计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20190109
56(6): 1161-1169, 2019

面向深度学习加速器的安全加密方法

左鹏飞[1,2,3] 华 宇[1,2] 谢新锋[3] 胡 杏[3] 谢 源[3] 冯 丹[1,2]
[1](武汉光电国家研究中心(华中科技大学) 武汉 430074)
[2](华中科技大学计算机学院 武汉 430074)
[3](加州大学圣芭芭拉分校 加利福利亚圣芭芭拉 美国 93106)
(pfzuo@hust.edu.cn)

A Secure Encryption Scheme for Deep Learning Accelerators

Zuo Pengfei[1,2,3], Hua Yu[1,2], Xie Xinfeng[3], Hu Xing[3], Xie Yuan[3], and Feng Dan[1,2]
[1](*Wuhan National Laboratory for Optoelectronics* (*Huazhong University of Science and Technology*), *Wuhan* 430074)
[2](*School of Computer*, *Huazhong University of Science and Technology*, *Wuhan* 430074)
[3](*University of California at Santa Barbara*, *Santa Barbara*, *California*, *USA* 93106)

Abstract With the rapid development of machine learning techniques, especially deep learning (DL), their application domains are wider and wider and increasingly expanded from cloud computing to edge computing. In deep learning, DL models as the intellectual property (IP) of model providers become important data. We observe that DL accelerators deployed on edge devices for edge computing have the risk of leaking DL models stored on them. Attackers are able to easily obtain the DL model data by snooping the memory bus connecting the on-chip accelerator and off-chip device memory. Therefore, encrypting data transmitted on the memory bus is non-trivial. However, directly using memory encryption in DL accelerators significantly decreases their performance. To address this problem, this paper proposes COSA, a COunter mode Secure deep learning Accelerator architecture. COSA achieves higher security level than direct encryption and removes decryption operations from the critical path of memory accesses by leveraging counter mode encryption. We have implemented COSA in GPGPU-Sim and evaluated it using the neural network workload. Experimental results show COSA improves the performance of the secure accelerator by over 3 times compared with direct encryption and causes only 13% performance decrease compared with an insecure accelerator without using encryption.

Key words machine learning; accelerator; edge device; security; bus snooping attack

摘 要 随着机器学习特别是深度学习技术的飞速发展,其应用场景也越来越广,并逐渐从云计算向边缘计算上扩展.在深度学习中,深度学习模型作为模型提供商的知识产权是非常重要的数据.发现部署在边缘计算设备上的深度学习加速器有泄露在其上存储的深度学习模型的风险.攻击者通过监听深度学习加速器和设备内存之间的总线就能很容易地截获到深度学习模型数据,所以加密该内存总线上的数据传输是非常重要的.但是,直接地在加速器上使用内存加密会极大地降低加速器的性能.为了解决这个问题,提出了一个有效的安全深度学习加速器架构称作 COSA.COSA 通过利用计数器模式加密不仅提高了加速器的安全性,而且能够把解密操作从内存访问的关键路径中移走来极大地提高加速器性能.

收稿日期:2019-03-01;**修回日期**:2019-04-11
基金项目:国家自然科学基金项目(61772212,61821003)
This work was supported by the National Natural Science Foundation of China (61772212, 61821003).
通信作者:华宇(csyhua@hust.edu.cn)

在 GPGPU-Sim 上实现了提出的 COSA 架构,并使用神经网络负载测试了其性能.实验结果显示 COSA 相对于直接加密的架构提升了 3 倍以上的性能,相对于一个不加密的加速器性能只下降了 13%左右.

关键词 机器学习;加速器;边缘设备;安全;总线监听攻击

中图法分类号 TP302

在过去的 10 年里,机器学习和边缘计算是 2 个被广泛关注的领域.一方面,机器学习特别是深度学习技术[1]最近取得了巨大的突破,在图片识别[2-3]、语音识别[4-5]、游戏[6]等领域实现了比人类更高的准确率和效率.另一方面,随着边缘设备计算能力和存储容量的不断提升,和不需要向云端远程传输数据等优势,边缘计算[7]也非常受欢迎.因此,越来越多的应用和产品开始探索和结合机器学习和边缘计算两者的优势,最典型的应用是无人驾驶汽车[8]和智能手机[9].通过在其上部署深度学习加速器,无人驾驶汽车可以进行本地计算来实时地对当前路况做出反应,而不用连接到高延迟的远程控制中心.

在深度学习中,深度学习模型是其核心,被认为是需要重点保护的机密信息[10].这是因为深度学习模型是由模型提供商花费大量财力训练出来的,是他们知识产权的重要组成部分和他们业务的核心竞争力来源.另外,深度学习模型通常是使用用户的隐私数据训练出来的,深度学习模型的泄露还会披露训练数据的相关隐私信息.更重要的是,当恶意的敌手获取到深度学习模型后,他可以有针对性地实施对该深度学习应用的对抗攻击[11].

