CiMSAT: Exploiting SAT Analysis to Attack Compute-in-Memory Architecture Defenses

Jianfeng Wang
Huazhong Yang
Shuwen Deng*
Xueqing Li*
Electronic Engineering, Tsinghua University
Beijing, China
wjf22@mails.tsinghua.edu.cn,{yanghz,shuwend,xueqingli}@tsinghua.edu.cn

Abstract

Compute-in-memory (CiM) architecture is an emerging energy-efficient processing paradigm that has attracted widespread attention in AI and Internet of Things (IoT) applications. To protect statically stored sensitive data in CiM, designers have implemented various hardware obfuscation techniques in CiM architectures. However, we observe that existing CiM obfuscation defense strategies are based on straightforward static-key deployment strategies, which pose vulnerabilities from the perspective of key-pruning algorithms for de-obfuscation.

This work proposes CiMSAT, a CiM de-obfuscation methodology based on Boolean satisfiability (SAT) theory. We conduct the first security analysis specifically tailored to the storage and mixed-signal computing features of CiM architecture, which are two key challenges to de-obfuscate existing state-of-the-art CiM defenses. To model storage units, we innovatively fit and utilize the "no-inference-value" obfuscated data for function approximation. To reconstruct mixed-signal circuits, we design bias-tolerant SAT to address the biases introduced by the approximation. With the proposed workflow, we investigate and evaluate all the existing 14 CiM obfuscation architectures using our de-obfuscation framework. We model a total of 176 defense vectors derived from different defense techniques and parameters, among which 158 (90%) can be de-obfuscated and returned the keys within 1,000 seconds and 172 (98%) defenses can be recovered within 10^5 seconds (approximately one day). We further reload the keys into CiM simulators with obfuscation, achieving an average of 97% and 95% accuracy recovery in widely adopted MNIST and CIFAR-10 classification applications in CiM obfuscation, respectively.

CCS Concepts

• **Security and privacy → Security in hardware**; **Hardware attacks and countermeasures**.

Keywords

Compute-in-Memory, Hardware Obfuscation, SAT

ACM Reference Format:
Jianfeng Wang, Huazhong Yang, Shuwen Deng*, and Xueqing Li*. 2024. CiMSAT: Exploiting SAT Analysis to Attack Compute-in-Memory Architecture Defenses. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690251

1 Introduction

With the rapid development of AI and IoT, data-intensive applications such as computer vision (CV) and natural language processing (NLP) have created a demand for energy-efficient processing. This has made the Compute-in-Memory (CiM) paradigm a widely researched and pursued direction currently [27, 49, 63]. By integrating computing into storage, CiM architecture effectively reduces the significant overhead caused by the frequent data movement between on-chip processing units and off-chip memory, lowering the energy consumption of data transfer by two orders of magnitude [51]. Consequently, it emerges as a viable technology for overcoming the storage wall bottleneck in traditional von Neumann architectures.

Although CiM architectures achieve energy-efficient processing of large-scale data, on-chip storage also brings security issues, introducing non-negligible CiM attacks and related defenses. Existing weight extraction attacks [15, 22, 28, 42, 57] pose threats to the sensitive data in CiM architecture, particularly those with high inference capabilities. CiM designers employ various protection methods, primarily hardware obfuscation strategies, to safeguard data confidentiality. Existing typical obfuscation structures can be roughly divided into three categories: multiplexers-based (MUX) obfuscation [3, 19, 55, 64, 66], XOR/XNOR logic-based obfuscation [7, 24, 35–37, 65], and network-based obfuscation [56, 67, 68]. By incorporating these structures into the CiM architecture and controlling them with keys, the protected weight data can be computed in obfuscated form, thus mitigating data piracy.

Nevertheless, we have identified a crucial weakness in existing CiM obfuscation protection mechanisms that cannot be easily fixed using existing methods. The major observation is that *the security of the existing CiM obfuscation originates from explicit key-related hardware insertion.* While increasing the key length can make it more challenging for adversaries to determine the correct key vectors, statically inserted keys, without high-overhead iterative encryption, are easily detectable and exploitable. This provides an opportunity

*Xueqing Li and Shuwen Deng are co-corresponding authors.

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690251

for the elimination of erroneous key possibilities. We also notice that existing de-obfuscation algorithms cannot easily break CiM obfuscation protection mechanisms. Studies [46, 48] have demonstrated rapid key pruning for combinational obfuscated logic circuits. However, *unique storage feature* and *analog computing feature* in CiM hardware make the combinational obfuscation methods unsuitable for CiM obfuscation architectures. These two CiM-related features make solving CiM de-obfuscation attacks non-trivial.

Based on the above observations, this paper proposes CiMSAT, a CiM de-obfuscation attack framework, aimed at breaking existing CiM obfuscation defenses. The de-obfuscation method for CiM is based on the SAT problem analysis, a Boolean formula satisfiable decision process in computer science. To accommodate the two key challenges of attack state-of-the-art CiM obfuscation defenses, which are the *storage property and analog/mixed-signal computing feature of CiM architectures*, we conduct a two-step de-obfuscation attack: function approximation reconstruction and bias-tolerant SAT analysis. In the first step, we reconstruct the digitized obfuscated architecture using the obfuscated weights and retain the key-controlled ports as the key vectors to be solved. In the second step, we propose two bias-tolerant SAT algorithms tailored for analog/mixed-signal CiM obfuscation to recover keys and accuracy in CiM-related applications.

From the evaluation results, CiMSAT successfully attacks the CiM obfuscation mechanism and nearly fully recovers the performance of CiM circuits. We evaluate the security of 14 state-of-the-art of CiM obfuscation architectures, and CiMSAT is capable of attacking all of them. Based on various defense techniques, features and parameters, we set a total of 176 defense vectors, among which 158 (90%) can be attacked and solved within 1,000 seconds to obtain the key. Furthermore, 172 (98%) defense vectors can be de-obfuscated within 10^5 seconds (approximately 1 day). We have subsequently reloaded the solved keys into CiM simulators with obfuscation, achieving average of 97% and 95% inference accuracy recovery for widely-adopted MNIST and CIFAR-10 classification applications in CiM obfuscation, respectively. We also summarize 9 sub-conclusions for different features and parameters, providing insights for future CiM security design.

In summary, we list our main contributions as follows:

- Systematically analyse existing CiM attack and defense scenarios, and discover the weakness in existing CiM obfuscation defense.
- Propose attack framework CiMSAT to break the current mainstream CiM defenses with a two-step de-obfuscation attack flow.
  - Overcome de-obfuscation difficulty on storage components in addition to computing circuits.
  - Implement attack framework to recover analog-domain CiM (take up 70% of all CiM circuits) for the first time, along with digital CiM.
- Successfully attack 14 state-of-the-art CiM defenses, recover sensitive obfuscated weight data and execute efficient key pruning for all of 176 different defense vectors in total, of which 158 (90%) can be attacked and solved within 1,000 seconds, and further recover averagely more than 95% accuracy on MNIST and CIFAR-10 applications.

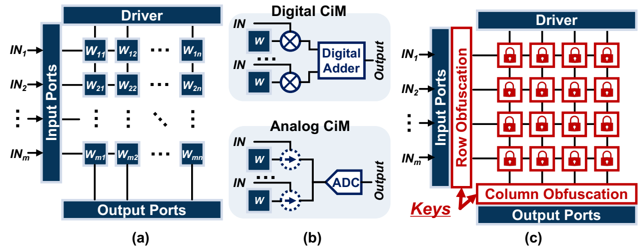


Figure 1: CiM hardware structure diagram. (a) Schematic of sub-array in CiM architecture, merging storage and computing in the crossbar. (b) Digital and analog CiM diagram. (c) Schematic of CiM obfuscation: key-controlled obfuscation and obfuscate weights.

2 Background and Related Works

This section sequentially introduces the background of CiM architecture, attacks and defenses of CiM, and SAT-related knowledge.

2.1 CiM Architecture

CiM architecture [1, 27, 49, 63], also known as IMC (in-memory computing) or PIM (process-in-memory), is a novel paradigm that differs from traditional von Neumann architecture. The most significant characteristic is the fusion of data storage and computing. As shown in Figure 1(a), CiM integrates data storage and computing through on-chip memory sub-array with crossbar nature. By storing a large amount of weight data in crossbars, CiM greatly reduces the energy overhead of data on/off-chip movements.

The crossbar array in CiM architecture can accelerate deep learning algorithms by supporting the primary operation of Vector-Matrix Multiplication (VMM). For instance, in Figure 1(a), the weights (W_{ij}) of a model can be mapped into the crossbar array of the CiM architecture, while input vectors (IN_i) are input through the row ports in parallel. The corresponding components (W_{ij} and IN_i) are multiplied within the same row i and accumulation within the same column j, i.e., $output_j = \sum_{i=1}^{m} IN_i \cdot W_{ij}$, with the computing results being output from the output ports.

Although both the inputs and outputs of the crossbar array are digital values, the intermediate computing processes can be divided into digital domain and analog/mixed-signal domain computing (hereafter referred to as digital and analog CiM). As illustrated in Figure 1(b), for digital CiM, architecture employs digital adders combined with storage arrays to perform VMM. For analog CiM, the architecture utilizes the accumulation of analog signals, such as current, in the column dimension to replace digital domain adders. The analog results are then quantified through an Analog-to-Digital Converter (ADC) to obtain digital results.

2.2 CiM Security: Attack and Defense

The security of CiM has yet to be widely formalized, to the best of authors' knowledge. Side-Channel Attacks (SCA) are mainly utilized to obtain sensitive data in CiM systems, enabling the analysis of the timing and power consumption of CiM operations, thereby threatening the confidentiality of the model [44, 45, 58]. In addition to data theft, Ensan et al. [15] achieved efficient reverse engineering using SCA. For CiM integrity, there is very limited attacks against the CIM architecture. Staudigl et al. [47] conducted a rowhammer attack on RRAM CiM, and Huang et al. [26] tailored Trojan designs for CiM computing patterns to implement covert attacks.

To mitigate the loss caused by attacking the confidentiality of CiM data, the most mainstream defense is data obfuscation. As shown in Figure 1(c), CiM obfuscation design mainly consists of two stages. Defenders first add key-controlled obfuscation logic to control the row or column, and then store ciphertext-form weight into the crossbar related to the keys. Once the data are obfuscated into ciphertext, they lose the application value unless the correct keys is adopted. There are studies [3, 7, 19, 24, 35–37, 55, 56, 64–68] that have made different efforts in various design metrics such as obfuscation techniques, overhead, and performance. In addition to obfuscation, system designers employ other defense techniques such as introducing disturbances like noise [8, 9], fine-tuning weights [25], or designing novel circuit structures [52]. However, the design complexity of these methods are much higher [43] and have not been widely discussed.

2.3 SAT Problem and SAT Attack

In logic and computer science, the Boolean satisfiability problem (SAT problem) is the problem of determining if there exists an interpretation that satisfies a given Boolean formula [59]. From the logic circuit perspective, a formula, built from logic variables and logic operators (AND, OR, NOT, etc.), is satisfiable if it can be made TRUE by assigning a set of logical values (i.e., TRUE, FALSE) to the variables. For example, the expression “A AND B” is satisfiable because when the SAT solver solves "A AND B = True", it can find a set of satisfying solutions: A=B=True.

