The Journal of Supercomputing (2024) 80:5815–5831
https://doi.org/10.1007/s11227-023-05671-9

Nacc-Guard: a lightweight DNN accelerator architecture for secure deep learning

Peng Li[1,2] · **Cheng Che**[1,2] · **Rui Hou**[1,2]

Accepted: 15 September 2023 / Published online: 7 October 2023

Abstract

Recent breakthroughs in artificial intelligence and deep neural networks (DNNs) have produced an explosive demand for computing platforms equipped with customized domain-specific accelerators. However, DNN accelerators have security vulnerabilities. Researchers have previously explored DNN attack and defense technologies that mainly focus on training and inference algorithms or model structure robustness. The problem of how to design a secure accelerator architecture has received relatively little attention, especially with the rapid development of FPGA-based heterogeneous computing SoCs. To mitigate this bottleneck, we propose Nacc-Guard, a lightweight DNN accelerator architecture which can effectively defend against neural network bit-flip attacks and memory Trojan attacks. By utilizing a linear randomization encryption algorithm based on stream cipher Trivium, interrupt signal confused coding, and hash-based message authentication code, Nacc-Guard can not only guarantee the integrity of the uploaded DNN file but also ensure buffer data confidentiality. To evaluate Nacc-Guard, NVDLA and a SIMD accelerator coupling with a RISC-V Rocket and ARM processor is implemented at RTL. Experimental evaluation shows that Nacc-Guard has a 3× hardware overhead reduction compared with conventional AES. Experiments on VGG, ResNet50, GoogLeNet, and YOLOv4-tiny validate that this framework can successfully ensure secure DNN inference with negligible performance loss. It achieves a 3.63× speedup and 35% energy reduction over the AES baseline.

Keywords DNN accelerator · Security · Trivium · RISC-V

✉ Peng Li
lipeng0629@iie.ac.cn

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 10093, China

[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 10049, China

1 Introduction

In the post-Moore era, deep learning technology has achieved great breakthroughs in various application fields, such as computer vision, NLP, and autonomous driving [1–3]. As the main component of deep learning, deep neural networks (DNNs) can address a wide range of AI challenges and even perform better than human beings in some specific domains [4]. The strong feature extraction and learning abilities are attributed to a DNN's robust network structure and high computational cost. The amount of floating-point operations can reach the order of ten billion or more [5].

To reduce the DNNs computational complexity and improve its training or inference speed, some algorithm-based optimization techniques have been proposed. These techniques include sparse neural network by dropout compression [6] and layer pruning [7–10]. Although these approaches have achieved great success in some specific networks, they still cannot meet the high computational requirement for some new and complex neural network structures, such as the transformer. In order to fundamentally solve this problem, the DNN accelerator was proposed. By quantizing long floating-point weight files to a short fixed-point format and mapping this to highly parallel PE arrays, a DNN accelerator can improve the throughput of data parallel processing.

However, DNN accelerators have crucial security vulnerabilities [11–13]. Attackers can capture the model structure or weight parameters through monitoring sensitive data or snooping the accelerator interrupt patterns [12, 13]. They can also conduct bit-flip attacks on the neural network weight files and this can significantly decrease a DNN model's inference accuracy [14].

Tackling these challenges requires hardware and software co-design, unifying the theoretical privacy algorithms with secure hardware architecture. In terms of software, researchers have tried their best to design robust network structures or planning more complicated training algorithms, such as adversarial training and GAN-aided training [15]. In parallel, hardware designers also design secure DNN accelerators by means of encrypting sensitive data or using trusted execution environment (TEE) technology [16, 17]. For example, Wang et al. proposed NPUFort which inserted a security unit into the existing FPGA-based heterogeneous computing accelerator by using the AES block cipher method [16]. Hanieh et al. presented DarKnight which utilized a customized data encoding strategy based on matrix masking to achieve data obfuscation on a TEE. DarKnight can guarantee data privacy and integrity in conventional GPUs [17].

However, these protection techniques have inherent performance limitations. For block cipher AES, the complex iterative mode (ECB, CTC, and CTR) can bring about heavy computing latency and lead to the potential risk of data conversion error [18]. Furthermore, to convert the long plaintext message to fixed-size blocks, AES needs some padding data and this can involve extra memory resource overhead. Third, to achieve accurate decision-making, a DNN requires extensive memory and computing resources which does not operate well in a TEE enclave with restricted memory space [19].

To address the above challenges, we propose Nacc-Guard, a lightweight security-enhanced DNN accelerator architecture. In Nacc-Guard, an improved stream cipher algorithm Trivium is adopted in DNN inference accelerators. With its less hardware consumption, Trivium is more suitable for encrypting resource-limited systems. Furthermore, the interrupt signals from the accelerator to a host CPU are refused. Conventional high and low level interrupt signals are recoded to two positive edge latency by 1B/4B algorithm; this confusion can avoid attacks based on monitoring interrupt signal patterns. Third, to ensure the integrity of the uploaded DNN weight file and achieve authentication, hash-based message authentication code (HMAC) is used.