然而我们发现相对于云计算,边缘计算上的机器学习系统产生了新的安全问题.这是因为边缘计算设备上的机器学习系统更容易地被物理访问.因此,边缘设备上的机器学习系统会受到基于物理访问的攻击,例如总线监听攻击(bus snooping attack)[12-13].攻击者通过监听深度学习加速器和内存之间总线上传输的数据,就很容易地获取到该机器学习系统中的深度学习模型.所以加密深度学习加速器的内存总线中传输的数据非常重要.一种直接的方法是,在深度学习加速器的片上加入加密(如 AES)逻辑电路[12].当加速器需要写数据到内存中时,先加密该数据再将密文写入;当加速器需要从内存中读数据时,先读密文数据到片上再解密.然而,这种直接加密的方法极大地降低了深度学习加速器的性能.主要是因为深度学习加速器(如 GPU)通过高并发来获得高的性能,其性能对内存访问吞吐量极其敏感.内存总线的吞吐量在 160 GBps 左右[14],但目前最好的硬件实现的加密电路吞吐量只有 10 GBps 左右[15],从而大大限制了数据访问的吞吐量和降低了加速器性能.

为了解决这个问题,本文提出一个高效的安全机器学习加速器架构 COSA,其利用计数器模式加密把解密操作从内存访问的关键路径中移走.具体地,当加速器在内存中读取密文数据的同时,COSA 使用一个秘钥、内存行地址和该行的计数器通过 AES 电路计算出一个临时数据.当密文数据从内存中读取出来后,COSA 把密文数据和这个临时数据进行异或操作就解密出了明文数据.可见.加密操作和内存读操作并行地执行了,只有一个异或操作在内存访问的关键路径上,大大提升了加速器性能.本文的主要贡献包括 3 方面:

1) 发现了边缘设备上深度学习加速器易受基于物理访问攻击的安全问题,并分析了加密对加速器性能的影响.

2) 提出了一个高效的安全深度学习加速器架构 COSA,通过把解密操作从内存访问的关键路径中移走来提升加速器性能.

3) 在 GPGPU-Sim 上实现了提出的 COSA 架构,并使用公开的神经网络负载进行测试,验证了本文提出方法的有效性.

1 背景及动机

本节首先介绍深度学习加速器的相关背景,然后介绍边缘设备上深度学习加速器的安全性问题,最后讨论一个保证深度学习加速器安全性的直接加密方法.

1.1 边缘计算

由于位于网络边缘的设备上产生的数据越来越多,数据的传输速度成为了云计算模式的瓶颈.巨大的带宽开销和高的响应延时极大地降低了云计算的效率.例如一辆自动驾驶汽车每秒产生的数据高达 1 GB[16],其很难通过云计算来处理数据做实时的决策.为了解决这个问题,边缘计算的概念被提出[7,17].边缘计算利用边缘设备上的硬件资源来直接在本地

进行计算,这将大大减少了数据处理延时和边缘设备的能耗开销.

1.2 深度学习加速器

深度学习技术在许多领域得到了广泛地应用,如图片识别、语音识别、自然语言处理、游戏等,甚至可以实现比人类更高的准确率和效率.但是,深度学习的高效性依赖于极大的计算开销.深度学习加速器的出现提高了边缘设备的算力,使得深度学习在边缘计算中的使用成为了可能.

GPU 是一种最常用的神经网络加速器[18],由于其强大的并行处理能力非常适用于神经网络中的大量矩阵/向量乘运算和浮点运算.FPGA 也常被用作神经网络加速器[19],由于其可编程的特性可支持灵活地设计新的硬件结构来匹配神经网络算法.另外还有一些专用的深度学习加速器产品,如 Google 的 TPU[20]、寒武纪的 Diannao[21]、麻省理工学院(MIT)提出的 Eyeriss[22]等.

大部分神经网络推理加速器都可以抽象为如图 1 所示的硬件架构.加速器片上会集成很多支持高并发计算的处理单元(processing element, PE),一块 cache 用来存储最近经常被访问的热数据.由于 cache 的容量很小,其不能存储全部的深度学习模型.整个的深度学习模型会存储在片外的大容量 DRAM 内存中,加速器需要通过 GDDR 总线连接片外的 DRAM.GDDR 总线是一种针对 GPU 等加速器专门设计的总线,其具有比传统 CPU 的 DDR 总线更高的带宽,从而可以支持更高的内存访问吞吐量.

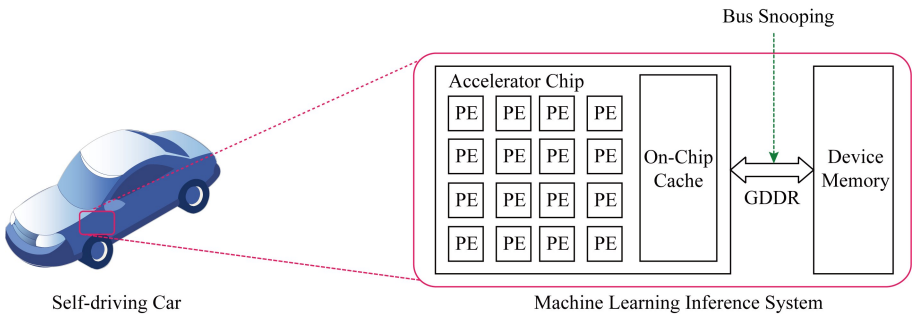

Fig. 1 The architecture of the deep learning inference accelerator on edge devices