Hardware security researchers have applied SAT-solving techniques to perform SAT attacks on obfuscated combination logic circuits, achieving significant success in efficient de-obfuscation [48]. Using SAT solving process can iteratively and rapidly eliminate incorrect keys, thereby avoiding inefficient brutal static key attempts. However, current SAT attacks primarily focus on digital logic circuits [31], leaving a research gap regarding emerging architectures such as CiM.

3 Understand the Challenges and Insights of CiM Confidentiality

In this section, we provide a systematic analysis of CiM security and position the widespread use of the CiM obfuscation protection mechanism. We then show that existing CiM obfuscation methods to protect the confidentiality of CiM weights are under serious threat due to the adoption of static key deployment strategies. We also provide insights of our attacks to actually break existing CiM obfuscation defenses.

3.1 Why is CiM important in both academia and industry?

The rapid growth of data volume in AI applications has made data processing and transfer increasingly energy-intensive [51]. CiM has served as an energy-efficient hardware solution, which can largely alleviate the storage wall bottleneck in traditional von Neumann

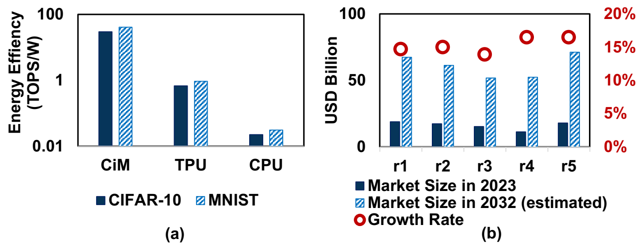


Figure 2: CiM performance and developing trend. (a) Energy efficiency simulation of CiM, TPU (estimated from [63]), and CPU (estimated from [30]). (b) Market size and growth rate of the CiM market reported by five market research reports. Source: r1(Imarcgroup), r2(MarketResearchGuru), r3(MarketReportsWorld), r4(VerifiedMarketResearch), r5(MarketsandMarkets).

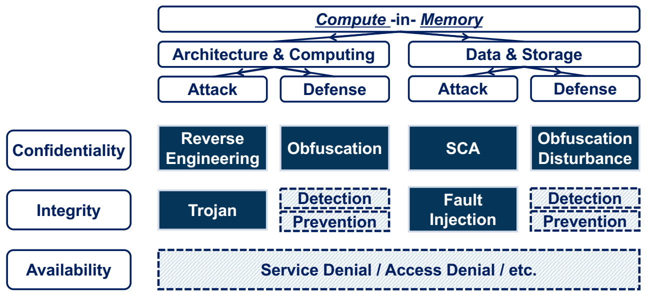


Figure 3: Systematic overview of CiM security. (Dark: CiM-related works. Dashed Blue: relevant techniques that can be applied to CiM in the future.)

architecture [6]. We used the neuromorphic simulator [5] to evaluate the comparison of energy consumption of CiM, TPU and CPU with the data support in [30, 63]. As shown in Figure 2(a), under the image classification of CIFAR-10 [32] and MNIST [11], CiM can exceed 30.38 TOPS/W of energy efficiency, which is significantly higher than that of TPU and CPU in [30, 63].

In the industry, in-memory computing is forecasted to achieve an annual growth rate of over 10%, with the market size expected to exceed 50 billion by 2032 (estimated by 5 reports) as depicted in Figure 2(b). On the product front, in 2021, Samsung launched HBM-PIM to enhance data efficiency in AI accelerator systems [14]. In 2022, SK Hynix developed GDDR6-AiM and a Computational Memory Solution (CMS) based on CXL [38].

3.2 Why is CiM security a critical concern?

With the growing trend of CiM development, understanding and guaranteeing the security of CiM is crucial for safeguarding data privacy and maintaining system stability. To the best of our knowledge, there has been limited systematic analysis of the attack and defense strategies specifically tailored for CiM.

Regarding this, we provide a novel abstraction and systematic analysis on CiM security, as shown in Figure 3. Our systematic review not only encompasses all attacks and defenses related to

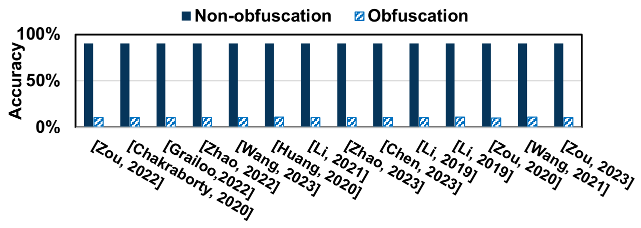


Figure 4: The accuracy of existing CiM obfuscation techniques on the CIFAR-10 [32] image classification task. The classification accuracy of the obfuscated CiM model drop to 10% compared to the non-obfuscated model.

CiM (dark boxes), but also incorporates potential hardware attacks and defenses [12, 39, 60] that could be deployed on CiM architectures (dashed blue boxes). From the security perspective, we use three cornerstone security properties [33] to categorize the CiM security: confidentiality, integrity, and availability. From the CiM architecture perspective, computing architecture and data storage are two crucial aspects as the name "CiM" suggests. Therefore, we summarize and analyse attacks and defenses based on these two dimensions.

At present, CiM confidentiality is currently the most extensively studied dimension, with both attacks and defenses being intensively studied on architecture and data. Works [15] have exploited reverse-engineering to recover CiM structure and [44, 45, 58] use SCA to threat data confidentiality. [3, 7–9, 19, 24, 25, 35–37, 55, 56, 64–68] have proposed various of defenses with obfuscation or disturbance. As for integrity, only a few works have discussed attack, yet we believe related detection and prevention defense is not a new research direction although it has not appeared in CiM related papers. As for availability, we have not found any literature related to CiM at present, but this is a direction worth exploring in the future.

3.3 Why is CiM obfuscation an effective defense mechanism for confidentiality?

From the systematic analysis presented in Section 3.2, we observe that CiM obfuscation has become a mainstream defense currently. To protect the confidentiality of on-chip data, obfuscation changes the data form from plaintext to ciphertext while enabling the system to perform ciphertext computing with the correct keys. The ciphertext form of weights significantly reduces comprehensibility. Furthermore, even if the data is leaked, obfuscated data holds no practical value without the keys.

We conduct image classification evaluations on all 14 state-of-the-art CiM obfuscation techniques, as listed in Table 1, using the extracted ciphertext data from their architectures. As shown in Figure 4, all 14 obfuscation techniques result in a significant reduction in inference accuracy on the CIFAR-10 [32] image classification dataset from 90% to 10% with VGG8 model. Therefore, obfuscated data fundamentally renders the extracted weights useless, without introducing significant overheads to the computing system [68]. Consequently, it has emerged as the predominant method for safeguarding confidentiality.

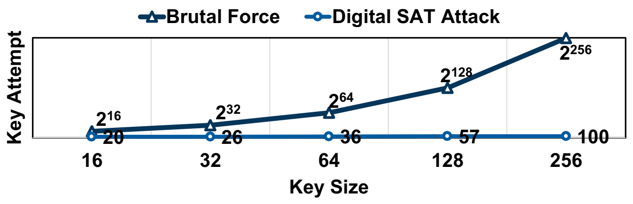


Figure 5: The attempts required for brute-force attack and SAT attack [48] to de-obfuscate the pure combination logic circuit. Benchmark: c1355 from ISCAS85[21].

3.4 Why does CiM obfuscation still exist vulnerability?

We analyzed the aforementioned 14 CiM obfuscation techniques and found that they share similar structural features. All 14 techniques utilize additional circuit modules (such as MUX or XOR) to add static keys to control the circuit behavior. They leverage the exponentially key space to defend against key attempts. However, these keys are not only static but also do not involve any iterative encryption techniques.

On the other hand, we know that static keys inserted in pure digital netlists are under the threat. For instance, the SAT-based de-obfuscation (SAT attack) proposed in [46, 48] enables rapid examination of static keys in digital netlists. As shown in Figure 5, we evaluate and compare between the attempts required for brute-force and SAT attacks to de-obfuscate a combinational logic obfuscated circuit. As the key length increases, the number of key attempts in brute-force attacks grows exponentially. However, SAT attacks leverage SAT solvers to rapidly prune the exponential key space and accomplish the search of the key space in polynomial complexity.

The significance of SAT attack lies in the fact that while solving SAT problems is typically NP-complete [13], those derived from boolean circuits can be efficiently solved by modern SAT solvers [29]. In other words, while defenders attempt to obfuscate boolean circuits using exponential key spaces, attackers can solve keys in polynomial time under the SAT attack model. Therefore, this inspires us to explore a novel de-obfuscation attack tailored specifically for CiM obfuscation.

3.5 How to attack CiM obfuscation: challenges and insights

CiM has unique architectural characteristics and computational features, making the straightforward application of existing SAT attacks to CiM obfuscation unfeasible. In this section, we analyse and summarize two major challenges to use SAT idea on to do CiM de-obfuscation attack, along with our insights.

Firstly, the storage characteristics of CiM architecture affect the extraction of function netlists, thereby challenging the SAT solving process. Traditional SAT attacks decompose circuits through scan chains, ultimately modeling and solving for combinational logic circuits, which do not contain storage units. In other words, the information in the netlist can represent the whole circuit functionality. However, CiM architecture is the fusion of storage and computing, meaning that the weights stored in crossbar also determine circuit

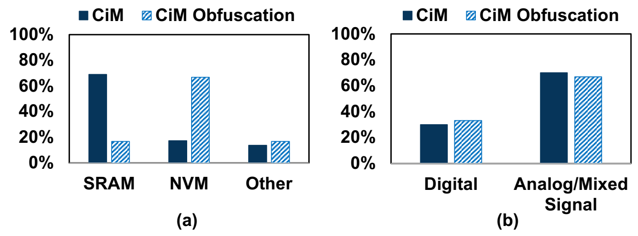


Figure 6: CiM and CiM obfuscation architecture research statistics [49]. (a) Proportion of cell devices including SRAM, NVM, and other types of memory devices. (b) Proportion of digital domain computing and analog/mixed signal domain computing.

functionality, rendering simple netlist extraction meaningless [19]. Additionally, as shown in Figure 6(a), among existing CiM obfuscation architectures, 60% of the work is based on non-volatile memory (NVM) with intrinsic data storage and non-CMOS characteristics, which further poses challenges for netlist modeling.

Our first critical observation is that even though the obfuscated weights in the CiM obfuscation architecture hold no application value, they can still be fitted and translated into a digital netlist form thereby converting storage elements that SAT attacks cannot handle into digital logic. While the circuit function is still determined by the key, the reconstructed digital netlist can be further subjected to SAT-based attacks without concerning storage units. Based on this observation, we first use regression to fit the obfuscated weights, and then reconstruct the pure obfuscated digital circuit. Therefore, we eliminate the storage component and obtain a pure logic netlist for further de-obfuscation.

Challenge: Storage feature cannot be directly modeled as a netlist usable for SAT attack.
Insight: The "no-inference-value" obfuscated weights can be used to model the CiM storage behavior with low bias.

