memristor devices move gradually through a RESET process, which means an memristor device can be tuned from low resistance state (LRS) to high resistance state (HRS) continuously. Consequently, an ideal analog memristor cell could be tuned into any arbitrary conductance state between LRS to HRS. Digital memristor devices can only be tuned to limited discrete resistance states (Prakash et al., 2014). Accordingly, multiple crossbars are used to represent high-precision weights (Cai et al., 2019b; Zhu et al., 2019; Zhu et al., 2020). An NN weight can be positive or negative, but the conductance of memristor devices can only be positive. To support negative weights, different mapping schemes were proposed. Two popular mapping schemes are biasing the original weights to be non-negative (bias-based mapping) Shafiee et al., 2016; Xue et al., 2020) and using the differential values of pairs of memristor devices to represent the original weights (differential mapping) (Chi et al., 2016; Zhu et al., 2019. Besides, NN algorithms are typically processed layer by layer, and the outputs of one layer are used as the inputs of the next layer. Large NN layers are assigned to multiple PEs, and the partial sums are aggregated in a global buffer (Long et al., 2019; Krishnan et al., 2021). To maximize the processing parallelism of memristor crossbars, some memristor computing systems (Zhu et al., 2020; Wan et al., 2022) proposed to directly transfer the partial outputs of a NN layer to the PEs where its next layer is located. Such inter-layer parallelism, however, may cause pipeline bubbles (Qiao et al., 2018).

2.2 Threat models and countermeasures

The current commercial memristive computing chips are embedded in boards with M.2 (Mythic, 2022b) or PCIe (Mythic, 2022c) interfaces. The memristive chips may be also equipped with I/Os ports such as GPIOs and I²C (Mythic, 2022a). In line
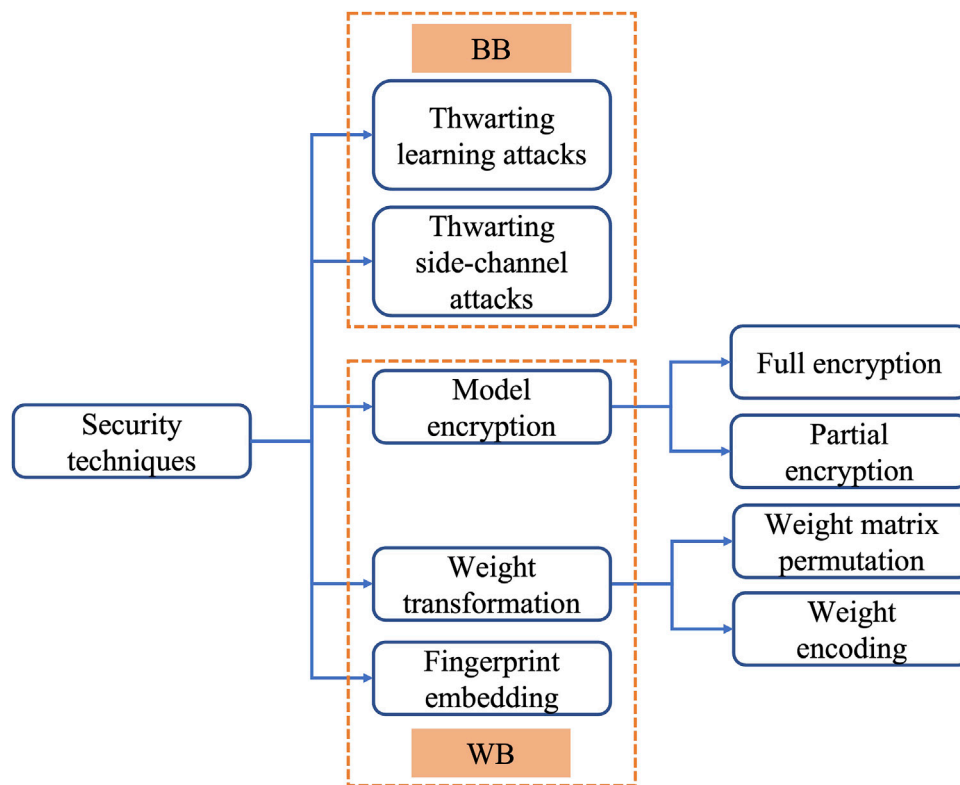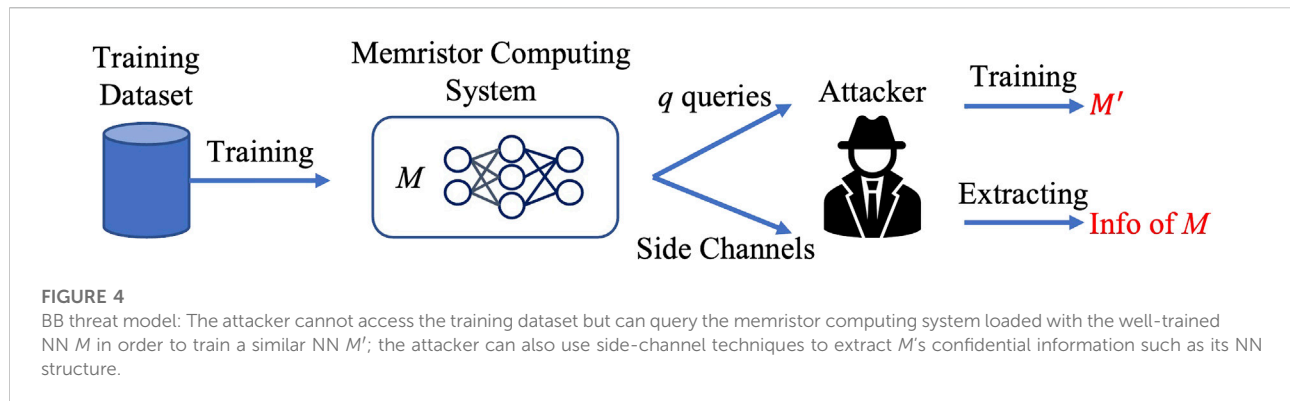
FIGURE 3
Security technique categorization: thwarting learning attacks (BB), thwarting side-channel attacks (BB), NN model encryption (WB), NN weight transformation (WB), and fingerprint embedding (WB).