图 1 边缘设备上深度学习推理加速器架构

1.3 安全问题和攻击模型

在深度学习中,深度学习模型作为模型提供商的知识产权是非常重要的数据.我们发现部署在边缘计算设备上的深度学习加速器有泄露其上存储的深度学习模型的风险.与云计算不同,边缘计算中的设备(如无人驾驶汽车)很容易被物理地访问.所以,边缘设备上的深度学习加速器就很容易受到基于物理访问攻击,如总线监听攻击[12].如图 1 所示,攻击者可以在 GDDR 内存总线上安装一个探听器,来截获片外内存与片上加速器之间传输的数据,进而获取到整个深度学习模型.另外,本文不考虑总线篡改攻击,其可以通过完整性校验技术抵御[23],与本文的研究工作是正交的.

1.4 直接加密

加密深度学习加速器的内存总线中传输的数据非常重要.如图 2 所示,一种直接的方法是,在深度学习加速器片上的内存控制器中加入加密逻辑电路,如高级加密标准(advanced encryption standard, AES)[24]逻辑电路.当内存控制器需要将一个内存行写回到片外 DRAM 内存中时,先将该内存行的数据加密再将密文写入;当加速器需要从内存中读数据时,先读密文数据到片上再解密出明文.

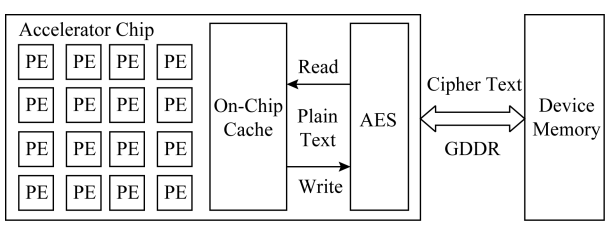

Fig.2 A secure accelerator architecture using direct encryption

图 2 一个使用直接加密的安全加速器架构

在这种直接加密(direct encryption)的架构中,每次内存读写都需要通过 AES 做加密或解密,并且加解密操作是在内存访问的关键路径上.然而,目前最好的硬件实现的加密电路吞吐量在 10 GBps 左右[15],其远低于内存总线的吞吐量(160 GBps 左右[14]).使得大量的读写请求在等待 AES 电路来做

加解密,大大增加了计算单元的等待周期从而降低了加速器的性能,如 3.2 节的实验评估所示.

2 COSA 安全加速器设计

本文提出一个高效的安全深度学习加速器架构(COunter mode Secure Accelerator for deep learning, COSA),其利用计数器模式加密(counter mode encryption)[23,25],把解密操作从内存访问的关键路径中移走来极大地提高安全加速器的性能,并且可以实现更高的安全性.本节首先介绍计数器模式加密方法,接着介绍如何在安全的深度学习加速器中有效地使用计数器模式加密,最后介绍 COSA 的实现细节.

2.1 计数器模式加密

加密内存行的一种简单的方法是用一个全局的秘钥(key)加密所有的内存行,如图 3(a)所示.但是,相同的内存行使用相同的秘钥总是生成相同的密文,那么攻击者简单地对比密文数据就可以知道哪些内存行的明文数据是一样的.因此,这种加密方法很容易受到字典攻击(dictionary attacks)和重播攻击(retry attacks)[26-27].一种改进的方法是,使用一个全局秘钥联合该内存行的地址(line addr)来加密该内存行的数据,如图 3(b)所示.那么,位于不同内存行地址的相同数据会被加密成不同的密文,可以避免不同内存行之间的字典攻击和重播攻击.但是,在相同行地址先后写入的相同数据的密文却是相同的.因此,这种加密方法会有对同一行地址的字典攻击和重播攻击的风险.

一种更安全的方法是计数器模式加密[25],其给每个内存行分配一个计数器(counter),并使用一个全局的秘钥、该内存行的地址和该内存行的计数器加密该内存行的数据,如图 3(c)所示.对于每次写,该内存行的计数器加一.那么,即使在相同行地址先后写入相同的明文数据,其产生的密文也是不同的,从而实现了高的安全性.

