XCRYPT：利用忆阻器交叉阵列加速基于格的密码学

Sarabjeet Singh Xiong Fan Ananth Krishna Prasad Lin Jia Anirban Nag 犹他大学 罗格斯大学 犹他大学 犹他大学 乌普萨拉大学 sarab@cs.utah.edu leofanxiong@gmail.com ananth@cs.utah.edu lin.jia@utah.edu anirban.nag@it.uu.se

Rajeev Balasubramonian 犹他大学 rajeev@cs.utah.edu

Mahdi Nazm Bojnordi 犹他大学 mnbojnordi@gmail.com

Elaine Shi 卡内基梅隆大学 runting@gmail.com

arXiv:2302.00095v1 [cs.AR] 31 Jan 2023

摘要——本文论证了使用忆阻器交叉阵列加速基于格的后量子密码学（PQC）的可行性，并表明，这些天生具有容错能力的算法非常适合交叉阵列中的噪声模拟乘累加操作。我们比较了美国国家标准与技术研究院（NIST）第三轮基于格的PQC候选方案，并发现SABER不仅在传统系统上执行时处于领先地位，也适合借助交叉阵列加速。SABER是一种基于模舍入学习（Module-LWR）的方法，通过舍入执行模多项式乘法。我们将SABER中的多项式乘法映射到交叉阵列上，并表明，与近期硬件方案相比，模拟点积可带来 $1.7-32.5\times$ 的性能和能量效率提升。这一初步设计融合了多项最先进工作的创新成果——SABER算法，以及ISAAC中提出的忆阻器加速原理（用于深度神经网络加速）。随后，我们找出这一初步设计中的瓶颈，并引入多种额外技术以提升效率。这些技术彼此协同，尤其受益于SABER的二次幂模运算。首先，我们表明，SABER中一些在中央处理器平台上效果显著的软件技术，在基于交叉阵列的加速器中却无济于事。采用更简单的算法可进一步将效率提升 $1.3-3.6\times$。其次，我们利用SABER计算的特性，在交叉阵列中错开操作，并共享少量可变精度模数转换器，从而使效率最高提升 $1.8\times$ 。第三，为进一步减轻模数转换器的负担，我们提出一种简单的模拟移位加法技术，使效率提升 $1.3-6.3\times$。总体而言，我们的设计相比初步设计效率提升 $3-15\times$，比以往工作高出 $3-51\times$。最后，模拟运算更容易出错；我们表明，模LWR计算的近似特性可以掩盖大部分此类误差。我们还表明，这类硬件引入的误差不会破坏算法的安全性保证。

I. 引言

近期涌现出多种量子计算系统——IBM Q 系统 [1], 英特尔 Tangle Lake [2],Google Bristlecone [3], 和 IonQ [4] 这提高了整数分解和离散对数问题在不久的将来变得可解的可能性，从而使若干现代密码学原语过时 [5], [6]。这促使人们开始关注其他密码学假设，因为已知算法（Shor [7] 和 Grover [8]）无法在量子系统上轻易破解这些假设。过去几年，美国国家标准与技术研究院（NIST）已征集并筛选出多种抗量子算法 [9]。目前正在考虑多种适用于此类后量子密码学（PQC）的数学方法，包括基于格、多变量、基于哈希、基于同源以及基于编码的密码学 [5], [9], [10]。其中，基于格的密码学（LBC）因在多项指标上具备高效率和安全性，成为领先方案 [5], [10]。因此，对这些后量子密码算法进行硬件加速的研究刻不容缓，因为这有助于推动该领域转向更适合硬件加速的算法。即便算法仍在不断演进，这项研究也正逢其时，因为大多数算法变体所包含的基本原语和计算已逐渐趋同。包括 Google、Microsoft、Digicert 和 Thales 在内的多家公司已经在测试部署后量子密码学（PQC）的影响 [11]。

现代基础设施建立在基于云的部署之上，采用对称密钥加密和非对称密钥加密相结合的方式，确保传输中的数据安全。通常，通过基于非对称密钥加密的握手协议来建立私钥。随后，使用该私钥保护传输中的数据。然而，RSA/ECC 等非对称密钥加密方案容易受到量子攻击；近期提出的后量子密码学对应方案可能会取代它们。正如我们将在下文中介绍的，基于带误差学习（LWE）问题 [12] （及其变体）的流行轻量级密码方案依赖大规模矩阵和向量运算，给硬件带来了显著负担。近期的工作 [13]–[20]在现场可编程门阵列/图形处理器/专用集成电路上实现了基于格的密码学算法，并报告称延迟达到几十至数百微秒。经典密码学功能通常可在现代硬件上于数微秒内执行 [21], [22]；例如，Intel QuickAssist 技术使用专用密码加速器 [23]，可在 5 微秒内完成 RSA 解密操作。因此，基于格的密码学的相关指标远远无法满足现代部署的要求。当前若干 HTTPS 应用需要在网络服务器与客户端之间建立密钥（通常每秒处理数百万次请求），并使用非对称密钥密码学原语。采用软件实现且耗时数百微秒的后量子密码算法 [24], 将使服务时间至少增加一个数量级。为了取代

该场景中使用的后量子密码算法，例如 RSA [25]和 Diffie–Hellman [26], ，后量子密码算法必须显著降低延迟和能耗。虽然提高并行度可以降低延迟，但也可能在面积、数据搬运和能耗方面产生高昂代价。因此，探索能够使所有相关效率指标提升数个数量级的变革性新型硬件技术至关重要。若没有这些进步，下一代隐私保护型部署将无法实现——无论是进一步扩大计算规模的同态加密，还是进一步压缩资源限制的医疗物联网设备。

本文探讨了一项前景广阔的技术——在电阻式存储器中进行模拟计算，并将其作为构建新型架构的基础，以高效执行一系列与基于格的密码学相关的算法。尽管这类技术此前已用于 [27], [28] 实现深度神经网络中的计算，我们展示了基于格的密码学为进一步提升这种存内计算方法的效率带来了新的机遇。虽然存内模拟运算能够高效执行点积，但将高分辨率电流转换为数字值的代价很高。模数转换器（ADC）是必须加以控制的主要开销。模拟电路面临的另一项挑战是容易受到噪声影响。我们通过利用后量子密码算法变体中的以下特性来克服这些挑战：（i）计算可以容忍一定噪声；（ii）它们执行 2 的幂次模运算。

我们重点研究 SABER，这是美国国家标准与技术研究院（NIST）的入围方案之一，采用模-LWR 算法。借鉴以往深度神经网络加速领域的最佳实践，我们设计了一个初始方案，其性能优于此前基于 FPGA 的后量子密码加速工作。随后，我们分析这一初始方案，并通过软硬件协同设计，进一步将解密的计算密度提升9.5×（加密提升11.8×），能量效率提升 15.7×（加密提升3.7×）。这些改进得益于引入了适用于加速器的多项式乘法软件技术、用于降低模数转换器压力的交错计算，以及一种新颖的模拟移位加法技术。图 1 展示了每项创新技术的概要及其影响。本文提出的技术还可扩展至所有以多项式运算为基础操作的后量子密码算法，例如数字签名和同态加密。

我们的主要贡献如下：

- 我们借鉴已有的最佳实践，构建了一种用于后量子密码学加密和解密算法的基础存内架构 [27], [28]。该设计作为本研究的基准方案，其效率高于近期的后量子密码加速器。
- 我们展示了多项软件技术，例如分解多项式次数和智能调度，以提高模数转换器共享率；与基准方案相比，这些技术可将能量效率提高 2.4–2.7×，并将计算效率提高 4.7×。
- 我们提出一种新颖的技术，利用交叉开关阵列执行无需写入的模拟域移位加法运算，从而使

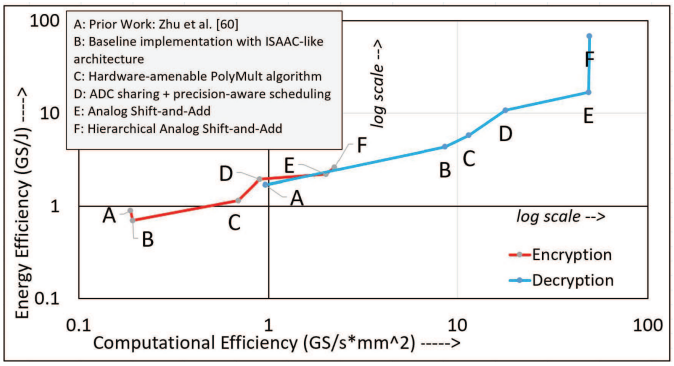


图 1：本文提出的技术及其影响概述。

我们能够在单元编程与模数转换器复杂度之间进行权衡。设计空间探索表明，计算效率和能量效率分别可提升至 2.7× 和6.3× 。

- 我们证明，基于格的方案能够容忍计算中由模拟器件/电路变化引入的少量误差。
- 总体而言，XCRYPT（基于交叉阵列的后量子密码加速器）在服务器部署中实现了 0.08 us 的解密延迟，在客户端部署中实现了 4 us 的加密延迟，芯片面积分别仅为 0.04 和 0.3 mm^2 。

II. 基于格的密码学背景

A. 带误差学习及其变体

鉴于基于格的方法广受欢迎，本文聚焦于基于格的密码学。格是 n 维空间中的一组点。每个点都是定义该格的一组基向量的整数线性组合。假设基向量很大，给定一个点，要确定距离它最近的格点，计算开销很高（需要非多项式时间）。直观而言，高维空间以及大型向量的众多可能线性组合，使得这一问题难以求解。最近向量问题 [29] 及其他相关问题（最短向量问题、最短线性无关向量问题）的困难性，构成了许多密码学原语的基础，例如公钥加密、数字签名和同态加密 [30]。

某些基于格的密码方案的安全性基于带误差学习（LWE [12]）问题的困难性。标准LWE问题指出，给定一个随机选取的、模 q的整数 n 维向量 $\vec{s}$，要从关于 $\vec{s}$ 的 $m \geq n$ 个近似随机线性方程中恢复$\vec{s}$是困难的。换句话说，我们从一个秘密值 $\vec{s}$ 开始；将 $\vec{s}$ 与 m 个随机向量相乘，并加入一个小误差 e_i，从而生成$\vec{b}$（公钥）。LWE方案的具体实现会因参数空间 (n,q,m) 以及用于对 $\vec{s}$ 和 e 进行采样的分布而异。

在环上带误差学习中， [31],整数样本被多项式样本取代。这里的主要计算是多项式

乘法（多项式乘法），可以使用数论变换高效实现。数论变换是快速傅里叶变换的一种变体。不过，与标准学习带误差相比，环上带误差学习可能会损害安全性 [32], [33]。为了在保留环上带误差学习所带来效率提升的同时，提供更高的安全性保证，人们提出了模学习带误差 [34] 。它使用一个由多项式构成的 l 维向量。通过采用舍入操作，而不是用随机生成的误差进行填充，可以进一步提高效率。这种变体称为舍入学习（模舍入学习） [35]。美国国家标准与技术研究院第三轮候选方案中有两个采用了模结构：Kyber *[34]*采用模学习带误差，SABER *[36]* 采用模舍入学习。

B. SABER

美国国家标准与技术研究院征集密钥封装机制（密钥封装机制）和数字签名算法。SABER [36] 是密钥封装机制中的入围方案。SABER 密钥封装机制由三种算法组成：密钥生成、加密和解密。

密钥生成使用基于 SHAKE-128 [37] 的伪随机数生成器确定一个由多项式组成的矩阵 A。通过从中心二项分布中采样，生成一个由多项式组成的秘密向量 $\vec{s}$。公钥由矩阵种子和取整后的乘积 $\vec{b} = A^T\vec{s}$组成。