with the different security techniques, the adversary with physical access to the boards is assumed to have different capabilities. As shown in Figure 3, the threat models are divided into BB models and WB models. In the BB model, as shown in Figure 1, the adversary can only access the inputs/outputs of the memristor computing systems and cannot access directly the intermediate states of the systems such as the intermediate NN layer outputs or NN weights. The adversary could, however, exploit side-channel information such as the memory access patterns or power consumption. One common BB model attack is a learning attack, which is based on collecting a certain number of the NN models' input/output pairs (Tramèr et al., 2016). Another potential type of attack is a side-channel attack that exploits the systems' covert information (Hua et al., 2018; Batina et al., 2019; Dubey et al., 2019; Yan et al., 2020). In the WB model, as shown in Figure 1, the adversary is assumed to be able to read the stored NN weights from the systems. This threat model, a result of the non-volatility of memristor devices, was first widely considered by existing works for the memristor devices' application as the main memory (Young et al., 2015; Awad et al., 2016; Awad et al., 2019; Zuo et al., 2019). For memristor computing systems, the adversary could exploit the universal interfaces of the boards

or ports of the chips, and hence the data theft threat is also viable. Besides, Huang et al. (2020) claimed it is also possible to able to read the stored NN weights from the systems through micro-probing the peripheral circuits of memristor crossbars, which further contributes to the strength of the WB model. The existing countermeasures for the WB model manage to prevent the adversary from reading the NN weights correctly. They are categorized into three classes: NN model encryption, NN weight transformation, and fingerprint embedding. In Section 3 and Section 4, we discuss the countermeasures for the BB model and WB model, respectively, in detail.

3 Black-box threats and countermeasures

For the BB threat model, memristor computing systems are akin to proverbial black boxes as far as concerns the attacker. As shown in Figure 4, the attacker cannot access the training dataset or the NN weights, but can manipulate the inputs and observe the outputs and exploit side-channel analysis.

FIGURE 4
BB threat model: The attacker cannot access the training dataset but can query the memristor computing system loaded with the well-trained NN M in order to train a similar NN M'; the attacker can also use side-channel techniques to extract M's confidential information such as its NN structure.

3.1 Thwarting learning attacks

Learning attacks are a powerful NN model theft technique. The adversary queries the system and observes the outputs. Once sufficient input/output pairs are collected, a similar functioning NN model could be trained using the input/output pairs (Tramèr et al., 2016). As shown in Figure 4, the training center trains a proprietary NN and implements it in an edge device as a service provider, which is accessible to the attacker. The attacker queries the devices q times to get q input/output pairs and trains an NN M' that is comparable to the original M.

To thwart this kind of attack, Yang et al. (2020) proposed to leverage the obsolescence effect of memristor devices to reduce the inference accuracy of the computing systems for unauthorized users. The obsolescence effect of memristor devices causes them to be in the HRS or LRS as a result of read voltage pulses across them. The authors of (Yang et al., 2020) enhanced the obsolescence rate of memristor devices by increasing the voltage amplitude so that the inference accuracy of the computing systems drops dramatically following a limited number of queries. To maintain the computing system's inference accuracy, the memristor devices need to be calibrated by writing the original weights on them to counter the obsolescence; however, the authors assumed the calibration process does not work when initiated by unauthorized users[1]. Thus, the attacker, as an unauthorized user, cannot collect sufficient input/output pairs to reverse-engineer the NN models loaded in the computing systems.

