SeDA: Secure and Efficient DNN Accelerators with Hardware/Software Synergy

Wei Xuan[†‡], Zhongrui Wang[⋆], Lang Feng[§], Ning Lin[¶], Zihao Xuan[†‡], Rongliang Fu[∥],
Tsung-Yi Ho[∥], Yuzhong Jiao[†‡], Luhong Liang[†‡]
[†]*ACCESS – AI Chip Center for Emerging Smart Systems, InnoHK Centers, Hong Kong Science Park, Hong Kong, China*
[‡]*The Hong Kong University of Science and Technology, Hong Kong, China*
[⋆]*Southern University of Science and Technology, Shenzhen, Guangdong, China*
[§]*Sun Yat-sen University Shenzhen Campus, Shenzhen, Guangdong, China*
[¶]*The University of Hong Kong, Hong Kong, China*
[∥]*The Chinese University of Hong Kong, Hong Kong, China*
Corresponding authors: wangzr@sustech.edu.cn, flang1994@outlook.com

***Abstract*—Ensuring the confidentiality and integrity of DNN accelerators is paramount across various scenarios spanning autonomous driving, healthcare, and finance. However, current security approaches typically require extensive hardware resources, and incur significant off-chip memory access overheads. This paper introduces SeDA, which utilizes 1) a *bandwidth-aware encryption mechanism* to improve hardware resource efficiency, 2) *optimal block granularity* through intra-layer and inter-layer tiling patterns, and 3) a *multi-level integrity verification mechanism* that minimizes, or even eliminates, memory access overheads. Experimental results show that SeDA decreases performance overhead by over 12% for both server and edge neural processing units (NPUs), while ensuring robust scalability.** [1]

***Index Terms*—Memory protection, secure DNN accelerators, confidentiality and integrity, deep neural networks**

I. Introduction

Securing Deep Neural Networks (DNNs) on neural processing units (NPUs) is increasingly vital for mission-critical applications in areas such as autonomous driving [1], healthcare [2], and finance [3]. The Artificial Intelligence (AI) hardware market was valued at USD 54.10 billion in 2023 and is expected to surge to USD 474.10 billion by 2030, reflecting a remarkable CAGR of 38.73% [4]. Protecting DNN accelerators by ensuring confidentiality and integrity is crucial for several reasons: *Data Confidentiality*: The sanctity of training data is non-negotiable. Protecting training data is essential to prevent unauthorized access and exploitation of private or sensitive information. *Resource Costliness*: The substantial investment in training resources necessitates stringent security protocols. Safeguarding these resources not only ensures their optimal utilization but also protects against financial losses and inefficiencies. *Vulnerability to Malicious Attacks*: Malicious intent poses a significant threat to DNN models. Protecting these models from potential tampering and attacks is critical to maintaining their integrity and functionality, safeguarding against adverse outcomes and ensuring the reliability of AI systems.

To protect traditional DNN accelerators from model theft and malicious tampering, as illustrated in Fig. 1(a) and Fig. 1(b), broad approaches [5]–[11] have been proposed to secure these accelerators, with the primary aim of minimizing off-chip memory access overhead for security metadata. They typically use the counter-mode encryption of Advanced Encryption Standard (AES-CTR) for confidentiality and the message authentication code (MAC) for integrity verification, as shown in Fig. 1(c). The counter value in AES-CTR concatenates the physical address (PA) and a ⓐ version number (VN) of a data block, with the VN stored off-chip and incremented with each write. To ensure the integrity of off-chip memory data, each data block is accompanied by a ⓑ message authentication code (MAC). Additionally, a ⓒ Merkle Tree (MT) [12] and its variants are often utilized, with the root stored on-chip to prevent replay attacks.

[1]SeDA source code: https://github.com/wayne4s/seda.git

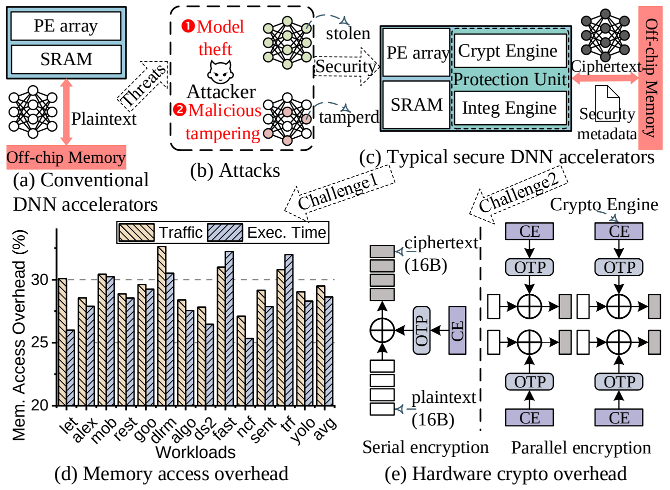


Fig. 1. Insight of typical secure accelerators. (a) Traditional DNN accelerators use untrusted off-chip memory and communication buses, risking model theft and malicious tampering, as shown in (b). (c) Secure DNN accelerators typically protect data confidentiality and integrity with memory protection schemes using AES encryption and Hash function. (d) Accessing security metadata in off-chip memory adds overhead. (e) Parallel hardware encryption incurs overhead to meet bandwidth needs of accelerators.