On the other hand, the analog domain computing feature affects the solving process of SAT solvers. Figure 6(b) demonstrates that both CiM and CiM obfuscation architectures have digital domain computing accounting for only about 30%, while analog/mixed-signal computing architectures account for close to 70%. Analog domain computing results need to be eventually quantized into digital results, inevitably introducing quantization noise. However, existing SAT solvers are designed only for digital circuit designs, so the solving process involves precise clause judgements. Therefore, the noise may cause this precise decision to stall or result in no solution.

Based on that, our second critical observation is that while quantization noise introduces biases, we can adjust the SAT solver's decision criteria based on the distribution of these biases. We incorporate the biases into the SAT-solving process, rendering the entire attack be bias-tolerant. We found that the majority of biases are concentrated at lower levels, thus will not bring a large overhead to the SAT solving process in clause decision criteria. This not only reduces design complexity but also has a negligible impact on solving time. Although this bias-tolerant SAT algorithm may lead to imprecise solutions, our application-oriented evaluation suggests that the overhead introduced by this approximation is acceptable.

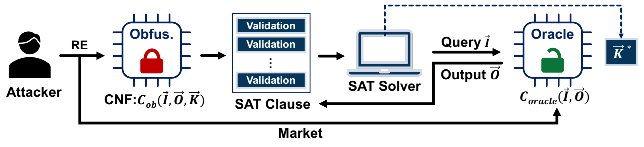


Figure 7: Oracle-guided threat model.

Challenge: Analog computing feature with quantization biases cannot be processed by SAT attack.
Insight: Adding tolerance to quantization biases can contribute to attack analog CiM.

Drawing from the insights above, we present our de-obfuscation attack strategy CiMSAT for CiM obfuscation architecture in Section 4.

4 CiMSAT Attack framework

In this section, we will discuss the threat model and the detailed attack methodology.

4.1 Threat Model and Overview

4.1.1 Threat Model. In our attack, as is shown in Figure 7, attacker is able to obtain the obfuscated netlist of the target circuit by reverse-engineering (RE). From the obfuscated netlist, attackers cannot know the keys but can acquire the $\vec{I}/\vec{O}$ pairs with arbitrary key attempts. In order to using SAT solvers for attack, the functionality of the obfuscated netlist is represented as a Conjunctive Normal Form (CNF). Attacker is also assumed to have access to an unlocked functional circuit f_{oracle} (without knowing keys) as an oracle, which can be obtained from the market in reality [17, 31]. This aligns with oracle-guided attack model in logic obfuscation [61]. The golden input ($\vec{I}$)-output ($\vec{O}$) pair can be obtained through querying the oracle through the testing structure of CiM [4, 16, 18, 20, 34, 40, 53, 54]. The attack use the SAT clause and SAT solver to analyse the keys, with the constraints iteratively added from the oracle.

The ultimate goal of the attacker is the key, which means that they need to find at least one key vector $\vec{K^*}$ that satisfies:

$$C_{ob}(\vec{I},\vec{O},\vec{K^*}) = C_{oracle}(\vec{I},\vec{O}), \forall \vec{I},\vec{O} \quad (1)$$

4.1.2 CiMSAT Overview. As outlined in Figure 8, our detailed de-obfuscation attack process contains two steps, functional approximation reconstruction (Figure 8(a)) and bias-tolerant SAT (btSAT) key recovery process (Figure 8(b)), which corresponds to the two challenges and insights discussed in Section 3.5. In the first step, we perform weight fitting on the crossbar hardware (Figure 8(a)(1)), and use the approximated weights to represent the functionality of the CiM obfuscation architecture (Figure 8(a)(2)), thereby reconstructing a pure-digital netlist format suitable for SAT solving (Figure 8(a)(3)). At this stage, the netlist is obfuscated since the key is still the target secret. In the second step, we address the bias

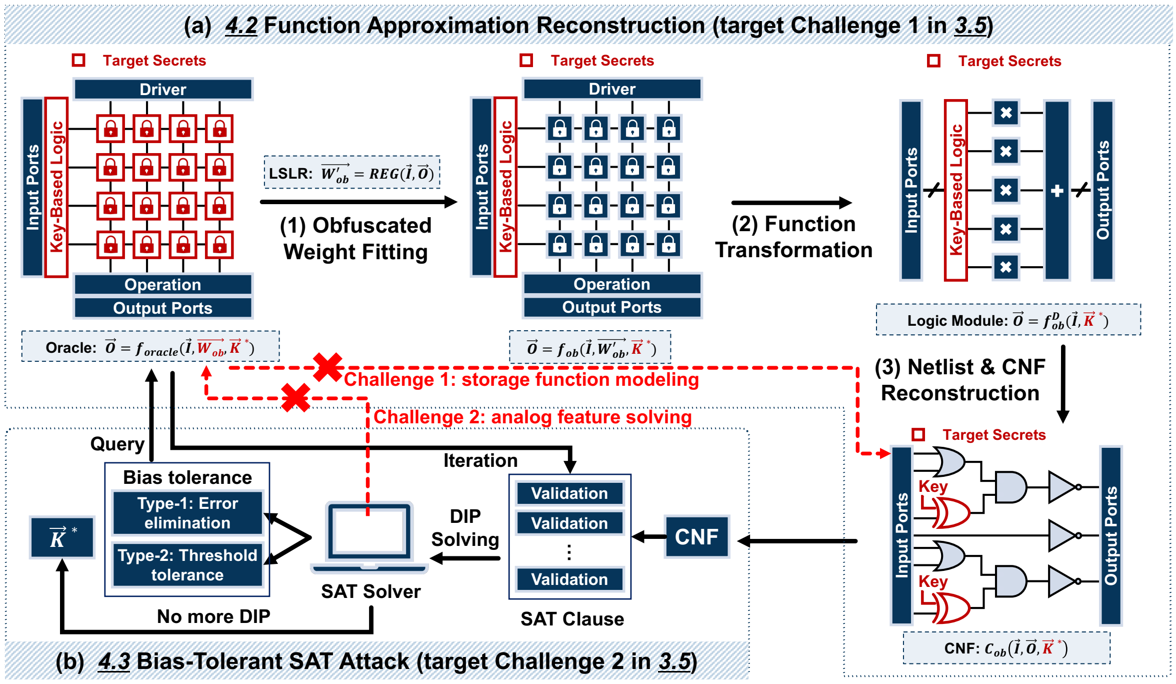

(a) 4.2 Function Approximation Reconstruction (target Challenge 1 in 3.5)
Target Secrets
Driver
Input Ports
Key-Based Logic
Operation
Output Ports
LSLR: $\overrightarrow{W'_{ob}} = REG(\vec{I},\vec{O})$
(1) Obfuscated Weight Fitting
(2) Function Transformation
Logic Module: $\vec{O} = f^D_{ob}(\vec{I},\vec{K}^*)$
Oracle: $\vec{O} = f_{oracle}(\vec{I},\overrightarrow{W_{ob}},\vec{K}^*)$
$\vec{O} = f_{ob}(\vec{I},\overrightarrow{W'_{ob}},\vec{K}^*)$
Challenge 1: storage function modeling
Challenge 2: analog feature solving
(3) Netlist & CNF Reconstruction
Query
Iteration
Bias tolerance
Type-1: Error elimination
Type-2: Threshold tolerance
$\vec{K}^*$
SAT Solver
DIP Solving
Validation
SAT Clause
CNF
No more DIP
Key
CNF: $C_{ob}(\vec{I},\vec{O},\vec{K}^*)$
(b) 4.3 Bias-Tolerant SAT Attack (target Challenge 2 in 3.5)

Figure 8: Overview of the CiMSAT (Dark red: target secrets for adversaries). (a) Function Approximation Reconstruction (4.2). (b) Bias-tolerant SAT attack (4.3).

generated in the previous reconstruction process and propose two bias tolerance methods to ultimately recover the target key vector $\overrightarrow{K^*}$ (Figure 8(b)

4.2 Function Approximation Reconstruction

This section introduces the reconstruction process from the CiM computing paradigm to digital computing paradigm, corresponding to Figure 8(a). The CiM obfuscation architecture employs keys and obfuscated weights $\overrightarrow{W_{ob}}$:

$$\vec{O} = f_{oracle}(\vec{I}, \overrightarrow{W_{ob}}, \overrightarrow{K^*}) \tag{2}$$

where f_{oracle} denotes the oracle function, $\vec{I}, \vec{O}$ represents the inputs and outputs, $\overrightarrow{W_{ob}}$ represent the obfuscated weights, and $\overrightarrow{K^*}$ refers to the key used for obfuscation. At this stage, both $\overrightarrow{W_{ob}}$ and $\overrightarrow{K^*}$ are target secrets to adversaries (shown in dark red). We first fit the obfuscated weights $\overrightarrow{W_{ob}}$ that do not possess any inference value but will help with the following reconstruction step.

4.2.1 Obfuscated Weight Fitting. The most important goal of this step is to approximate a set of obfuscated weights $\overrightarrow{W'_{ob}}$ that are consistent with $\overrightarrow{W_{ob}}$ functionality. As shown in Figure 8(a)(1), we adopt least squares linear regression (LSLR) with $\vec{I}/\vec{O}$ samples from the scan chain to get $\overrightarrow{W'_{ob}}$. We take controllable points as the crossbar input ports and observable points as the crossbar output ports. For analog/mixed domain CiM architecture, observable points are registers after the ADC quantization.

To perform LSLR on CiM array, we first model the MAC operations in crossbar:

$$\vec{O} \leftarrow \overrightarrow{W_{ob}}^T \cdot \vec{I} + \epsilon \tag{3}$$

where $\vec{O}$ is the target binary-form output vector obtained by registers (digital CiM) or ADCs (analog CiM), $\vec{I}$ is an $m \times 1$ input vector query from input ports, and $\overrightarrow{W_{ob}}$ is an $m \times 1$ vector representing weights) of the model. ϵ represents quantization noise, which exists while using ADCs for analog/mixed signal computing. We use a left arrow "$\leftarrow$" to indicate that this is a hardware-centric computational process. Please note that from hardware perspective, $\vec{O}$ is represented with n bits, and can be recognized as an n-dimensional output vector.

We adopted random sample input vectors $\vec{I}$ (10,000 in this work) and get the corresponding outputs $\vec{O}$, which served as the train set for the LSLR analysis. The goal of the LSLR is to find the optimal obfuscated weight ($\overrightarrow{W'_{ob}}$, in this case), that minimizes the sum of squared errors e between the predicted values $\hat{O}$ and the output value given by output registers O:

$$e = \min_{\overrightarrow{W'_{ob}}} \|O - \hat{O}\|_2^2 = \min_{\overrightarrow{W'_{ob}}} \|O - \overrightarrow{W'_{ob}}^T \cdot \vec{I}\|_2^2 \tag{4}$$

where $\|\cdot\|_2$ denotes the Euclidean norm.