(Rajasekharan et al., 2021) proposed to utilize the innate stochasticity in super-paramagnetic magnetic tunnel junctions (s-MTJs) to thwart learning attacks. s-MTJs are unstable and need to be refreshed periodically to keep the weights mapped on them. MTJs, different from s-MTJs, are stable and do not need refreshing operations. By only storing
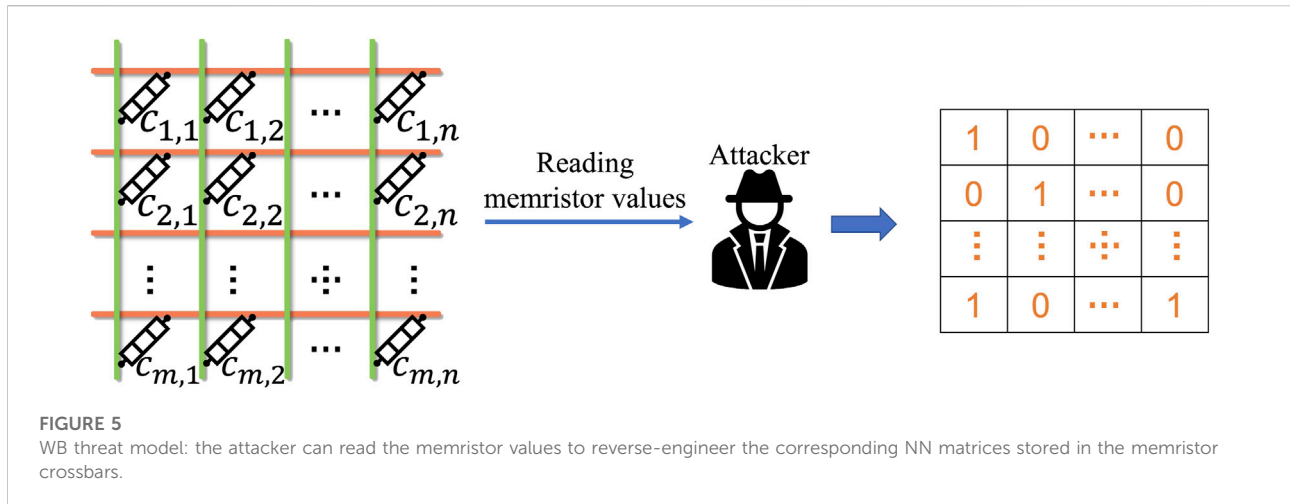
significant weights on s-MTJs and the other weights on MTJs, the number of cells requiring periodic refreshing is reduced. Similar to (Yang et al., 2020), the authors in Rajasekharan et al. (2021) assumed that only authorized users can keep the refreshing operations executing. The induced switching in the s-MTJs causes the systems' performance to deteriorate in a very short time if the s-MTJs are not refreshed. Thus, unauthorized users are thwarted in their effort to collect sufficient input/output pairs.

3.2 Thwarting side-channel attacks

Side-channel attacks are another type of non-invasive attack that treat the device under attack as a black box. These methods can exploit side channels, such as power, latency, and electromagnetism, to discover confidential information, as shown in Figure 4.

Some works (Hua et al., 2018; Batina et al., 2019; Dubey et al., 2019; Yan et al., 2020) have already explored how to steal the NN models through side-channel attacks. These attacks either target conventional computing architectures or FPGAs, which are different from memristor computing systems in implementing VMM operations. Some of these attacks, however, could be applied to memristor computing systems directly, such as that proposed by Hua et al. (2018), which is focused on memory accessing patterns. Hua et al. (2018) assumed that all the intermediate layer outputs are stored in memory and transferred to the computing unit as inputs of the next layer. The memory accessing patterns leak the NN structure information. For some memristor computing systems, a similar layer-by-layer processing technique is used (Qiao et al., 2018; Krishnan et al., 2021). Thus, the memory accessing patterns could also be a side-channel vulnerability that adversaries can exploit in memristor computing systems. To counter this attack, the authors referred to techniques for hiding the memory accessing patterns (Goldreich and Ostrovsky, 1996) that depend on oblivious RAM algorithms.

---

1 In their paper, authorized and unauthorized users refer to those who can access the computing systems legally and illegally, respectively.

FIGURE 5
WB threat model: the attacker can read the memristor values to reverse-engineer the corresponding NN matrices stored in the memristor crossbars.

4 White-box threats and countermeasures

For the white-box threat model, as shown in Figure 5, the attacker could read the NN weights stored in the memristor crossbar by exploiting the non-volatility of memristor devices (Young et al., 2015; Awad et al., 2016; Awad et al., 2019; Zuo et al., 2019). Alternatively, the attacker could use micro-probing techniques (Huang et al., 2020) so that the NN weight matrices stored in the memristor crossbar can be reverse-engineered.

4.1 Model encryption

Methods based on encrypting the NN weights are straightforward. Such methods encrypt the NN weights and decrypt them each time they are used. The methods require additional write operations to the memristor devices. To reduce the latency and high energy consumption produced by the encryption and decryption, Li et al. (2019); Lin et al. (2020) focused on accelerating and optimizing these processes and (Cai et al., 2019a) focused on reducing the necessary encryption/decryption data. One security concern of this type of method is that the NN weights on the memristor crossbars executing the NN algorithms are decrypted and thus exposed to the adversary.