Motivation ❶ ► *Costly Off-Chip Memory Access Overhead for Integrity Verification*. Accessing security metadata (e.g., ⓐⓑⓒ) to ensure the confidentiality and integrity of untrusted off-chip memory significantly increases memory access overhead, as shown in Fig. 1(d). Existing approaches propose several optimizations: using Bonsai Merkle Tree (BMT) [13] instead of traditional MT for smaller version numbers (VNs); employing coarser-grained protection units, such as 512B data blocks instead of 64B cachelines; and dynamically updating VNs based on DNN model state information to eliminate off-chip memory access [8], [9]. Securator [11] uses a layer-level MAC to reduce off-chip memory access for integrity checks. However, it overlooks the overlap of intra-layer tiles and distinct tiling patterns across layers, potentially leading to redundant encryption/decryption and integrity verification overhead. Moreover, not considering inter-layer tiling pattern differences may result in false negatives.

Motivation ❷ ► *High Hardware Overhead for Confidentiality Protection*. Due to the AES Engine's limitation of en/decrypting only a 128-bit data block at one time, typical solutions to meet the high bandwidth demands of DNN accelerators involve increasing the number of AES Engines. For example, Securator [11] uses four AES engines to en/decrypt a 64B data block, as shown in Fig. 2(c). However, this approach adds strain on resource-limited accelerators, requiring a careful balance between hardware resource allocation and security performance. As illustrated in Fig. 1(e), a single Crypto Engine can only perform serial encryption, failing to meet the high bandwidth demands of DNN accelerators, while multiple AES engines can satisfy these demands but result in significant hardware encryption overhead.

In this paper, we introduce SeDA, a novel secure and efficient DNN accelerator architecture with reduced hardware resource consumption and near-zero performance overhead for integrity verification, while maintaining the same level of security and practical applicability. This paper makes the following major contributions:

- **Insight.** Providing an in-depth insight of limitations of current memory protection strategies for DNN accelerators highlights two critical concerns: the substantial *hardware overhead for encryption* and the expensive *off-chip memory access for integrity verification*.
- **Solution.** Through hardware/software synergy, we present SeDA, a secure and efficient accelerator architecture. It incorporates a *bandwidth-aware encryption mechanism* that utilizes a single AES engine with adjustable encryption granularity, minimizing hardware resource overhead. Furthermore, its *multi-level integrity verification mechanism* significantly reduces or eliminates off-chip memory access overhead.
- **Evaluation.** Conducting extensive experiments using cycle-accurate simulators for DNN accelerators, memory protection schemes, and off-chip memory accesses. Experimental results of SeDA demonstrate a performance improvement, reducing overhead by 12.26% for server NPU and 12.29% for edge NPU, while also providing robust scalability with minimal hardware overhead to meet the bandwidth demands of accelerators.

This paper is structured as follows: Section II reviews related work on memory protection and the threat model for our study. In Section III, we provide a detailed introduction to the proposed SeDA architecture. We then conduct extensive experiments on various DNN models in Section IV. Finally, Section V concludes our work.

II. Related Work and Threat Model

Based on the corresponding threat model, secure DNN accelerators typically provides robust confidentiality and integrity guarantees in untrusted environments.

A. Confidentiality Protection

Confidentiality protection typically uses AES-CTR mode due to several advantages: 1) it utilizes the same AES engine for en/decryption, reducing hardware overhead (see Fig. 2(a)); 2) One time pad (OTP) generation can be parallelized with communication and DNN computation, minimizing time overhead; and 3) its block-based streaming mode does not require prior knowledge of data size and scale. The AES-CTR encryption requires a non-repeating and incrementing counter to produce a OTP for each encryption/decryption under the same key. The counter concatenates the physical address of a data block that will be encrypted and version number that is incremented on each write memory operation. Here, let $K_e, \mathcal{P}, \mathcal{C}$ be the AES encryption key, plaintext, ciphertext, respectively. The AES-CTR mode for encryption and decryption can be formulated as following Eq. 1 and Eq. 2, where $AES\text{-}CTR_{K_e}(PA||VN)$ produces

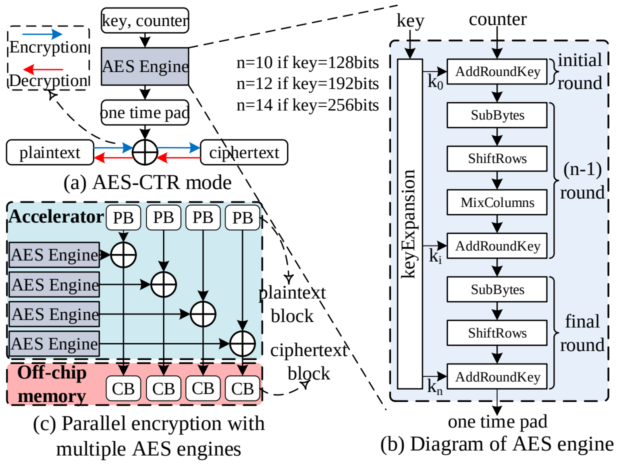


Fig. 2. Summary of AES-CTR mode. (a) Reusing AES engine for encryption and decryption in AES-CTR mode. (b) Diagram of the AES engine, featuring the AddRoundKey, SubBytes, ShiftRows, MixColumns, and keyExpansion modules. (c) Utilization of multiple AES engines for parallel encryption to boost high-bandwidth capabilities.

