

后量子密码算法的软件实现研究*

周 天¹, 郑昉昱², 林璟铨^{1,3}, 魏 荣², 唐文煦¹

1. 中国科学技术大学 网络空间安全学院, 合肥 230026

2. 中国科学院大学 密码学院, 北京 100049

3. 北京中科研究院, 北京 100193

通信作者: 林璟铨, E-mail: linjq@ustc.edu.cn

摘 要: 基于公钥密码算法的各类安全技术已经成为了目前网络空间安全的重要基石, 在构建信任体系、保护传输安全、身份鉴别等方面发挥着举足轻重的作用. 然而, 利用 Shor 算法, 量子计算机可以在多项式时间内破解 RSA、ECC 等当前主流公钥密码体制所依赖的数学困难问题. 随着量子计算技术的发展, 这一安全威胁日益迫近, 所以学术界和工业界都致力于研制抵御量子计算机攻击的公钥密码体制. NIST 后量子密码标准化项目是目前最具代表性的项目之一, 它于 2017 年征集了数量众多、种类多样的后量子密码方案, 目前已产生阶段性结果. 除了算法安全性分析之外, 各后量子密码算法的软件实现性能也是该项目评价其优劣的重要指标之一. 本文围绕六类主要的后量子密码算法, 以密码算法软件实现的视角, 详细分析了每一类的代表方案及其主要计算负载、优化算法以及软件实现特点; 并基于 x86 和 ARM 平台, 全面梳理和总结了各文献中提出的、特定类型算法的加速技术和通用优化方法.

关键词: 公钥密码; 后量子密码; 软件实现; 格密码算法

中图分类号: TP309.7 **文献标识码:** A **DOI:** 10.13868/j.cnki.jcr.000681

中文引用格式: 周天, 郑昉昱, 林璟铨, 魏荣, 唐文煦. 后量子密码算法的软件实现研究[J]. 密码学报 (中英文), 2024, 11(2): 308-343. [DOI: 10.13868/j.cnki.jcr.000681]

英文引用格式: ZHOU T, ZHENG F Y, LIN J Q, WEI R, TANG W X. On software implementations of post-quantum cryptography[J]. Journal of Cryptologic Research, 2024, 11(2): 308-343. [DOI: 10.13868/j.cnki.jcr.000681]

On Software Implementations of Post-Quantum Cryptography

ZHOU Tian¹, ZHENG Fang-Yu², LIN Jing-Qiang^{1,3}, WEI Rong², TANG Wen-Xu¹

1. School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230026, China

2. School of Cryptology, University of Chinese Academy of Sciences, Beijing 100049, China

3. Beijing Research Institute, University of Science and Technology of China, Beijing 100193, China

Corresponding author: LIN Jing-Qiang, E-mail: linjq@ustc.edu.cn

Abstract: Various fundamental technologies based on the public-key cryptography become an important cornerstone of cyberspace security, and play a pivotal role in building a system of trust, provide data transmission security and authentication, etc. However, with the development of quantum

* 基金项目: 国家重点研发计划 (2020YFB1005803); 国家自然科学基金 (61902392)

Foundation: National Key Research and Development Program of China (2020YFB1005803); National Natural Science Foundation of China (61902392)

收稿日期: 2022-12-18 定稿日期: 2024-01-27

computing technology, with Shor's algorithm, quantum computers are able to solve many mathematically difficult problems in polynomial time, these problems are the bases for the security of current mainstream public key cryptosystems such as RSA and ECC, which means that these public key cryptosystems are no longer safe. Faced with this looming threat, both academia and industry are working to develop public-key cryptosystems that are resistant to attacks from quantum computers. The NIST Post-Quantum Cryptography Standardization Project is one of the most representative projects. In 2017, it solicited the research community to propose a large number of post-quantum cryptographic schemes of different variety, and phased results have been produced. In addition to the basic security analysis, the performance of the software implementation is one of the important metrics when evaluating each post-quantum cryptographic algorithm. This paper focuses on six main types of post-quantum cryptosystems from the perspective of software implementation of cryptographic algorithms, and analyzes in detail the representative schemes of each type, their main computing load, optimization algorithms, and software implementation characteristics. Based on x86 and ARM platforms, this paper systematically lists and comprehensively summarizes the algorithm-specific acceleration techniques and general optimization methods proposed in various literature.