The main contributions are summarized as follows:

- We propose a novel secure DNN accelerator architecture named Nacc-Guard which can defend DNNs against memory Trojan and neural network bit-flip attacks.
- Different from conventional AES, an improved linear randomization algorithm Trivium is first adopted in DNN accelerators. Experiments show that Trivium is more suitable for encrypting hardware resource-limited DNN accelerators.
- Interrupt signals from the accelerator to the host CPU are recoded from the high/low level to positive edge latency; this can help to avoid attacks from monitoring the interrupt signal patterns. Furthermore, hash-based message authentication code is involved to ensure the integrity of the uploaded DNN weight file and achieve authentication.
- The Nacc-Guard prototype is implemented in NVDLA and SIMD DNN accelerator coupling with RISC-V Rocket and ARM CortexA9 at the RTL level. Runtime evaluation shows that this architecture can successfully ensure secure DNN inference. Experiments on VGG, ResNet50, GoogLeNet, and YOLOv4-tiny shows that Nacc-Guard can bring about 3× hardware overhead reduction and 3.63× performance improvement over the AES baseline with negligible extra power consumption.

The rest of this paper is organized as follows. Section 2 discusses related work and motivation. Section 3 introduces the design. In Sect. 4, the experimental evaluation and result analysis are presented. In Sect. 5, we conclude the paper.

2 Related work and motivation

2.1 DNN accelerator

The DNN accelerator was first invented to mitigate the bottleneck between a neural network's heavy computational requirements and its training and inference speed [20]. DNN accelerators (such as GPU, FPGA, and CGRA) have denser parallel computing micro-architectures and can offer higher data throughput bandwidth without significant accuracy loss. It can result in orders of magnitude

Table 1 Several typical DNN accelerators

DNN accelerator	Stage	Year
DianNao [21]	Inference	2014
Graphcore [22]	Inference/Training	2017
TPU [23, 24]	Training	2018
NVDLA [25]	Inference	2018
Equinox [26]	Inference/Training	2021
Transformer accelerator [27]	Inference/Training	2022

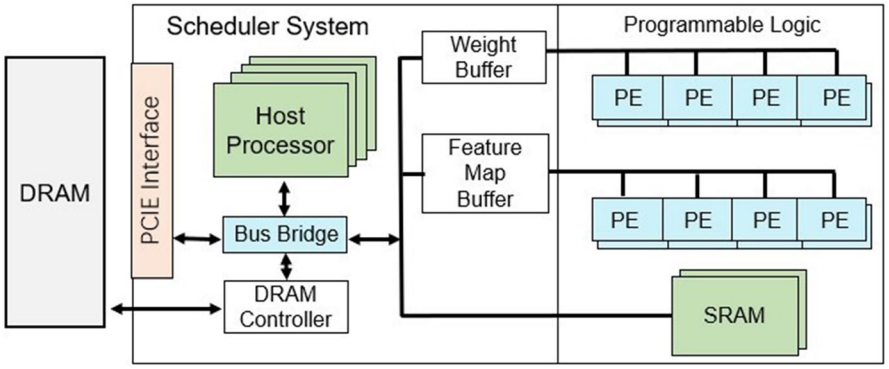

Fig. 1 DNN accelerator architecture

improvement of computational density with higher power efficiency. Several typical DNN accelerators are shown in Table 1.

In terms of DNN stage, there are training and inference accelerators. For example, the Graphics Processing Unit (GPU) is a DNN accelerator which is used for neural network training. A GPU is designed to provide a high-performance computing platform with large data throughput. In parallel, the DNN inference accelerator is also a young but quickly developing technology. In order to meet the fast changing and flexible characteristics of deep neural networks, an inference accelerator's micro-architecture should always be designed to be both scalable and reconfigurable. This requirement makes FPGA the best choice. Furthermore, with the development of heterogeneous computing technology of a CPU coupling with FPGA in one SoC, FPGA-based DNN inference accelerators are becoming more and more popular. FPGA-based DNN accelerator can be shown in Fig. 1.

A well-trained DNN model can be mapped onto an inference accelerator to realize real-time prediction. A pre-trained network weight file always contains long floating-point types. FPGA is better at fixed-point and bit-shift calculations. To deploy a DNN model on FPGA-based inference accelerators, the weight file should be first converted to a fixed-point type (such as 8-bit or 16-bit) through

quantization strategies [28, 29]. Short fixed-point representations of weights and feature maps can significantly reduce the computational cost with negligible accuracy loss [30–32].

2.2 Trivium and message authentication code

In general, cryptography involves symmetry cryptograms and public key cryptograms. Public keys are often used for key distribution while symmetric cryptography is used for sensitive data encryption. Block ciphers and stream ciphers are the main components of symmetric cryptography [18, 33].

A block cipher algorithm encrypts a fixed size of data (such as 128 bits as a block) at one time. The 128-bit and 256-bit are the most widely used cipher blocks [18]. For example, a 128-bit plaintext will be encrypted into a 128-bit ciphertext. In cases where the plaintext data is shorter than the block size, a bit padding scheme will be called into use. Advanced Encryption Standard (AES, Rijndael) is now the most commonly used block encryption algorithm. Specifically, AES treats the 128-bit plaintext block as 16 bytes, and these 16 bytes are arranged in four columns and four rows for the next step of processing as a matrix. For the processing, AES uses 10 transformation rounds for 128-bit keys and 14 transformation rounds for 256-bit keys. Each round involves four transforming steps: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

In this paper, an improved linear randomization Trivium algorithm is deployed for buffer data encryption and decryption. Trivium consists of three interconnected nonlinear feedback shift registers (NLFSR) of length 93, 84, and 111 bits [34]. First, Trivium