a OTP, || and ⊕ represents a bit-wise concatenation and XOR operator, respectively.

$$\mathcal{C} = \mathcal{P} \oplus AES\text{-}CTR_{K_e}(PA \,||\, VN) \tag{1}$$

$$\mathcal{P} = \mathcal{C} \oplus AES\text{-}CTR_{K_e}(PA \,||\, VN) \tag{2}$$

When an accelerator reads a data block from off-chip memory, the protection module uses the VN to generate an OTP and decrypts the block by XORing it with the OTP (red path in Fig. 2(a), Eq. 2). Similarly, when writing a data block to off-chip memory, the module increments the VN to generate an OTP and encrypts the block by XORing it with the OTP (blue path in Fig. 2(a), Eq. 1). Fig. 2(b) illustrates the diagram of the AES engine along with its key functional modules.

B. Integrity Verification

To ensure data integrity against off-chip memory tampering, an integrity verification engine computes a message authentication code (MAC) by concatenating a data block with its physical address and version number. This MAC is generated during every write operation and verified during reads to ensure data authenticity. Relying solely on MAC for a data block fails to guarantee freshness and opens the door to replay attacks. To counter this threat, prior works utilize an Integrity Tree for hierarchical MAC verification, such as MT [12] and Bonsai Merkle Tree (BMT) [13], with the root stored on-chip to prevent malicious tampering. The overhead of integrity verification is non-trivial since it requires traversing both MACs and the nodes of the Integrity Tree stored in the off-chip memory to prevent the replay attack.

C. Secure DNN Accelerators

Intel SGX [5] creates a secure enclave using CPU hardware mechanisms, using AES-CTR mode and MT with its root in the Trusted Computing Base (TCB). SEAL [6] introduces a criticality-aware smart encryption scheme that selectively bypasses the encryption engine for partial data and colocates data with corresponding counters. GuardNN [7] proposes a secure accelerator architecture by using small TCB and low overhead for privacy-preserving deep learning. MGX [8] introduces a secure DNN accelerator with application-specific version number management, utilizing on-chip status for version number

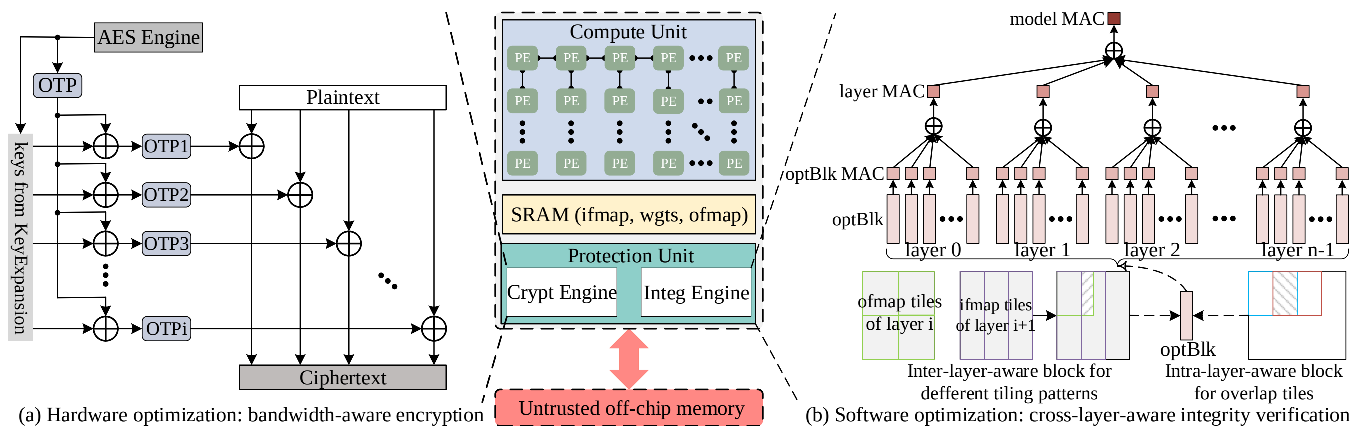

AES Engine
OTP
keys from KeyExpansion
OTP1
OTP2
OTP3
OTPi
Plaintext
Ciphertext
(a) Hardware optimization: bandwidth-aware encryption
Compute Unit
PE
SRAM (ifmap, wgts, ofmap)
Protection Unit
Crypt Engine
Integ Engine
Untrusted off-chip memory
model MAC
layer MAC
optBlk MAC
optBlk
layer 0
layer 1
layer 2
layer n-1
ofmap tiles of layer i
ifmap tiles of layer i+1
Inter-layer-aware block for defferent tiling patterns
optBlk
Intra-layer-aware block for overlap tiles
(b) Software optimization: cross-layer-aware integrity verification

Fig. 3. Overview of SeDA architecture. (a) The Crypt Engine optimizes hardware by XORing keys from the AES Engine's KeyExpansion module with OPT, creating bandwidth-sensitive encryption granularity and reducing hardware overhead compared to using multiple AES Engines. (b) The Integ Engine optimizes software by analyzing overlapping tiles within a layer and patterns across layers to determine the optimal block size, optBlk, for integrity verification. This leads to a multi-level integrity verification mechanism with optBlk MAC, layer MAC, and model MAC.

generation and coarse-grained integrity verification to reduce off-chip memory overhead. In a similar vein, TNPU [9] generates all version numbers within an on-chip cache. Securator [11] introduces a layer-level integrity checks by XORing all message authentication codes (MACs) within a layer.