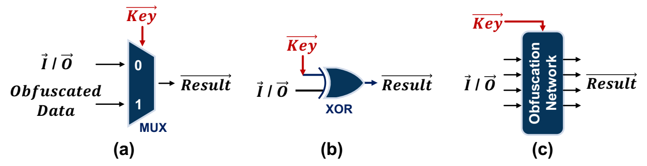


Figure 9: Key-related logic module summary. (a) Mux form (light overheads) [3, 19, 55, 64, 66]. (b) XOR form (light overheads) [7, 24, 35–37, 65]. (c) Obfuscation network form (heavy overheads) [56, 67, 68].

After extracting $\overrightarrow{W'_{ob}}$ by LSLR, as shown in the Figure 8(b), the target secrets are reduced to only the key $\overrightarrow{K^*}$:

$$\overrightarrow{O} = f_{ob}(\overrightarrow{I}, \overrightarrow{W'_{ob}}, \overrightarrow{K^*}) \tag{5}$$

Although the SAT solver still cannot model the CiM paradigm in Eq. (5) at this stage, we can transform it to a new pure-digital logic netlist with the function brought by $\overrightarrow{W'_{ob}}$, which does not contain any storage or non-digital logic. We will detail this in the next part.

4.2.2 Function Transformation. In this sub-section, we focus on the paradigm shifting from CiM to digital logic (Figure 8(b)(2)). Since we use LSLR to fit $\overrightarrow{W'_{ob}}$, Eq. (5) actually represents a completely pure digital system f_{ob}, which takes digital vectors $\overrightarrow{I}$ and $\overrightarrow{K^*}$(target) as input, undergoing operations modeled by Eq. (3) involving W'_{ob}, and finally producing digital vector $\overrightarrow{O}$ as output. In other words, if we consider the internal operations as a black box and only look at the inputs and outputs, this system perform a purely digital to digital function. Therefore, we adopt high-level hardware description language (HDL) to transform f_{ob} into digital logic module f^D_{ob}:

$$\overrightarrow{O} = f^D_{ob}(\overrightarrow{I}, \overrightarrow{K^*}) \tag{6}$$

where f^D_{ob} embeds the operations with $\overrightarrow{W'_{ob}}$, $\overrightarrow{I}$ and $\overrightarrow{O}$ are digital input and output.

For $\overrightarrow{K^*}$ in Eq. (6), we incorporate the knowledge from RE (align with Section 4.1) to transform key-related logic. As shown in Figure 9, we categorize the logic modules in current CiM obfuscation architectures into three main types, and we define N as the number of I/O ports controlled by each key in key-related logics. Among them, those based on MUX and XOR are more commonly adopted (Figure 9(a-b)) with relatively low hardware overheads, and typically $N = 1$. Modules based on obfuscation networks, e.g., a N-to-N Benes network [23] , shown in Figure (9)(c), provide obfuscation involved multiple input/output ports ($N > 1$), resulting in enhanced correlation between each port despite higher overheads.

We synthesized the HDL with added key logic and evaluated the 2-input logic gate number in a 64×64 crossbar scenario as an example. As shown in Figure 10, designs employing obfuscation networks exhibit significantly higher hardware overheads, while MUX and XOR works cost significant lower hardware resources.

4.2.3 Netlist & CNF Reconstruction. Finally, as shown in Figure 8(a)(3), we perform synthesis and reconstruct the digital netlist based on Eq.

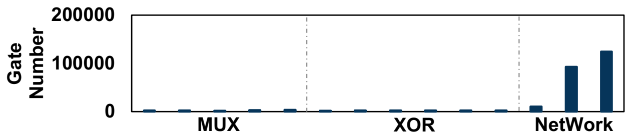


Figure 10: Gate number (unified to 2-input gate) of the reconstructed netlist (Condition: crossbar=64×64, Ref: same as Figure 9).

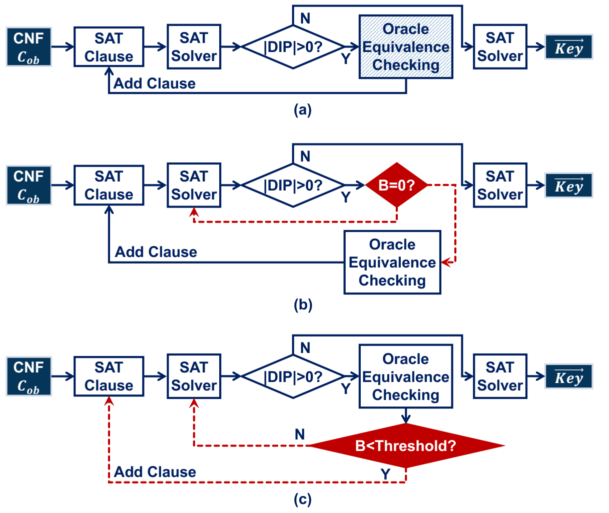


Figure 11: SAT attack procedure. (a) SAT attack for digital circuits. (b) Proposed type-1 btSAT for CiM. (c) Proposed type-2 btSAT for CiM.

(6). Since the netlist is now purely digital, it can be reconstructed into CNF formula that SAT solvers can resolve:

$$CNF : C_{ob}(\overrightarrow{I}, \overrightarrow{O}, \overrightarrow{K^*}) \tag{7}$$

where $\overrightarrow{K^*}$ is the target secret to be solved in the next section.

Back to Figure 8(a), we present functional approximation reconstruction through three subsections, primarily addressing the storage issue in Challenge 1 (Section 3.5). In the next section, we show how to tackle Challenge 2 and use the reconstructed CNF to recover the key $\overrightarrow{K^*}$.

4.3 Bias-Tolerant SAT Attack (btSAT)

In this section, we use C_{ob} in CNF formula to conduct bias-tolerant SAT attack to obtain the final key $\overrightarrow{K^*}$ (Figure 8(b)).

4.3.1 Traditional SAT Attack and Limitation. As shown in Figure 11(a), in the digital SAT attack, the C_{ob} can be transformed into an iteratively solvable SAT clause, and SAT solver solves distinguishable input patterns (DIPs) through SAT clauses. Each discovery of a DIP implies that the key space can be partitioned into two subsets: incorrect keys classified by the current DIP and oracle, and undecided keys. The DIP is inputted into an oracle for equivalence checking, obtaining a set of golden I/O pairs as a constraint

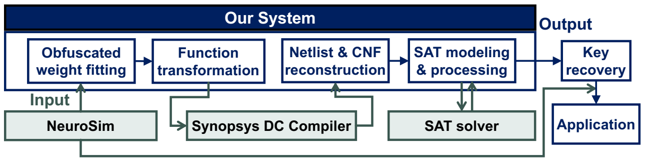


Figure 12: CiMSAT platform with NeuroSim [5], Synopsys DC compiler [50], and SAT solver [48].

clause, which are then iterated into SAT clauses. In this way, DIPs can efficiently prune the key space by identifying and removing a certain number of incorrect keys. When no DIP can be found by the SAT solver, it indicates that all key vectors in the current key space are the solution.

However, this may not work for CiM de-obfuscation. In the equivalence checking step (shaded square in Figure 11(a)) , oracle provides golden I/O pairs for SAT clause, in other words, the oracle provides precise and accurate constraints to C_{ob}. However, in Section 4.2, C_{ob} is an approximation reconstruction, thus there is a potential bias between the oracle and C_{ob}. Therefore, the precise constraint conditions from oracle may lead to unsolvability of C_{ob}.

4.3.2 Our Solution. We propose two bias-tolerant SAT (btSAT) key recovery methodologies, as is shown in Figure 8(b-c). We define a variable B as the output bias between the oracle and reconstructed C_{ob} under the same input $\overrightarrow{I}$. The following is a detailed introduction to two techniques.

Type-1 btSAT. To avoid $B \neq 0$ in the oracle equivalence checking, we add a judgement to select DIPs. As shown in Figure 11(b), we check the B produced by the solved DIP. If $B = 0$, this DIP does not cause conflicts with the CNF constraints; if $B \neq 0$, the DIP is discarded and SAT solver calculates a new DIP. The advantage of type-1 btSAT is that the final solved key ($\overrightarrow{K^*}$) does not introduce any bias. The drawback is the increased time cost of SAT solving, as many DIPs need to be discarded, even if they have the ability to prune incorrect keys. However, this time overhead can be acceptable, mainly because we find that for the solver proposed in work [48], the total time of CNF construction, SAT clause construction, and equivalence checking incurred a greater time proportion than the DIP solving time. Therefore, the time overhead introduced by type-1 btSAT is tolerable.

Type-2 btSAT. Type-2 approach is to add bias threshold during equivalence checking, allowing the SAT solver to approximately solve some clauses rather than precisely recognizing them. As shown in Figure 11(c), we set a threshold for judgement and adjust the criteria of the SAT solver correspondingly. If $B < Threshold$, we add this clause served as an approximate constraint condition. Consequently, the SAT solver will not fail when $B \neq 0$, but instead implements an approximate solution within a certain threshold. Approximating MAC operations (Section 4.2) provides benefits to perform approximate equivalence checks for the majority of DIPs while ensuring the threshold is small. The advantage of this method is that it reduces the attack time compared to type-1 btSAT. The drawback is that there may be an extra bias on the left side of Eq. (1) caused by the approximate constraint.

Algorithm 1 btSAT De-Obfuscation Algorithm

/* This algorithm is applicable when the oracle and the obfuscated netlist perform biased outputs. We present two types of btSAT and user can select either one. */
Input: C_{ob}, oracle, Threshold, {Type-1,Type-2} = {1,0} or {0,1}
Output: $\overrightarrow{K^*}$

```
1:  $i := 1$
2:  $F_i := C_{ob}(\overrightarrow{I}, \overrightarrow{Key_1}, \overrightarrow{O_1}) \land C_{ob}(\overrightarrow{I}, \overrightarrow{Key_2}, \overrightarrow{O_2})$
3:  while $sat[F_i \land (\overrightarrow{O_1} \neq \overrightarrow{O_2})]$ do
4:      $\overrightarrow{DIP_i} := sat\_solver_{\overrightarrow{DIP}}[F_i \land (\overrightarrow{O_1} \neq \overrightarrow{O_2})]$
5:      $\overrightarrow{O_i^{DIP}} := oracle(\overrightarrow{DIP_i})$
6:      $B := ||\overrightarrow{O_i^{DIP}} - f_{ob}^D(\overrightarrow{DIP})||$    ▹ Get the bias.
7:      if (Type-1 & $!B == 0$) then
8:          continue    ▹ Type-1: Skip and find another DIP.
9:      end if
10:     $F_{i+1} := F_i \land C_{ob}(\overrightarrow{DIP_i}, \overrightarrow{Key_{1,2}}, \overrightarrow{O_i^{DIP}})$
11:     if (Type-2 & $B < Threshold$) then
12:         $F_i^{Th} := \bigvee_d C_{ob}\left(\overrightarrow{DIP_i}, \overrightarrow{Key_{1,2}}, \overrightarrow{O_i^{DIP} + Threshold_d}\right)$
13:         $F_{i+1} := F_{i+1} \land F_i^{Th}$    ▹ Type-2: Add approximate CNF
14:     end if
15:     $i := i + 1$
16: end while
17: return $\overrightarrow{K^*} := sat\_solver_{Key_1}(F_i)$
```

However, this bias can also be acceptable, and our insight is as follows: We believe that CiM de-obfuscation differs from traditional digital circuit de-obfuscation. For digital logic, de-obfuscation must be precise, because attackers need to accurately recover the functionality, such as an ALU or multiplier module. In contrast, CiM is widely applied in energy-efficient AI applications at the edge, where the fault tolerance of deep learning shows advantageous. In other words, even if bias causes minor imprecision, it merely results in partial performance degradation rather than rendering the application entirely non-functional. Thus, within the realm of CiM applications, we view bias presence as an acceptable reality.