Key words: public-key cryptography; post-quantum cryptography; software cryptographic implementation; lattice-based cryptography

1 背景介绍

目前, 基于 RSA、ECC 的公钥基础设施构建了互联网的信任基础, 然而量子计算技术的不断提升, 对 RSA、ECC 等传统公钥密码算法的安全性带来了极大的冲击. 根据量子计算机的基础数据单元——量子比特的叠加态原理^[1], 一些传统计算机难以求解的数学问题可以被量子计算机高效解决. 根据 Shor 算法^[2], 量子计算机可在多项式时间内求解整数分解和离散对数问题, 从而破解 RSA 和 ECC 等主流公钥密码算法. 考虑到回溯攻击^[3] (即敌手存储当前网络中加密数据流, 待大规模量子计算机出现后利用其来取得秘密数据), 迫切需要在大规模量子计算机出现之前建立可抵抗量子攻击的公钥密码算法体制.

后量子密码 (post-quantum cryptography, PQC) 是指可以同时抵抗来自于传统计算机和量子计算机攻击的密码体制 (cryptosystem). 它包括用于公钥加密 (public-key encryption, PKE)/密钥封装机制 (key encapsulation mechanism, KEM) 和数字签名 (digital signature) 两类算法. 为了构建后量子密码, 就要研究哪些困难问题是量子计算机无法高效求解的. 自从 1994 年 Shor 算法提出, 经过数十年研究, 密码学界逐渐确定了六类具有量子困难性的问题, 在此基础上构建了基于格的密码、基于编码的密码、基于多变量的密码、基于同源的密码、基于杂凑的密码和基于零知识证明的密码.

1.1 NIST 后量子密码标准化项目

为了对现行“量子不安全”的密钥建立 (SP 800-56A^[4] 和 SP 800-56B^[5]) 标准和数字签名 (FIPS 186-4^[6]) 标准向后量子算法演进, 美国国家标准技术研究院 (National Institute of Standards and Technology, NIST) 从 2016 年开始启动后量子密码标准化项目, 并在 2017 年征集到了 69 个“完整且合适 (complete and proper)”的候选算法. 经过三轮筛选, 最终保留了 15 个算法. 其中 9 个属于 PKE/KEM 类^[7], 6 个方案是数字签名算法. 此外, 每类方案中又分为入围方案 (finalist) 和备选方案 (alternate). NIST 在第二轮阶段报告^[8]中写道, “入围方案是 NIST 认为最有希望适合大多数用途的方案, 并将在第三轮结束不久后开始标准化. 而备选方案则被认为是潜力进行标准化, 将会接受另一轮评估.”

NIST 在后量子密码标准化项目中对各个后量子密码方案进行筛选, 从多个方面评估各个方案之间的优劣, 而评估的指标包括: 安全性、密钥参数和 (软件) 实现性能以及其他实现特征 (一些候选算法具有目前 NIST 已标准化的 RSA、ECC 算法所不具备的新特点, 例如基于杂凑的数字签名方案 SPHINCS+ 高度模块化, 可以有多种不同的杂凑函数实例化).

2022 年 7 月 5 日, NIST 公布了 PQC 标准化新一轮的进展, 包括四个选作后量子密码标准的算法: Kyber、Dilithium、FALCON 和 SPHINCS+, 以及参与第四轮评估的算法: BIKE、Classic McEliece、

HQC 和 SIKE.

为了全面展现后量子密码的多样性和发展历程, 本文以 NIST 第三轮入围及部分备选算法作为每个密码类型的代表分别介绍, 包括安全性受到影响的 Rainbow 和 SIKE, 具体情况见表 1.