D. Threat Model

Similar to the threat model proposed in literature [8], [9], we assume that the accelerator itself is secure, meaning that its internal state cannot be directly observed or tampered with. Any external device connected to the accelerator is considered untrusted, such as off-chip memory and communication buses, meaning that attackers could potentially access or manipulate any information stored in these devices through physical or software attacks. Notably, we do not consider other physical side-channel attacks to infer DNN model architectures [14], such as power channel attacks and electromagnetic attacks. Additionally, adversarial attacks [15] against machine learning algorithms are also not within scope to ensure that decryption is restricted to authorized users possessing the key.

III. SeDA Architecture

This section initially provides an overview of SeDA. We then subsequently explores the scalability challenges of cryptographic hardware and the expensive off-chip memory access overhead in integrity verification, providing corresponding solutions for each.

A. Overview

SeDA provides a secure and efficient environment for DNN accelerators by hardware/software co-optimization. The overview architecture of the framework is presented in Fig. 3. The AES Engine's limitation to encrypting only a 128-bit data block at one time poses a challenge in meeting the high bandwidth demands of DNN accelerators. SeDA addresses these issues by utilizing bandwidth-aware encryption mechanism to optimize hardware resources impacted by security concerns, as illustrated in Fig. 3(a). Additionally, the inclusion of security metadata by secure DNN accelerators results in storage and access overheads in off-chip memory. For example, using an 8B MAC to represent a 64B data block alone leads to a 12.5% increase in memory traffic. To mitigate this, SeDA introduces a multi-level integrity verification software optimization mechanism, showcased in Fig. 3(b). Furthermore, securely housing these limited MACs directly

Algorithm 1: Attack and defense of SECA

```
Attack of SECA
Input  : OTP: one time pad for a data block ($blk$);
         $most\_value\_p$: most used plaintext in $blk$.
Output : $value\_p$: all plaintext of $blk$
1 $most\_value\_c \longleftarrow$ CalcFreqValue($blk$)
2 OTP $\longleftarrow most\_value\_p \oplus most\_value\_c$
3 for each encrypted element value_c of blk do
4     $value\_p \longleftarrow value\_c \oplus$ OTP

Defense of SECA
Input  : PA, VN: physical address and version number of $blk$.
Output : OTP$_i$: generate multiple OTPs for $blk$
5 OTP $\longleftarrow$ AES-CTR$_{K_e}$(PA ‖ VN)
6 for each 128-bit $key_i$ in keyExpansion of AES-CTR do
7     OTP$_i \longleftarrow$ (OTP $\oplus$ $key_i$)
```

in on-chip SRAM can eliminate off-chip memory access overheads entirely.

B. Bandwidth-aware encryption mechanism

To meet the high bandwidth demands of accelerators, increasing the number of AES engines incurs significant overhead, while using the same OTP for all data in a block introduces security risks.This dilemma often leads to a suboptimal trade-off between security and computational efficiency.

Challenge 1: High Hardware Resources for Confidentiality Protection. While maintaining an equivalent level of security, numerous approaches have bolstered encryption parallelism by stacking multiple AES engines. For example, Securator [11] uses four AES-128 engines to concurrently encrypt a 64B data block, as shown in Fig. 2(c). Nevertheless, the drawback of sacrificing hardware resources to achieve performance improvements in encryption and decryption is apparent, particularly for edge DNN accelerators constrained by limited hardware capabilities. This situation emphasizes the crucial equilibrium needed to optimize security and operational efficiency in these systems.

Challenge 2: Security Threat for Shared OTP. A straightforward approach is to use each engine once per data block, with each 128-bit segment within this data block sharing the same OTP. Nevertheless, assigning a OTP to a individual data block poses security risks and could be vulnerable to a Single-Element Collision Attack (**SECA**). The principle of a SECA attack when a data block shares the same OTP is outlined in lines 1-4 of Algorithm 1. By employing the CALCFREQVALUE function, we can identify the most frequently encrypted data within the block, denoted as $most\_value\_c$. Through comprehensive data analysis, we can infer the most common plaintext data within the block, denoted as $most\_value\_p$ (e.g., 0). Referring to the AES-CTR encryption formula Eq. 1, we can calculate the OTP for that block as $most\_value\_c \oplus most\_value\_p$, as shown in line 2 of Algorithm 1. Since the block shares this OTP, once obtained, we can derive all plaintext values within the encrypted block, as shown in lines 3-4 of Algorithm 1. Extending this principle further, we can extract the data values from every DNN layer and even the entire model.

Solution. We introduce a novel bandwidth-aware encryption mechanism to meet the high bandwidth requirements of accelerators without compromising the security of encryption. This method leverages the inherent features of AES-CTR encryption mode, using just a single AES engine and a minimal number of XOR logic gates, as illustrated in Fig. 3(a). To defend against SECA attacks, we first employ a standard AES-CTR encryption engine to create a shared OTP for a data block, as shown in line 5 of Algorithm 1. Subsequently, utilizing the keys array produced by the keyExpansion module within the AES-CTR encryption engine, we can generate multiple distinct OTPs by XORing the OTP with keys, as shown in lines 6-7 of Algorithm 1. This ensures that each 128-bit data segment within the data block corresponds to a unique OTP, thereby thwarting SECA attacks to safeguard security. Moreover, this mechanism significantly conserves hardware resources by utilizing a small number of XOR logic gates instead of an equivalent number of AES engines, commonly used by traditional methods.