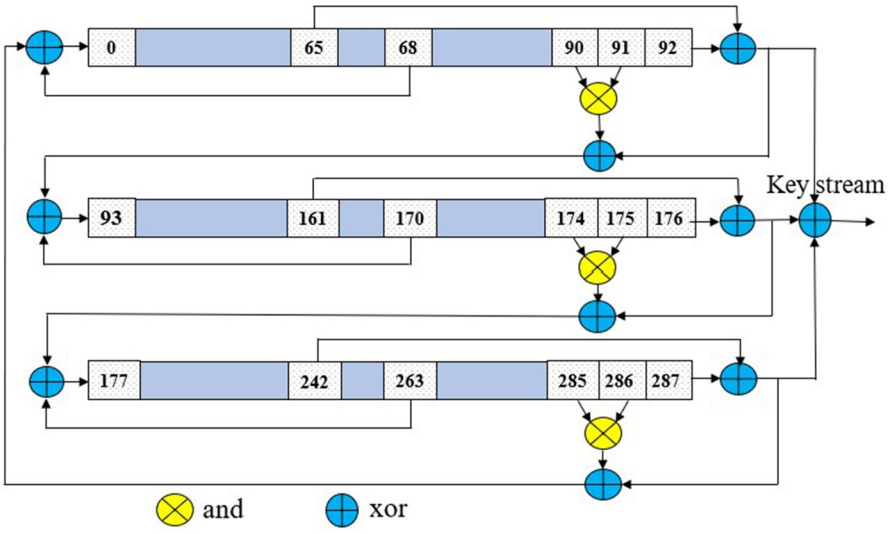

Fig. 2 Register framework of the stream cipher Trivium

requires a pre-owned 80-bit key and an 80-bit initialization vector (IV). This key and vector are used to pad the above state registers. Then, 1152 steps of the clocking shift procedure are required before Trivium keystream generation. Finally, we can encrypt the plaintext data by the bitwise exclusive or operation (XOR) with the generated keystream. The register structure and keystream generation process are illustrated in Fig. 2. Trivium was originally designed to meet the high data throughput requirement in hardware resource-limited systems. For its deployment, it can guarantee data confidentiality without an undue increase in the hardware resource overhead. For this reason, Trivium is more suitable for DNN inference accelerators than conventional AES techniques.

Message Authentication Code (MAC) is a tag which can be attached to the original files to ensure the integrity and authenticity of the pre-transmitted data. To protect the integrity of the DNN weight files and the DNN model authentication, a hash-based MAC (HMAC) is used in Nacc-Guard. HMAC is generated by applying a hash function to the original data message in combination with a key. A one-bit data change will produce a different HMAC, hence HMAC can guarantee that the DNN weight files are legitimate and do not contain harmful code.

2.3 Vulnerability and motivation

DNNs have security vulnerabilities [35, 36]. Attackers can intentionally implant backdoor or malicious programs in DNN files. They can steal, modify, or even destroy the whole neural network system. To be specific, there are algorithm-based and hardware-based attacks. Algorithm-based attacks include adversarial example attacks [37], model inversion attacks [38], model extraction attacks [39], and data poisoning attacks [40]. In parallel, hardware-based attacks include memory Trojan attacks [41], side-channel information leakage attacks [42], and neural network bit-flip attacks [14, 43].

In this paper, we focus on DNN accelerator memory Trojan attacks and bit-flip attacks. For example: (1) a hacker can change weight file key bit positions that are critical to the DNN model inference accuracy. Malicious programs can also be uploaded together with files to the accelerator without authentication; (2) an eavesdropper can capture the neural network structure via snooping interrupt signal patterns [12]; (3) Trojan attackers can also make an accelerator not work properly through hijacking sensitive data stored in the massive on-chip buffers [40].

To address this challenge, previous approaches used AES encryption in DNN accelerators. For example, Wang et al. presented a secure accelerator architecture named NPUFort [16]. They added AES-based data en/decryption modules and instruction en/decryption modules into DNN accelerators. To some extent, this offers an alternative method that can guarantee accelerator security. Unfortunately, this method only ensures data confidentiality rather than integrity. In addition, AES could bring dramatically higher hardware overhead and data processing latency. Another defense method is to build a TEE in the DNN accelerators [17]. However, to achieve accurate predictions, a DNN requires extensive memory and computing resources which does not operate well in TEE enclaves which have restricted memory space [19].

3 Nacc-Guard

3.1 Threat model

DNN models reuse accelerator hardware resources layer by layer; after a layer is finished, the accelerator will give an interrupt to inform the host CPU to dispatch the next layer. Certain neural network models are accompanied by certain interruption signal patterns. By monitoring the interrupt signal patterns, an attacker can capture a DNN model structure.

Furthermore, due to its special memory hierarchy and high parallel micro-architecture, DNN accelerators have massive on-chip buffers and this makes it more vulnerable to buffer Trojan attacks. By monitoring the plaintext buffers, an attacker can launch hardware Trojan attacks.

3.2 Overview

Nacc-Guard is realized on FPGA-based heterogeneous computing SoC platforms. Figure 3 depicts the anatomy of the whole architecture and its main functional modules. In this figure, an on-chip host processor is deployed coupling with the DNN accelerator. Specifically, the host processor connects the accelerator IP core as an IO device through the on-chip bus (such as AXI, or AXI to APB bus). The accelerator IP core registers are mapped to the Linux process virtual address space via the mmap function and the accelerator can communicate with DRAM through the DMA controller. The host processor orchestrates the neural network inference process, including uploading the remote DNN model files from DRAM or clouds and configuring control registers. PE arrays perform feature map matrix multiplication and addition. By using local on-chip memory access patterns, this tightly coupled micro-architecture can significantly reduce the latency of data traffic. All neural network layers reuse the accelerator computing systolic arrays. The scheduler executes the neural network operator layer by layer. Ping-pong buffers are used to alleviate