Algorithm. In Algorithm 1, we introduce the pseudo code implementation of type-1 or type-2 btSAT. The algorithm takes an obfuscated netlist as input, and our threat model assumes the oracle can provide the golden I/O pairs. We also require a threshold as a criterion for the type-2 btSAT, which can be pre-defined by checking the distribution of B. .

From line 2 to line 5, the algorithm first uses a SAT solver to find the current DIP under the condition of the SAT clause F_i, consistent with Subramanyan et al. [48]. $\overrightarrow{Key_{1,2}}$ are variables instead of certain specific key vectors. In line 6, we get the bias B between the oracle and the obfuscated netlist under the input of $\overrightarrow{DIP_i}$. From line 7 to line 9, we deploy the type-1 btSAT. If there is a bias $B! = 0$, we judge that this DIP is not suitable for iterative inclusion in the SAT clause. If there is no bias, the DIP can be included in the SAT clause through line 10. From line 11 to line 14, we deploy the type-2 btSAT. When the bias is less than the given threshold, we include a set of CNF expressions with biased outputs, denoted as F_i^{Th}, in the existing SAT clause. Since line 12 is a "or" expression, with an appropriate threshold selection, the probability of unsolvability of the SAT clause can be reduced to an acceptable degree. Finally, when all DIPs are found, it returns the recovered key vector $\overrightarrow{K^*}$.

5 Evaluation

In this section, we evaluate the effectiveness of CiMSAT attack framework proposed in and perform experiment on different parameter settings. We further show that in widely adopted neural-network applications such as MNIST [11] and CIFAR-10 [32] image classifications, CiMSAT can effectively recover the obfuscated keys and reconstruct averagely 97% and 95% accuracy compared with original applications.

5.1 Experiment Settings

CiMSAT Platform. As shown in Figure 12, we have developed a system using Python to implement obfuscated weight fitting (Section 4.2.1), function transformation ((Section 4.2.2), netlist & CNF Reconstruction (Section 4.2.3), and SAT modeling (Section 4.3). Our system comprises three interfaces interacting with: NeuroSim simulator [5], Synopsys DC Compiler [50], and a SAT solver [41]. Specifically, the NeuroSim simulator provides application-level simulation for in-memory computing hardware, the DC compiler translates high-level language into netlists, and the SAT solver is utilized for btSAT analysis. These three simulation tools, combined with our system, form an analysis platform for de-obfuscation.

Hardware Setup. The function reconstruction using DC Compiler are executed on a 24-core Intel Xeon Gold 5318Y CPU running at 2.1GHz. Other simulations are executed on a 18-core Intel Core i9-10980XE CPU running at 3.00GHz and NVIDIA GeForce RTX 3090.

Existing CiM obfuscation defenses. Our focus is primarily on the architectures that obfuscate the computing process, i.e., performing on-chip computing using obfuscated ciphertext. As is shown in Table 1, We thoroughly searched for obfuscation techniques that can be applied to CiM for performing ciphertext computing and evaluated them. In the table we summarized the existing defenses, including the obfuscation type, cell device, computing modes, application datasets, and evaluation details considered in each work.

5.2 CiM De-Obfuscation Evaluation and Analysis

In this section, we first present the de-obfuscation attack results for the state-of-the-art obfuscation defenses in Table 1 under a typical parameter vector defined in the following Eq. (8). Subsequently, we generalize the different elements of the parameter vector and further implement CiMSAT to evaluate CiM security. To facilitate readers in accessing the conclusions, we label all the sub-conclusions starting from ❶.

Parameter Vector. For ease of discussion, we define a parameter vector:

$$\overrightarrow{Parameter} = (S, N, W, M, B) \quad (8)$$

Table 1: Existing CiM Obfuscation Defenses

Defense	Obfuscation Type	Cell Device	Computing Mode	Application Dataset	Evaluation Aspects
Zou et al. [66]	MUX	RRAM	Analog	CIFAR-10	Accuracy Drop Overheads
Chakraborty et al. [3]	MUX	/	Digital	MNIST CIFAR-10 SVHN	Accuracy Drop
Grailoo et al. [19]	MUX	/	Digital	MNIST SVHN	Accuracy Drop Overheads
Zhao et al. [64]	MUX	RRAM	Analog	CIFAR-10 MNIST ImageNet	Accuracy Drop Overheads Speedup
Wang et al. [55]	MUX	NVM	Analog	CIFAR-10	Accuracy Drop Overheads
Huang et al. [24]	XOR	SRAM	Analog	ImageNet	Accuracy Drop Overheads HW Performance
Li et al. [36]	XOR	RRAM	Analog	CIFAR-10	Accuracy Drop Overheads HW Performance
Zhao et al. [65]	XOR	FeFET	Analog	ImageNet	Accuracy Drop Device Overheads
Chen et al. [7, 35, 37]	XOR	SRAM/ eDRAM	Analog	MNIST CIFAR-10	Accuracy Drop Device HW Performance Overheads
Zou et al. [67]	Network	RRAM	Analog/ Digital	CIFAR-10	Accuracy Drop Overheads
Wang et al. [56]	Network	RRAM	Analog	MNIST	Accuracy Drop Overheads
Zou et al. [68]	Network	RRAM	Analog/ Digital	CIFAR-10	Accuracy Drop Overheads

where S (scale) refers to the size of crossbar. N (Number of key-controlled ports) is defined in Section 4.2.2 to describe the key-controlled I/O port number. W (width) represents data bit width, M (mode) indicates digital or analog/mixed signal computing mode, and B (bias) denotes reconstruction bias.

We applied parameter vectors to 14 defense techniques, resulting in 176 distinct defense vectors. Among them, we merged [7, 35, 37, 65] as one defense as they exhibited similar circuit characteristics. Our subsequent evaluation showed that 158 vectors (90%) could be de-obfuscated within 1000 seconds, and 172 vectors (98%) could be de-obfuscated within 1 day.

5.2.1 Effectiveness of CiMSAT Attack @ (16, 1, 8, Digital, 0). We show the attack effectiveness of CiMSAT under a typical parameter setting of 16×16 crossbar, row/column-wised key ports ($N = 1$), and the 8-bit data width [49]. We first analyze the digital CiM obfuscation, hence the bias is 0. Our de-obfuscation results are summarized in Table 2. Consistent with Table 1, we categorize the obfuscation types into three major classes: MUX, XOR, and Network. For each defense, we outline the specific locations where they deploy key ports. We provide the gate-level netlist size after functional transformation, uniformly represented in terms of two-input logic gates, along with the iteration count for SAT-based key solving, equivalence checks results, and the final de-obfuscation time. We can see that CiMSAT are able to de-obfuscated all of them within 12 iterations and 781 seconds.

From Table 2, we summarize two conclusions. ❶ Firstly, regarding the technique of obfuscation, the de-obfuscation time is approximately positively correlated with the gate number, i.e. reconstructed netlist size. The gate number of the Network type is significantly higher than MUX and XOR, leading to the higher de-obfuscation

Table 2: CiMSAT executing results on CiM obfuscation defenses with . Parameter vector = (16,1,8,Digital,0)

Obfuscation Type	MUX					XOR			Network		
Obfuscation Technique	[3]	[66]	[19]	[64]	[55]	[24]	[36]	[7, 35, 37, 65]	[67]	[56]	[68]
Obfuscation Position	Column	Column	Column	Row	Row	Row	Row	Row	Row	Column	R & C
Gate Number*	462	435	628	695	815	400	673	654	2560	6499	10531
Iteration	1	1	1	3	11	5	7	4	12	1	8
Checking Equivalence	Match	Match	Match	Match	Match	Match	Match	Match	Match	Match	Match
De-obfuscation Attack Time (s)	0.429	0.438	0.626	1.527	2.848	0.891	1.848	0.896	71.654	5.932	781.108

*Unified as two-input logic gates.

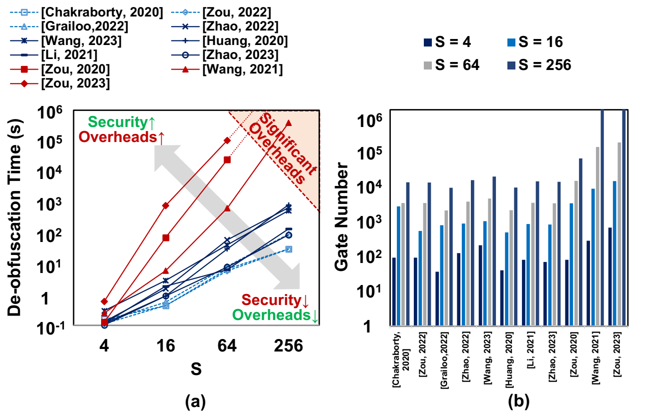


Figure 13: CiMSAT executing results with different S. Parameter vector = (S ∈ {4,16,64,256}, 1, 8, Digital, 0). (a) De-obfuscation time. (b) Gate Number.