When it comes to securing the entire DNN model, using a single key in the AES-CTR keyExpansion process is typically sufficient to ensure security. We proceed with the assumption that multiple keys are generated through the keyExpansion module using just one key. The keys produced by key expansion are inherently secure. These keys can be stored on-chip or generated in real-time. By XORing the shared OTP from a data block with these different keys, new secure OTPs can be generated. When generating OTPs from multiple keys derived from a single key input for keyExpansion doesn't meet the accelerator's bandwidth requirements, expanding the key expansion input to $key \oplus$ (PA ‖ VN) can solve the issue. This approach produces enough OTPs for a data block, meeting both security and bandwidth needs simultaneously.

C. Multi-level integrity verification mechanism

The granularity of integrity verification, whether too large or too small, can be problematic. Small granularity increases security metadata and off-chip memory overhead, affecting performance. Larger granularity reduces metadata overhead but can lead to redundant processing due to tiling overlaps. SeDA addresses this with a multi-level integrity verification mechanism that combines the flexibility of fine granularity, which avoids redundant security computations, with near-zero overhead of coarse granularity, significantly reducing or eliminating off-chip memory access overhead.

Challenge 1: Expensive Off-Chip Memory Access for Integrity Checks. While recent research efforts [8], [9] have successfully eliminated the off-chip memory access overhead of VNs and Merkle Trees, the overhead introduced by MACs still remains to be adequately addressed. Securator [11] proposed a layer-level freshness and integrity check method to reduce the MACs overhead by XORing MACs of all blocks within a layer, with a block size granularity of 32 bytes. The approach didn't consider tile overlaps within a layer, causing redundant computations like repeated integrity checks, which adds costs. Securator also ignored different tiling patterns between layers. Fig. 3(b) shows that $ofmap$ in layer i and $ifmap$ in layer i+1 use different strategies, which can vary in size and direction. Not addressing inter-layer tiling properly can lead to extra costs or disrupt the model due to inaccurate integrity checks.

```
Algorithm 2: Attack and defense of RePA
Attack of RePA
Input  : MAC$_i$: the MAC of a data block in one layer.
Output : $plaintext\_e$: error plaintext.
1 SUM_MAC ⟻ $\sum_{i=1}^{n} \oplus$ MAC$_i$
2 SHUFFLEORDER(MACs)
3 SUM_MAC_shuffle ⟻ $\sum_{i=1}^{n} \oplus$ MAC$_i^{'}$
4 if True = VERIFYINTEG(SUM_MAC, SUM_MAC_suffle) then
5     for each encrypted data block blk of one layer do
6         $plaintext\_e$ ⟻ DECRYPT($blk$)
Defense of RePA
Input  : $layer_{id}$, $fmap_{idx}$, $blk_{idx}$: layer number, feature
         map index and block index of $layer_{id}$.
Output : Secure layer MAC.
7 for each encrypted data block blk of $layer_{id}$ do
8     MAC$_i$ ⟻ HASH$_{K_h}$($blk$ ‖ PA ‖ VN ‖ $layer_{id}$ ‖ $fmap_{idx}$
        ‖ $blk_{idx}$)
```

TABLE I
COMPARISON OF MULTI-LEVEL INTEGRITY VERIFICATION GRANULARITY.

Granularity	Flexibility	Off-chip Access Overhead	Storage
optBlk	●	◔	Off-chip
layer	◍	◔	Off/On-chip
model	◌	○	On-chip

*Note: Color depth shows intensity, while coverage area indicates ratio.

Challenge 2: Security Threat for XOR-MAC Scheme. The XOR-MAC scheme offers parallelizability, incrementality, and provable security, with its security being on par with that of chaining MAC [16]. However, XORing all MACs generated by directly hashing the ciphertext within a layer to produce a unique layer MAC could lead to a Re-Permutation Attack (RePA), as illustrated in lines 1-6 of Algorithm 2. SHUFFLEORDER rearranges the order of data blocks in the current layer and computes SUM_MAC_shuffle for that layer using XOR operations. Because XOR is commutative, meaning the order of operands doesn't affect the result, an attacker can successfully pass the VERIFYINTEG(SUM_MAC, SUM_MAC_shuffle) verification. This rearrangement of data blocks within a layer can hinder correct decryption of ciphertext, posing a security risk due to RePA vulnerability.

Solution. To reduce off-chip memory access overhead from integrity verification, we propose a secure multi-level mechanism. It combines block-level and layer-level checks to minimize memory access by leveraging deterministic memory access patterns of DNNs, while ensuring security. We use the scheduling search strategy proposed in the SecureLoop [10] to obtain the optimal authentication block

TABLE II
DNN SIMULATION CONFIGURATIONS [20], [21].