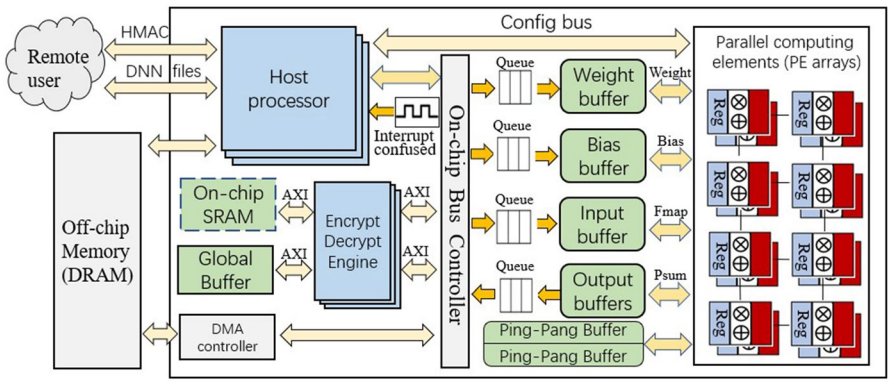

Fig. 3 Overview of the Nacc-Guard micro-architecture

the bottleneck between the higher parallel computing speed and the lower intermediate data access speed.

To ensure security, two keys are used for accelerator data encryption and DNN weight file verification. For the accelerator, key-1 is used for on-chip data encryption, and for the host processor, key-2 is used for generating the message authentication code. To be specific, the accelerator IP core should first store an 80-bit secret key-1 in a register for the improved Trivium engine. Then, we can distribute the other key-2 for the on-chip processor through public key cryptography. Meanwhile, the off-chip or remote users who can upload DNN models files to Nacc-Guard should share the same key-2 with the host processor for generating the message authentication code.

3.3 Encryption engine

A DNN accelerator has a Domain-Specific Architecture (DSA). The PE has locally coupled memory structures and decentralized on-chip buffers. Hackers can attack a DNN accelerator by monitoring the plaintext of intermediate data or inference result on the massive on-chip buffers through memory Trojans [41]. This characteristic means that it has crucial security vulnerabilities and we should encrypt sensitive data in massive on-chip buffers. However, these accelerators often suffer from stringent on-chip hardware resource limitations. This problem is more severe in real-time systems and mobile devices. This challenge makes the stream cipher Trivium more suitable for accelerator encryption than the conventional block cipher AES. In Nacc-Guard, to guarantee the confidentiality of data that are offloaded from the accelerator to the on-chip SRAM or critical buffers, an improved linear randomization encryption algorithm Trivium is used for the AXI buses. This design can support parallel computing in DNN accelerators and can meet the high throughput data requirement. We deployed the Trivium engine in the AXI bus controller to selectively encrypt data before it is transmitted to the data write channel and decrypt data that are coming from the data read channel. The linear randomization encryption Trivium implementation details can be seen in Fig. 4.

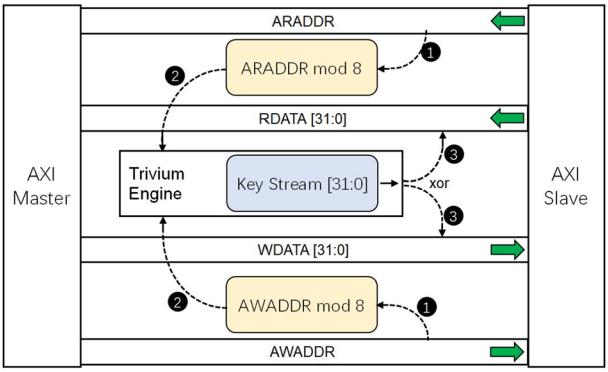

Fig. 4 Schematic diagram of linear randomization encryption Trivium algorithm in AXI bus controller

According to the AXI protocol, before the master port initiates a data read or write request in the data channel, a corresponding address value should be sent to the address read or write channels. First, the address values are captured and the modulus-8 operation is applied (ADDR mod 8). Then, if the result is 0, the Trivium engine will encrypt the write data or decrypt the read data from the data channels. This lightweight encryption scheme will not bring excessive hardware overhead or performance loss.

$$plaintext = m = m_1 m_2 m_3 ..., \tag{1}$$

$$keystream = s = s_1 s_2 s_3 ..., \tag{2}$$

$$cipher = c = m \, xor \, s, \tag{3}$$

$$plaintext = c \, xor \, s = m \, xor \, s \, xor \, s, \tag{4}$$

For data encryption and decryption, the process can be seen in Eqs. (1) to (4). In (1) and (2), m and s are the plaintext message and keystream, respectively. As shown in (3) and (4), the ciphertext can be obtained by applying the XOR operation to the plaintext and keystream. If the ciphertext applies the XOR operation again to the keystream, we will recover the plaintext data. As a result, if a malicious memory Trojan snoops on data from the SRAM or on-chip buffer, they will get the randomized encrypted data and have no ability to launch an attack according to the pre-set plaintext information. This operation can not only guarantee on-chip data confidentiality but also can avoid DNN model inversion attacks. The Nacc-Guard accelerator stores the pre-distributed key-1 in a secure register, while the IV(The 80 bit initialization vector in Trivium nonlinear-feedback shift register) can be generated by a pseudorandom number generator.