4.1.1 Full encryption

Li et al. (2019) proposed to encrypt the whole NN model based on XOR before implementing it in the computing systems. The encrypted NN model is stored in off-chip memory and copied to the eDRAM buffer and then to the computing units. The decryption is done and accelerated in the eDRAM with modifier sense circuits. Lin et al. (2020) proposed to encrypt the NN weights based on chaotically exchanging the weight

positions; however, the encyption/decryption needs additional hardware and is not compatible with memristor crossbars.

4.1.2 Partial encryption

Cai et al. (2019a) proposed to select the most significant weights of each NN layer. The most significant weights are defined as the weight with the largest gradient when their sign bits are flipped. The authors showed that only encrypting the MSW can cause the NN models to become useless for inference tasks. Compared to the full encryption methods, the partial encryption method reduces the number of encryption/decryption weights significantly (e.g., only 20 per layer for ResNet-101).

4.2 NN weight transformation

In the NN weight transformation protection methods, the protected NN weights are stored in the computing units and there is no need to rewrite the weights so that the energy and latency brought by the weight rewriting operations are eliminated. Thus, this class of methods aims to minimize the imposed hardware overhead. Moreover, the weights are always protected – which is a significant benefit over the model encryption methods.

4.2.1 Weight matrix permutation

Zou et al. (2020) proposed to obfuscate the crossbar row connections between positive and negative crossbars. During inference, the obfuscating module is configured with the correct keys. Without the correct keys, the theft NN model cannot function properly. Wang et al. (2021) proposed to obfuscate the column connections between consecutive crossbars. The crossbars are first divided into smaller groups and then MUX and DEMUX are used to configure the groups of each two

consecutive crossbars. Huang et al. (2020)a[2] also considered obfuscating the crossbar column connections. Their implementation is based on SRAM arrays. By configuring the SRAM arrays with one-hot coding for each row, the SRAM arrays could permute the output channels into any order.

4.2.2 Weight encoding

Li et al. (2021) proposed to encode a weight w as its XORed result w_e, which is expressed as

$$w_e = w \oplus k, \tag{3}$$

where k is a key. The memristor crossbars store both w_e and its complementary $\hat{w}_e$ in the same crossbar column. The dot-product between input i and w can be implemented as

$$i \cdot w = i \cdot (w_e \oplus k) = (i \cdot k) \cdot \hat{w}_e + (i \cdot \hat{k}) \cdot w_e, \tag{4}$$

where $\hat{k}$ is k's complementary. Thus, the decoding for weight w could be done on the input, leaving the VMM operations on the memristor crossbar unaffected. Nevertheless, this method requires double the number of memristor devices compared with other methods.

Zou et al. (2022) proposed to selectively encode some columns of weights as their 1's complement and leave the other untouched. The attacker does not know which columns of weights are encoded so the actual representation of the weights is hidden. The authors designed and implemented the protection methods for bias-based mapping and differential mapping, respectively.

4.3 Fingerprint embedding

Methods based on fingerprint embedding use the hardware variation of the computing systems as a fingerprint and embed it inside the NN weights. The directly stolen NN model cannot function well without the fingerprint. The fingerprint is hardware dependent and hard to copy.

Huang et al. (2020)b observed that the ADC offset plays an important role in the model accuracy and the NN model needs to be refined to improve the accuracy. Thus, the intrinsic ADC offset pattern of a specific chip could be the fingerprint embedded in the NN model. The directly extracted NN models do not contain the fingerprint so that the NN models are well-protected.

---

2 In Huang et al. (2020), the authors proposed two protection techniques. To distinguish between them, we denote the two techniques as Huang et al. (2020)a and Huang et al. (2020)b, respectively.

5 Cross-comparison

We now present a cross-comparison of the limitations of the above security techniques. The results are summarized in Table 1.

5.1 Ability to protect from attacks by an authorized adversary

The adversary could be an authorized or an unauthorized user. As an authorized user, the adversary could carry out an unlimited number of inference tasks on a memristor computing system. In this scenario, the countermeasure of thwarting the learning attacks Yang et al. (2020) Rajasekharan et al. (2021) would not stop the adversary. In the other countermeasures, the keys of the protection methods can be configured as internal and not accessible by any users; thus they would work regardless of whether or not the adversary is authorized.