Metrics	Server (Google TPU v1)	Edge (Samsung Exynos 990)
PE	256 x 256 in systolic array	32 x 32 in systolic array
Bandwidth	20 GB/s with 4 channels	10 GB/s with 4 channels
Frequency	1 GHz	2.75 GHz
SRAM	24 MB	480 KB
Precision	1-B for per element	1-B for per element

(optBlk), which synergizes with our work orthogonally. We propose a novel multi-level integrity verification mechanism involving three granularities of MAC types: optBlk MAC, layer MAC, and model MAC, as shown in Fig. 3(b). Table I compares their features in detail. The optBlk MAC offers flexibility by accounting for intra-layer tiling overlap and diverse inter-layer tiling patterns, avoiding redundant computations from repeated integrity checks. By XORing all optBlk MACs within a layer to create the layer MAC, we achieve a significant reduction in the number of MACs needed for DNN model integrity verification, despite a slight delay due to layer-specific checks. This allows layer MACs to be stored in on-chip SRAM, eliminating off-chip memory access costs. The Model MAC uses a single MAC to represent the entire model weights on-chip, further reducing off-chip memory costs and conserving on-chip SRAM, with verification results available only at the end of model inference. To prevent RePA attacks, we associate each encrypted optBlk data block with specific location details like PA, VN, $layer_{id}$, $fmap_{idx}$, and blk_{idx}. We then compute the corresponding MAC using hashing, as detailed in lines 7-8 of Algorithm 2. In summary, our multi-level integrity verification mechanism can remove off-chip memory overhead from security metadata, ensuring security while maintaining integrity guarantee.

IV. EVALUATION

A. Experimental Setup

Accelerators. In order evaluate DNN inference behaviours, we use an open-source cycle-level DNN simulator SCALE-Sim2 [17], [18] developed by ARM Research to 1) study the inference execution for various DNN models; 2) analyse the performance overhead of different memory protection schemes. The DNN accelerator can generate detailed computation information of systolic array, and DRAM access traces. After obtaining the DRAM traces, we use various memory protection mechanisms to calculate execution time and bandwidth usage, producing the total DRAM traces after running the security simulator. Finally, we use the DRAM simulator Ramulator2 [19] to simulate the total DRAM access traces.

Configurations. Table II presents the DNN simulation configurations, featuring a server NPU (Google TPU v1) and an edge NPU (Samsung Exynos 990). To balance DNN computation and memory bandwidth, we simulate four 64-bit DDR channels for both the server and edge NPUs.

Benchmarks. For DNN accelerators, we evaluate SeDA across various DNN models, including Lenet (let), Alexnet (alex), Mobilenet (mob), ResNet18 (rest), GoogleNet (goo), DLRM (dlrm), AlphaGoZero (algo), DeepSpeech2 (ds2), FasterRCNN (fast), NCF_recommendation (ncf), Sentimental_seqCNN (sent), Transformer_fwd (trf), Yolo_tiny (yolo). These models are selected from diverse machine learning domains, such as computer vision, speech recognition, natural language processing, gaming, and personalized recommendation.

Memory Protection Simulation. We implemented accelerators based on Intel SGX, MGX, and SeDA security mechanisms, using an unprotected accelerator as a benchmark, and set the size of protected

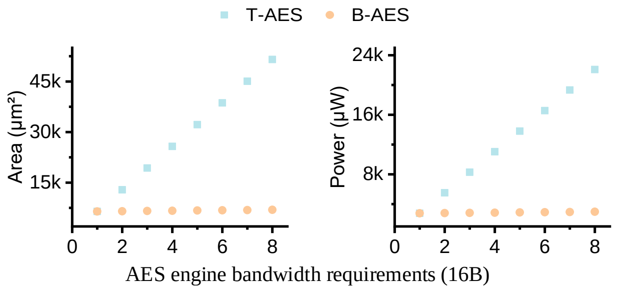
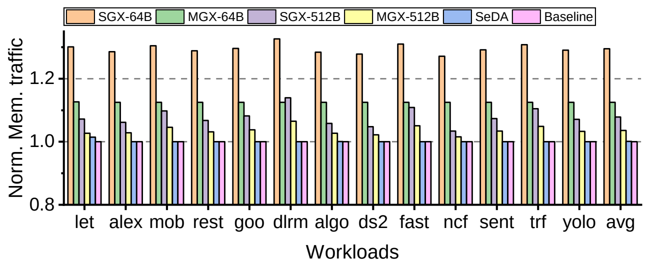
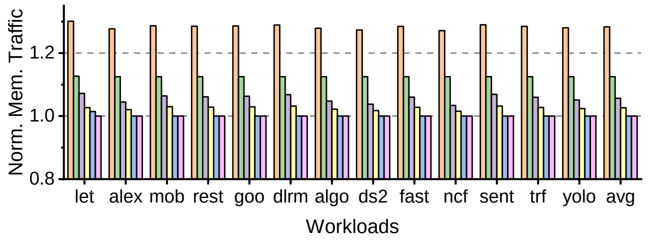


Fig. 4. The area and power with increasing AES engine bandwidth requirements.



(a) Server NPU



(b) Edge NPU

Fig. 5. The normalized memory traffic of memory protection schemes for various workloads

memory to 16GB. SGX uses a multi-level Integrity Tree with 56-bit VNs and 64-bit MACs, along with a 16KB VN cache and 8KB MAC cache, both using an LRU replacement policy for write-back and write-allocate strategies. We use two different protection granularities: 64B and 512B. To ensure fairness, SeDA stores layer MACs off-chip.