3.4 Interrupt signal confused coding

DNN accelerators have a specific working mode. The CPU scheduler executes the neural network operator layer by layer, and DNN models reuse accelerator PE resources layer by layer. After computing a layer, the accelerator will give an interrupt to inform the host CPU to dispatch the next layer. By monitoring the interrupt signal pattern through side channels, attackers can capture a DNN model layer number and structure. To solve this problem, in Nacc-Guard, interrupt signals from the accelerator to the host CPU are recoded; conventional high/low level interrupt signals are recoded to the positive/negative edge latency by using the mB/nB coding algorithm (1B/4B). As shown in Fig. 5, a specific clock cycle delay represents a real interrupt. This confusion can avoid attacks from snooping the interrupt signal patterns.

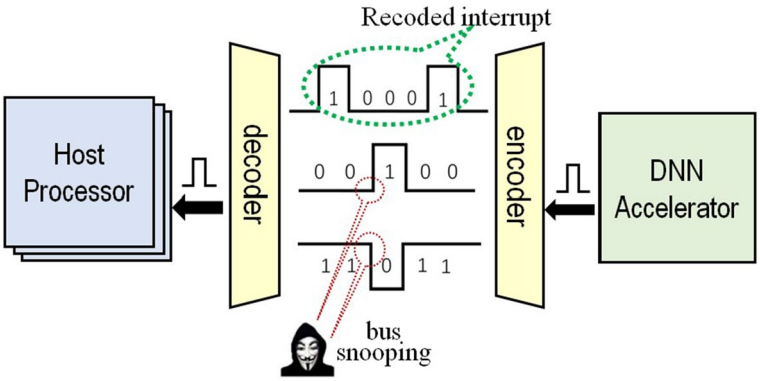

Fig. 5 Interrupt signal coding by using the mB/nB algorithm

3.5 Message authentication code

In order to achieve security isolation between the DNN accelerator platform and the user DNN model files, we build an authentication mechanism by using the SHA-256 algorithm to verify the uploaded DNN file integrity and make the authentication. HMAC-SHA256 is a hash-based message authentication code using the SHA-256 algorithm. Compared with the traditional message digest, HMAC can not only guarantee the data integrity, but can also ensure that the DNN model files come from legitimate users.

In Nacc-Guard, the SHA-256 algorithm uses the DNN weight file to generate a message authentication code HMAC. This process is shown in (5) and (6). In Equation (5) and (6), H denotes the SHA-256 function and m is the binary bit data of a DNN weight file.

The HMAC mechanism can make the accelerator computing platform have a secure channel between the accelerator SoC and off-chip users. To be specific, before the SoC host processor fetches DNN weight files from off-chip users, the off-chip user should first calculate a message authentication code HMAC-1 according to the pre-distributed key-2 and the weight file data. This HMAC-1 will be sent to the SoC host processor together with the DNN model weight file. After file transmission, the host processor will calculate another HMAC-2 according to the shared key-2 and compare it with HMAC-1. If a hacker inserted poisoned data or malicious programs into the DNN model file, an error will be reported in the verification process before DNN model inference. The warning will be issued and be handled by the host processor. As a result, this malicious or modified DNN file will be forbidden and the accelerator will deny service. Furthermore, this message authentication code verification scheme can also avoid weight data loss or error during data transmission. The whole process is illustrated in Figs. 6 and 7.

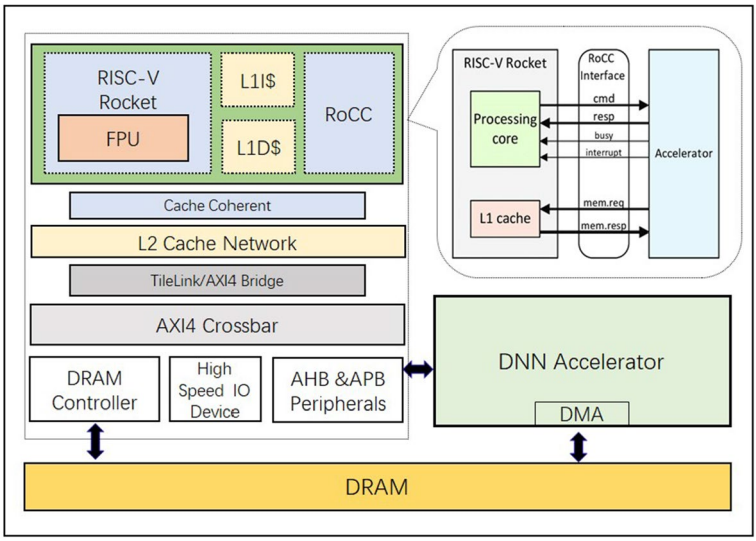

Fig. 6 Architecture of RISC-V Rocket and DNN accelerator; a RoCC coprocessor is used for accelerating HMAC generation

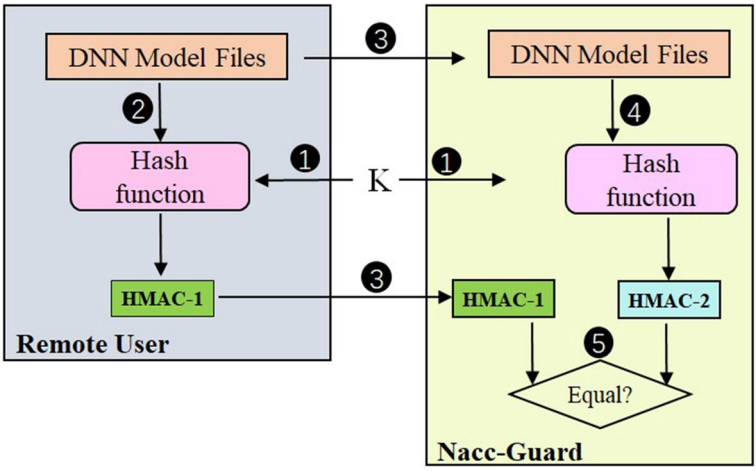

Fig. 7 Verification procedure of HMAC in Nacc-Guard

4 Evaluation

4.1 Experiment setup

Nacc-Guard is developed in the SIMD [44] and NVDLA [45] open source DNN accelerators at RTL. The SIMD accelerator is coupled with an ARM Cortex ®-A9