5.2 NN structures and NN parameters

The secrecy of a NN model includes both its structure and its parameters. For memristor computing systems, the countermeasure referred by Hua et al. (2018) protect only the NN structures. Some NN weight transformation methods, Zou et al. (2020); Wang et al. (2021); Huang et al. (2020)a; Zou et al. (2022) exploit the padded fake rows or columns to hide the exact size of weight matrices with fewer rows or columns than the size of the crossbar. As a result, the structures of these NN layers with small weight matrices are protected. Other countermeasures protect only the NN parameters. Note that learning attacks would not succeed easily without knowledge of the NN structures (Batina et al., 2019). The learning attacks thus could be treated as a way to steal the NN parameters. Hence, the methods proposed by Yang et al. (2020), Rajasekharan et al. (2021) are deemed to protect the NN parameters only.

5.3 Requiring retraining?

Huang et al. (2020)b requires the NN parameters to be retrained so that they are compatible with the hardware variation. This kind of method requires information about the target devices in advance of mapping the NN models to them. Retraining the NN models for each specific hardware systems, however, may be time-consuming and not practical. In the other countermeasures, retraining NN parameters is not necessary.

5.4 Additional memristor write operations

Both full (Li et al., 2019; Lin et al., 2020) and partial (Cai et al., 2019a) encryption techniques require additional write operations to the memristor

TABLE 1 Cross-comparison of the security techniques.

	Stop authorized adversary?	Structures (A) or parameters (B)?	Require retraining?	Memristor writing?	Analog (A) or digital (B)?	Full-time protection?
Yang et al. (2020)	No	B	No	Yes	A	N/A
Rajasekharan et al. (2021)	No	B	No	Yes	A and B	N/A
Hua et al. (2018)	Yes	A	No	No	A and B	N/A
Li et al. (2019)	Yes	B	No	Yes	A and B	No
Lin et al. (2020)	Yes	B	No	Yes	A and B	No
Cai et al. (2019a)	Yes	B	No	Yes	A and B	No
Zou et al. (2020)	Yes	B and partial A	No	No	A and B	Yes
Wang et al. (2021)	Yes	B and partial A	No	No	A and B	Yes
Huang et al. (2020)a	Yes	B and partial A	No	No	B	Yes
Li et al. (2021)	Yes	B	No	No	B	Yes
Zou et al. (2022)	Yes	B and partial A	No	No	A and B	Yes
Huang et al. (2020)b	Yes	B	Yes	No	A and B	Yes

devices for each inference. In the techniques suggested by Yang et al. (2020) and Rajasekharan et al. (2021) the memristor devices must be periodically written because of the devices' resistance drifting. Additional memristor write operations not only consume high energy and introduce long latency into the systems Chang et al. (2014); Yao et al. (2017), they also shorten the lifetime of the memristor computing systems due to the limited endurance of memristor devices Wang et al. (2019) The other countermeasures do not require additional memristor write operations.

5.5 Analog or digital memristor devices

In Yang et al. (2020), the authors only investigated the obsolescence effect in analog memristor devices. Determining whether this technique works for digital memristor devices needs further justification since the resistance states of digital memristor devices have larger margins than analog devices. Huang et al. (2020)a only works for digital devices because the SRAM arrays can only output binary results at the BLs. The technique proposed by Li et al. (2021) also only works for digital devices because the XOR-based encoding is only applicable for single-bit-precision devices. The other countermeasures would work for both kinds of devices.

5.6 Full-time protection

Full-time protection means that the systems are always protected whenever the adversary carries out an attack.

Among the countermeasures of the WB model, the model encryption methods (Cai et al., 2019a; Li et al., 2019; Lin et al., 2020) decipher the NN weights and store them as plaintext when they are involved in NN computing. The adversary may analyze the systems' side channels, such as power consumption, to pinpoint when the NN weights are involved in computing. Once the specific point in time is determined, the adversary could turn off the system and then read the unprotected NN weights from non-volatile memristors. Though (Cai et al., 2019a) narrowed the attack window by exposing only one NN layer at a time to the adversary, the threat is not thwarted. Conversely, the methods based on NN weight transformation leave no such attack window and protect the NN weights all the time.

6 Discussion

We now discuss the developing directions of security techniques for protecting NN models in memristor computing systems.

6.1 Countermeasures against comprehensive attacks

Each reviewed countermeasure can only prevent one type of specific attack. Sophisticated attackers, however, may maximize their attack strategies by combining a number of different

attacks. For example, the adversary may use side-channel attacks to infer the NN structures and micro-probing techniques to extract the NN weights. Simply adding more and more defense techniques may impose a heavy hardware overhead and even open up new opportunities for the adversary to exploit. Thus, more thoughtful countermeasures are required both to deliver protection for multiple attacks and ensure low hardware overhead.

6.2 More investigations into side-channels attacks

Side-channel attacks against other NN computing systems, such CPUs and FPGAs (Dubey et al., 2019; Yan et al., 2020), have attracted much attention. To date, we have seen few The side-channel attacks against memristor computing systems. Though different from other NN computing systems, memristor computing systems have a lot in common with them such as activation functions, pooling functions, and storing intermediate results in buffers. Side-channel attacks focusing on those attack surfaces in other NN computing systems may also be feasible in memristor computing systems. Thus, more investigations into side-channel attacks are needed.