$\vec{}$加密会生成一个新的秘密 s′，并将消息 m（一个系数为 $\in\{0,1\}$ 的多项式）添加到公钥 b 与 s′的内积 $\vec{}$ $\vec{}$中，从而形成密文的第一部分。第二部分通过对乘积 As′进行取整来隐藏加密秘密$\vec{}$。

第三种算法是解密，它使用私钥$\vec{}$s从密文的两个部分中提取编码的消息。

总体而言，SABER 执行 24 次多项式乘法、14 次多项式模运算和 8 次多项式舍入操作——每次多项式乘法包含 256^2 次整数乘法，模运算和舍入操作则对每个系数执行。SABER 在计算中加入了一些常数，因此可以用简单的位移代替舍入操作。

在基于模格的密码学中，多项式乘法对整体性能起着关键作用 [38]。在我们的实验中，我们发现多项式乘法内核占用了> 90%的执行时间。在实现多项式乘法时，数论变换的渐近时间复杂度最快，为 O(n logn)，但要求 q 为素数，这又会导致模运算的复杂度不容忽视。另一方面，SABER 选择了 2 的幂作为 q，从而加快了模运算（只需舍弃最高有效位）。由于数论变换并不可行，SABER 采用 Toom-Cook-4 算法 [39] 将每个 256 次多项式乘法降为 7 个 64 次多项式乘法，然后再使用 Karatsuba 算法将它们进一步降为 32 次多项式乘法。这一选择（以及对 AVX2 的支持）使多项式乘法的占比达到 57%，SABER 的性能超过了采用更快的基于数论变换的多项式乘法器的实现（如 Kyber）。借助 AVX2 支持，SABER 可在数十微秒内完成计算 [36]。与其他基于格的方案的现场可编程门阵列/专用集成电路实现相比，SABER 的各种软硬件协同设计在使用更少资源的同时，实现了更低的延迟 [40]–[42], [43], [44]。此外，

表 I：美国国家标准与技术研究院第三轮基于格的密码学方案比较

后量子密码方案	密文/明文比率	Haswell 周期		
		密钥生成	封装	解封装
Kyber-768 [34]	37.00	85K	112K	108K
SABER [36]	34.00	101K	125K	129K
NTRU [46]	40.03	307K	48K	67K

SABER 的密文开销在候选方案中属于最低之列。我们在表 I 中比较了 3 种基于格的PQC候选方案及其 NIST安全等级1 密钥封装机制实现。鉴于 SABER 在速度以及较低的硬件和密文复杂度方面具有优势，我们选择 SABER 作为硬件加速的目标。

除了多项式乘法之外，使用 SHAKE-128 生成伪随机数也是一项开销很大的操作。众所周知，作为 SHAKE-128 核心的 Keccak核心在硬件平台上效率很高 [41]。我们发现 Keccak 函数占执行时间的 $< 30\%$， [41]也证实了这一点。此外，Keccak核心可以加速 [45]，并与其他算术运算流水线化。因此，在这项工作中，我们聚焦于基于格的密码学的主要瓶颈：多项式乘法。

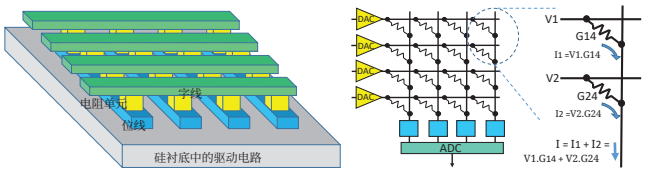
C. 现有的PQC实现

目前，已有工作着手在硬件中实现后量子密码算法 [10]。早期的大多数工作都聚焦于 FPGA 实现 [13]–[16]。为说明研究动机，我们在此讨论其中一些重要工作。

Nejatollahi等人 [10]的综述论文全面讨论了后量子密码学在硬件和软件方面的工作 [17]–[20], [43], [47]–[63] 。他们指出，大部分硬件开销可归因于乘法单元；高斯采样器单元也会带来不可忽视的硬件成本。Basu等人 [13] 使用高层次综合为后量子密码算法生成 FPGA电路。报告的延迟处于微秒范围内，其中相当大一部分硬件开销来自多项式乘法器（即使采用了 O(n logn) NTT算法）。 [13], [61] 在其软硬件协同设计实现中评估了多个美国国家标准与技术研究院候选算法。 [63] 则聚焦于多项式乘法的算法优化。需要注意的是，高效的 FPGA 实现需要采用一套专门技术，以减少瓶颈组件——查找表（LUT）的开销。近期研究强调了使用图形处理器 [17] 或采用专用集成电路设计 [18]–[20]。我们将在第 IV-C 节中将基本设计与其中许多方案进行比较。

D. KEM部署的影响

在现代云服务中，每分钟都有数百万新客户端请求服务，而每个客户端都需要使用密钥封装机制建立对称密钥。SABER 等量子安全密钥封装机制例程的复杂度显著高于传统例程，从而降低了服务质量。尽管本文重点关注密钥封装机制，但密钥封装机制加速所带来的益处也将适用于其他基于多项式乘法的应用。



(a) 电阻式交叉开关阵列的物理视图 (b) 交叉开关阵列的逻辑视图 (c) 乘积累加

图 2：电阻式交叉开关阵列（Xbar）电路及其执行模拟点积运算的方式 [27], [77]

CRYSTALS-DILITHIUM [64]（美国国家标准与技术研究院最终入围方案）等量子安全数字签名方案基于多项式乘法。基于格的密码学也是同态加密（HE）的基础。FV/BFV [65], [66] 等流行的同态加密方案（由 CryptoExperts [67] 提供的 FV-NFLlib 和 Microsoft [68] 提供的 SEAL 等多个库实现），以及 GSW [69], 、CKKS [70], 和 TFHE [71]，都将信息编码在多项式中。我们的工作可以扩展到同态加密应用，其中多项式运算同样是关键瓶颈。此外，当前的同态加密方案使用噪声预算；当预算耗尽时，需要使用 SABER 或类似 SABER 的方案对密文进行解密和加密（这也是本文的目标应用）。Cheetah [72] 展示了使用同态加密提供安全机器学习服务的方案，并且每一层之后都需要对每个输出神经元进行解密/加密，从而增加了密钥封装机制方案的贡献。对同态加密和数字签名部署的深入分析将留待未来工作。

III. 加速器背景

过去十年间，工业界和学术界对电阻式存储技术投入了大量资源 [73]–[76]。电阻式存储单元利用其电阻值来存储信息。电阻式存储阵列具有若干优势：密度极高、非易失性，以及具有竞争力的读取延迟和能耗。但它们也面临一项主要挑战：写入操作成本高昂，因为这类操作消耗更多能耗和时间，并会造成器件磨损。经过多年研究，第一代商用电阻式存储产品近几年已问世 [76]。最近，多个研究团队 [27], [28], [77]–[79]发现，电阻式存储阵列可配置为执行模拟运算。这类存内处理技术有望实现高并行度和低数据搬运开销。

电阻式存储器阵列以交叉开关阵列的形式实现，即由存储单元组成的网格，如图 2a 所示。位于 X 维导线（字线）与 Y 维导线（位线）之间的是电阻式存储单元（具有可编程电阻的材料）。图 2a 展示了一个小型交叉开关阵列的物理布局。图 2b 展示了该交叉开关阵列的逻辑表示，而图 2c 则放大展示了其中一条位线上的运作过程。如果将电压 V_1 和 V_2 施加到前两条字线上，则每条位线上都会注入电流，其大小与相应行中每个存储单元的电导成正比。最后一条位线中的电流为 $V_1 \times G_{14} + V_2 \times G_{24}$,，其中 G_{ij} 表示单元 {i,j} 的电导。因此，每条位线中的电流都是两个向量点积的模拟表示——施加到字线上的电压向量，以及预先编程到一列存储单元中的电导向量。

此外，字线电压会广播至所有列；每一列都使用同一个电压向量、但采用不同的电导向量，独立并行地执行点积。基尔霍夫定律的基本方程被用于设计模拟向量-矩阵乘法电路：当向字线施加输入电压向量时，该电路可在单步内产生位线输出电流向量。对存储单元进行编程是一个耗时的过程。执行向量-矩阵乘法相当于进行一次交叉阵列读取操作，随后进行模拟感测；这一操作快得多（仅需 10^{-7} 秒量级）。先前的研究 [27], [28], [77] 表明，与等效的数字电路相比，这种模拟矩阵-向量乘法单元可将每次操作能耗降低数个数量级（具体取决于模拟感测的类型）。部分原因在于，计算是在“原位”进行的，也就是说，矩阵作为操作数无需搬运；此外，利用自然现象（基尔霍夫定律）即可完成复杂的乘法和加法运算。

每条位线输出的模拟信号都必须转换为数字值。模数转换（ADC）电路会消耗大量功率，而且功耗会随模数转换器分辨率的提高呈指数级增长。如何管理模数转换器分辨率和功耗，是充分发挥这项新兴技术潜力的一大挑战。[28], [78], [80] 也探索了其他电路。

近期研究 [27], [28], [79], [81]–[83]将这些模拟电阻电路用作深度神经网络（DNN）加速器的核心组件，而深度神经网络能够容忍计算中的少量噪声。这推动了多个项目的开展，其中包括商业初创公司的相关尝试 [84]–[87]。本文借鉴了这些先前架构的见解，并探讨密码学应用能否利用这项新兴技术。特别是，基于格的密码学（LBC）的近似特性，使其有望成为模拟电阻电路的“杀手级应用”。

近期的原位深度神经网络加速器凸显了将模拟信号转换为数字值所带来的巨大开销。ISAAC [27] 将计算分散到不同的时间和区域，以将模数转换器精度降至 8 位，从而使模数转换器开销降至功耗的 58% 和硅片面积的 31%。PRIME [28]采用感应放大器（SA）而非传统模数转换器，以缩小面积。然而，由于一个感应放大器一次只能判定一位，因此将 8 位信号转换需要 2^8 个周期。CASCADE [82] 将输出信号写入另一个忆阻器交叉阵列，以便在将信号转换为数字值之前执行移位与加法。这使模数转换器的调用次数减少了近 25×。不过，CASCADE 只有在存储单元编程开销较低时才有效；鉴于其需要 6 位分辨率和较低的写入电压，这种情况不太可能出现 [88], [89]。

IV. 忆阻器交叉阵列上的基于格的密码学

大多数关于轻量级密码加速的既有研究都聚焦于 FPGA 实现 [13], [40]–[42], [61], [62]，其延迟和能耗分别为 5us-1ms 和 10uJ-10mJ,

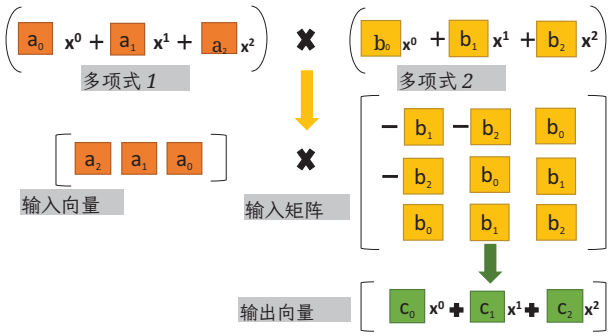


图3：将模多项式乘法表示为向量-矩阵乘法（以次数=3为例）。

近期文献表明，基于阻变存储器的计算能够提升点积密集型深度神经网络应用 [27], [28], [79], [82], [90] 的效率。下面我们将介绍，忆阻器交叉阵列如何成为适用于基于格的密码学的高密度、低能耗计算的理想载体。