表 1 本文选作代表的密码算法
Table 1 Cryptographic algorithms selected as representatives in this paper

功能	基于的困难问题类别	后量子密码算法	当前状态
PKE/KEM	格	CRYSTALS-KYBER (下文简称 Kyber)	将被标准化
		NTRU	第三轮入围, 第四轮落选
		SABER	第三轮入围, 第四轮落选
	编码	Classic McEliece	第四轮入围
	同源	SIKE	第四轮入围, 但已不安全
数字签名	格	CRYSTALS-DILITHIUM (下文简称 Dilithium)	将被标准化
		FALCON	将被标准化
	杂凑	SPHINCS+	将被标准化
	多变量	Rainbow	第三轮入围, 第四轮落选
	ZKP	Picnic	第三轮备选, 第四轮落选

1.2 后量子密码安全模型

NIST 对 PKE/KEM 方案的安全性要求是达到对选择密文攻击“语义安全”(semantically secure), 即选择密文攻击下的不可区分性(indistinguishability under adaptive chosen ciphertext attack, IND-CCA2) [7]. 此外, 对仅加密临时密钥的 PKE/KEM 方案的安全性要求则可以宽松至选择明文攻击下的不可区分性(indistinguishability under chosen plaintext attack, IND-CPA) [7]. NIST 对数字签名方案的安全性要求是达到对自适应选择消息攻击存在不可伪造(existential unforgeability under chosen message attack, EUF-CMA) 的数字签名 [7].

大部分后量子 PKE/KEM 方案中公钥加密方案(PKE) 都设计为达到 IND-CPA 安全. 但仅具有 IND-CPA 安全性的方案, 当密钥重用的时候易遭到选择密文攻击 [9–11]. 1999 年 Fujisaki 和 Okamoto [12] 提出了一种将弱对称和弱非对称加密方案转换为 CCA 安全的混合方案. 其中变换方案称为 Fujisaki-Okamoto (FO) 变换. FO 变换可将 IND-CPA 安全方案转换为 IND-CCA 安全. 经过 FO 变换后的方案可以检测出无效或恶意构造的密文, 并在检测出后直接返回失败, 这样, 攻击者精心选择的恶意密文将会被拒绝, 也就得不到选择密文解密的任何有效信息 [13]. 因此, 大部分后量子 PKE 方案会在达到 IND-CPA 安全的基础上, 对量子随机预言机模型采用 FO 变换, 从而构建出 IND-CCA2 安全的 KEM 方案, 但代价是降低了解密速度.

1.3 后量子密码安全强度

NIST 官方在征集提案文档 [7] 中对征集的后量子密码提出了量化的安全强度要求, 具体安全等级如表 2.

NIST 在征集文档 [7] 中指出“除非在标准化过程中出现一些突破性技术进展, 否则 NIST 将假设五种安全强度在实际安全性方面是顺序排序的 (例如, NIST 假设对 SHA256 的暴力碰撞攻击比对 AES192 的暴力密钥搜索攻击所需的计算量更小)”. 尽管 NIST 对后量子密码列出了五个安全等级, 大多数方案中仅提供了安全等级 1 (低安全性)、安全等级 3 (中安全性) 和安全等级 5 (高安全性) 的参数集.

表 2 NIST 定义的后量子密码安全等级
Table 2 Post-quantum cryptography security levels defined by NIST

安全等级	描述	算法举例
1	攻破相关安全定义的任何攻击必须需要多于对 128 位密钥分组密码密钥搜索所需的计算资源	AES128
2	攻破相关安全定义的任何攻击必须需要多于对 256 位杂凑函数寻找碰撞所需的计算资源	SHA256/SHA3-256
3	攻破相关安全定义的任何攻击必须需要多于对 192 位密钥分组密码密钥搜索所需的计算资源	AES192
4	攻破相关安全定义的任何攻击必须需要多于对 384 位杂凑函数寻找碰撞所需的计算资源	SHA384/SHA3-384
5	攻破相关安全定义的任何攻击必须需要多于对 256 位密钥分组密码密钥搜索所需的计算资源	AES256

2 后量子密码

本节简要介绍各类后量子密码及典型方案. 此外, 根据 pqm4 基准测试中杂凑计算占比评估数据, 一些方案的主要计算负载是杂凑计算 (且随着其他负载的不断优化, 杂凑计算占比还会进一步提高). 而杂凑计算优化不是后量子密码软件实现优化的首要目标, 这主要是因为目前大多数平台会支持专门的指令集扩展以加速杂凑计算, 例如 Intel 2013 年引入了 SHA 扩展^[14]支持 SHA 系列杂凑算法等. 本节主要关注后量子密码算法 (除基于杂凑的类别) 中杂凑计算以外的最主要负载.