6.3 Alternative approaches to thwarting learning attacks

The current learning attack countermeasures can only prevent attacks by unauthorized users. Authorized users, however, are neither barred nor limited in their access to the input/output pairs. Limiting authorized users' access to input/output pairs would impose an inconvenience for them, making this avenue an unrealistic option. Alternative approaches to thwarting learning attacks must consider learning attacks by authorized users. We suggest looking for alternate solutions from the perspectives of NN model training strategies or training data manifestation.

7 Conclusion

In recent years, memristor computing systems have dramatically improved the energy efficiency of NN algorithms. The security issues of memristor computing systems, however, should be addressed before they become widely used commercially. Specifically, the NN models loaded in memristor computing systems face potential theft threats because of memristors' non-volatile nature. In this paper, we reviewed the existing security techniques

for protecting NN models for memristor computing systems. Two threat models assume the adversary has different attack capabilities, i.e., the BB model and the WB model. For the threat models, based on the attack method, we classified the existing countermeasures into five sub-classes. Those countermeasures are limited because of their defense capabilities and hardware implementation. For example, the countermeasures against learning attacks can only protect the memristor computing systems from unauthorized users; the protection methods based on NN model encryption leave an attack window open when NN weights are involved in computing. We presented a cross-comparison of the limitations of the existing security techniques and suggested future research directions for this field.

Author contributions

MZ: Proposing the methodology, investigating the related works, designing the security metrics, and writing the original manuscript. ND: Proposing the methodology and reviewing and editing the manuscript. SK: Reviewing and editing the manuscript and supervising this project.

Funding

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Awad, A., Manadhata, P., Haber, S., Solihin, Y., and Horne, W. (2016). Silent shredder: Zero-cost shredding for secure non-volatile main memory controllers. *SIGPLAN Not.* 51, 263–276. doi:10.1145/2954679.2872377

Awad, A., Ye, M., Solihin, Y., Njilla, L., and Zubair, K. A. (2019). "Triad-nvm: Persistency for integrity-protected and encrypted non-volatile memories," in Proceedings of the 46th International Symposium on Computer Architecture, Phoenix, AZ, USA, 21-26 June 2019 (IEEE), 104–115.

Batina, L., Bhasin, S., Jap, D., and Picek, S. (2019). "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in In 28th USENIX Security Symposium, 16-18 July 2016 (Austin, TX: USENIX Security), 515–532.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., et al. (2020). Language models are few-shot learners. *Adv. neural Inf. Process. Syst.* 33, 1877–1901.

Cai, Y., Chen, X., Tian, L., Wang, Y., and Yang, H. (2019a). "Enabling secure in-memory neural network computing by sparse fast gradient encryption," in In 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 13 October 2019 (IEEE), 1–8. doi:10.1109/ICCAD45719.2019.8942041

Cai, Y., Tang, T., Xia, L., Li, B., Wang, Y., and Yang, H. (2019b). Low bit-width convolutional neural network on rram. *IEEE Trans. Comput. -Aided. Des. Integr. Circuits Syst.* 39, 1414–1427. doi:10.1109/tcad.2019.2917852

Chang, M.-F., Wu, J.-J., Chien, T.-F., Liu, Y.-C., Yang, T.-C., Shen, W.-C., et al. (2014). "19.4 embedded 1Mb ReRAM in 28nm CMOS with 0.27-to-1V read using swing-sample-and-couple sense amplifier and self-boost-write-termination scheme," in In 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), San Francisco. CA, USA, 19 January 2014 (IEEE), 332–333. doi:10.1109/ISSCC.2014.6757457

Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., et al. (2016). Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. *SIGARCH Comput. Archit. News* 44, 27–39. doi:10.1145/3007787.3001140

Dong, S., Wang, P., and Abbas, K. (2021). A survey on deep learning and its applications. *Comput. Sci. Rev.* 40, 100379. doi:10.1016/j.cosrev.2021.100379

Dubey, A., Cammarota, R., and Aysu, A. (2019). Maskednet: A pathway for secure inference against power side-channel attacks. *arXiv preprint arXiv:1910.13063.*

Goldreich, O., and Ostrovsky, R. (1996). Software protection and simulation on oblivious rams. *J. ACM (JACM)* 43, 431–473. doi:10.1145/233551.233553

Hu, X., Liang, L., Chen, X., Deng, L., Ji, Y., Ding, Y., et al. (2022). A systematic view of model leakage risks in deep neural network systems. *IEEE Trans. Comput.* 09, 3148235. doi:10.1109/tc.2022.3148235

Hua, W., Zhang, Z., and Suh, G. E. (2018). "Reverse engineering convolutional neural networks through side-channel information leaks," in 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco. CA, USA, 24-28 June 2018 (IEEE), 1–6.