B. Experimental Results

We compare the memory traffic, which refers to the data exchanged between off-chip memory and accelerators, and performance across a unprotected baseline and five protection schemes: SGX-64B, SGX-512B, MGX-64B, MGX-512B, and our proposed SeDA, as shown in Table III. All results are normalized to the baseline without protection.

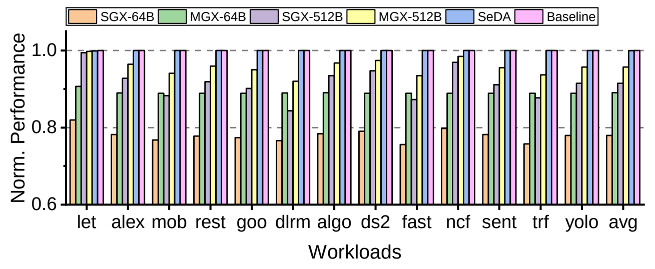
Area and Power. We developed a simulator based on 28nm technology to evaluate area and power, utilizing the AES engine implementations detailed in [22]. We refer to the bandwidth-aware encryption mechanism as B-AES, and other traditional methods using multiple AES engines as T-AES. Through simulation, we assessed the changes in area and power. The x-axis represents the bandwidth required by the encryption engine to match the accelerator's bandwidth needs, expressed as multiples of the bandwidth provided by a single AES engine. As shown in Fig. 4, our proposed B-AES demonstrates strong scalability, with minimal increases in area and power consumption as the accelerator bandwidth increases.

Memory traffic. Fig. 5 compares the memory traffic overhead using the DNN simulation configurations listed in Table II. On average,

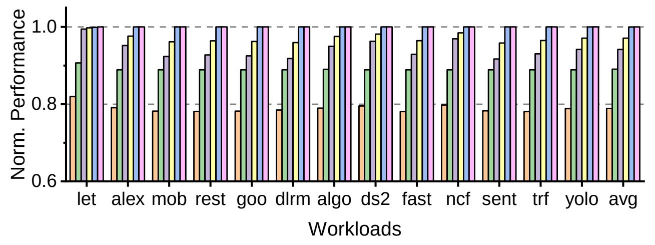
TABLE III
COMPARISON OF MEMORY PROTECTION SCHEMES.

Protection Scheme	Encryption Granularity	Integrity Granularity	Off-chip Memory Access	DNN Tiling Pattern	Encryption Scalability
SGX-64B	16B	64B	MAC,VN,IT	✗	✗
SGX-512B	16B	512B	MAC,VN,IT	✗	✗
MGX-64B	16B	64B	MAC	✗	✗
MGX-512B	16B	512B	MAC	✗	✗
SeDA	bandwidth-aware	multi-level	minimal to no cost	✓	✓

*Note: "IT" signifies integrity tree.



(a) Server NPU



(b) Edge NPU

Fig. 6. The normalized performance of memory protection schemes for various workloads

SGX-64B increases memory traffic by 30% for Server NPU and 28.29% for Edge NPU. In contrast, MGX-64B, without the overhead from additional VNs and MT, increases traffic by only 12.51% and 12.63%, respectively. Increasing the protection granularity from 64B to 512B significantly reduces memory traffic. SGX-512B cuts traffic by 7.83% on Server NPU and 5.13% on Edge NPU. MGX-512B achieves reductions of 3.59% and 2.39% on these devices, respectively. Expanding the protection granularity to 512 bytes reduces memory traffic by minimizing security metadata. However, using larger data blocks can lead to inefficiencies due to misalignment with intra-layer tiling overlaps and varying inter-layer tiling patterns. This mismatch can hinder memory access and resource utilization, emphasizing the need for a balanced approach in selecting granularity to improve system performance and efficiency. Our proposed SeDA uses a multi-level integrity verification mechanism by XORing all optBlk MACs into a layer MAC. Fig. 5 shows that SeDA introduces near-zero memory traffic, with an overhead of only 0.12% for Server NPU and 0.03% for Edge NPU, highlighting the superiority of our proposed protection mechanism.

Performance. Fig. 6(a) presents a performance comparison analysis among the baseline and five memory protection mechanisms on Server NPU. SGX-64B is 22.04% slower, MGX-64B is 10.93% slower, SGX-512B is 8.49% slower, and MGX-512B is 4.28% slower than the unprotected baseline. In contrast, our proposed SeDA impacts performance by less than 1%, making it nearly negligible. This is primarily attributed to proposed multi-level integrity verification mechanism, which can reduce or even completely eliminate the performance overhead of off-chip memory accesses caused by integrity verification. By minimizing the storage and retrieval of MACs in off-chip memory, SeDA effectively manages the security metadata overhead from integrity verification. Additionally, by storing these small layer or model MACs directly in on-chip SRAM, we can remove the performance overhead associated with off-chip access for integrity verification. For Edge NPU, Fig. 6(b) presents similar findings. Compared to the baseline, SGX-64B, MGX-64B, SGX-512B, and MGX-512B slow down by 21.10%, 10.95%, 5.84%, and 2.90%, respectively. Remarkably, our proposed SeDA results in an almost imperceptible performance drop.