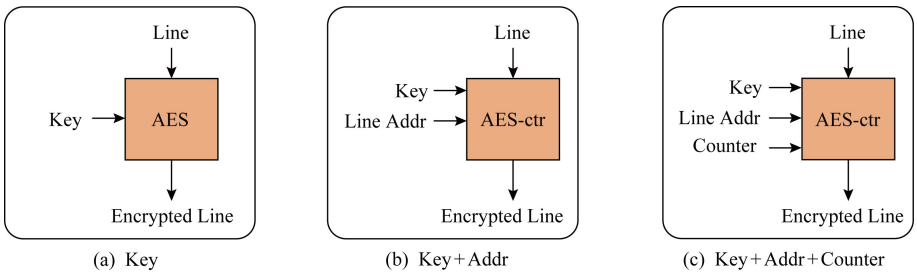

Fig. 3 Different encryption methods

图 3 不同的加密方法

2.2 COSA 安全加速器架构

通过使用计数器模式加密,本文提出了一种高效的安全深度学习加速器架构 COSA,如图 4 所示.COSA 在片外内存中分配了一块空间用来存储所有内存行对应的计数器值,并在加速器片上增加了一个计数器 cache(counter cache)来存储最近经常被访问的内存行的计数器值.COSA 并不是用 AES 电路逻辑直接对内存行的数据进行加解密,而是使用一个全局秘钥、内存行地址和内存行计数器通过 AES 电路逻辑计算出一个一次性的填充数据(one-time pad, OTP).该填充数据和内存行的密文数据进行简单的异或操作就生成了明文数据;同样地,该填充数据和内存行的明文数据进行异或操作就生成了密文数据[12].

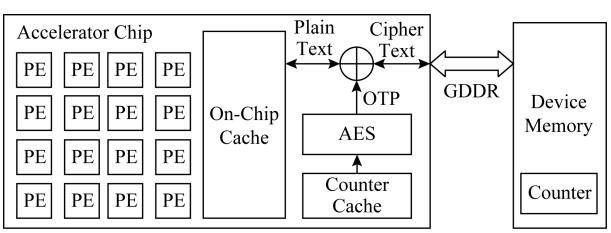

Fig. 4 The COSA hardware architecture

图 4 COSA 安全深度学习加速器硬件架构

这样做的好处是可以并行地执行 AES 计算和数据读操作.具体地,当一个读请求到 DRAM 内存中取数据的同时,COSA 使用该内存行的地址、计数器值和全局秘钥通过 AES 电路逻辑计算出一个一次性填充数据(OTP).等密文内存行被读取到片上时,COSA 直接将该填充数据(OTP)和密文内存行异或就解密出了明文数据.可见,AES 计算步骤被成功地从读操作的关键路径上移走,只有异或操作的延迟被加在了读操作的关键路径上,如图 5 所示:

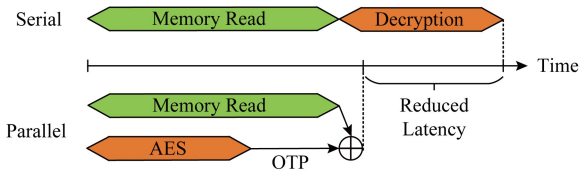

Fig. 5 The parallel encryption in COSA

图 5 COSA 中的并行加密操作

需要注意的是,只有当读请求将要访问的内存行所对应的计数器缓存在片上计数器 cache 中时,AES 计算才能和 DRAM 内存中的读操作并行执行.如果其所对应的计数器没有缓存在计数器 cache 中,COSA 必须到内存中先读取计数器来形成 OTP,再异或数据密文来解密.这样的话,AES 计算延迟依然在读请求的关键路径上.

为了解决这个问题,COSA 增强 DRAM 内存中计数器存储的空间局部性并通过预取来提高计数器 cache 的命中率.具体地,每个内存行对应的计数器是个 8 B 的值,而加速器(如 GPU)的内存行大小通常是 128 B.COSA 把连续的 16 个物理地址所对应内存行的计数器存储在一个 128 B 的内存行中.那么,当一个计数器被访问时,连续的 16 个内存行的计数器会在一次内存访问中被预取到计数器 cache 中.接下来如果有对剩余 15 个计数器的访问,都会命中 cache,从而提升了计时器 cache 的命中率.

由于每个 128 B 的内存行需要 8 B 的计数器,COSA 只需要 6.25%(=8 B/128 B)的内存空间来存储全部的计数器.

2.3 实现细节

本节介绍安全深度学习加速器 COSA 的实现细节,如图 6 所示. 位于片上的加速器会对片外的 DRAM 内存发送读写操作请求,这些读写请求都会先放入如图 6 中下半部分所示的 L2 到 DRAM 队列(L2→DRAM Queue)中.另一方面,片外的 DRAM 内存会返回读请求的数据到片上加速器(图 6 中的上半部分所示).