我们首先通过改造专为深度神经网络设计的 ISAAC架构 *[27],* ，构建一个强大的基准方案。接下来，我们论证 SABER 的多项式乘法算法（Toom-Cook-*4* + Karatsuba 算法）并不总是适用于忆阻器交叉阵列，并探索不同的乘法算法。然后，我们展示利用 SABER 的 2的幂模运算实现的技术，以降低模数转换器的开销。在下一节中，我们将扩展该设计，使其能够在模拟域中执行移位与加法，从而进一步降低对模数转换器的需求。

与当今处理器中的 AES 加密引擎类似，XCRYPT加速器对程序员也是透明的，并会在软件需要发送安全消息时由硬件调用。需要注意的是，所提出的加速器基于忆阻器（阻变存储器），与互补金属氧化物半导体兼容，并且可以与处理器集成在同一芯片裸片上。

A. 将模多项式乘法表示为向量-矩阵乘法

从本质上说，NIST 后量子密码学中的几乎所有基于格的候选方案都要进行模多项式乘法。这里的“模”是指，两个 n 次多项式 $p_1, p_2 \in R_q = \mathbb{Z}_q[x]/(x^n+1)$ 相乘所得的多项式中，$x^i (i \geq n)$ 项将除以 $-x^n$，以使次数保持为 n。这就是多项式 $\mathbb{Z}_q[x]$ 除以 (x^n+1) 所表示的含义。多项式乘法要求 p_1 的每个系数都与 p_2 的每个系数相乘。以三次多项式为例，$p_1 = a_0x^0 + a_1x^1 + a_2x^2$ 和 $p_2 = b_0x^0 + b_1x^1 + b_2x^2$；$p_1 \times p_2 = (a_0b_0 - a_1b_2 - a_2b_1)x^0 + (a_0b_1 + a_1b_0 - a_2b_2)x^1 + (a_0b_2 + a_1b_1 + a_2b_0)x^2$。如图 3 所示，可以将其看作向量 (a_2, a_1, a_0) 与一个矩阵相乘，该矩阵的列分别为 $(-b1, -b_2, b_0)$、$(-b_2, b_0, b_1)$、(b_0, b_1, b_2)。末项 x^2 对应的列为 (b_0, b_1, b_2)，其余各列则是该列向下逐项移位得到的版本，并在列首带有一个负值。

B. 方法论

在介绍我们提出的加速器 XCRYPT 之前，我们先说明方法论。这有助于

表II：32纳米下的 XCRYPT-Schoolbook（X-SB）参数。

组件	数量	功耗（uW）	面积（um^2）
交叉阵列 (128x128)	1	300	25
数模转换器 1bit	128	3.9	0.16
S+H 6bit	128	0.007	0.029
模数转换器 6bit	16	945	435
模数转换器 7bit	1	1365	628.33
总计（1 个阵列）		**123.13**	**7737.557**
X-SB = **1** 加密 + 1 个解密单元 = **2** x （48 个阵列）+（IR+OR+SA）		11.92 mW	0.743 mm^2

结合相关结果讨论 XCRYPT 的设计选择。我们采用了 ISAAC 架构中引入的许多基本构件 [27]，并采用了与该工作非常相似的评估方法。忆阻器交叉阵列（包括移位加法交叉阵列）的能耗和面积模型基于 Hu 等人 [77] 的模型。RRAM 单元模型源自 [88],，其写入延迟为 25 ns，写入能耗为 0.1 pJ/单元/位；我们使用 NVSim [91] 提取阵列级数据。读取延迟由模数转换器读出决定，因为交叉阵列的 RC 延迟通常小于 1 ns [77], [92], [93]。由于 RRAM 是一种新兴技术，研究界接受了许多面向不同实现方案的 RRAM 器件参数。我们已确认，所提出的技术不受这些参数值的影响，并且在广泛的参数范围内都能带来显著改进。RRAM 单元的读取能耗比写入能耗低四个数量级。移位加法、采样保持和 1位数模转换器电路的面积和能耗参数取自 ISAAC [27]。我们考虑了一种能效高、面积效率高的自适应模数转换器，其采样速率可达每秒十亿次采样 [94]。该自适应模数转换器包含三个主要组件——基于电荷的数模转换器、比较器和控制器。为了得到不同模数转换器位精度下的功耗和面积，我们对基于电荷的数模转换器的功耗和面积按指数缩放，对其余部分则按线性比例缩放。初始设计（X-SB，下一小节将予以介绍）的详细参数列于表II。需要指出的是，我们提出了多个 XCRYPT 版本，其交叉阵列/模数转换器尺寸各不相同，因此参数也有所不同。我们还根据 Chou等人 [82]提及的参数对 CASCADE 组件进行建模。我们修改了美国国家标准与技术研究院竞赛参赛方案 SABER [36] 的代码，以运行 XCRYPT 的各种设计特性。我们还扩展了代码，对 RRAM 单元方差进行建模，以计算该实现的噪声容限；相关内容将在第VI节中详细介绍。我们针对一百万组随机生成的密钥对进行仿真，用这些密钥对加密和解密 32B 明文，并报告解密失败率。我们采用两个关键指标来评估 XCRYPT 的效率：

1）计算效率（CE）：每平方毫米面积每秒执行的 1 位明文/密文操作数（Gbits/s × mm^2）。2）能量效率（EE）：每 1 J 能耗执行的 1 位明文/密文操作数（Gbits/J）。

我们考虑一种 SABER 变体，其后量子安全级别与 AES-192 相近。在 SABER 中，解密

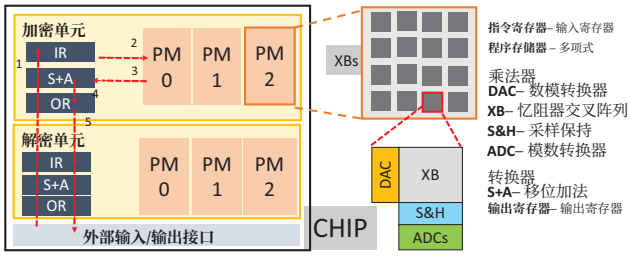


图 4：XCRYPT-SchoolBook 或 X-SB 架构。箭头表示加密路径。

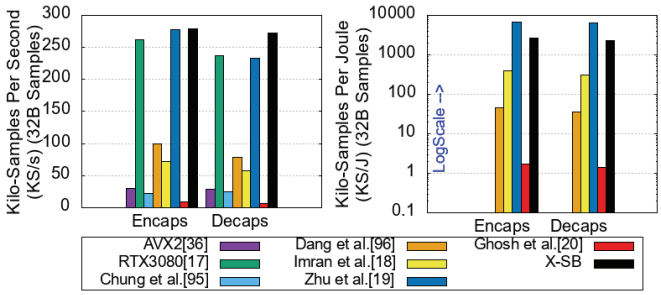


图 5：采用忆阻器交叉阵列实现的 SABER 与已发表的 SABER 实现方案对比。 [18]–[20]为专用集成电路。

需要进行向量-向量乘法，而加密则需要进行向量-向量乘法和矩阵向量乘法。一个向量包含 l 个 $= 3$ 次数为 $n = 256$的多项式，而一个矩阵由 $l^2 = 9$ 个多项式组成。因此，向量-向量乘法需要进行 3 次多项式乘法，随后将结果相加；矩阵向量乘法则需要进行 9 次多项式乘法，以得到一个由 3 个多项式组成的向量。SABER 的参数规定，矩阵向量乘法中多项式的系数为 $\log q = 13$ 位。进行乘法运算时，输入多项式会经过舍入，使其位数为 $\log p = 10$ 位。在所有多项式乘法中，¯其中一个多项式为 ¯s 或 s′，二者均为从中心二项分布中采样得到的秘密多项式，其每个系数需要 4 位。

C. 将 SABER 映射到忆阻器交叉阵列

在本小节中，我们首先遵循 ISAAC 深度神经网络设计的原则 [27]，构建一个基于交叉阵列的高性能基准方案来执行 SABER。由于阻变存储器的写入操作是该设计的瓶颈，我们提出了几种技术来减轻这一开销。随后，我们将此基准方案与此前的后量子密码学加速工作进行比较，并进一步解决基准方案中的瓶颈。

在云服务平台中，每秒都会处理多个客户端请求。为确保安全通信，客户端和服务器通过 SABER 的公钥加密（PKE）建立对称密钥（在非量子领域则使用 RSA）。因此，服务器会为每个客户端生成一次密钥。利用密钥对，服务器通过反复调用本文重点讨论的加密和解密算法，与客户端建立连接。

我们采用与 ISAAC [27]非常相似的方法论，将所需计算映射到交叉开关阵列。由于在服务器的大部分运行时间内，私钥 ¯保持不变，因此我们将其指定为要编码到交叉阵列单元中的操作数。我们需要 72 128×128 1 位交叉开关阵列来存储密钥 ¯及其移位版本（PolyMult 所需）。我们的设计中的一个瓦片（如图 4 所示）包含多个交叉开关阵列，可执行 3 次多项式乘法，以满足向量-向量乘法的要求。采用 ISAAC 的翻转编码方案，每一列都会产生 6 位电流。对于本研究，我们使用一种面积高效的模数转换器 [94]，可按 1 GSps 的频率转换采样值。为进一步降低面积开销，我们让一个模数转换器供一个交叉开关阵列的 8 列共用，因此交叉阵列读取周期时间为 8 ns。处理阻变存储器写入操作。如图 4 所示，我们设计了两种不同的瓦片——加密瓦片和解密瓦片。这是因为我们面向两种不同的部署场景——云服务器和客户端设备。在典型的握手协议中，服务器和客户端使用非对称密钥加密（例如后量子时代的 SABER）建立私钥。在协议过程中，客户端使用服务器的公钥加密，而服务器使用其私钥解密。因此，我们分别为客户端和服务器设计了 XCRYPT 加密/解密瓦片。

对忆阻器单元进行编程会带来较高的能耗、性能开销，并影响耐久性。通常，基于忆阻器的存储单元在发生故障前可承受 10^{12} 次写入操作。由于服务器仅定期更换私钥（以保证新鲜性），忆阻器的写入次数足够少，因此能够维持合理的使用寿命。例如，假设私钥每秒更新一次这一较为保守的情况，加速器寿命仍超过 3 万年。在客户端，每次建立新连接都需要进行一次新的忆阻器写入操作。不过，客户端通常建立的连接数量较少。即使每天建立 10^5 次连接，客户端加速器的寿命也将超过 2.7 万年。

与基准方案比较。由此，我们基于 ISAAC 的设计原则和高效的写入策略，定义了一个初始 XCRYPT 设计。我们将该架构称为 XCRYPT-SchoolBook（或 X-SB），因为它采用 Schoolbook 算法进行多项式乘法。我们在图 5 中将这一初始设计与现有的 SABER 实现进行比较，以展示这一基础设计在基于格的密码学加速方面优于以往工作。大多数以往工作报告封装/解封装的性能数据。这些函数的底层内核是加密/解密——封装会调用一次加密，而解封装会调用一次加密和一次解密。为进行公平比较，我们在图 5 中比较封装/解封装。不过，在全文中，我们报告底层独立内核（加密和解密）的结果。

中央处理器实现结果引自 SABER 论文 [36]，，该实现运行于一台支持 AVX2、主频为 3.1 GHz 的 Intel Haswell 机器上。Lee 等人 [17] 评估了图形处理器运行后量子密码学的能力（图中展示了 RTX3080 的结果）。Chung 等人 [95] 调整了 SABER 的参数，以支持数论变换，并利用快速的 16 位