2.1 具有竞争力的后量子密码

根据 NIST 选作标准和入围第四轮评选的后量子密码算法, 三类极具竞争力的后量子密码脱颖而出, 本节将分别介绍它们.

2.1.1 基于格的后量子密码

格 (lattice) 是由一组线性无关的基向量的线性组合生成, 从直观上看, 它在 n 维空间中呈现网格状结构. 基于格的密码由于其设计结构, 具有基于格问题的最坏情况安全证明. 此类密码在提出之初就受到了理论界的高度关注. 经过了数十年的广泛研究, 基于格的困难问题被广泛认为具有量子抵抗性. 其中目前主流的两个困难问题分别是基于容错学习 (learning with errors, LWE)^[15]和舍入学习 (learning with rounding, LWR)^[16]. 此外, 可对格附加一些结构来加速其计算, 例如环上容错学习 (ring-LWE, RLWE)^[17]、模容错学习 (module-LWE, MLWE)^[18]和模舍入学习 (module-LWR, MLWR)^[19].

在入围 NIST 后量子算法征集第三轮方案中, 基于格的密码方案占比最高: KEM 类的有 Kyber、NTRU 和 Saber, 签名类的有 Dilithium 和 FALCON. 考虑到 Kyber 方案各个方面 (密钥对长度和实现速度等) 都较为优秀、没有明显短板, 且相关研究最为广泛. 基于以上两点本节选择 Kyber 作为代表来介绍基于格的密码.

Kyber 是由 Bos 等人^[20]于 2017 年提出的基于格的 KEM 方案. 它的安全性基于 MLWE 问题. 设计者指出, 模格结构结合了 LWE 构造的安全性优势和 RLWE 构造的效率优势¹.

Kyber 计算过程如下, 其中 sk 表示私钥、 pk 表示公钥、 m 表示明文以及 c 表示密文:

- **密钥生成:** 均匀分布采样得到矩阵 A , 二项分布采样得到向量 s 和 e , 计算 $pk = As + e$. 公开公钥 pk , 秘密持有私钥 $sk = s$.
- **封装:** 通过种子 μ 由二项分布采样得到向量 r, e_1, e_2 , 计算 $u = A^T r + e_1$, 计算 $v = pk^T r + e_2 + m$. u 和 v 分别经过压缩得到 u' 和 v' , 令密文 $c = (u', v')$.
- **解封装:** 计算 $M = v - s^T u$, M 经过压缩后得到明文 m .

Kyber 的主要计算负载分解如图 1 所示, 负载分解图中的随机数由伪随机函数产生 (pseudo-random function, PRF), 经过二项分布采样转化为噪声参与密钥生成和封装过程中计算. 此外, 密钥生成、封装和解封装过程中都需要进行大量以数论变换 (number theoretical transformation, NTT) 加速的多项式乘法运算.

¹<https://pq-crystals.org/>

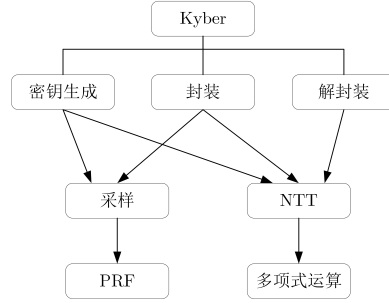


图 1 Kyber 计算负载分解

Figure 1 Kyber computing load decomposition

Kyber 采用加速多项式运算的策略是使用 NTT 将多项式从一般域转换到频域, 计算完成后结果再从频域转换回一般域. 频域中多项式乘法是以逐点乘法 (point-wise multiplication) 计算进行, 可以大大降低乘法运算的计算量. NTT 是快速傅里叶变换 (fast Fourier transformation, FFT) 在有限域上的推广, 具有 $O(n \log n)$ 渐进最快时间复杂度^[21]. 实现中 NTT 部分常以蝶形结构的迭代算法代替 NTT 的递归版本. FFT 算法构造主要由两种形式: Cooley-Tukey^[22] (时间抽取) 和 Gentleman-Sande^[23] (频率抽取) 蝶形运算^[24].