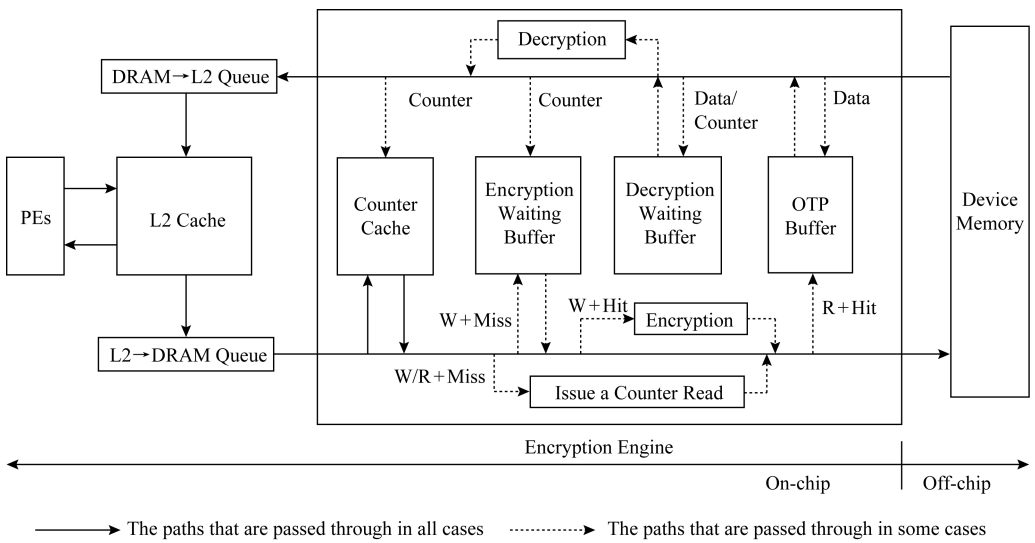

Fig. 6 The workflow of counter mode encryption in COSA

图 6 COSA 中计数器模式加密工作流程

首先介绍片上加速器向片外内存发送请求的流程.发送的请求包括读(R)和写(W)两种请求,不管是读请求还是写请求都需要先查询计数器 cache.这是因为写请求需要获取对应的计数器值来加密写入的数据,读请求需要获取对应的计数器值来并行地执行 AES 计算作解密.根据请求的类型和计数器 cache 是否命中,COSA 需要分别处理4 种情况:

1) 写请求+cache 不命中(W+Miss).当一个内存写请求对应的计数器值不在计数器 cache 中时,COSA 生成一个对该计数器值的读请求到片外内存,并把该写请求加到解密等待缓存(encryption waiting buffer)中.如果解密等待缓存已满的话,该写请求会等待,直到解密等待缓存非满时再被加入.

2) 写请求+cache 命中(W+Hit).当一个内存写请求对应的计数器值在计数器 cache 中时,COSA 使用该计数器通过 AES 计算出一个 OTP 来加密该写请求,然后把该写请求发送到片外内存.

3) 读请求+cache 不命中(R+Miss).当一个读请求对应的计数器值不在计数器 cache 中时,COSA 生成一个对该计数器值的读请求到片外内存,然后

继续把该读请求发送到片外内存.

4) 读请求+cache 命中(R+Hit).当一个读请求对应的计数器在计数器 cache 中时,COSA 把该读请求发送到片外内存,同时,使用它的计数器通过 AES 计算出一个 OTP,并把该 OTP 放到 OTP 缓存中.

再介绍 COSA 处理片外内存返回到片上的读请求的流程.返回的读请求可能是数据行或计数器行,所以分为 2 种情况:

1) 数据行(data).当读请求返回的是数据行时,COSA 首先查询 OTP 缓存中是否有其对应的 OTP.如果有,则直接解密该数据行,并删除 OTP 缓存中对应的该数据的 OTP;如果没有,继续查询计数器 cache 中是否存在其对应的计数器.如果计数器 cache 命中,则解密该数据行;如果 cache 不命中,COSA 将该数据行放入解密等待缓存(decryption waiting buffer)中.

2) 计数器行(counter).当读请求返回的是计数器行时,COSA 先查询解密等待队列中是否有其对应的数据行.如果有,则解密该数据行并将解密后的数据发送到 DRAM→L2 队列.COSA 继续查询加密等待队列中是否有其对应的数据行.如果有,则加密该数据行,并将该被加密的数据行写入到内存.最后,COSA 将该计数器写入到计数器 cache 中.

3 性能评测

为了不失一般性,本文使用 GPU 一种最常用的深度学习加速器来实现提出的 COSA 方案.然而,本文提出的安全加密方法 COSA 实现在深度学习加速器的内存控制器中,从而也适用于其他的深度学习加速器,如基于 FPGA 和 ASIC 的加速器.

3.1 实验配置

1) GPU 配置.通过使用 GPGPU-Sim[28],实现了本文提出的安全深度学习加速器方案 COSA.GPGPU-Sim 是一个被广泛使用的周期级的商用 GPU 性能模拟器,其可以支持运行 CUDA[29] 和 OpenCL[30] 程序.在测试中,使用 GPGPU-Sim 模拟的 GPU 的配置如表 1 所示.在 GPU 中的每个内存控制器中加入了一个 16 阶流水线的 AES 电路,其加密块大小是 128 bit,对一个内存行(128 B)的加解密延迟是 40 cycles[31].COSA 中 encryption waiting buffer,decryption waiting buffer 和 OTP buffer 的大小都为 16.