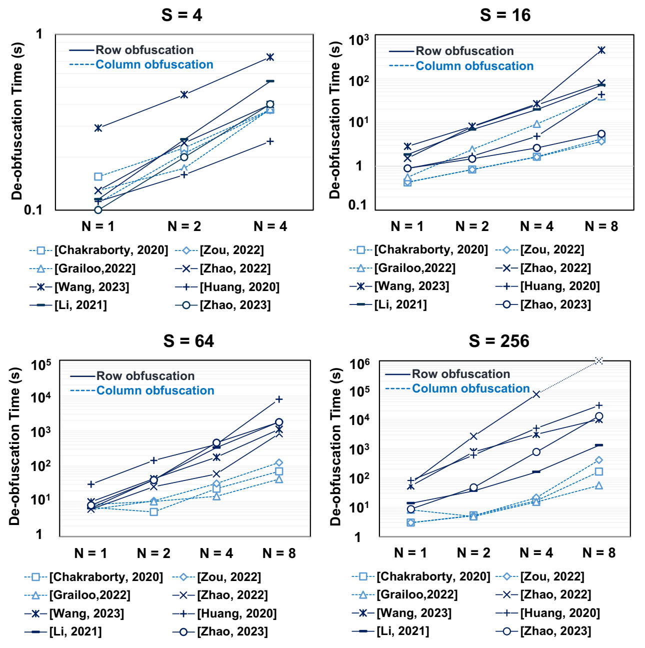


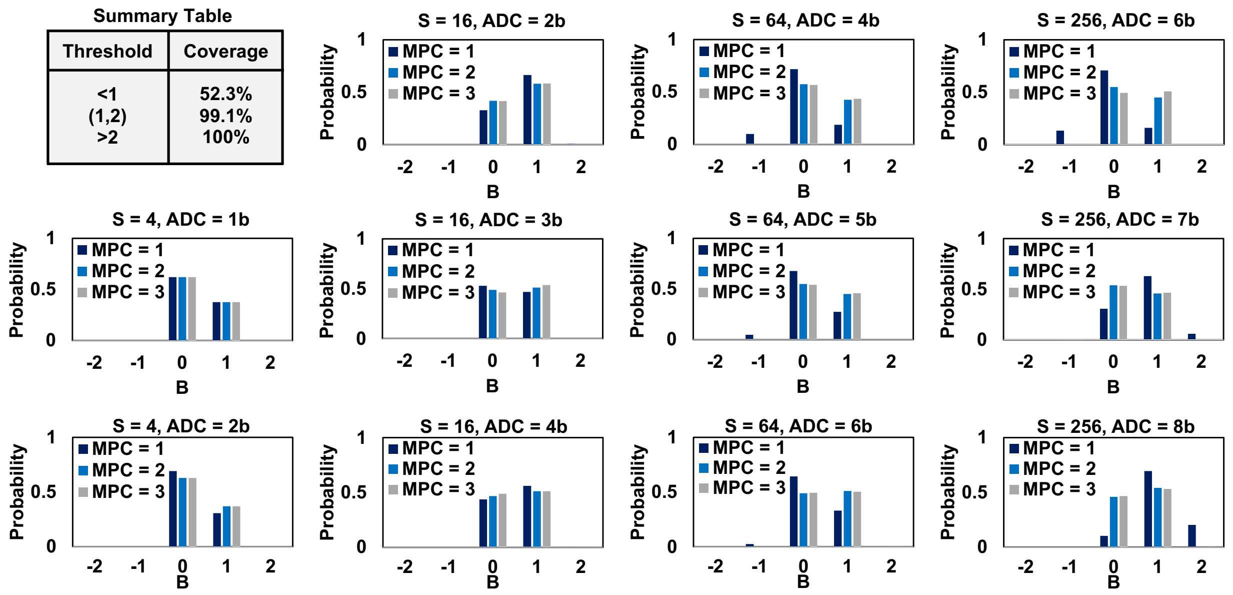
Figure 14: CiMSAT executing results with different S and N. Parameter vector = (S ∈ {4, 16, 64, 256} , N ∈ {1, 2, 4, 8}, 8, Digital, 0).

time. This reveals a design trade-off: the security of CiM obfuscation comes with hardware costs. ❷ Secondly, the iteration count for column obfuscation (key ports deployed in column dimension) is significantly lower than other types. This is because, for CiM test structures, columns are a common observation point for scan chains and are much easier to attack.

5.2.2 *Scale Analysis @ (**S**, 1, 8, Digital, 0).* This section evaluates the existing CiM obfuscation defenses on different array scale S. We select four S values from 4 to 256, covering mainstream designs [49]. We apply the existing obfuscation approaches to different array sizes. As the S increases, the scale of the reconstructed netlist will also increase, and lead to more time cost of the de-obfuscation process. Figure 13(a) illustrates the de-obfuscation time of CiMSAT under different S conditions, while Figure 13(b) shows the gate-level netlist scale of existing works after functional transformation. ❸ We observe a trade-off between de-obfuscation time and hardware cost depicted figures. For MUX/XOR types, the de-obfuscation time is significantly lower compared to the Network type, and correspondingly, the gate number of netlist is also lower. The Network type offers higher security demonstrated from de-obfuscated time, but entails more significant hardware overheads.

Furthermore, we can roughly predict the security boundaries of the three obfuscation types under static key insertion strategy, providing guidance to the CiM designers. For MUX/XOR types, even if the array scale is extended to the existing maximum S of 1024, the predicted CiMSAT de-obfuscation time will not exceed one day. ❹ Therefore, the critical conclusion is that low-cost obfuscation, highly regarded by hardware designers, is actually vulnerable from a security perspective. As for the Network type, if defenders are willing to bear the high hardware cost introduced by key insertion, resistance capability exceeding one day can be achieved in arrays with parallelism scales above 256. However, according to the latest CiM survey [49], the mainstream parallelism of existing MAC calculations still concentrates on 4 to 64. Therefore, static key insertions within mainstream parallelism ranging from 4 to 64 are under the security threats of keys being recovered.

5.2.3 *Key-based Module Analysis @ (**S**, **N**, 8, Digital, 0).* The relatively low hardware overhead of MUX/XOR types results in a much lower de-obfuscation time compared to Network types. In this section, we increase the key-controlled number of rows/columns and illustrate the MUX/XOR types obfuscation security under high key-related hardware overhead. We set the number of key-controlled


Summary Table

Threshold	Coverage
<1	52.3%
(1,2)	99.1%
>2	100%

Figure 15: Bias evaluation with 10,000 samples. The summary table indicates that over 99% of the values of B are less than 2. Parameter = (P ∈ {4, 16, 64, 256}, 1, 8, Analog, B)

row/columns from 1 to 8, consistent to the typical crossbar computing scale.

As shown in Figure 14, we evaluated the de-obfuscation time of eight types of MUX/XOR-based CiM obfuscation techniques under conditions where S is 4, 16, 64, and 256, and N ranges from 1 to 8 (with N reaching a maximum of 4 when S is 4). In each sub-figure, the de-obfuscation time increases with the increase of N, as the SAT solver needs to evaluate larger reconstructed netlist sizes. Comparing the four subplots, it can be observed that defense techniques with obfuscation positions in columns (light blue dashed lines) require less time to be attacked compared to those with obfuscation positions in rows (dark blue solid lines). In summary, 119 of 120 (99%) defense test points are de-obfuscated within 10^6 seconds (approximately 12 days), with 117 test points (98%) de-obfuscated within one day. ❺ The conclusion here is that increasing key-related hardware resources can indeed increase the de-obfuscation time, but SAT solvers are still able to recover the key in polynomial time. Additionally, security of column obfuscation remains lower than row obfuscation security, especially under conditions of large-scale arrays.

5.2.4 *Data Bit Width Analysis @ (**S**, 1, **W**, Digital, 0).*

For hardware design of data accelerators, data width is an important parameter that influences the performance and overheads. We conducted analysis of the data width (W) under S equal to 16 and 64. As shown in Figure 16, the de-obfuscation time grows approximately linear with the growth of W. ❻ Therefore, the conclusion here is that the length of the data width does not fundamentally affect the security of CiM obfuscation, yet longer data widths can increase the de-obfuscation time.

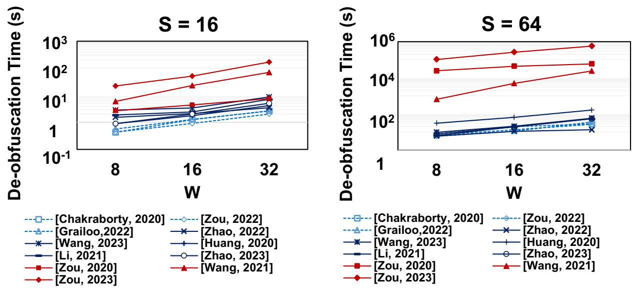


Figure 16: CiMSAT executing results with different S and W. Parameter vector = (S ∈ {16, 64} , 1, W ∈ {8, 16, 32}, Digital, 0).

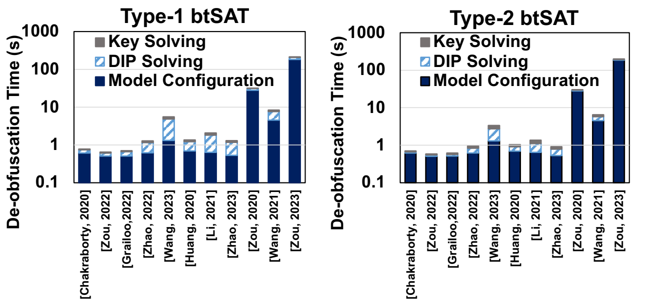


Figure 17: The time discrepancy between two types of btSAT. Parameter vector = (16, 1, 8, Analog, B≤2).

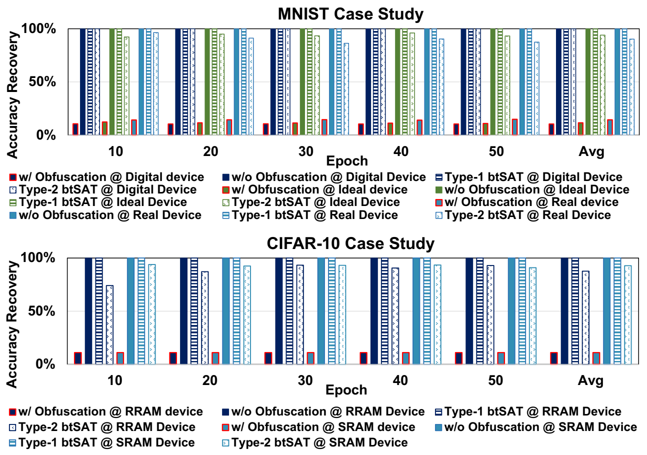


Figure 18: Accuracy recovery with keys from CiMSAT .

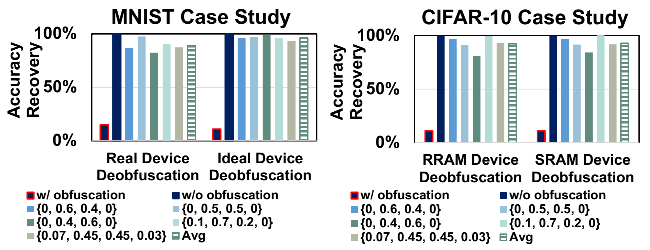


Figure 19: Accuracy recovery with different bias distribution. Legend represents {p(B=-1),p(B=0),p(B=1),p(B=2)}.

5.3 De-obfuscation on Analog CiM

*5.3.1 Bias Evaluation @ (**S**, 1, 8, Analog, **B**).* In this section, we evaluate the reconstruction bias B. We conduct random sampling on the reconstructed circuit and the original CiM circuit to obtain the distribution of B. As shown in Figure 15, for each sub-figure, we calculate the probability distribution of B under certain settings of S and ADC quantized bits. Firstly, for the digital CiM, B is entirely zero, indicating that MAC calculations in the digital domain do not introduce bias, consistent with our theoretical expectations. Secondly, we analyze B for four S values and observe that over 90% of B are concentrated near zero (0 or 1). Thirdly, we evaluate the number of data bits per memory cell of the crossbar (MPC) and observe that in most cases, MPC is not the main deterministic factor in CiM designs B. ❼ The conclusion drawn here is that the reconstruction of MAC exhibits low-error characteristics (no error for digital CiM), which is reasonable since CiM is essentially a hardware accelerator for MAC operations, and its operation pattern naturally approaches the linear calculation characteristics of real MAC expressions.