数论变换硬件乘法器。Dang 等人 [96] 还提出了一种基于数论变换的全新 Saber 设计，并在现场可编程门阵列上进行了演示。Imran 等人 [18] 通过智能流水线和 SABER 内核间的逻辑共享，对专用集成电路进行了设计空间探索。他们的基准架构移植自一项 SABER 硬件现场可编程门阵列方案 [41]。Zhu 等人 [19] 采用 8 层级联的 Karatsuba 框架实现多项式乘法，并使用任务调度器将资源利用率降低了多达 90%。其布局后芯片的尺寸与我们的 X-SB 处理器大致相同，因此可以进行公平比较。Ghosh 等人 [20]展示了首个经硅验证的 Saber 专用集成电路实现。与现有工作相比，阻变存储器可实现 1.*17*×更高的吞吐量预算，如图 5 所示。尽管最先进的专用集成电路 [19] 在能量效率方面比 X-SB 高 2.4×，但本文展示的技术可使 XCRYPT 的性能超越 [19] 两个数量级。忆阻器交叉阵列利用基尔霍夫定律进行乘积累加，效率很高。模拟点积的 RC 延迟低于纳秒级，并以外围电路的数字转换频率进行读出。正如先前工作 [27],[77], [82], 所观察到的，该模数转换器电路是瓶颈，占总能耗的 70%–90%。

CryptoPIM [80] 是首篇提出针对基于格的方案进行存内计算加速的论文。该研究展示了相比现有现场可编程门阵列工作的吞吐量提升，但延迟显著更高。该工作低估了 RRAM 单元的编程开销。其方案中的每项操作都需要向 RRAM 单元写入数据，并假设延迟为 1.1ns。然而实际上，每个 RRAM 单元的编程耗时为 25ns [82]。即使假设采用并行写入驱动器，同时向整行或整列的存储单元写入数据，我们估计 CryptoPIM 的多项式乘法延迟仍比我们的基准方案 X-SB 设计高 5 个数量级，同时能耗也更高 13× 。此外，基于电阻式随机存取存储器的器件容易受到非理想特性的影响，而 CryptoPIM 忽略了这一点。CiM [97] 也针对 PolyMult 提出了存内计算方案，目标是同态求值（HE）；与公钥密码学相比，同态求值通常采用大得多的参数集。因此，他们专注于采用 ComputeCache实现 [98]的静态随机存取存储器基底。另一方面，XCRYPT 在面积高效的阻变存储器件上实现了参数规模更小的 SABER，并针对模数转换器瓶颈进行了设计。

D. 多项式乘法算法的影响

在本小节中，我们将探讨不同的乘法算法，以找出最适合交叉阵列加速的实现方式。与矩阵乘法类似，多项式乘法也能从各种低复杂度算法中受益 [10]。图 6 对这些算法的计算效率和能效进行了量化。我们先介绍用于相乘两个 256 次多项式的标准 $O(n^2)$ 复杂度 Schoolbook 算法，记为 X-SB。Karatsuba 算法（$O(n^{1.58})$ 复杂度）将一次 256 次多项式乘法分解为 3 次 128 次多项式乘法，从而将所需的 128×128 交叉阵列数量从 32 个减少到 24 个。如 X-

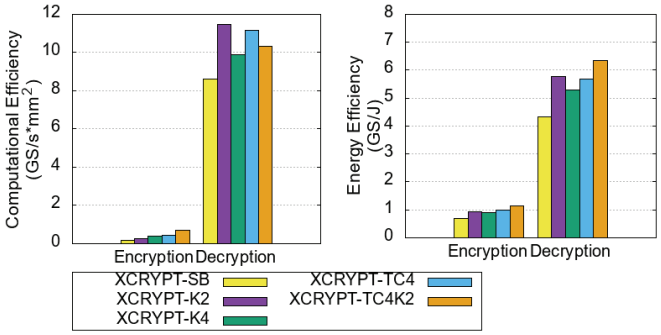


图 6：在忆阻器交叉阵列上运行 SABER 时的计算效率和能效，评估了多种多项式乘法算法。

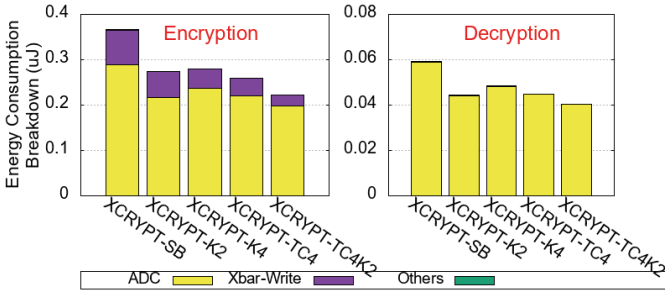


图 7：XCRYPT 在采用不同乘法算法进行加密和解密时的能耗分解。

图 7 中的 K2 所示，这按比例降低了模数转换器和交叉阵列写入能耗。请注意，由于固定密钥 ¯s，解密不需要交叉阵列写入，因此图 7 中几乎所有解密能耗都来自模数转换器。加密的计算效率也低于解密，因为它需要更多交叉阵列、更多输入周期，并且需要较长的初始交叉阵列编程步骤（占端到端加密延迟的 90%）。

X-K4 进一步将 3 次 128 阶乘法缩减为 9 次 64 阶多项式乘法，因此所需的 64×64 交叉开关阵列更小。虽然 X-K4 降低了阶数和每个交叉阵列的 ADC 采样数，但也增加了多项式乘法的总次数。因此，ADC 采样总数和交叉阵列写入操作都会增加，这就是 X-K4 在解密方面的改进幅度低于 X-K2 的原因。请注意，尽管交叉阵列写入操作更多，X-K4 映射到更小的 64×64 交叉开关阵列，因此写入能耗占比低于 X-K2。

ToomCook-4（$O(n^{1.4})$ 复杂度）将 256 次多项式乘法简化为 7 次 64 次多项式乘法。尽管 ToomCook-4 的渐近速度快于 K2，但它会生成更多多项式，因此需要更多 ADC 采样和更高的交叉开关阵列需求；这解释了在解密时 X-TC4 的表现为何不如 X-K2。不过，在加密时，TC4 表现良好，因为较小交叉开关阵列带来的优势（更低的写入延迟和能耗）超过了对交叉开关阵列数量的更高要求。SABER [36] 的软件实现使用 ToomCook-4 + Karatsuba-2（图中标记为 X-TC4K2）执行 21 次 32 次多项式乘法；由于交叉开关阵列写入延迟和能耗较低，它在加密时表现最佳。

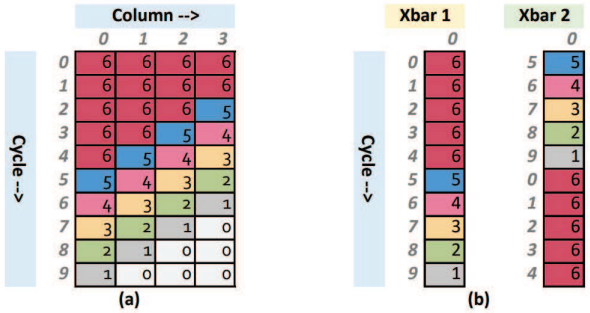


图 8：（a）对最终系数有贡献的位数。（b）重新排列交叉开关阵列中的计算，以便共享最高精度的模数转换器。

对于加密，X-TC4K2 相较于 X-SB 分别将计算效率和能效提升了 3.6× 和 1.6×。对于解密，写入操作不会发生，因此将次数降至 128 以下并不能提高效率。由此可见，交叉开关阵列加速器上理想的 SABER算法因任务而异；本文后续选择 X-K2 用于解密，选择 X-TC4K2 用于加密。

E. 使用模运算降低模数转换器开销

SABER 在忆阻器交叉阵列上的 X-K2 实现主要受模数转换器开销的限制。加密过程中，模数转换器占据 90% 的面积，并消耗 78% 的能耗。其余能耗来自交叉开关阵列写入。以往关于深度神经网络应用的研究也报告了类似的模数转换器开销 [27], [77], [82]。在 128×128交叉阵列中，采用 ISAAC 的翻转编码方案，需要使用 6 位精度模数转换器将模拟值转换为数字值 [27]。由于计算分布在一行中的多个存储单元上，并跨多个周期进行，因此必须经过适当移位后，将 6 位模数转换器的结果累加。SABER 的参数规定，输出多项式的系数必须经过模运算2^{10} 或 2^{13}，，使系数保持为 10 位或 13 位。因此，鉴于模运算，并非所有 6 位 ADC 采样中的所有位都会对最终输出产生贡献。与深度神经网络计算中最高有效位承载最多信息的情况不同，我们的计算中有些最高有效位并不起作用。例如，第 0 周期最低有效位列的全部 6 位都需要，因为它们会累加到输出系数的最低 6 位上。然而，第 9 周期最低有效位列的输出在左移 9 位后才累加到同一系数上，也就是说，采样 ADC 值的最低有效位才会对最终的 10 位结果产生贡献。图 8(a) 展示了解密过程中的有效位数。秘密 ¬的每个系数为 4 位，存储在一行的 4 个存储单元中，最低有效位存于第 0 列。因此，4 列的输出必须经过适当移位后相加。此外，不同周期的数值也要经过移位后相加，如算法 1 所示。从图中可以看出，所需位数及相应的模数转换器精度会因（周期+列）而异。对于向量-向量乘法，只有在（周期+列）≤4时才需要完整精度（6 位）。

算法 1：多项式乘法的伪代码，按系数计算，使用交叉开关阵列

每个系数使用交叉开关阵列进行运算，采用 crossbar

```
系数 = 0;
for(周期=0; 周期<10; ++周期)
    对于（列=0；列<4；  ++列）
        系数 +=位线值[周期][列]≪（周期+列）；
系数 = （系数）模 2^10
```

我们利用这种灵活性，对另一个交叉阵列的计算进行重新排序，使得在任一给定周期内，最多只有 1 个交叉阵列产生精度最高为 6 位的输出。这一点如图 8(b) 所示，其中展示了 2 个交叉阵列的第 0 列输出精度要求。对第二个交叉阵列的计算进行重新排序，使 5th 在第 0th 个周期执行。这种错峰安排确保每个周期只有一个交叉阵列产生 6 位输出。通过在交叉阵列之间共享 2 个模数转换器（一个 6 位，另一个 5 位），相比使用两个 6 位模数转换器的基准方案，我们降低了模数转换器开销。每个模数转换器负责一半的工作负载。进一步应用这一方案，可以在略微增加面积的同时降低能耗。5 位模数转换器的工作负载可拆分给一个 5 位和一个 4 位模数转换器。4 位模数转换器处理拆分后近 90% 的工作负载，从而节省能耗。通过让 10 个交叉阵列共享 5 位模数转换器（因为该转换器只需处理一小部分转换任务），可以降低额外模数转换器带来的面积开销。由于模数转换器开销会随精度呈指数增长，这种利用共享的低精度模数转换器的技术，使我们在解密中相较于 X-K2，将能效提高了 1.8× ，将计算效率提高了 1.5× ，如图 9 所示。由于该技术不会减少交叉开关阵列写入操作的次数，因此在加密中的收益较低：能效提高 1.7×，计算效率提高 1.08×。