Table 1 Configuration Parameters of the Simulated GPU

表 1 GPU 硬件配置参数

Configuration Items	Configuration Parameters
Number of Shader Cores	28
Warp Size	32 threads
SIMD Pipeline Width	8
Number of Threads per Core	1 024
Number of CTAs per Core	8
Number of Registers per Core	16 384
Shared Memory per Core	16 KB
Constant Cache Size per Core	8 KB
Texture Cache Size per Core	64 KB
L1 Cache	16 KB
L2 Cache	128 KB×6
GDDR3 Memory Timing (cycle)	CCD=2, RRD=8, RCD=12, RAS=25, RP=10, RC=35, CL=10, WL=7, CDLR=6, WR=11
Bandwidth per Memory Module	8 Byte/cycle
DRAM Request Queue Capacity	32 requests
Memory Controller	×6, FRFCFS

2) 数据集.使用 SPASS2009-Benchmarks[28] 中的神经网络负载(neural network, NN)来评估 COSA 的性能.该 NN 负载使用卷积神经网络(convolutional neural network)来识别手写数字.已经训练好的神经网络权重数据和待识别的手写数字输入已经预先放在片外 DRAM 内存中.该 NN 负载允许同时识别多个手写数字来提高并行性,总共识别来自美国国家标准技术局(National Institute of Standards Technology, NIST)数据库的 28 个手写数字.

3) 实验对比.实验对比了一个没安全保证的神经网络加速器(Un-encr),和一个如 1.3 节描述的采用直接加密(direct encryption)的神经网络加速器.

3.2 实验结果

实验测试了不同架构的深度学习加速器的每周期执行指令数(instructions per cycle, IPC),如图 7 所示.由于本文提出的安全深度学习加速器 COSA 的片上部署了一个计数器 cache,不同的计数器 cache 大小会影响其性能.所以实验也评估了 COSA 在不同计数器 cache 大小情况下的性能.

从实验结果中发现,相对于一个没有加密的加速器(Un-encryption),使用直接加密(direct encryption)的加速器性能大大地降低了,其 IPC 降低高达 80%,如图 7 所示.这是因为在使用直接加密

的加速器中加密操作是在内存访问的关键路径上,加密电路逻辑相对非常低的吞吐量严重影响了加速器的性能.本文提出的 COSA,相对于直接加密方法极大地提高了加速器的性能.随着每个内存控制器中计数器 cache 的大小从 8 KB 增加到 64 KB,COSA 相对于直接加密方法实现了 3.1 倍到 5.0 倍的性能提升.当计数器 cache 的大小为 64 KB,COSA 只比不加密的安全加速器慢 13%左右.这是因为 COSA 利用计数器模式加密把很多读请求的解密步骤从内存访问的关键路径中移走了.COSA 会产生一些额外的内存访问来读取计数器,但是这不会显著影响性能,因为内存有足够高的带宽.另外,更大的片上计数器 cache 容量可以实现更高的性能提升,这是由于更大的计数器 cache 有更高的 cache 命中率,如图 8 所示.

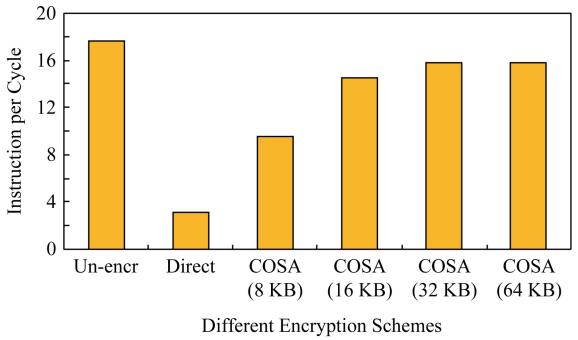

Fig. 7 IPCs of accelerators with different encryption schemes

图 7 不同加密方法的加速器 IPCs

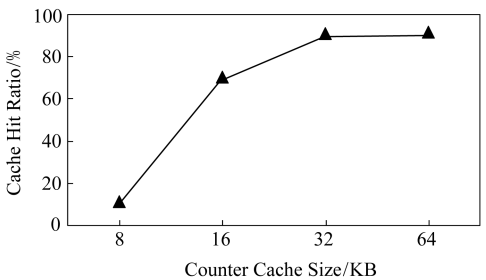

Fig. 8 The hit ratio of the counter cache with different sizes

图 8 不同 cache 大小的计数器 cache 命中率

4 相关研究工作

本节首先从算法和架构层面讨论机器学习模型安全性相关的研究工作,然后讨论内存加密相关的研究工作.

1) 模型提取攻击.①算法层面.现有的一些工作把机器学习模型看做为一个黑盒子,然后提出一些算法通过观察该黑盒子的输入和输出数据来提取或推测模型相关的信息.Tramer 等人[32]假设机器学习系统输出的分类标签的置信分数(confidence scores)和模型结构(model architecture)是已知的,并证实可以通过置信分数来推测模型参数相关的信息.Oh 等人[33]假设模型结构是未知的,然后提出一个元模型(metamodel)方法来获取模型结构相关的信息.Wang 等人[34]提出了一个获取机器学习模型的超参数(hyperparameter)的方法.超参数一般被用于平衡目标函数中的正则项和损失函数,知道超参数可以进一步帮助攻击者获取模型参数.②系统和架构层面.现有的一些方法通过利用操作系统或体系架构的一些信息来推测机器学习模型的相关信息.Naghibijouybari 等人[35]提出利用操作系统中的侧信道信息,如内存分配 API、GPU 硬件的性能计数器和时序等,来推测神经网络模型的相关信息如神经元的个数.Hua 等人[36]提出利用机器学习加速器体系架构中的侧信道信息如内存访问模式,来推测神经网络的结构信息.