and NVDLA is coupled with an ARM Cortex®-A9 and RISC-V Rocket host processor. The EDA tool Vivado (version 2019.1) is used to synthesize and implement this lightweight accelerator prototype. The correctness of Nacc-Guard functionality has been verified on runtime platform Zedboard (SIMD) and ZCU-102 (NVDLA) on DNN models of YOLOv2 and ResNet-18. Performance evaluation is made using ModelSim SE and DNN accelerator simulator MAESTRO [46]. When not specifically stated, the DNN accelerators are all evaluated under a 100 MHz clock frequency. The detailed information of the two DNN accelerators is shown in Table 2.

4.2 Hardware overhead

The ARM + SIMD and RISC-V Rocket + NVDLA DNN accelerator SoC platforms are synthesized and implemented on Zedboard and VC709 SoCs. The detailed on-chip hardware resource consumption of BRAMs (36 Kb Block-RAM Blocks), FF (CLB Flip-Flops), LUT (Look-Up Tables), and DSP (18x25 MACCs DSP Slices) are calculated before and after Nacc-Guard is deployed in the two DNN accelerator platforms. The result is shown in Figs. 8 and 9. From this result, we can clearly see that Nacc-Guard architecture results in negligible on-chip resource overhead in the SIMD and NVDLA accelerators.

Furthermore, we also compared the hardware resource consumption of the AES-based encryption method (AES-128, ECB) in ARM + SIMD and RISC-V Rocket + NVDLA heterogeneous computing SoCs. The experimental results show that Nacc-Guard stream cipher-based encryption achieves a 3× hardware overhead reduction compared with AES-based encryption for the two DNN accelerators.

4.3 Performance evaluation

An excellent security environment should bring a small performance overhead to the original DNN accelerator platform. To this end, we evaluate the Nacc-Guard

Table 2 The accelerator platform configuration information

DNN accelerator (SIMD) [44]	
Clock	100 MHz
MAC units (INT-16)	120
Buffer Size (KB)	220 (Ping-Pong buffer)
Memory bandwidth	400 MB/s
NVDLA accelerator [45]	
Clock	100 MHz
MAC units (INT-8)	256
Buffer Size (KB)	128
Memory bandwidth	400 MB/s

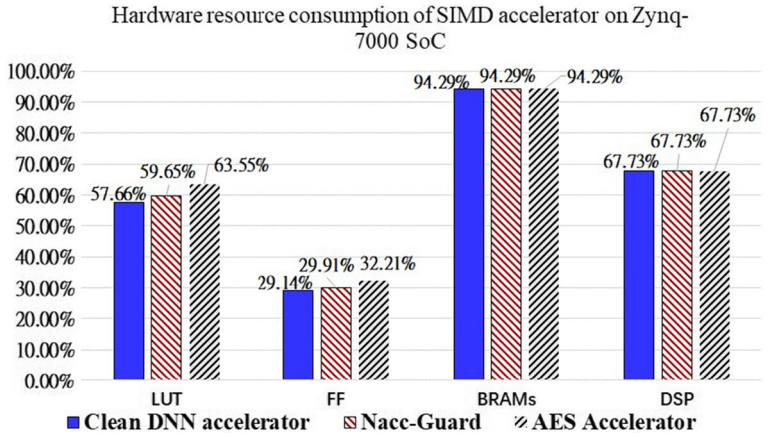

Fig. 8 Hardware resource consumption before and after Nacc-Guard is implemented in SIMD on ZedBoard (percentage of total SoC resource)

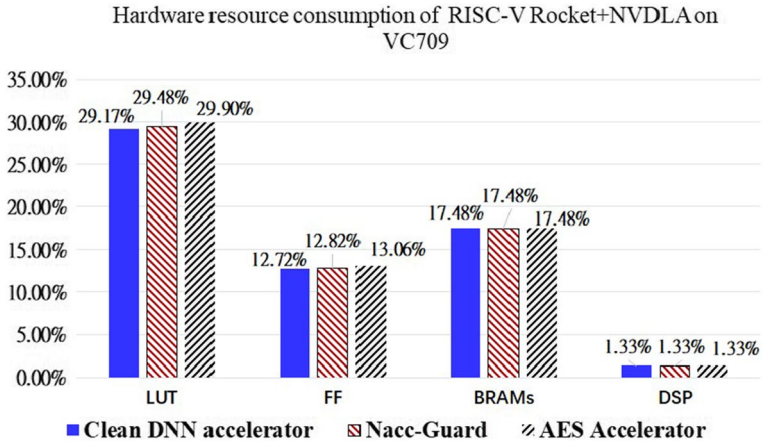

Fig. 9 Hardware resource consumption before and after Nacc-Guard is implemented in NVDLA on VC709 (percentage of total chip resource)

performance overhead through the VGG, ResNet50, and GoogLeNet deep neural network models.