这凸显了选择 2 的幂作为模数的重要性，由此引出了上述 ADCShare 技术，并为 SABER 带来了显著优势。其他基于 FFT 的后量子密码方案的模数为大素数，因此必须先计算所有中间值，然后才能进行模运算。 [36], [97] 已证明，在不影响安全性的前提下，将方案修改为采用 2 的幂模运算能够带来优势；因此，ADCShare 技术可应用于广泛的基于格的后量子密码算法。ADCShare 技术不适用于深度神经网络，因为丢弃中间位可能会影响最终结果。另一方面，SABER 通过对 2^t 执行模运算来缩减输出，这意味着任何会影响 $(t+1)^{th}$ 位的中间结果都是不必要的。

V. 移位加法交叉开关阵列（SACS）

使用交叉开关阵列进行多项式乘法会生成许多中间模数转换器读出值，之后通过移位相加得到最终输出多项式。例如，解密过程中的每次多项式乘法都会在 10 个周期内每周期生成 4 个值，再将这些值适当移位相加，从而得到一个输出系数值（算法 1）。中间模拟值读出的成本很高，因为需要使用模数转换器。在

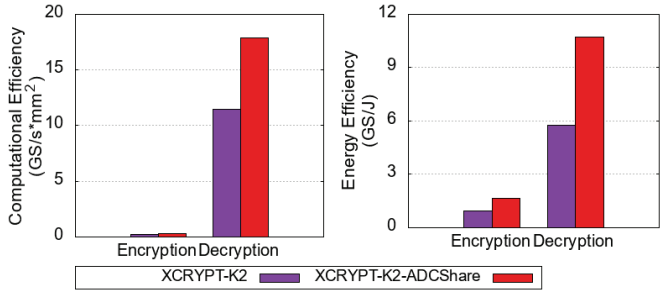


图 9：通过重新排序计算，利用低精度计算和模数转换器共享带来的改进。

本节中，我们探讨以模拟方式执行移位相加操作，从而延迟模数转换器读出的可行性。

A. 现有的模拟内移位加法实现

CASCADE [82] 提出了一种模拟内中间值移位加法方案：将交叉开关阵列各列的输出写入另一个称为缓冲器 RRAM 交叉阵列的交叉开关阵列。在给定周期内，列输出被写入缓冲器交叉阵列某一行中相邻的存储单元；跨周期的值则通过适当移位写入不同的行。采用这种映射方式，只需读出缓冲器交叉阵列，即可对所有中间值执行移位加法。通过在模拟域中执行移位加法，CASCADE 将模数转换器读出延迟到最终单个值。然而，CASCADE 假设 RRAM 单元的写入能耗开销低于近期文献报道的水平。尽管多个写入驱动器可以并行进行单元编程，但预计写入延迟约为 25 ns。RRAM 单元的写入能耗也比读出能耗高 4 个数量级（0.1 pJ/单元/位 [88]）[88], [89], [99], [100]。接下来，我们将讨论一种效率高于 CASCADE 的替代模拟移位加法技术。

B. 使用 SAC 实现无需写入的模拟域移位加法

我们提出一种在模拟域执行移位加法的新技术。我们引入移位加法交叉阵列（SAC）——一种微型单列交叉阵列，其存储单元预先编程为存储连续的 2 的幂。SAC 的基本原理是：给定一个输入向量，将其输入 SAC 后，各个值分别与存储单元的值相乘，再汇总结果。实际中，已展示出精度最高达 6 位的 RRAM 单元。因此，乘数因子最高为 $1\ll5$ 的 SAC 只能对输入执行最多 5 次移位。不过，通过分层部署 SAC 可以克服这一限制。

我们首先从在一个周期内使用 SAC 开始。由于密文 s 写入一行中的 4 个 1 位存储单元，因此每个周期都必须对 4 列的输出进行适当移位后相加。在基准方案中，这一步加法是在模数转换器读出后以数字方式完成的。在我们的工作中，我们提出使用单个 SAC 来执行这些移位加法，如图 10 所示。要将交叉阵列的输出电流馈入 SAC 的数模转换器，必须先将其转换为成比例的电压信号，这通常由跨阻放大器（TIA）完成。我们

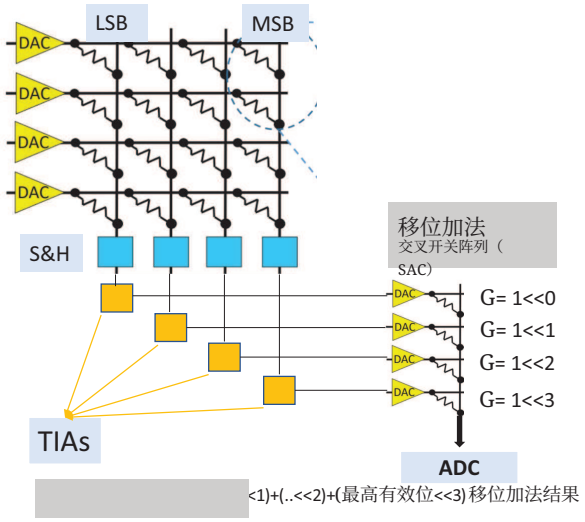


图 10：使用移位累加交叉阵列（SAC）对 4 列的输出执行移位加法。

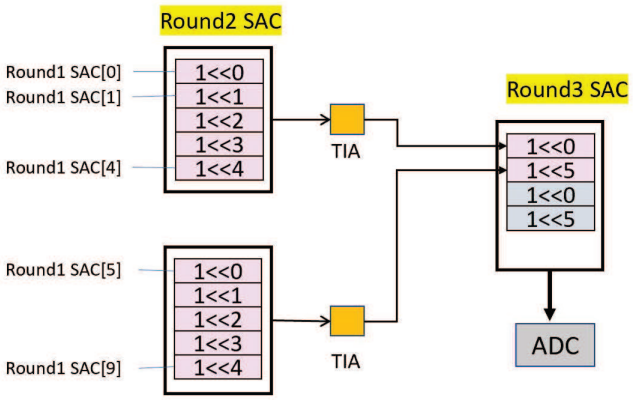
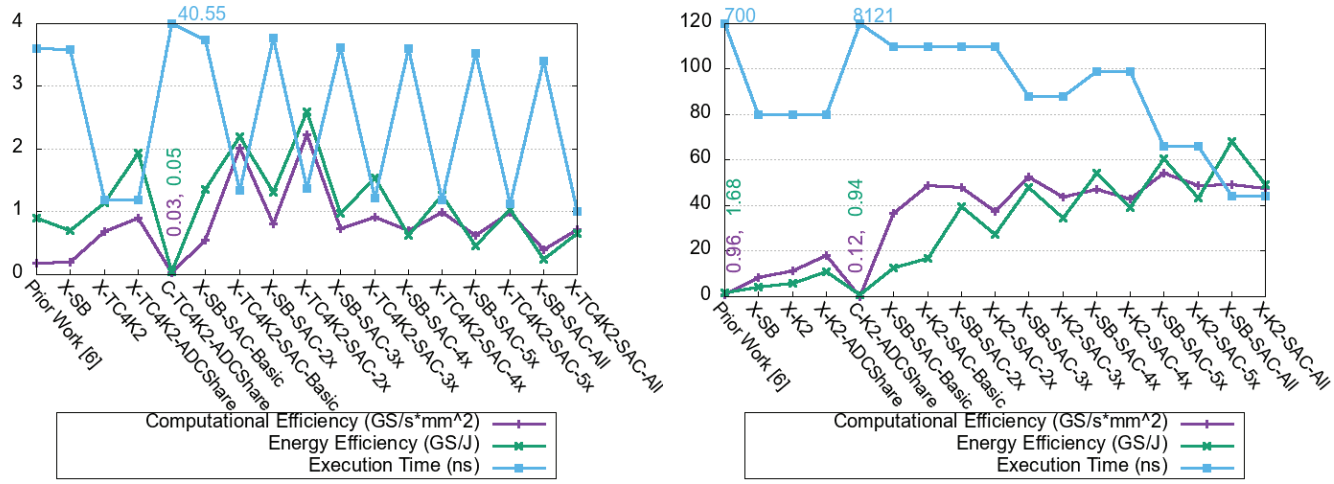


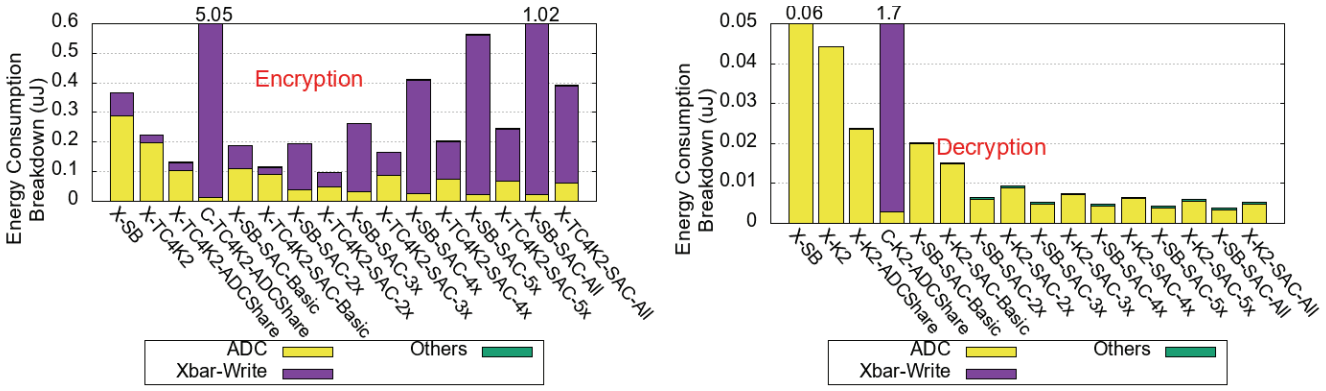
图 11：分层SAC：针对 SAC-All，对每个输出多项式系数在所有周期内进行模拟域累加。

采用 CASCADE [82] 提出的快速跨阻放大器电路（感测+传输时间为 11 ns）。请注意，SAC 的输出是 4 个 6 位值的移位加法结果，因此得到一个 10 位值。因此，需要更昂贵的 10 位精度模数转换器。然而，延迟模数转换器读出可以转换 4× 更少的样本，并允许在交叉开关阵列中实现更多共享。此外，随着周期推进，累加值中对最终输出系数有贡献的位数会逐渐减少，如 IV-E 节所述。这使得提高模数转换器共享更加灵活，因为整个点积仅生成一个 10 位样本。另一方面，在非SAC实现中，许多周期内都会生成4× 6位样本，如图 8 所示。SAC 显著减少了样本数量，从而提升了 IV-E 节所述模数转换器共享和智能调度的效果。我们将此设计称为 SAC-Basic。SAC-Basic 是一种协同技术，它利用已知的模拟电路（如跨阻放大器电路）、SABER 模运算所提供的灵活性以及资源共享



（a）加密　　（b）解密

图 12：各种 XCRYPT 设计实现的比较。X- 和 C- 分别表示 XCRYPT 和 CASCADE。



（a）加密　　（b）解密

图 13：各种 XCRYPT 设计的能耗分解。X- 和 C- 分别表示 XCRYPT 和 CASCADE。

通过交叉开关阵列上的智能调度实现。

C. 跨周期模拟域累加

在上一小节中，我们展示了如何在一个周期内以模拟方式累加列输出，从而延迟模数转换器读出。由于输入值也以每周期 1 位的形式流入，不同周期的模数转换器输出也必须在数字域中进行移位相加。由于我们的设计不会将周期输出写入交叉开关阵列（如 CASCADE），因此无法轻易实现跨周期累加。不过，我们可以通过并行执行多个周期来克服这一障碍。