在本文讨论的全部五种 KEM 方案中, 基于格的 KEM 方案为 Kyber、NTRU 和 Saber, 其中 Kyber 的私钥较长, NTRU 较为耗时. 另一方面, 相较于其他类型的签名方案, 以 Dilithium 和 FALCON 为代表的基于格的签名方案密钥和签名长度都较短. 此外, FALCON 还具有签名和验证过程耗时短的特点. 总体来说, 基于格的密码方案综合性能优秀, 普遍内存占用量小.

2.1.2 基于杂凑的后量子密码

杂凑函数是密码学基本原语之一, 它通过单向函数将任意长度的输入转换为固定长度的输出. 而基于杂凑的签名方案在公钥密码发展的早期就已被提出. 1979 年 Lamport^[25] 提出了一种一次签名方案. 一个 OTS 方案例子如下^[26]:

给定一个杂凑函数 H , H^K 表示进行 K 次迭代杂凑计算, 令密钥 $sk = (sk_1, sk_2)$, 模 T 剩余集为 $\{0, 1, \dots, T-1\}$. 公钥是: $pk = (pk_1, pk_2) = (H^T(sk_1), H^T(sk_2))$. 将待签名消息 m 映射为模 T 剩余集中元素 $t \in \{0, 1, \dots, T-1\}$, 则其签名是:

$$\sigma = (\sigma_1, \sigma_2) = (H^t(sk_1), H^{T-t}(sk_2)).$$

验证者仅需检查:

$$(\sigma_1^{T-t}, \sigma_2^t) = (pk_1, pk_2).$$

由于 OTS 方案的一对密钥仅能对一条消息进行签名, 为了提高签名的效率, 研究者在 OTS 的基础上提出了各种多次签名方案. 这些方案可分为两类: 有状态签名和无状态签名.

有状态签名由 Merkle^[27] 于 1989 年提出, 要求使用者记录使用过的子密钥. 这样一个使用要求带来了许多操作上的不便, 例如网络中密钥被多个服务器持有, 需要维护共同的使用记录. 而无状态签名由 Goldreich^[28] 于 1986 年提出, 无需记录使用信息, 但与前者相比, 签名时间和签名长度都长得多.

2015 年 Bernstein 等人^[29] 提出了实用无状态签名方案 SPHINCS, 在具有无状态性的同时, 相较于此前的无状态签名, 签名时间和签名长度方面优势显著.

SPHINCS+^[30] 是 SPHINCS 的变体, 作为签名备选算法入围了第三轮评比, 2022 年被 NIST 选为三个标准签名算法之一. 在众多参与标准化评比的基于杂凑的后量子密码算法中, SPHINCS+ 具有无状态的突出特点 (之前在 NIST SP 800-208^[31] 中标准化的 XMSS、LMS 均为有状态的基于杂凑的密码算法). 其基本思想是使用一种称为超树 (hypertree) 的结构来认证大量 FTS (few-time signature) 密钥对. FTS 方案是一种密钥对产生少量签名 (例如, SPHINCS+ 中一个 FTS 密钥对仅能生成 10 个签名^[32]) 的

签名方案. 对消息进行签名时, 随机选择一个 FTS 密钥对来签名此消息. 则签名中包括 FTS 签名和对此 FTS 密钥对的认证信息. 此认证信息类似于超树签名, 即, 使用 Merkle 树签名的认证树的签名.

如图 2 所示, SPHINCS+ 虚拟结构 (virtual structure) 采用的超树是一个基于杂凑的 MTS (many-time signature) 树^[32]. 而这种结构被称为虚拟结构的原因是此结构在计算中并不会完整地恢复出来. 超树中限定一个 MTS 密钥对只能对 N 条消息进行签名, 这些 MTS 密钥对构成了 d 层 N 叉树. 超树顶端的第 $d-1$ 层是单个 MTS 密钥对, 它对 N 个 MTS 密钥对中的公钥进行签名, 这 N 个 MTS 密钥对就组成了超树的第 $d-2$ 层. $d-2$ 层中的每个 MTS 密钥对中每一个都对另外 N 个 MTS 公钥进行签名, 此 N^2 个 MTS 密钥对构成了超树的第 $d-3$ 层. 如此递推, 直到底层共有 N^{d-1} 个密钥对分别对 N^d 个 FTS 公钥进行签名, 得到了 N^d 个经过认证的 FTS 密钥对. 因此, 对 FTS 密钥对的认证信息包括从超树的叶子节点到根节点路径上的 d 个 MTS 签名.