First, runtime experiments on Zedboard (SIMD) and ZCU-102 (NVDLA) show that Nacc-Guard can successfully ensure private DNN inferences. Then, the cycle accurate simulations are made in ModelSim SE and the DNN accelerator simulator MAESTRO [46]. This experimentation shows that the performance overhead in Nacc-Guard mainly comes from buffer data en/decryption and interrupt signal recoding. Interrupt signal coding using the mB/nB coding algorithm (1B/4B) will bring 3 cycles latency for each interrupt signal, and this means each DNN layer will have a 3 clock cycle delay. The performance results for the VGG, ResNet50,

and GoogLeNet DNN models are presented in Fig. 10. As a comparison, the AES(ECB)-128 en/decryption model is also embedded in the original DNN accelerator. Through the analysis of the results, we can see that Nacc-Guard has a significantly better performance than the conventional AES en/decryption scheme in DNN accelerators. It achieves a 3.63× performance improvement on average.

4.4 Power consumption

Finally, to evaluate the energy consumption of Nacc-Guard, the SoC power is estimated from the implemented netlists in the Vivado EDA tool 2019.1 after on-chip synthesis and implementation (place and route). Due to the fact that ARM has a stable and solidified hardware micro-architecture in Zynq, we chose ARM + NVDLA on Xilinx ZCU102 as the accelerator power evaluation platform. The increment rate of SoC dynamic power consumption with the accelerator working frequency is shown in Fig. 11. We find that the Nacc-Guard encryption engine introduces less energy overhead than AES-based secure DNN accelerators. From this figure, we can also see that Nacc-Guard has a more robust power performance in terms of working frequency. The increasing rate of power with frequency is lower than AES.

5 Conclusion

In this paper, we proposed a lightweight DNN accelerator architecture named Nacc-Guard. This architecture aims to defend against memory Trojan attacks and neural network bit-flip attacks. Experimental results show that Nacc-Guard has a 3.63× performance improvement with 3× less hardware resource cost than AES-based en/decryption. Furthermore, the experimental evaluation shows that Nacc-Guard has

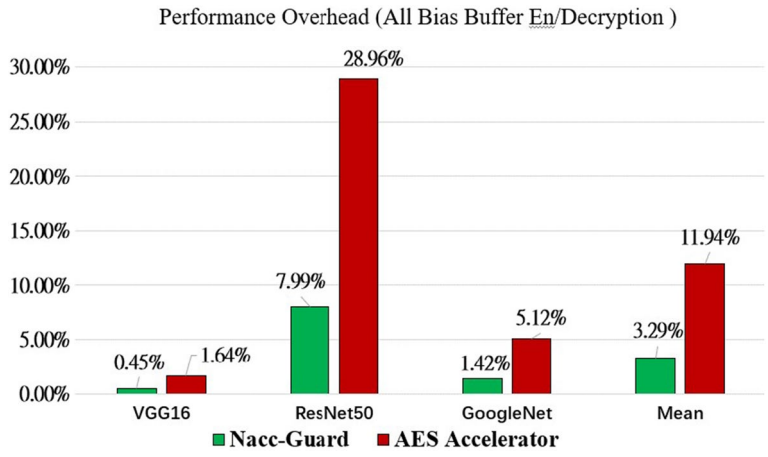

Fig. 10 Performance of Nacc-Guard and AES-based accelerator on different DNN models

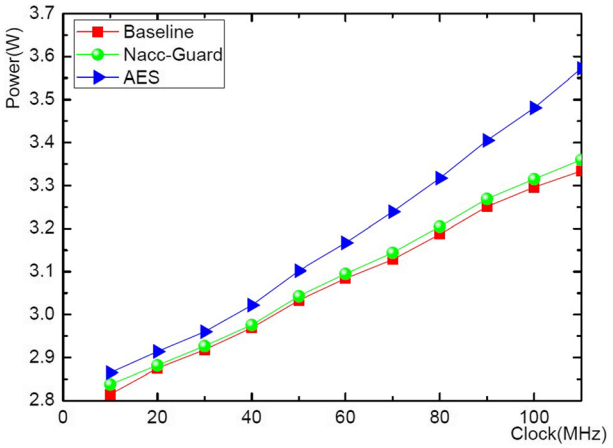

Fig. 11 Dynamic power consumption of original accelerator, Nacc-Guard, and AES-based accelerator in different working frequency

low and robust power consumption. Cryptoanalysis shows that the system is secure as long as the key is secured.

For the next step and in our future work, we will deploy Nacc-guard in transformer accelerators and we will deploy it in the autonomous vehicles, such as a tesla car.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have declared that they have no conflicts of interest that are relevant to the content of this work.

References

1. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
2. Foote Keith D (2017) A brief history of deep learning
3. MLK (2019) Brief History of Deep Learning from 1943 to 2019 [Timeline]
4. Zou Z, Shi Z, Guo Y, Ye J (2019) Object detection in 20 years: a survey. arXiv:1905.05055
5. Ham TJ, Jung SJ, et al. (2020) A3: accelerating attention mechanisms in neural networks with approximation. In: HPCA, pp 328–341
6. Mishra R, Gupta HP, Dutta T (2020) A survey on deep neural network compression: challenges, overview, and solutions. arXiv:2010.03954
7. Chen T, Ji B, Shi Y, Ding T, Fang B, Yi S, Tu X (2020) Neural network compression via sparse optimization. arXiv:2011.04868
8. Xu S, Huang A, Chen L, Zhang B (2020) Convolutional neural network pruning: a survey. In: Proceedings of the 39th Chinese Control Conference, pp 7458–7463
9. Blalock D, Gonzalez Ortiz JJ, Frankle J, Guttag J (2020) What is the state of neural network pruning?. arXiv:2003.03033