跨周期累加：SAC 要求同时向不同的行馈入输入，以便对其进行移位相加。因此，要累加两个周期的输出，就需要并行执行它们的计算。这需要 2×计算/存储资源，但可减少 2×ADC 采样——这是合理的权衡，因为模数转换器占据了大部分能耗/面积。

我们将每个交叉开关阵列中单周期的 SAC 称为 Round1-SAC。与 SAC-Basic 不同，Round1-SAC 的输出会被重定向到另一个使用 TIA 的 SAC。多个对同一输出系数有贡献的交叉阵列共享这个 SAC，我们称之为 Round2-SAC。这个经过 2 个周期累加的列值，最终会在 Round2-SAC 结束时由模数转换器读出。我们将这种设计称为 SAC-2x，即在读出前累加 2 个周期。SAC-2x 虽然减少了模数转换器开销，但增加了并行运行的交叉阵列数量，进而提高了整体交叉阵列写入成本。因此，我们通过设计空间探索来评估这一权衡，同时逐步增加模拟域累加的周期数。作为极端方案，SAC-All 会对每个输出多项式系数对应的所有列和所有周期进行模拟域累加，类似于 CASCADE，因此每个输出系数仅生成 1 个模数转换器采样值。

分层SAC：由于 RRAM 单元的最高分辨率为 6 位，因此更大规模的计算采用分层方式执行，从而扩展到 Round3 和 Round4 SAC，如图 11 所示。请注意，SAC 技术也适用于其他基于格的PQC方案以及同态加密等算法。

D. 结果

SAC-Basic 结果：我们在图 12 中比较了基础 XCRYPT 设计、采用类似技术的 CASCADE，以及我们提出的新设计——在一个周期内通过 SAC 累加各列的值（SAC-Basic）；其中，前两者均采用 ADC 共享技术。SAC 设计的周期时间由 TIA 的感测+传输时间（=11ns）决定。在一个周期内，输入被流式传送至第一个交叉开关阵列，由 TIA 感测，然后作为输入发送至 SAC。在第二个周期内，TIA 的输出经 SAC 流式传送，生成最终值，再由模数转换器进行采样。由于周期数和周期时间增加，相较于基础 XCRYPT，端到端延迟有所增加。然而，通过延迟模数转换器读出，SAC-Basic 相较于 X-K2-ADCShare（X-TC4K2-ADCShare），计算效率提高了 2.7× （加密时为 2.4×），能效提高了 1.5× （1.1×）。另一方面，CASCADE（C-K2-ADCShare）的表现不如基础设计，这凸显了每个周期都执行交叉开关阵列写入操作所带来的开销。深度神经网络等其他应用也能从 SAC 设计中受益。

并行 SAC 结果：我们还在图 12 中比较了各种 SAC-* 设计。随着我们提高并行度以减少 ADC 采样，瓶颈会从模数转换器转移到交叉开关阵列写入操作。我们在图 13 中绘制了能耗分解结果。SAC 设计在模数转换器能耗与交叉开关阵列写入能耗之间进行了权衡。从 SAC-Basic 到 SAC-All，瓶颈从模数转换器（占总能耗的 77%）转移到交叉开关阵列写入操作（占总能耗的 83%）。减少 ADC 采样所带来的收益，无法抵消增加交叉开关阵列写入操作所产生的能耗开销；因此，在加密方面，SAC-All 的能效相比 X- TC4K2-ADCShare 下降了 66%。与此同时，由于解密在启动时写入一次后无需再写入交叉开关阵列，提高并行度会提升计算效率和能效。

我们还发现，采用这些电路技术会改变最适合加速的软件算法，例如，在某些情况下，Schoolbook 算法的表现优于 Karatsuba 算法。总体而言，最佳解密设计（X-SB-SAC-All）相比 X-K2-ADCShare，计算效率和能效分别提升 2.7× 和 6.3× 。对于加密，X- TC4K2-SAC-2x 相比 X-TC4K2-ADCShare 的提升分别为 2.5× 和 1.3× 。结果总结：总体而言，最佳解密设计（X- SB-SAC-All）相比 X-K2-ADCShare，计算效率和能效分别提升 4.5× 和 6.3× 。对于加密，X-TC4K2-SAC-2x 相比 X- TC4K2-ADCShare 的提升分别为 2.5× 和 1.3× 。与最先进的 ASIC 相比， [19], XCRYPT 的效率高出 3-51×，加速比达到 2.6-16×。单个瓦片每秒可执行 0.7M/22.7M 次加密/解密，面积预算为 $0.09/0.07\text{mm}^2$ 。从更广泛的密钥封装机制操作来看，封装可在 1.3 us 内完成，面积预算为 0.08 mm^2 ，能耗为 0.16 uJ；解封装则可在 1.3 us 内完成，占用 0.16 mm^2 面积，能耗为 0.17 uJ。

VI. 噪声分析

由于器件的非理想行为和电路问题，存内 RRAM 计算容易出错 [77], [92],

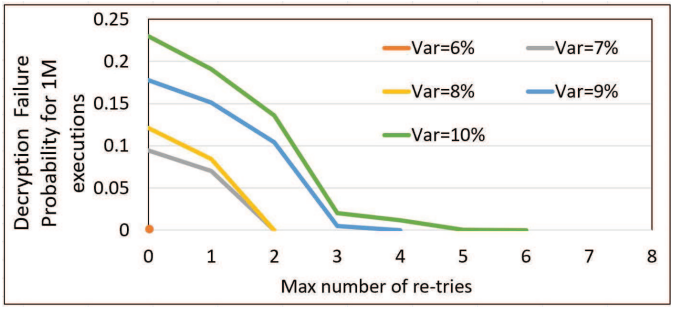


图 14：在 100 万次 SABER 调用中，为实现无故障解密所需的重试次数，单元方差可变。

[101]–[106]。不过，我们首先指出，尽管多项式乘法过程中产生的噪声会影响解密的正确性，但不会泄露有关私钥的信息。在 SABER 解密算法的第一步中，v 并非计算为 $\langle \vec{b}', \vec{s} \rangle$（忽略舍入操作），而是实际设为 $\langle \vec{b}', \vec{s} \rangle + e$，其中 e 是单元变化导致的噪声。由于解密失败，计算中的误差与 e 的总和超过了舍入参数，其中 e 占总和的大部分，且远大于私钥的范数。因此，含噪值 $\langle \vec{b}', \vec{s} \rangle + e$ 与私钥 $\vec{s}$“基本独立”。这有时称为“噪声淹没” [107]，因为噪声 e“淹没”了与私钥相关的值，从而将其影响降至最低。正如 SABER 规范 [108] 中所述，在 CPA 安全的 SABER 下，基于解密的攻击并不适用。

许多方法主要用于机器学习应用，会采用校准信号 [77], [92], [103], [104] 、故障感知映射/编码 [101], [102], ，或两者结合 [105], [106]。Hu 等人 [92] 展示了使用 TIA 进行简单线性校准的方法，以便在实际部署中实现低噪声。我们模拟了 SAC-All 设计，其中单元方差最高达 10%，并在图14中绘制了 SABER 公钥算法执行 100 万次时的失效概率。单元方差是引入输出非理想性的分布函数的方差。具体而言，5% 的单元方差表示其输出会在理想单元输出电流的 5% 范围内。我们在每次乘加操作（单个 RRAM单元）层面对交叉阵列误差进行建模，并从概率分布函数中抽样来模拟变化。我们保守地假设跨阻放大器电路中的噪声为 2%。我们对 SAC 电路中由单元编程和变化引起的非理想特性进行了建模，并假设单元变化恒定；当有大电流流过器件时，这一假设可能并不成立 [103]。然而，与传统移位加法不同，后者的误差可能因数字值中的位移而放大，而 SAC 中累积的误差体现为电流误差，只有在模数转换器读出时才会显现。因此，将一定范围的电流转换为数字值的读出过程，其误差低于传统移位加法。单个存储单元故障会影响该方案的失效概率。不过，

此类问题很容易识别并重新映射 [101], [102]。

为检测解密是否失败，数据会附带其循环冗余校验（CRC） [109]。如果解密失败，我们会重新执行该计算。图 14展示了解密失效概率随单元方差和允许的最大重试次数变化的情况。从图中可以看出，当单元方差为 5–6% 时，我们的加速器能够实现无失效解密。超过这一范围后，存储单元的变化会导致位线电流跃出一个区间，从而产生错误系数，进而导致解密失败，失效概率最高可达 0.23。经过少量重试，也可以纠正这些错误。实际应用中，5% 的单元方差是合理的 [77], [104], [110], [111], ，并且采用近期的校准技术还可以进一步降低。除此之外，采用器件良率更高、导线电阻更低且工作电导更低的最先进制造工艺，也能进一步减小噪声的影响。

VII. 结论

本研究评估了使用忆阻器交叉阵列加速基于格的后量子密码学（PQC）的可行性。我们表明，即使是 SABER 的简单实现，其速度也快于现有的 SABER 硬件方案。通过利用 SABER 的算法特性（例如其以 2 的幂为模数的模运算），我们还可以进一步提升加速器的效率。我们确定，多项式乘法是基于格方案中的关键运算，并指出基于交叉阵列的设计可能无法从现有的一些乘法软件技术中获益。我们提出了 SABER 专用的可变精度模数转换器，结合计算重排序，可实现高水平的模数转换器共享。为进一步降低模数转换器开销，我们提出了简单的模拟移位加法技术。总体而言，我们提出的加速器在计算效率/能量效率方面比最先进的 ASIC 高出 12-51×/3-40× 倍。最后，我们修改了 SABER 的 NIST 提交代码，以模拟含噪交叉阵列，并针对 XCRYPT 的设计点进行了噪声分析。

参考文献

[1] IBM。IBM量子计算，2020年（访问）。https://www.ibm.com/quantum-computing/learn/what-is-ibm-q。[2] Jeremy Hsu。2018年消费电子展：英特尔的49量子比特芯片瞄准量子霸权。IEEE Spectrum Tech Talk，第1–6页，2018年。[3] Julian Kelly。Bristlecone预览：Google的新型量子处理器。谷歌研究博客，5，2018年。[4] B Wang。IonQ拥有最强大的量子计算机，配备79个囚禁离子量子比特和160个存储量子比特，2018年。https://www.nextbigfuture.com/2018/12/ionq-has-the-most-powerful-quantum-computers-with-79-trapped-ion-qubits-and-160-stored-qubits.html。[5] Daniel J Bernstein和Tanja Lange。后量子密码学。《自然》，549(7671):188–194，2017年。[6] N. Wolchover。通往量子安全加密的艰难之路，2015年。https://www.quantamagazine.org/quantum-secure-cryptography-crosses-red-line-20150908/。[7] Peter W Shor。量子计算机上用于素因数分解和离散对数的多项式时间算法。《SIAM评论》，41(2):303–332，1999年。[8] Lov K Grover。量子力学助力在干草堆中寻找一根针。《物理评论快报》，79(2)，1997年。