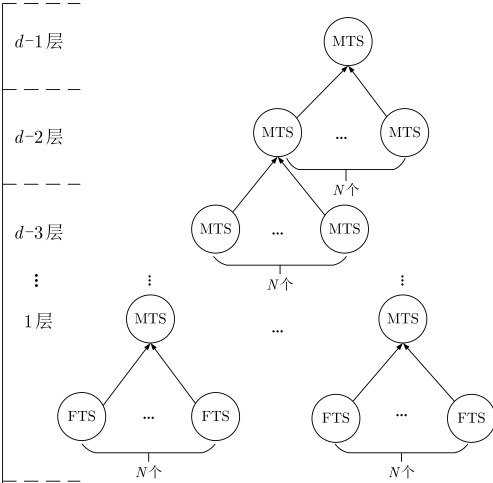


图 2 SPHINCS+ 虚拟结构
Figure 2 SPHINCS+ virtual structure

SPHINCS+ 的公钥是图 2 中根节点 MTS 的公钥, 实现中还包括与根节点同样公开种子值. 而私钥是一个秘密种子值, 通过它可伪随机生成所有 OTS 和 FTS 密钥, 这些 OTS 和 FTS 密钥共同确定了 SPHINCS+ 密钥对的虚拟结构. 实现中, 私钥还包括一个与秘密种子大小相同的附加秘密值以及公钥的副本.

如图 3 所示, SPHINCS+ 方案的结构比较复杂, 但计算种类较为单一, 计算负载集中于杂凑函数计算.

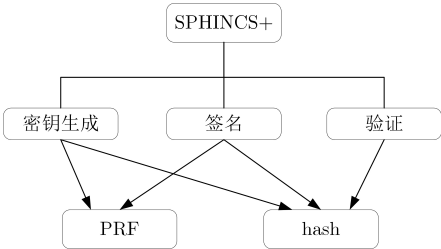


图 3 SPHINCS+ 计算负载分解
Figure 3 SPHINCS+ computing load decomposition

安全等级相同时, 与其他种类的签名方案相比, 基于杂凑的签名方案密钥较短, 但签名长度很长 (仅短于 Picnic). 实现速度方面, 密钥生成和签名过程都非常耗时.

2.1.3 基于编码的后量子密码

基于编码的密码有着几十年研究历史, 它的安全性基于编码理论中的困难问题, 例如校验子解码 (syndrome decoding) 和 LPN (learning parity with noise) 问题. 此类密码系统基于纠错码来构建具有量子抵抗性的单向函数^[33]. 数字通信中数据传输是以 0 和 1 的二进制格式进行的, 过程中需要对传输错误进行检测和纠正, 用于编码和解码的技术就是纠错码. 基于编码的密码的安全性基于解码一般线性码的困难性, 该困难问题被认为是 NP 困难问题^[34].

McEliece^[35] 于 1978 年提出了 McEliece 密码系统. 在此基础上, 基于 McEliece 密码系统的 Niederreiter 变体^[36], Albrecht 等人^[37] 提出了 Classic McElieceKEM 方案.

Classic McEliece 基于二进制 Goppa 纠错码^[38]. 其中, 公钥指定了一个随机二进制 Goppa 码, 密文是加入了随机错误的码字, 而私钥是一个具有高效纠正特定数量错误 (此数量称为该纠错码的纠错能力) 能力的字符串. Classic McEliece 通过向明文码字增加 (纠错能力内的) 错误完成加密过程, 利用私钥对密文进行纠错完成解密过程.