与这些现有的攻击相比,本文关注的是一个更加危险的攻击也就是总线监听攻击.通过总线监听攻击,攻击者不用推测就可以直接获取到机器学习模型的所有数据,包括网络结构和权重参数等.为了抵御这种攻击,本文提出了一个有效的安全加速器架构 COSA.

2) 内存加密.内存加密被广泛地应用在安全 CPU 系统中[12-13,23,26-27].由于 CPU 系统中的 DDR 内存总线带宽(10~30 GBps)和 AES 加密逻辑的带宽(10 GBps)差不多,所以在 CPU 系统中使用内存加密并不会带来太大的性能损失.然而,机器学习加速器系统如 GPU 的 GDDR 总线带宽远高于 AES 加密逻辑的带宽,并且加速器的性能是带宽敏感的,内存加密会极大地影响加速器的性能.本文提出了 COSA 安全加速器架构来有效地解决这个问题.

5 总 结

我们发现部署在边缘计算设备上的深度学习加速器有泄露其上存储的深度学习模型的风险.攻击者通过监听深度学习加速器和设备内存之间的总线就能很容易地截获到模型数据.为了解决这个问题,本文提出了一个有效的安全深度学习加速器架构

称作 COSA.COSA 通过利用计数器模式加密不仅提高了加速器的安全性,并且能够把解密操作从内存访问的关键路径中移走来极大地提高加速器性能.我们在 GPGPU-Sim 上实现了提出的 COSA 架构,并使用神经网络负载测试了其性能.实验结果显示 COSA 相对于直接加密的架构提升了 3 倍以上的性能,相对于一个不加密的加速器性能只下降了 13%左右.

参 考 文 献

[1] LeCun Y, Bengio Y, Hinton G. Deep learning [J]. Nature, 2015, 521(7553): 436-444

[2] Krizhevsky A, Sutskever I, Hinton G. ImageNet classification with deep convolutional neural networks [C] //Proc of the 25th Int Conf on Neural Information Processing Systems (NerIPS). New York: ACM, 2012: 1097-1105

[3] He Kaiming, Zhang Xiangyu, Ren Shaoqing, et al. Deep residual learning for image recognition [C] //Proc of the IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2016: 770-778

[4] Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning [C] //Proc of the 25th Int Conf on Machine Learning (ICML). New York: ACM, 2008: 160-167

[5] Xiong W, Droppo J, Huang Xuedong, et al. The microsoft 2016 conversational speech recognition system [C] //Proc of the 2017 IEEE Int Conf on Acoustics Speech and Signal Processing (ICASSP). Piscataway, NJ: IEEE, 2017: 5255-5259

[6] Silver D, Huang A, Maddison C J, et al. Mastering the game of go with deep neural networks and tree search [J]. Nature, 2016, 529(7587): 484-489

[7] Shi Weisong, Cao Jie, Zhang Quan, et al. Edge computing: Vision and challenges [J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646

[8] Kim J, Kim H, Lakshmanan K, et al. Parallel scheduling for cyber-physical systems: Analysis and case study on a self-driving car [C] //Proc of the ACM/IEEE 4th Int Conf on Cyber-physical Systems. New York: ACM, 2013: 31-40

[9] Song I, Kim H J, Jeon P B. Deep learning for real-time robust facial expression recognition on a smartphone [C] //Proc of the 2014 IEEE Int Conf on Consumer Electronics (ICCE). Piscataway, NJ: IEEE, 2014: 564-567

[10] Shokri R, Shmatikov V. Privacy-preserving deep learning [C] //Proc of the 22nd ACM SIGSAC Conf on Computer and Communications Security (CCS). New York: ACM, 2015: 1310-1321

[11] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets [C] //Proc of the Int Conf on Neural Information Processing Systems (NerIPS). New York: ACM, 2014: 2672-2680

[12] Zuo Pengfei, Hua Yu, Zhao Ming, et al. Improving the performance and endurance of encrypted non-volatile main memory through deduplicating writes [C] //Proc of the 2018 51st Annual IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2018: 442-454

[13] Zuo Pengfei, Hua Yu. SecPM: A secure and persistent memory system for non-volatile memory [C] //Proc of the 10th USENIX Workshop on Hot Topics in Storage and File Systems. Berkeley, CA: USENIX Association, 2018: 1-7

[14] Navarro C A, Hitschfeld-Kahler N, Mateu L. A survey on parallel computing and its applications in data-parallel problems using GPU architectures [J]. Communications in Computational Physics, 2014, 15(2): 285-329

[15] Mathew S K, Sheikh F, Kounavis M, et al. 53Gbps native GF(24)2composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors [C] //Proc of the IEEE Symp on VLSI Circuits (VLSIC). Piscataway, NJ: IEEE, 2010: 169-170