[9] Gorjan Alagic、Jacob Alperin-Sheriff、Daniel Apon、David Cooper、Quynh Dang、Yi-Kai Liu、Carl Miller、Dustin Moody、Rene Peralta 等。《NIST 后量子密码学标准化进程第一轮状态报告》。美国商务部、美国国家标准与技术研究院，2019年。[10] H. Nejatollahi、N. Dutt、S. Ray、F. Regazzoni、I. Banerjee 和 R. Cammarota。《基于格的后量子密码学实现：综述》。《ACM计算调查》，第 51 卷，第 6 期，2019年。[11] 迈向传输层安全协议中的后量子密码学。https://blog.cloudflare.com/towards-post-quantum-cryptography-in-tls/。访问日期：2021-08-13。[12] Oded Regev。《关于格、带误差学习、随机线性码与密码学》。《ACM期刊》(JACM)，2009年。[13] Kanad Basu、Deepraj Soni、Mohammed Nabeel 和 Ramesh Karri。《NIST 后量子密码学——硬件评估研究》。IACR 密码学电子印本档案，2019年，2019年。[14] J. Howe、C. Moore、M. O'Neill、F. Regazzoni、T. Guneysu 和 K. Beeden。《硬件中标准格上的基于格的加密》。载于 DAC 会议论文集，2016年。[15] R. Agrawal、L. Bu、A. Ehret 和 M. Kinsy。《后量子密码硬件原语的开源 FPGA 实现》。载于 FPL 会议论文集，2019年。[16] Hamid Nejatollahi、Nikil D Dutt、Indranil Banerjee 和 Rosario Cammarota。《理想格公钥协议的领域专用加速器》。IACR 密码学电子印本档案，2018年。[17] Wai-Kong Lee、Hwajeong Seo、Seong Oun Hwang、Angshuman Karmakar、Jose Maria Bermudo Mera 和 Ramachandra Achar。《DPCrypto：使用图形处理器上的点积指令加速后量子密码算法》。密码学电子印本档案，2021年。[18] Malik Imran、Felipe Almeida、Jaan Raik、Andrea Basso、Sujoy Sinha Roy 和 Samuel Pagliarini。《65nm 专用集成电路中 SABER 的设计空间探索》。载于第五届硬件安全攻击与解决方案研讨会论文集，2021年。[19] Yihong Zhu、Min Zhu、Bohan Yang、Wenping Zhu、Chenchen Deng、Chen Chen、Shaojun Wei 和 Leibo Liu。《Lwrpro：面向模舍入学习的高能效可配置密码处理器》。《IEEE电路与系统汇刊 I：常规论文》，2021年。[20] Archisman Ghosh、JMB Mera、Angshuman Karmakar、Debayan Das、Santosh Ghosh、Ingrid Verbauwhede 和 Shreyas Sen。《一款 334uW、0.158 mm^2 基于 Saber 舍入学习的后量子密码加速器》。arXiv 预印本 arXiv:2201.07375，2022年。[21] Ciaran Mclvor、Maire McLoone 和 John V McCanny。《快速 Montgomery 模乘法及 RSA 密码处理器架构》。载于第三十七届阿西洛马会议：信号、系统与计算机，2003年。[22] Soram Ranbir Singh、Ajoy Kumar Khan 和 Soram Rakesh Singh。《RSA 与椭圆曲线密码学的性能评估》。载于当代计算与信息学国际会议 (IC3I)，2016年。[23] 英特尔架构平台上的集成密码与压缩加速器。https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/integrated-cryptographic-compression-accelerators-brief.pdf。访问日期：2021-08-13。[24]Leonardo ADS Ribeiro、Jos´e Paulo da Silva Lima、Ruy JGB de Queiroz、Amirton Bezerra、Jonysberg P Chagas、Fabio QB da Silva、Andre LM Santos、Jos´e Roberto M Ribeiro Junior。《Saber 后量子密钥封装机制 (KEM)：评估其在移动设备上的性能并提出若干改进建议》。[25] Ronald L Rivest、Adi Shamir 和 Leonard Adleman。《一种获得数字签名和公钥密码系统的方法》。《ACM通讯》，21(2):120–126，1978年。[26] Whitfield Diffie 和 Martin Hellman。《密码学的新方向》。《IEEE信息论汇刊》，22(6):644–654，1976年。[27] A. Shafiee、A. Nag、N. Muralimanohar、R. Balasubramonian、J. Strachan、M. Hu、R.S. Williams 和 V. Srikumar。《ISAAC：一种采用交叉阵列原位模拟运算的卷积神经网络加速器》。载于 ISCA 会议论文集，2016年。[28] Ping Chi、Shuangchen Li、Ziyang Qi、Peng Gu、Cong Xu、Tao Zhang、Jishen Zhao、Yongpan Liu、Yu Wang 和 Yuan Xie。《PRIME：一种基于电阻式随机存取存储器主存的神经网络计算新型存内处理架构》。载于 ISCA-43 会议论文集，2016年。[29] Oded Goldreich、Daniele Micciancio、Shmuel Safra 和 J-P Seifert。《近似最短格向量并不比近似最近格向量更难》。《信息处理快报》，71(2):55–

61，1999年。[30] C. Gentry。《全同态加密方案》。博士论文，斯坦福大学，2009年。[31] Vadim Lyubashevsky、Chris Peikert 和 Oded Regev。《关于理想格与环上的误差学习》。载于密码技术理论与应用国际年会论文集，第 1–23 页，2010年。[32] Yara Elias、Kristin E Lauter、Ekin Ozman 和 Katherine E Stange。《环上带误差学习的可证明弱实例》。载于密码学年会，2015年。[33] Wouter Castryck、Ilia Iliashenko 和 Frederik Vercauteren。《重新审视环上带误差学习的可证明弱实例》。载于密码技术理论与应用国际年会论文集，2016年。[34] Joppe Bos、L´eo Ducas、Eike Kiltz、Tancr`ede Lepoint、Vadim Lyuba- shevsky、John M Schanck、Peter Schwabe、Gregor Seiler 和 Damien Stehl´e。《CRYSTALS-Kyber：一种抗选择密文攻击的模格密钥封装机制》。载于 IEEE 欧洲安全与隐私研讨会，2018年。[35] Jo¨el Alwen、Stephan Krenn、Krzysztof Pietrzak 和 Daniel Wichs。《重新审视舍入学习》。载于密码学年会，2013年。[36] Jan-Pieter D'Anvers、Angshuman Karmakar、Sujoy Sinha Roy 和 Frederik Vercauteren。《Saber：基于模舍入学习的密钥交换、抗选择明文攻击加密及抗选择密文攻击的密钥封装机制》。载于非洲密码学国际会议，2018年。[37] Morris J Dworkin。《SHA-3 标准：基于置换的哈希与可扩展输出函数》。2015年。[38] Matthias J Kannwischer、Joost Rijneveld、Peter Schwabe 和 Ko Stoffelen。《pqm4：在 ARM Cortex-M4 上测试和评测 NIST 后量子密码》。2019年。[39] Stephen A Cook 和 St˚al O Aanderaa。《关于函数的最小计算时间》。《美国数学会汇刊》，1969年。[40] Viet B Dang、Farnoud Farahmand、Michal Andrzejczak 和 Kris Gaj。《采用软硬件协同设计实现并评测三种基于格的后量子密码算法》。载于 2019 年现场可编程技术国际会议 (ICFPT)，2019年。[41] Sujoy Sinha Roy 和 Andrea Basso。《用于基于格的密钥封装机制的高速指令集协处理器：硬件实现的 Saber 》。《 IACR 密码硬件与嵌入式系统汇刊》，2020 年。[42] Jose Maria Bermudo Mera、Furkan Turan、Angshuman Karmakar、Sujoy Sinha Roy 和 Ingrid Verbauwhede。《用于加速基于模格的密钥封装机制的紧凑型领域专用协处理器》。IACR 密码学电子印本档案，2020年。[43] Utsav Banerjee、Tenzin S Ukyab 和 Anantha P Chandrakasan。《Sapphire：面向基于格的后量子协议的可配置密码处理器》。arXiv 预印本 arXiv:1910.07557，2019年。[44] James Howe、Tobias Oder、Markus Krausz 和 Tim G¨uneysu。《嵌入式设备上的标准基于格的密钥封装》。IACR 密码硬件与嵌入式系统汇刊，2018年。[45] Keccak 团队。《VHDL 中的 Keccak：高速核心》，2019年。[46] Jeffrey Hoffstein、Jill Pipher 和 Joseph H Silverman。《NTRU：一种基于环的公钥密码系统》。载于国际算法数论研讨会，1998年。[47] Norman G¨ottert、Thomas Feller、Michael Schneider、Johannes Buchmann 和 Sorin Huss。《现代基于格的加密方案之硬件构建模块设计》。载于国际密码硬件与嵌入式系统研讨会，2012年。[48] Thomas P¨oppelmann 和 Tim G¨uneysu。《面向可重构硬件上高效基于格的密码学算术》。载于拉丁美洲密码学与信息安全国际会议，2012年。[49] Thomas P¨oppelmann 和 Tim G¨uneysu。《迈向可重构硬件上实用的基于格的公钥加密》。载于密码学选定领域国际会议，2013年。[50] Aydin Aysu、Cameron Patterson 和 Patrick Schaumont。《低成本、高面积效率的基于格的密码学 FPGA 实现》。载于 2013 年 IEEE 硬件安全与信任国际研讨会 (HOST)，2013年。[51] Sujoy Sinha Roy、Frederik Vercauteren、Nele Mentens、Donald Donglong Chen 和 Ingrid Verbauwhede。《紧凑型环上带误差学习密码处理器》处理器。在国际密码硬件与嵌入式系统研讨会上，2014年。[52] David Bruce Cousins、John Golusky、Kurt Rohloff 和 Daniel Sumorok。基于现场可编程门阵列协处理器的同态加密实现。发表于 2014 IEEE高性能极限计算会议（HPEC），2014年。[53] Wei Wang、Zhilu Chen 和 Xinming Huang。利用图形处理器加速分级全同态加密。发表于 2014 IEEE国际电路与系统研讨会（ISCAS），2014年。[54] Wei Wang、Yin Hu、Lianmu Chen、Xinming Huang 和 Berk Sunar。探索全同态加密的可行性。《IEEE计算机汇刊》，2013年。[55] Chaohui Du 和 Guoqiang Bai。面向基于格的密码学的一系列可扩展多项式乘法器架构。发表于 2015 IEEE Trustcom/BigDataSE/ISPA，2015年。[56] Tamas Gy¨orfi、Octavian Cret 和 Zalan Borsos。在现场可编程门阵列中实现模快速傅里叶变换——基于格的密码学的基本模块。发表于 2013 Euromicro数字系统设计会议，2013年。[57] Chaohui Du 和 Guoqiang Bai。面向基于格的密码学的高效多项式乘法。发表于 2016 IEEE国际电路与系统研讨会（ISCAS），2016年。[58] Chaohui Du 和 Guoqiang Bai。面向环上带误差学习公钥密码系统的高效多项式乘法器架构。发表于 2016 IEEE国际电路与系统研讨会（ISCAS），2016年。[59] Hamid Nejatollahi、Rosario Cammarota 和 Nikil Dutt。面向环上带误差学习基于格密码学的灵活数论变换加速器。发表于 2019 IEEE第37届国际计算机设计会议（ICCD），2019年。[60] Hamid Nejatollahi、Sina Shahhosseini、Rosario Cammarota 和 Nikil Dutt。探索利用阵列处理器实现能效型抗量子信号处理。发表于 ICASSP 2020-2020 IEEE国际声学、语音与信号处理会议（ICASSP），2020年。[61] Farnoud Farahmand、Viet Ba Dang、Michal Andrzejczak 和 Kris Gaj。采用软硬件协同设计方法实现并基准测试七种第二轮基于格的密钥封装机制。发表于第二届后量子密码标准化会议论文集，2019年。[62] Andrea Basso 和 Sujoy Sinha Roy。面向后量子密钥封装机制 SABER 的优化多项式乘法器架构。[63] Jose Maria Bermudo Mera、Angshuman Karmakar 和 Ingrid Ver-bauwhede。Toom-Cook 乘法中的时间-存储折衷：在基于模格的密码学中的应用。《IACR 密码硬件与嵌入式系统汇刊》，2020年。[64] L´eo Ducas、Tancrede Lepoint、Vadim Lyubashevsky、Peter Schwabe、Gregor Seiler 和 Damien Stehl´e。CRYSTALS–dilithium：源自模格的数字签名。2018年。[65] J. Fan 和 F. Vercauteren。实用型全同态加密。发表于《密码学电子印本档案》报告 2012/144，2012年。[66] Zvika Brakerski。无需模数切换、基于经典间隙最短向量问题的全同态加密。发表于密码学年会，2012年。[67] CryptoExperts。FV-NFLlib，2016年。https://github.com/CryptoExperts/ FV-NFLlib/。[68] Microsoft Research。简单加密算术库（SEAL），2016年。https://github.com/microsoft/SEAL/。[69] Craig Gentry、Amit Sahai 和 Brent Waters。基于带误差学习的同态加密：概念更简单、渐近速度更快且支持属性基。发表于密码学年会，2013年。[70] Jung Hee Cheon、Andrey Kim、Miran Kim 和 Yongsoo Song。面向近似数运算的同态加密。发表于国际密码学与信息安全理论及应用会议，2017年。[71] Ilaria Chillotti、Nicolas Gama、Mariya Georgieva 和 Malika Iz-abach`ene。TFHE：环面上的快速全同态加密。《密码学杂志》，2020年。[72] B. Reagen、W.-S. Choi、Y. Ko、V.T. Lee、H.-H.S. Lee、G.-Y. Wei 和 D. Brooks。Cheetah：针对隐私推理优化并加速同态加密。发表于第27届国际高性能计算机体系结构会议（HPCA），2021年。[73] G. W. Burr、M. J. Breitwisch、M. Franceschini、D. Garetto、K. Gopalakrishnan、B. Jackson、B. Kurdi、C. Lam、L. A. Lastras、A. Padilla、B. Rajendran、S. Raoux 和 R. S. Shenoy。相变存储器技术，2010年。http://arxiv.org/abs/1001.1164v1。[74] Pascal O Vontobel、Warren Robinett、Philip J Kuekes、Duncan R Stewart、Joseph Straznicky 和 R Stanley Williams。写入和