V. CONCLUSION

In this work, we introduce SeDA, a secure and efficient DNN accelerator architecture designed to ensure confidentiality and integrity in untrusted environments. Using bandwidth-aware encryption and multi-level integrity verification mechanisms, SeDA provides excellent scalability with minimal overhead to match computational bandwidth of accelerators, and significantly reduces or even eliminates off-chip memory access overhead. Experimental results show that proposed SeDA meets bandwidth requirements with near-zero hardware overhead, and significantly reduces the performance overhead compared to the state-of-the-art approaches.

ACKNOWLEDGMENT

This research was partially supported by the National Key R&D Program of China (Grant No. 2022YFB3608300). It was supported in part by ACCESS – AI Chip Center for Emerging Smart Systems, sponsored by the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government. It was also supported in part by Shenzhen Science and Technology Innovation Commission (Grant No. SGDX20220530111405040), Beijing Natural Science Foundation (Grant No. Z210006), Hong Kong Research Grant Council (Grant Nos. 27209621, 17205922, 17212923), the National Natural Science Foundation of China under Grant 62204111.

References

[1] D. Parekh, N. Poddar, A. Rajpurkar, M. Chahal, N. Kumar, G. P. Joshi, and W. Cho, "A review on autonomous vehicles: Progress, methods and challenges," *Electronics*, vol. 11, no. 14, p. 2162, 2022.

[2] G. Rong, A. Mendez, E. B. Assi, B. Zhao, and M. Sawan, "Artificial intelligence in healthcare: review and prediction case studies," *Engineering*, vol. 6, no. 3, pp. 291–301, 2020.

[3] F. Rundo, F. Trenta, A. L. Di Stallo, and S. Battiato, "Machine learning for quantitative finance applications: A survey," *Applied Sciences*, vol. 9, no. 24, p. 5574, 2019.

[4] "Artificial intelligence AI hardware market size and forecast," https://www.verifiedmarketresearch.com/product/global-artificial-intelligence-ai-hardware-market/, 2024.

[5] V. Costan and S. Devadas, "Intel SGX explained," *Cryptology ePrint Archive*, 2016.

[6] P. Zuo, Y. Hua, L. Liang, X. Xie, X. Hu, and Y. Xie, "Sealing neural network models in encrypted deep learning accelerators," in *ACM/IEEE Design Automation Conference*, 2021, pp. 1255–1260.

[7] W. Hua, M. Umar, Z. Zhang, and G. E. Suh, "Guardnn: secure accelerator architecture for privacy-preserving deep learning," in *ACM/IEEE Design Automation Conference*, 2022, pp. 349–354.

[8] W. Hua, M. Umar, Z. Zhang, and G. E. Suh, "MGX: Near-zero overhead memory protection for data-intensive accelerators," in *International Symposium on Computer Architecture*, 2022, pp. 726–741.

[9] S. Lee, J. Kim, S. Na, J. Park, and J. Huh, "TNPU: Supporting trusted execution with tree-less integrity protection for neural processing unit," in *International Symposium on High-Performance Computer Architecture*, 2022, pp. 229–243.

[10] K. Lee, M. Yan, J. Emer, and A. Chandrakasan, "SecureLoop: Design space exploration of secure DNN accelerators," in *IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 194–208.

[11] N. Shrivastava and S. R. Sarangi, "Securator: A fast and secure neural processing unit," in *International Symposium on High-Performance Computer Architecture*, 2023, pp. 1127–1139.

[12] B. Gassend, G. E. Suh, D. Clarke, M. Van Dijk, and S. Devadas, "Caches and hash trees for efficient memory integrity verification," in *International Symposium on High-Performance Computer Architecture*, 2003, pp. 295–306.

[13] B. Rogers, S. Chhabra, M. Prvulovic, and Y. Solihin, "Using address independent seed encryption and bonsai merkle trees to make secure processors os-and performance-friendly," in *IEEE/ACM International Symposium on Microarchitecture*, 2007, pp. 183–196.

[14] M. Yan, C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures," in *USENIX Security Symposium*, 2020, pp. 2003–2020.

[15] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018.

[16] M. Bellare, R. Guérin, and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions," in *Annual International Cryptology Conference*, 1995, pp. 15–28.

[17] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "SCALE-Sim: Systolic CNN accelerator simulator," *arXiv preprint arXiv:1811.02883*, 2018.

[18] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A systematic methodology for characterizing scalability of DNN accelerators using scale-sim," in *International Symposium on Performance Analysis of Systems and Software*, 2020, pp. 58–68.

[19] H. Luo, Y. C. Tuğrul, F. N. Bostancı, A. Olgun, A. G. Yağlıkçı, and O. Mutlu, "Ramulator 2.0: A modern, modular, and extensible DRAM simulator," *Computer Architecture Letters*, 2023.

[20] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *International Symposium on Computer Architecture*, 2017, p. 1–12.

[21] J. Song, Y. Cho, J.-S. Park, J.-W. Jang, S. Lee, J.-H. Song, J.-G. Lee, and I. Kang, "7.1 An 11.5TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC," in *IEEE International Solid-State Circuits Conference*, 2019, pp. 130–132.

[22] U. Banerjee, "Energy-efficient protocols and hardware architectures for transport layer security," Ph.D. dissertation, Massachusetts Institute of Technology, 2017.