Huang, S., Peng, X., Jiang, H., Luo, Y., and Yu, S. (2020). New security challenges on machine learning inference engine: Chip cloning and model reverse engineering. *arXiv:2003.09739 [eess]* ArXiv: 2003.09739

Károly, A. I., Galambos, P., Kuti, J., and Rudas, I. J. (2020). Deep learning in robotics: Survey on model structures and training strategies. *IEEE Trans. Syst. Man. Cybern. Syst.* 51, 266–279. doi:10.1109/tsmc.2020.3018325

Krishnan, G., Mandal, S. K., Chakrabarti, C., Seo, J.-S., Ogras, U. Y., and Cao, Y. (2021). Impact of on-chip interconnect on in-memory acceleration of deep neural networks. *ACM J. Emerg. Technol. Comput. Syst.* 18, 1–22. doi:10.1145/3460233

Li, B., Gu, P., Shan, Y., Wang, Y., Chen, Y., and Yang, H. (2015). Rram-based analog approximate computing. *IEEE Trans. Comput. -Aided. Des. Integr. Circuits Syst.* 34, 1905–1917. doi:10.1109/tcad.2015.2445741

Li, W., Huang, S., Sun, X., Jiang, H., and Yu, S. (2021). "Secure-rram: A 40nm 16kb compute-in-memory macro with reconfigurability, sparsity control, and embedded security," in 2021 IEEE Custom Integrated Circuits Conference (CICC), Austin, TX, USA, 25-30 April 2021 (IEEE), 1–2.

Li, W., Wang, Y., Li, H., and Li, X. (2019). "P3M: A PIM-based neural network model protection scheme for deep learning accelerator," in Proceedings of the 24th Asia and South Pacific Design Automation Conference, 19-21 April 2019 (Tokyo Japan: ACM), 633–638. doi:10.1145/3287624.3287695

Lin, N., Chen, X., Lu, H., and Li, X. (2020). Chaotic weights: A novel approach to protect intellectual property of deep neural networks. *IEEE Trans. Comput. -Aided. Des. Integr. Circuits Syst.* 40, 1327–1339. doi:10.1109/tcad.2020.3018403

Long, Y., Kim, D., Lee, E., Saha, P., Mudassar, B. A., She, X., et al. (2019). A ferroelectric fet-based processing-in-memory architecture for dnn acceleration. *IEEE J. Explor. Solid-State Comput. Devices Circuits* 5, 113–122. doi:10.1109/jxcdc.2019.2923745

Lyu, B., Hamdi, M., Yang, Y., Cao, Y., Yan, Z., Li, K., et al. (2022). Efficient spectral graph convolutional network deployment on memristive crossbars. *IEEE Trans. Emerg. Top. Comput. Intell.* 5, 1–11. doi:10.1109/tetci.2022.3210998

Marković, D., Mizrahi, A., Querlioz, D., and Grollier, J. (2020). Physics for neuromorphic computing. *Nat. Rev. Phys.* 2, 499–510. doi:10.1038/s42254-020-0208-2

Mythic (2022a). *M1076 analog matrix processor product brief.* Austin, TX: Mythic.

Mythic (2022b). *MM1076/ME1076 M.2 accelerator card product brief.* Austin, TX: Mythic.

Mythic (2022c). *MP10304 quad-AMP PCIe card product brief.* Austin, TX: Mythic.

Oseni, A., Moustafa, N., Janicke, H., Liu, P., Tari, Z., and Vasilakos, A. (2021). Security and privacy for artificial intelligence: Opportunities and challenges. *arXiv preprint arXiv:2102.04661*

Prakash, A., Park, J., Song, J., Woo, J., Cha, E.-J., and Hwang, H. (2014). Demonstration of Low Power 3-bit Multilevel Cell Characteristics in a TaO<sub><italic>x</italic></sub>-Based RRAM by Stack Engineering. *IEEE Electron Device Lett.* 36, 32–34. doi:10.1109/led.2014.2375200

Qiao, X., Cao, X., Yang, H., Song, L., and Li, H. (2018). "AtomLayer: A universal ReRAM-based CNN accelerator with atomic layer computation," in 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco. CA, USA, 24-28 June 2018 (IEEE), 1–6. doi:10.1109/DAC.2018.8465832

Rajasekharan, D., Rangarajan, N., Patnaik, S., Sinanoglu, O., and Chauhan, Y. S. (2021). SCANet: Securing the weights with superparamagnetic-MTJ crossbar array networks. *IEEE Trans. Neural Netw. Learn. Syst.* 21, 1–15. doi:10.1109/TNNLS.2021.3130884

Rathi, N., Chakraborty, I., Kosta, A., Sengupta, A., Ankit, A., Panda, P., et al. (2022). Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware. *ACM Comput. Surv.* 11, 3571155. doi:10.1145/3571155

Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., et al. (2016). Isaac: A convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars. *SIGARCH Comput. Archit. News* 44, 14–26. doi:10.1145/3007787.3001139

Sharir, O., Peleg, B., and Shoham, Y. (2020). The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900.*

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243.*

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. (2016). "Stealing machine learning models via prediction APIs," in 25th USENIX Security Symposium (USENIX Security 16), 16-18 July 2016 (Austin, TX: USENIX Association), 601–618.