5.3.2 btSAT Time Evaluation @ (16, 1, 8, Analog, B≤2). This section evaluates the difference of de-obfuscation time between two types of btSAT attacks. We implement both types of btSAT on the existing CiM defenses with parameters = (16, 1, 8, Analog, B≤2). As shown in Figure 17, firstly, modeling and constructing the circuit scripts incurs a significant amount of time overhead (proportional to the reconstructed netlist size). Therefore, there is an approximate double time cost relationship difference in the DIP solving process, but the overall time difference depends on the proportion of DIP solving time. The larger the proportion, the more significant the time overhead of type-1 btSAT. Secondly, for different CiM obfuscation techniques, the additional time overhead generated by type-1 varies. The main reason is that the number of iteration rounds for DIP solving differs for each defense. The more the iteration rounds, the longer the additional time overhead generated by type-1 btSAT. ❽ The conclusion is that the time overhead of type-1 btSAT depends on the specific design of the DIP iterative solving process, and the time overhead of constructing the model cannot be ignored. For type-2 btSAT, because the bias of B is concentrated around 0, the threshold can be sufficiently small (≤ 2) as well as covering most DIPs. However, while DIP solving time is saved, it introduces solving bias. We evaluate this issue in Section 5.4.

5.4 Application Evaluation

We conducted CiMSAT evaluations with two case studies on typical image classification tasks MNIST [11] and CIFAR-10 [32] adopted by CiM obfuscation defense works, using CiMSAT Platform as shown in Figure 12. We uploaded models trained for 10 to 50 epochs onto CiM hardware and loaded the keys solved by CiMSAT into the key ports to evaluate the inference accuracy recovery.

Our evaluation results are presented in Figure 18. First, a two-layer perceptron for MNIST image classification is employed. We found that for three different device models (digital, ideal, real), type-1 btSAT can recover the classification accuracy by 100% compared with original CiM circuits, while type-2 btSAT can recover an average of 100%, 93%, and 89% of the classification accuracy, respectively. We then employ a VGG-8 model for CIFAR-10 classification. For two different device models (RRAM and SRAM), type-1 btSAT can recover the original classification accuracy by 100% compared with original CiM circuits, while type-2 btSAT can recover an average of 88% and 93% of the classification accuracy, respectively.

To further estimate type-2 btSAT in analog CiM de-obfuscation, in Figure 19, we extracted the probability distributions of typical B biases based on Figure 15 and evaluated the accuracy recovery. We denote the bias distribution as $\{p(B=-1), p(B=0), p(B=1), p(B=2)\}$. For MNIST application, the average accuracy recovery for real devices and ideal devices is 87% and 96% compared with original CiM circuits, respectively. For CIFAR-10 application, the average accuracy recovery for RRAM devices and SRAM devices is 91% and 92% compared with original CiM circuits, respectively.

❾ To conclude, in the typical application scenario of CiM obfuscation, our attack can achieve de-obfuscation across various devices and computing modes due to key-related logic and deployment algorithm that determine the CiM system security rather than device types. Additionally, since our de-obfuscation capability basically comes from the approximate modeling of CiM architecture and data, we believe CiMSAT can be extended to other applications.

5.5 End-to-End Demonstration and Analysis

In Section 4, we have conducted the following end-to-end analysis: target hardware and oracle definition, approximate netlist reconstruction, and btSAT analysis (the core algorithm). To make the

CiMSAT demonstration more comprehensive, we further evaluated its performance in a real-chip environment. We employed a mixed-signal compute-in-memory chip published in JSSC'23 [62], with specifications of (32, 1, 8, Analog, 2). We tested representative clock frequencies for DIP queries, with results showing that de-obfuscation remains effective, achieving over 95% accuracy. The maximum delay observed was 40ns per query, which constitutes only a negligible portion (<0.0001%) of the de-obfuscation time, as illustrated in Fig. 17.

The minimal time overhead observed is not an unexpected outcome, as the primary bottleneck in the hardware de-obfuscation process lies in the complexity of solving SAT-related algorithms. In the entire end-to-end demonstration process, the time associated with "queries" tied closely to the real hardware is minimal and deterministic. An attacker needs only to control the input-output interfaces of the logic module to acquire input-output pairs within a certain number of clock cycles, which can then be fed back into the SAT solver. Meanwhile, the SAT solver must iterate and solve a large number of SAT clauses, making it the primary contributor to the time overhead in the end-to-end demonstration. Therefore, we recommend that defenders focus on anti-SAT algorithm design during netlist development.

6 Discussion

6.1 Result Interpretation and Analysis

This work reveals that under the same obfuscation defenses, array parallelism is the primary parameter influencing the solving time and complexity. We investigated the trend in parallelism growth among different CiM architectures. Notably, only current-mode CiM architectures exhibit a clear trend of increasing parallelism. A significant milestone in this trend is the 512-parallelism design, which was published in 2023 [10]. However, despite this progression, extremely high-parallelism designs remain uncommon in mainstream applications. This is largely due to the substantial hardware overhead associated with such designs, which poses practical challenges that limit their widespread adoption.

To further understand the implications of high parallelism on security, we simulated the first three defense mechanisms listed in Table 1 using a parallelism level of 4096. The results revealed an average de-obfuscation time of approximately 3000 seconds. This outcome is consistent with the expectations derived from polynomial fitting models, suggesting that the de-obfuscation complexity scales predictably with increased parallelism, albeit at the cost of significantly longer computation times.

Therefore, while increasing the array size may offer some defensive advantages, it does not fundamentally enhance the netlist's resistance to SAT attacks. Moreover, as SAT-related algorithms continue to evolve, merely enlarging the array size is not an effective defense strategy. We advocate for and anticipate innovation at the CiM obfuscation algorithm level as a more robust approach to enhancing security.

6.2 Further Defense Strategy

We have demonstrated the effectiveness of CiMSAT and found that the static key insertion alone is insufficient to achieve exponential key complexity. Considering the defenses, there are mainly two directions. First is to break the foundation of the oracle-guided attack model, for example, by setting up secure scan chains for CiM. However, modifying the test structure implies utilizing larger logic circuit overhead and additional test pins [2], which cannot be automated using EDA tools. The second direction is to develop collaborative hardware and algorithm defenses under existing attack models. This requires defenders to combine the CiM design and anti-SAT algorithms. However, reducing the additional hardware overheads introduced by the algorithm while maintaining energy efficiency within the CiM array architecture remains to be a challenge. No related approaches have been proposed to the best of our knowledge.

6.3 Adaptability and Scalability Beyond VMM Operations

Currently, existing CiM architectures primarily focus on accelerating Vector-Matrix Multiplication (VMM), which provides prior knowledge for regression analysis. In the future, CiM may accelerate other computational paradigms, requiring updates to the algorithms for data extraction as discussed in Section 4.2. Our critical observation is that, compared to combination logic circuits, CiM typically serves as an accelerator in computing architecture, supported by underlying mathematical principles. If researchers can utilize these mathematical principles as prior knowledge, developing reverse fits may not be as challenging as SAT problems encountered in combination circuits. Existing work has already proposed corresponding algorithms [28, 42] that are fully compatible with our analysis framework.

7 Acknowledgement

This work is supported in part by NSFC (U21B2030, 92264204), LFET, and BNRist.

8 Conclusion

This work proposes CiMSAT attack that can break the security of existing CiM obfuscation defense. CiMSAT comprises functional approximation reconstruction and bias-tolerant SAT attack (btSAT), firstly focusing on modeling and processing the unique storage and mixed-signal computing feature of CiM architectures. We set up 176 defense vectors derived from all 14 existing CiM obfuscation techniques and evaluated security, 90% of which were de-obfuscated within 1,000 seconds and 98% within a day. We applied the de-obfuscated model to today's mainstream CiM confounding applications, recovering an average accuracy of 97% and 95% on MNIST and CIFAT-10 classifications. We also mark nine detailed sub-conclusion in the evaluation section to provide insights for the security of CiM and even similar new hardware architectures.

References

[1] Kazi Asifuzzaman, Narasinga Rao Miniskar, Aaron R Young, Frank Liu, and Jeffrey S Vetter. 2023. A survey on processing-in-memory techniques: Advances and challenges. *Memories-Materials, Devices, Circuits and Systems* 4 (2023), 100022.

[2] Kimia Zamiri Azar, Hadi Mardani Kamali, Houman Homayoun, and Avesta Sasan. 2021. From cryptography to logic locking: A survey on the architecture evolution of secure scan chains. *IEEE Access* 9 (2021), 73133–73151.

[3] Abhishek Chakraborty, Ankit Mondai, and Ankur Srivastava. 2020. Hardware-assisted intellectual property protection of deep learning models. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[4] Arjun Chaudhuri, Chunsheng Liu, Xiaoxin Fan, and Krishnendu Chakrabarty. 2021. C-testing and efficient fault localization for AI accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 7 (2021), 2348–2361.

[5] P.-Y. Chen, X. Peng, and S. Yu. 2017. NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures. In *IEEE International Electron Devices Meeting (IEDM)*. San Francisco, USA.

[6] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. *ACM SIGARCH computer architecture news* 44, 3 (2016), 367–379.

[7] Zhuojun Chen, Ming Wu, Yifeng Zhou, Renlong Li, Jinzhe Tan, and Ding Ding. 2023. PUF-CIM: SRAM-Based Compute-In-Memory With Zero Bit-Error-Rate Physical Unclonable Function for Lightweight Secure Edge Computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2023).

[8] S. K. Cherupally, J. Meng, A. S. Rakin, and et al. 2022. Improving the accuracy and robustness of RRAM-based in-memory computing against RRAM hardware noise and adversarial attacks. *Semiconductor Science and Technology* 37, 3 (2022), 034001.

[9] Sai Kiran Cherupally, Adnan Siraj Rakin, Shihui Yin, Mingoo Seok, Deliang Fan, and Jae-sun Seo. 2021. Leveraging Noise and Aggressive Quantization of In-Memory Computing for Robust DNN Hardware Against Adversarial Input and Weight Attacks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 559–564. https://doi.org/10.1109/DAC18074.2021.9586233

[10] Peter Deaville, Bonan Zhang, and Naveen Verma. 2023. A 256-kb Fully Row/Column-parallel 22nm MRAM In-Memory-Computing Macro with Differential Readout for Robust Parallelization and Scale-up. In *ESSCIRC 2023-IEEE 49th European Solid State Circuits Conference (ESSCIRC)*. IEEE, IEEE.

[11] L. Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[12] C. Dong, Y. Xu, X. Liu, and et al. 2020. Hardware trojans in chips: A survey for detection and prevention. *Sensors* 20, 18 (2020), 5165.

[13] Niklas Eén and Niklas Sörensson. 2003. An extensible SAT-solver. In *International conference on theory and applications of satisfiability testing*. Springer, 502–518.

[14] Samsung Electronics. 2023. HBM-PIM. https://semiconductor.samsung.com/news-events/tech-blog/hbm-pim-cutting-edge-memory-technology-to-accelerate-next-generation-ai/.

[15] Sina Sayyah Ensan, Karthikeyan Nagarajan, Mohammad Nasim Imtiaz Khan, and Swaroop Ghosh. 2021. SCARE: Side channel attack on in-memory computing for reverse engineering. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, 12 (2021), 2040–2051.

[16] Moritz Fieback, Mottaqiallah Taouil, and Said Hamdioui. 2022. Structured test development approach for computation-in-memory architectures. In *2022 IEEE International Test Conference in Asia (ITC-Asia)*. IEEE, 61–66.