[16] Finnegan M. Self-Driving cars will create 2 petabytes of data, what are the big data opportunities for the car industry? [OL]. (2016-12-07) [2019-02-26]. http://www.computerworlduk.com/news/data/boeing-787screate-half-terabyte-of-data-per-flight-says-virgin-atlantic-3433595/

[17] Mao Yuyi, You Changsheng, Zhang Jun, et al. A survey on mobile edge computing: The communication perspective [J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358

[18] Chetlur S, Woolley C, Vandermersch P, et al. CUDNN: Efficient primitives for deep learning [J]. arXiv preprint arXiv:1410.0759, 2014

[19] Zhang Chen, Li Peng, Sun Guangyu, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks [C] //Proc of the 2015 ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2015: 161-170

[20] Jouppi N P, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit [C] //Proc of the 2017 ACM/IEEE 44th Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2017: 1-12

[21] Chen Tishi, Du Zidong, Sun Ninghui, et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning [C] //Proc of the 19th Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS). New York: ACM, 2014: 269-284

[22] Chen Y H, Krishna T, Emer J, et al. 14.5 Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks [C] //Proc of the 2016 IEEE Int Solid-State Circuits Conf (ISSCC). Piscataway, NJ: IEEE, 2016: 262-263

[23] Yan C, Englender D, Prvulovic M, et al. Improving cost, performance, and security of memory encryption and authentication [C] //Proc of the 33rd Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2006: 179-190

[24] Daemen J, Rijmen V. The Design of Rijndael: AES-the Advanced Encryption Standard [M]. Berlin: Springer Science & Business Media, 2013

[25] Lipmaa H, Rogaway P, Wagner D. CTR-mode encryption [C] //Proc of the 1st NIST Workshop on Modes of Operation. Gaithersburg, MD: NIST, 2000: 1-4

[26] Awad A, Manadhata P, Haber S, et al. Silent shredder: Zerocost shredding for secure non-volatile main memory controllers [C] //Proc of the 21st Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS). New York: ACM, 2016: 263-276

[27] Young V, Nair P J, Qureshi M K. DEUCE: Write-efficient encryption for non-volatile memories [C] //Proc of the 20th Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS). New York: ACM, 2015: 33-44

[28] Bakhoda A, Yuan G L, Fung W W, et al. Analyzing CUDA workloads using a detailed GPU simulator [C] //Proc of the 2009 IEEE Int Symp on Performance Analysis of Systems and Software. Piscataway, NJ: IEEE, 2009: 163-174

[29] Nvidia Corporation. Nvidia CUDA C programming guide [J]. Nvidia Corporation, 2011, 120(18): 1-175

[30] Stone J E, Gohara D, Shi G. OpenCL: A parallel programming standard for heterogeneous computing systems [J]. Computing in Science & Engineering, 2010, 12(3): 66-73

[31] Liu Sihang, Kolli A, Ren Jinglei, et al. Crash consistency in encrypted non-volatile main memory systems [C] //Proc of the 2018 IEEE Int Symp on High Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2018: 310-323

[32] Tramèr F, Zhang F, Juels A, et al. Stealing machine learning models via prediction apis [C] //Proc of the 25th USENIX Security Symp. Berkeley, CA: USENIX Association, 2016: 601-618

[33] Oh S J, Augustin M, Schiele B, et al. Towards reverse-engineering black-box neural networks [J]. arXiv preprint arXiv:1711.01768, 2017

[34] Wang Binghui, Gong N Z. Stealing hyperparameters in machine learning [C] //Proc of the 2018 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2018: 36-52

[35] Naghibijouybari H, Neupane A, Qian Zhiyun, et al. Rendered insecure: GPU side channel attacks are practical [C] //Proc of the 2018 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2018: 2139-2153

[36] Hua Weizhi, Zhang Zhiru, Suh G E. Reverse engineering convolutional neural networks through side-channel information leaks [C] //Proc of the 2018 55th ACM/ESDA/IEEE Design Automation Conf (DAC). Piscataway, NJ: IEEE, 2018: No.4

Zuo Pengfei, born in 1992. PhD candidate. Student member of CCF. His main research interests include memory system and architecture, storage system, security and deep learning.

Hua Yu, born in 1978. PhD, professor, PhD supervisor. Distinguished member of CCF, senior member of ACM and IEEE. His main research interests include cloud storage systems, non-volatile memory, big data analytics, artificial intelligence hardware and software infrastructure, etc.

Xie Xinfeng, born in 1995. PhD candidate. His main research interests include computer architecture, programming language, and heterogeneous system.

Hu Xing, born in 1988. PhD. Her main research interests include emerging memory systems, neural network acceleration and security.

Xie Yuan, born in 1973. PhD, professor, PhD supervisor. IEEE Fellow. His main research interests include VLSI design, electronic design automation, computer architecture, and embedded systems design.

Feng Dan, born in 1970. PhD, professor, PhD supervisor. Distinguished member of CCF. Her main research interests include computer architecture, massive storage systems, and parallel file systems.