Wan, W., Kubendran, R., Schaefer, C., Eryilmaz, S. B., Zhang, W., Wu, D., et al. (2022). A compute-in-memory chip based on resistive random-access memory. *Nature* 608, 504–512. doi:10.1038/s41586-022-04992-8

Wang, C., Feng, D., Tong, W., Liu, J., Li, Z., Chang, J., et al. (2019). Cross-point resistive memory: Nonideal properties and solutions. *ACM Trans. Des. Autom. Electron. Syst.* 24, 1–37. doi:10.1145/3325067

Wang, Y., Jin, S., and Li, T. (2021). "A low cost weight obfuscation scheme for security enhancement of ReRAM based neural network accelerators," in Proceedings of the 26th Asia and South Pacific Design Automation Conference, 19-21 April 2019 (Tokyo Japan: ACM), 499–504. doi:10.1145/3394885.3431599

Wen, S., Chen, J., Wu, Y., Yan, Z., Cao, Y., Yang, Y., et al. (2020). Ckfo: Convolution kernel first operated algorithm with applications in memristor-based convolutional neural network. *IEEE Trans. Comput. -Aided. Des. Integr. Circuits Syst.* 40, 1640–1647. doi:10.1109/tcad.2020.3019993

Wen, S., Wei, H., Yan, Z., Guo, Z., Yang, Y., Huang, T., et al. (2019). Memristor-based design of sparse compact convolutional neural network. *IEEE Trans. Netw. Sci. Eng.* 7, 1431–1440. doi:10.1109/tnse.2019.2934357

Wu, W., Wu, H., Gao, B., Yao, P., Zhang, X., Peng, X., et al. (2018). "A methodology to improve linearity of analog rram for neuromorphic computing," in 2018 IEEE Symposium on VLSI Technology, Honolulu, HI, USA, 18-22 June 2018 (IEEE), 103–104.

Xue, C.-X., Huang, T.-Y., Liu, J.-S., Chang, T.-W., Kao, H.-Y., Wang, J.-H., et al. (2020). "15.4 a 22nm 2mb reram compute-in-memory macro with 121-28tops/w for multibit mac computing for tiny ai edge devices," in 2020 IEEE International

Solid-State Circuits Conference-(ISSCC), San Francisco. CA, USA, 16-20 Feburary 2020 (IEEE), 244–246.

Yan, M., Fletcher, C. W., and Torrellas, J. (2020). "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures," in 29th USENIX Security Symposium (USENIX Security 20), 16-18 July 2016 (Austin, TX: USENIX Association), 2003–2020.

Yang, C., Liu, B., Li, H., Chen, Y., Barnell, M., Wu, Q., et al. (2020). Thwarting replication attack against memristor-based neuromorphic computing system. *IEEE Trans. Comput. -Aided. Des. Integr. Circuits Syst.* 39, 2192–2205. doi:10.1109/TCAD.2019.2937817

Yao, P., Wu, H., Gao, B., Eryilmaz, S. B., Huang, X., Zhang, W., et al. (2017). Face classification using electronic synapses. *Nat. Commun.* 8, 15199. doi:10.1038/ncomms15199

Young, V., Nair, P. J., and Qureshi, M. K. (2015). Deuce: Write-efficient encryption for non-volatile memories. *SIGPLAN Not.* 43, 33–44. doi:10.1145/2775054.2694387

Zhu, Z., Sun, H., Lin, Y., Dai, G., Xia, L., Han, S., et al. (2019). "A configurable multi-precision cnn computing framework based on single bit rram," in 2019 56th

ACM/IEEE Design Automation Conference (DAC), San Francisco. CA, USA, 24-28 June 2018 (IEEE), 1–6.

Zhu, Z., Sun, H., Qiu, K., Xia, L., Krishnan, G., Dai, G., et al. (2020). "Mnsim 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," in Proceedings of the 2020 on Great Lakes Symposium on VLSI, Nicosia, Cyprus, 04-06 July 2020 (IEEE), 83–88.

Zou, M., Zhou, J., Cui, X., Wang, W., and Kvatinsky, S. (2022). "Enhancing security of memristor computing system through secure weight mapping," in 2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Nicosia, Cyprus, 04-06 June 2022 (IEEE), 182–187.

Zou, M., Zhu, Z., Cai, Y., Zhou, J., Wang, C., and Wang, Y. (2020). "Security enhancement for RRAM computing system through obfuscating crossbar row connections," in 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 09-13 March 2020 (IEEE), 466–471. doi:10.23919/DATE48585.2020.9116549

Zuo, P., Hua, Y., and Xie, Y. (2019). "Supermem: Enabling application-transparent secure persistent memory with low overheads," in Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, 09-12 October 2019 (Tokyo Japan: ACM), 479–492.