[17] Jugal Gandhi, Diksha Shekhawat, M Santosh, and Jai Gopal Pandey. 2023. Logic locking for IP security: A comprehensive analysis on challenges, techniques, and trends. *Computers & Security* (2023), 103196.

[18] Anteneh Gebregiorgis and Mehdi B Tahoori. 2019. Testing of neuromorphic circuits: Structural vs functional. In *2019 IEEE International Test Conference (ITC)*. IEEE, 1–10.

[19] Mahdieh Grailoo, Uljana Reinsalu, Mairo Leier, and Tooraj Nikoubin. 2022. Hardware-assisted neural network ip protection using non-malicious backdoor and selective weight obfuscation. In *2022 IEEE 15th Dallas Circuit and System Conference (DCAS)*. IEEE, 1–6.

[20] Said Hamdioui, Moritz Fieback, Surya Nagarajan, and Mottaqiallah Taouil. 2019. Testing computation-in-memory architectures based on emerging memories. In *2019 IEEE International Test Conference (ITC)*. IEEE, 1–10.

[21] Mark C. Hansen et al. 1999. Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering. *IEEE Design & Test of Computers* 16, 3 (1999), 72–80.

[22] Weizhe Hua, Zhiru Zhang, and G Edward Suh. 2018. Reverse engineering convolutional neural networks through side-channel information leaks. In *Proceedings of the 55th Annual Design Automation Conference*. 1–6.

[23] L. Huang and J. Walrand. 2013. A Benes packet network. In *2013 Proceedings IEEE INFOCOM*. IEEE, 1204–1212.

[24] Shanshi Huang, Hongwu Jiang, Xiaochen Peng, Wantong Li, and Shimeng Yu. 2020. XOR-CIM: Compute-in-memory SRAM architecture with embedded XOR encryption. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–6.

[25] S. Huang, H. Jiang, and S. Yu. 2021. Mitigating adversarial attack for compute-in-memory accelerator utilizing on-chip finetune. In *2021 IEEE 10th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*. IEEE, 1–6.

[26] Shih-Hsu Huang, Wei-Che Cheng, and Jin-Fu Li. 2023. Hardware Trojans of Computing-In-Memories: Issues and Methods. In *2023 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. 1–6. https://doi.org/10.1109/DFT59622.2023.10313529

[27] Chuan-Jia Jhang, Cheng-Xin Xue, Je-Min Hung, Fu-Chun Chang, and Meng-Fan Chang. 2021. Challenges and trends of SRAM-based computing-in-memory for AI edge devices. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 5 (2021), 1773–1786.

[28] Shui Jiang, Seetal Potluri, and Tsung-Yi Ho. 2023. Scalable Scan-Chain-Based Extraction of Neural Network Models. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–6.

[29] Yier Jin and Gang Qu. 2021. *Hardware Security* (1st ed.). China Industry and Information Technology Publishing and Media Group.

[30] N. P. Jouppi, C. Young, N. Patil, and et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*. 1–12.

[31] Hadi Mardani Kamali, Kimia Zamiri Azar, Farimah Farahmandi, and Mark Tehranipoor. 2022. Advances in logic locking: Past, present, and prospects. *Cryptology ePrint Archive* (2022).

[32] A. Krizhevsky et al. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Technical Report.

[33] R. Lee. 2013. Threat-Based Design. In *Security Basics for Computer Architects*. Springer, Cham. https://doi.org/10.1007/978-3-031-01742-1_1

[34] Jin-Fu Li, Tsai-Ling Tsai, Chun-Lung Hsu, and Chi-Tien Sun. 2020. Testing of configurable 8T SRAMs for in-memory computing. In *2020 IEEE 29th Asian Test Symposium (ATS)*. IEEE, 1–5.

[35] Wen Li et al. 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*.

[36] Wantong Li, Shanshi Huang, Xiaoyu Sun, Hongwu Jiang, and Shimeng Yu. 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)*. IEEE, 1–2.

[37] W. Li, Y. Wang, H. Li, and X. Li. 2019. Leveraging Memory PUFs and PIM-based encryption to secure edge deep learning systems. In *2019 IEEE 37th VLSI Test Symposium (VTS)*. IEEE, 1–6.

[38] Euicheol Lim. 2022. *PIM and Various Computational Memory Solutions*. http://prism.sejong.ac.kr/dossa-4/dossa_paper/PIM_and_Various_Computational_Memory_Solutions-Euicheol_Lim_0402.pdf

[39] S. Mittal, H. Gupta, and S. Srivastava. 2021. A survey on hardware security of DNN models and accelerators. *Journal of Systems Architecture* 117 (2021), 102163.

[40] Sarath Mohanachandran Nair, Christopher Münch, and Mehdi B Tahoori. 2020. Defect characterization and test generation for spintronic-based compute-in-memory. In *2020 IEEE European Test Symposium (ETS)*. IEEE, 1–6.

[41] Lucas Nestor. 2020. sat_attack. https://github.com/lnestor/sat_attack.

[42] Seetal Potluri and Aydin Aysu. 2021. Stealing neural network models through the scan chain: A new threat for ml hardware. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–8.

[43] C. Qian, M. Zhang, Y. Nie, and et al. 2023. A Survey of bit-flip attacks on deep neural network and corresponding defense methods. *Electronics* 12, 4 (2023), 853.

[44] James Read, Wantong Li, and Shimeng Yu. 2022. A Method for Reverse Engineering Neural Network Parameters from Compute-in-Memory Accelerators. In *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 302–307. https://doi.org/10.1109/ISVLSI54635.2022.00066

[45] Brojogopal Sapui and Mehdi B. Tahoori. 2024. Power Side-Channel Analysis and Mitigation for Neural Network Accelerators based on Memristive Crossbars. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 612–617. https://doi.org/10.1109/ASP-DAC58780.2024.10473828

[46] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z Pan, and Yier Jin. 2017. AppSAT: Approximately deobfuscating integrated circuits. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 95–100.

[47] Felix Staudigl, Hazem Al Indari, Daniel Schön, Hsin-Yu Chen, Dominik Sisejkovic, Jan Moritz Joseph, Vikas Rana, Stephan Menzel, Amelie Hagelauer, and Rainer Leupers. 2024. It's Getting Hot in Here: Hardware Security Implications of Thermal Crosstalk on ReRAMs. *IEEE Transactions on Reliability* (2024), 1–15. https://doi.org/10.1109/TR.2024.3371589

[48] Pramod Subramanyan, Sayak Ray, and Sharad Malik. 2015. Evaluating the security of logic encryption algorithms. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 137–143.

[49] Wenyu Sun, Jinshan Yue, Yifan He, Zongle Huang, Jingyu Wang, Wenbin Jia, Yaolei Li, Luchang Lei, Hongyang Jia, and Yongpan Liu. 2023. A Survey of Computing-in-Memory Processor: From Circuit to Application. *IEEE Open Journal of the Solid-State Circuits Society* (2023).

[50] Synopsys. 2022. DC Compiler Version T-2022.03-SP2 for linux64. May 25, 2022.

[51] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* 105, 12 (2017), 2295–2329.

[52] S. Tripathi, S. Choudhary, and P. K. Misra. 2023. An 8T PA attack resilient NVSRAM for in-memory-computing applications. *IEEE Transactions on Circuits and Systems I: Regular Papers* (2023).

[53] Tsai-Ling Tsai, Jin-Fu Li, Chun-Lung Hsu, and Chi-Tien Sun. 2019. Testing of in-memory-computing 8T SRAMs. In *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 1–4.

[54] Tsai-Ling Tsai, Jin-Fu Li, Chun-Lung Hsu, and Chi-Tien Sun. 2021. Testing of in-memory-computing memories with 8 T SRAMs. *Microelectronics Reliability* 123 (2021), 114215.

[55] Jianfeng Wang, Zhonghao Chen, Yiming Chen, Yixin Xu, Tianyi Wang, Yao Yu, Vijaykrishnan Narayanan, Sumitha George, Huazhong Yang, and Xueqing Li. 2023. WeightLock: A mixed-grained weight encryption approach using local decrypting units for ciphertext computing in DNN accelerators. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 1–5.

[56] Yuhang Wang, Song Jin, and Tao Li. 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*. 499–504.

[57] Ziyu Wang, Fan-hsuan Meng, Yongmo Park, Jason K Eshraghian, and Wei D Lu. 2023. Side-channel attack analysis on in-memory computing architectures. *IEEE Transactions on Emerging Topics in Computing* (2023).

[58] Z. Wang, Y. Wu, Y. Park, and et al. 2023. PowerGAN: A Machine Learning Approach for Power Side-Channel Attack on Compute-in-Memory Accelerators. *Advanced Intelligent Systems* 5, 12 (2023), 2300313.

[59] Wikipedia. 2024. *Boolean satisfiability problem.* https://en.wikipedia.org/wiki/Boolean_satisfiability_problem

[60] Q. Xu, M. T. Arafin, and G. Qu. 2021. Security of neural networks from hardware perspective: A survey and beyond. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. 449–454.

[61] M. Yasin, A. Sengupta, M. T. Nabeel, and et al. 2017. Provably-secure logic locking: From theory to practice. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1601–1618.

[62] Guodong Yin, Yiming Chen, Mufeng Zhou, Wenjun Tang, Mingyen Lee, Zekun Yang, Tianyu Liao, Xirui Du, Vijaykrishnan Narayanan, Huazhong Yang, et al. 2023. Cramming More Weight Data Onto Compute-in-Memory Macros for High Task-Level Energy Efficiency Using Custom ROM With 3984-kb/mm{2} Density in 65-nm CMOS. *IEEE Journal of Solid-State Circuits* (2023).

[63] Shimeng Yu, Hongwu Jiang, Shanshi Huang, Xiaochen Peng, and Anni Lu. 2021. Compute-in-memory chips for deep learning: Recent trends and prospects. *IEEE circuits and systems magazine* 21, 3 (2021), 31–56.

[64] Lei Zhao, Youtao Zhang, and Jun Yang. 2022. SRA: a secure ReRAM-based DNN accelerator. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 355–360.

[65] Zijian Zhao, Yixin Xu, James Read, Po-Kai Hsu, Yixin Qin, Tzu-Jung Huang, Suhwan Lim, Kijoon Kim, Kwangsoo Kim, Wanki Kim, et al. 2023. In-Situ Encrypted NAND FeFET Array for Secure Storage and Compute-in-Memory. In *2023 International Electron Devices Meeting (IEDM)*. IEEE, 1–4.

[66] Minhui Zou, Junlong Zhou, Xiaotong Cui, Wei Wang, and Shahar Kvatinsky. 2022. Enhancing security of memristor computing system through secure weight mapping. In *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 182–187.

[67] Minhui Zou, Zhenhua Zhu, Yi Cai, Junlong Zhou, Chengliang Wang, and Yu Wang. 2020. Security enhancement for rram computing system through obfuscating crossbar row connections. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 466–471.

[68] Minhui Zou, Zhenhua Zhu, Tzofnat Greenberg-Toledo, Orian Leitersdorf, Jiang Li, Junlong Zhou, Yu Wang, Nan Du, and Shahar Kvatinsky. 2023. TDPP: Two-Dimensional Permutation-Based Protection of Memristive Deep Neural Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023).