10. Molchanov P, Tyree S, Karras T, Aila T, Kautz J (2016) Pruning convolutional neural networks for resource efficient inference. arXiv:1611.06440
11. Mittal S, Gupta H, Srivastava S (2021) A survey on hardware security of DNN models and accelerators. J Syst Archit 117:1–30
12. Hu X, Zhao Y, Deng L, Liang L, Zuo P, Ye J, Lin Y, Xie Y (2020) Practical attacks on deep neural networks by memory trojaning. In: Proceedings of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems
13. Zuo P, Hua Y, Liang L, et al. (2020) Sealing neural network models in secure deep learning accelerators. Arxiv: 2008.03752
14. Rakin AS, He Z, Fan D (2019) Bit-flip attack: crushing neural network with progressive bit search. In ICCV, pp 1211–1220
15. Cai Q, et al. (2018) Curriculum adversarial training. In: IJCAI
16. Wang X, Hou R, Zhu Y, et al. (2019) NPUFort: a secure architecture of DNN accelerator against model inversion attack. In: Proceedings of the 16th ACM International Conference on Computing Frontiers
17. Hashemi H, Wang Y, Annavaram M (2021) DarKnight: an accelerated framework for privacy and integrity preserving deep learning using trusted hardware. In: MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture
18. Stinson DR (2005) Cryptography: theory and practice, 3rd edn. Chapman Hall Press, London
19. Xu C, Lai S (2021) Accelerating TEE-based DNN inference using mean shift network pruning. In: 17th EAI International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, pp 25–41
20. Capra M (2020) Hardware and software optimizations for accelerating deep neural networks: survey of current trends, challenges, and the road ahead. IEEE Access 8:225134–225180
21. Chen T, Zidong D, Sun N et al (2014) DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. ACM SIGARCH Comput Archit News 42:269–284
22. Graphcore (2019) Introduction to the IPU architecture. [Online]. Available: https://www.graphcore.ai/. Accessed 6 Aug 2019
23. Cloud TPU, Accessed: 2018-01-31. [Online]. Available: https://cloud.google.com/tpu
24. Tearing Apart Google's TPU 3.0 AI coprocessor, Accessed: 2018-05-15. [Online]. Available: https://www.nextplatform.com/2018/05/10/tearing-apart-googles-tpu-3-0-ai-coprocessor
25. NVIDIA (2018) Hardware architectural specification
26. Drumond M, Coulon L, Pourhabibi A et al. (2021) Equinox: training (for free) on a custom inference accelerator. In: MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture
27. Fengbin T, Zihan W, Yiqi W, et al (2022) A 28 nm 15.59uJ/token full-digital bitline-transpose CIM-based sparse transformer accelerator with pipeline/parallel reconfigurable modes. In: 2022 IEEE International Solid-State Circuits Conference (ISSCC)
28. Shan L, Zhang M, Deng L, et al. (2016) A dynamic multi-precision fixed-point data quantization strategy for convolutional neural network. In: CCF National Conference on Computer Engineering and Technology, pp 102–111
29. Lin D, Talathi S, Sreekanth V (2016) Fixed point quantization of deep convolutional networks. In: International conference on machine learning
30. Qiu J, Wang J, Yao S, et al. (2016) Going deeper with embedded FPGA platform for convolutional neural network. In: Proceedings of the 2016 ACM/SIGDA international symposium on field-programmable gate arrays
31. Cong J, Fang Z, Lo M, et al. (2018) Understanding performance differences of FPGAs and GPUs. In: 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)
32. Wang X, Hou R, Zhao B, et al. (2020) DNNGuard: an elastic heterogeneous DNN accelerator architecture against adversarial attacks. In: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems
33. [Online]. Available: https://www.thesslstore.com/blog/block-cipher-vs-stream-cipher/
34. Cannière C (2006) Trivium: a stream cipher construction inspired by block cipher design principles
35. Gan Y, Qiu Y, Leng J, Guo M, Zhu Y (2020) Ptolemy: architecture support for robust deep learning. In: 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp 241–255

36. Rouhani BD, Samragh M, Javaheripi M, Javidi T, Koushanfar F (2018) Deepfense: online accelerated defense against adversarial deep learning. In: 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp 1–8
37. Szegedy C, Zaremba W, Sutskever I, et al. (2013) Intriguing properties of neural networks. ArXiv: 1312.6199
38. Zhang Y, Jia R, Pei H, Wang W, Li B, Song D (2020) The secret revealer: generative model-inversion attacks against deep neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 250–258
39. Hu X, Liang L, Deng L, Li S, Xie X, Ji Y, Ding Y, Liu C, Sherwood T, Xie Y (2019) Neural network model extracion attacks in edge devices by hearing architectural hints. arXiv:1903.03916
40. Sun G, Cong Y, Dong J, et al. (2020) Data poisoning attacks against federated learning systems. arXiv:2004.10020
41. Liu Z, Ye J, Hu X, et al. (2020) Sequence triggered hardware trojan in neural network accelerato. In: 2020 IEEE 38th VLSI Test Symposium (VTS)
42. Lyu Y, Mishra P (2018) A survey of side-channel attacks on caches and countermeasures. J Hardw Syst Secur 2:33–50
43. Rakin AS, He Z, Li J, et al. (2021) T-BFA: targeted bit-flip adversarial weight attack. In: Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence
44. [Online]. Available: https://github.com/dhm2013724/yolov2_xilinx_fpga
45. [Online]. Available: https://github.com/nvdla/
46. [Online]. Available: https://maestro.ece.gatech.edu/