读取基于忆阻器的纳米级交叉阵列存储器。《纳米技术》，第20卷，2009年。 [75] B. Lee、E. Ipek、O. Mutlu 和 D. Burger。将相变存储器设计为可扩展的动态随机存取存储器替代方案。发表于 ISCA 论文集，2009年。[76] Joseph Izraelevitz、Jian Yang、Lu Zhang、Juno Kim、Xiao Liu、Amirsaman Memaripour、Yun Joon Soh、Zixuan Wang、Yi Xu、Sub-ramanya R Dulloor 等。英特尔傲腾 DC 持久内存模块的基本性能测量。arXiv 预印本 arXiv:1903.05714，2019年。[77] Miao Hu、John Paul Strachan、Zhiyong Li、Emmanuelle Merced Grafals、Noraica Davila、Catherine Graves、Sity Lam、Ning Ge、R. Stanley Williams 和 Jianhua Yang。用于神经形态计算的点积引擎：通过编程 1T1M 交叉阵列加速矩阵向量乘法。发表于 DAC-53 论文集，2016年。[78] S. Kvatinsky、D. Belousov、S. Liman、G. Satat、N. Wald、E.G. Friedman、A. Kolodny 和 U.C. Weiser。MAGIC——忆阻器辅助逻辑。《IEEE 电路与系统汇刊 II：简报》，61(11)，2014年。[79] Linghao Song、Xuehai Qian、Hai Li 和 Chen Yiran。PipeLayer：用于深度学习的流水线式阻变存储器加速器。发表于 HPCA 论文集，2017年。[80] H. Nejatollahi、S. Gupta、M. Imani、T.S. Rosing、R. Cammarota 和 N. Dutt。CryptoPIM：基于存内加速的基于格的密码硬件。发表于 DAC 论文集，2020年。[81] B. Feinberg、S. Wang 和 E. Ipek。提高忆阻神经网络加速器的可靠性。发表于 HPCA 论文集，2018年。[82] Teyuh Chou、Wei Tang、Jacob Botimer 和 Zhengya Zhang。CASCADE：连接阻变存储器，以在端到端存内处理范式中扩展模拟数据流。发表于第52届 IEEE/ACM 微体系结构国际研讨会论文集，第114–125页，2019年。[83] S. Narayanan、A. Shafiee 和 R. Balasubramonian。INXS：弥合脉冲神经网络的吞吐量与能耗差距。发表于 IJCNN 论文集，2017年。 [84] D. Fick。用于数据中心级人工智能推理的小芯片闪存模拟计算，2018年。Hot Chips 演讲，https://www.mythic-ai.com/mythic-hot-chips-2018/。[85] Crossbar Inc。Crossbar 阻变存储器：重新思考简约设计，2020年（访问日期）。https://www.crossbar-inc.com/technology/reram-overview/。[86] Memryx。Memryx Argo，2020年（访问日期）。https://www.memryx.com/home。[87] Syntiant。由定制人工智能芯片驱动的全天候语音，2020年（访问日期）。https://www.syntiant.com/。[88] Furqan Zahoor、Tun Zainal Azni Zulkifli 和 Farooq Ahmad Khanday。阻变随机存取存储器（RRAM）：材料、开关机制、性能、多级单元（MLC）存储、建模及应用综述。《Nanoscale Research Letters》，2020年。[89] Zongjie Shen、Chun Zhao、Yanfei Qi、Wangying Xu、Yina Liu、Ivona Z Mitrovic、Li Yang 和 Cezhou Zhao。阻变存储器件的进展：阻变机制、材料及仿生突触应用。《纳米材料》，2020年。 [90] A. Nag、R. Balasubramonian、V. Srikumar、R. Walker、A. Shafiee、J. Strachan 和 N. Muralimanohar。Newton：迈向交叉阵列加速的物理极限。《IEEE Micro》忆阻器计算特刊，2018年。[91] X. Dong 等。NVSim：新兴非易失性存储器的电路级性能、能耗与面积模型。技术报告，2012年。[92] Miao Hu、Catherine E Graves、Can Li、Yunning Li、Ning Ge、Eric Montgomery、Noraica Davila、Hao Jiang、R Stanley Williams、J Joshua Yang 等。基于忆阻器的模拟计算和利用点积引擎进行神经网络分类。《先进材料》，2018年。[93] Yu Ji、Youyang Zhang、Xinfeng Xie、Shuangchen Li、Peiqi Wang、Xing Hu、Youhui Zhang 和 Yuan Xie。FPSA：面向可重构阻变存储器神经网络加速器架构的全系统栈解决方案。发表于第二十四届编程语言与操作系统体系结构支持国际会议论文集，2019年。[94] K.D. Choo、J. Bell 和 M.P. Flynn。面积高效的 1GS/s 6b 逐次逼近型模数转换器，采用基于电荷注入单元的数模转换器。发表于 ISSCC 论文集，2016年。[95] Chi-Ming Marvin Chung、Vincent Hwang、Matthias J Kannwischer、Gregor Seiler、Cheng-Jhih Shih 和 Bo-Yin Yang。面向不适用于数论变换的环的数论变换乘法。《IACR 密码硬件与嵌入式系统汇刊》，2021年。

[96] Viet Ba Dang、Kamyar Mohajerani 和 Kris Gaj。CRYSTALS-Kyber、NTRU 和 Saber 的高速硬件架构与公平的 FPGA 基准测试。[97] Dayane Reis、Jonathan Takeshita、Taeho Jung、Michael Niemier 和 Xiaobo Sharon Hu。面向高性能、低能耗同态加密的存内计算。IEEE超大规模集成电路系统汇刊，2020年。[98] S. Aga、S. Jeloka、A. Subramaniyan、S. Narayanasamy、D. Blaauw 和 R. Das。计算缓存。载于 HPCA 论文集，2017年。[99] Wei-Hao Chen、Kai-Xiang Li、Wei-Yu Lin、Kuo-Hsiang Hsu、Pin-Yi Li、Cheng-Han Yang、Cheng-Xin Xue、En-Yu Yang、Yen-Kai Chen、Yun-Sheng Chang 等。用于二值深度神经网络人工智能边缘处理器的 65 nm、1 Mb 非易失性存内计算 ReRAM 宏，支持低于 16 ns 的乘加运算。载于 2018年 IEEE 国际固态电路会议（ISSCC），2018年。[100] Basma Hajri、Hassen Aziza、Mohammad M Mansour 和 Ali Chehab。RRAM器件模型：比较分析与实验验证。IEEE Access，2019年。[101] Injune Yeo、Myonglae Chu、Sang-Gyun Gi、Hyunsang Hwang 和 Byung-Geun Lee。基于忆阻器交叉阵列的神经网络固定故障容错方案。IEEE电子器件汇刊，2019年。[102] Ching-Yi Chen、Hsiu-Chuan Shih、Cheng-Wen Wu、Chih-He Lin、Pi-Feng Chiu、Shyh-Shyuan Sheu 和 Frederick T Chen。基于 March 测试和新型挤压搜索方案的 RRAM 缺陷建模与失效分析。IEEE计算机汇刊，2014年。[103] Indranil Chakraborty、Mustafa Fayez Ali、Dong Eun Kim、Aayush Ankit 和 Kaushik Roy。GENIEx：利用神经网络模拟忆阻器交叉阵列非理想性的一种通用方法。载于 2020年 第57届 ACM/IEEE设计自动化会议（DAC），2020年。[104] Beiye Liu、Miao Hu、Hai Li、Zhi-Hong Mao、Yiran Chen、Tingwen Huang 和 Wei Zhang。用于基于忆阻器交叉阵列的模拟神经形态计算引擎的数字辅助降噪训练。载于 2013年 第50届 ACM/EDAC/IEEE设计自动化会议（DAC），2013年。[105] Gouranga Charan、Jubin Hazra、Karsten Beckmann、Xiaocong Du、Gokul Krishnan、Rajiv V Joshi、Nathaniel C Cady 和 Yu Cao。利用不准确的 RRAM 器件实现准确推理：统计数据、模型迁移与在线自适应。载于 2020年 第57届 ACM/IEEE设计自动化会议（DAC），2020年。[106] Shubham Jain 和 Anand Raghunathan。CxDNN：电阻式交叉阵列系统中深度神经网络的软硬件补偿方法。ACM嵌入式计算系统汇刊，2019年。[107] Shafi Goldwasser、Yael Tauman Kalai、Chris Peikert 和 Vinod Vaikuntanathan。带误差学习假设的鲁棒性。2010年。[108] SABER：基于模数学习带舍入的密钥封装机制（第三轮提交）。https://www.esat.kuleuven.be/cosic/pqcrypto/saber/files/saberspecround3.pdf/。[109] William Wesley Peterson 和 Daniel T Brown。用于错误检测的循环码。IRE会刊，1961年。[110] Lixue Xia、Boxun Li、Tianqi Tang、Peng Gu、Pai-Yu Chen、Shimeng Yu、Yu Cao、Yu Wang、Yuan Xie 和 Huazhong Yang。MNSIM：基于忆阻器的神经形态计算系统仿真平台。IEEE集成电路与系统计算机辅助设计汇刊，2017年。[111] Shinhyun Choi、Yuchao Yang 和 Wei Lu。氧化物基电阻式存储器的随机电报噪声与电阻开关分析。Nanoscale，2014年。