Classic McEliece 各个过程主要计算负载如图 4 所示, 矩阵运算是每个过程的基本运算. 密钥生成过程中的主要计算负载是生成随机数的 PRF 和将矩阵化为标准型的高斯消去法 (Gaussian reduction). 此外, 解封装过程中的主要计算是校正子计算 (syndrome computation)、求根 (root finding) 和纠错 (error correction).

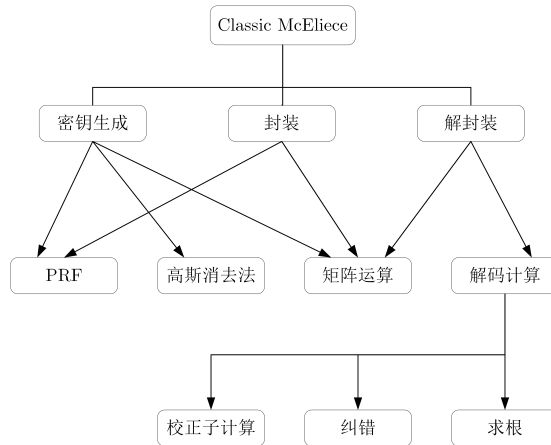


图 4 Classic McEliece 计算负载分解

Figure 4 Classic McEliece computing load decomposition

校正子计算是解码线性码中的关键计算. 对 Classic McEliece 需要使用加性 FFT (additive FFT)^[39] (n 个代入点构成一个加法群, 而不是传统的乘法群) 进行大矩阵乘法. 加法 FFT 运算是计算校正子最高效 (且时间恒定) 的方法^[40]. 实现中求根用加法 FFT, 校正子计算用转置加法 FFT^[41].

纠错是码字解码的下一阶段. 标准解码阶段是找到错误定位多项式并找到多项式的所有根 (或错误), 从而对码字进行纠错. 解码 Goppa 码需要唯一码字, 一般是使用 Berlekamp 方法^[42] 找到此码字. 然后用 Berlekamp-Massey 算法计算错误定位多项式, 用 Chien 搜索算法来找到错误定位根. 其中, Berlekamp-Massey (BM) 算法是二进制 Goppa 编码解码的关键. 该算法采用错误向量的校正子多项式 $S(x)$, 并返回错误定位多项式 $\sigma(x)$, 后续计算中可使用该多项式找到错误向量. BM 算法的描述参见文献^[43]. 算法的每次迭代中, 必须进行求有限域中逆元运算. 为了减少计算量, 可使用 Xu 提出的免求逆 (inversion-free) 算法^[44].

与其他类别的 PQC KEM 方案相比, Classic McEliece 的优势是封装和解封装较快, 密文较短. 主要劣势是密钥规模较大 (同等级长度是其他算法的五倍以上)、内存占用量非常大, 密钥生成也非常耗时. 根据 OQS 基准测试数据¹, Classic McEliece 的密钥生成过程比封装和解封装过程多耗时约三个数量级.

¹https://openquantumsafe.org/benchmarking/visualization/speed_kem.html

2.2 作为补充的后量子密码算法类别

本节将介绍的后量子密码种类都有代表性算法参与了 NIST 第三轮评选, 各自因为不同的原因遭到淘汰. 为了更全面地介绍后量子密码的发展情况, 本节将分别介绍三类后量子密码作为补充了解.

2.2.1 基于同源的后量子密码

基于同源的后量子密码的典型代表是 SIKE (supersingular isogeny key encapsulation) 算法, 它入围了 NIST 第四轮评选. 但 SIKE 的安全性基于 SIDH (supersingular isogeny Diffie-Hellman), 后者已被鲁汶大学的学者 Castryck 和 Decru 破解^[45]. 他们的破解方法 (下文简称为 Castryck-Decru 攻击) 利用了 Alice 和 Bob 进行交互时传输的额外点, 以及同源的阶是公开的这两个条件来恢复出私钥. 因此, SIKE 也被认为是不安全的. 但其他基于同源的密码系统, 例如 CSIDH^[46] 和 SQISign^[47] 等, 不具备 Castryck-Decru 攻击的先决条件, 仍保持了安全性. 故对 SIKE 实现的大量研究对其他基于同源的密码研究也有非常大的借鉴意义.

同源关系是两条椭圆曲线之间的非平凡代数映射, 也是一个群同态. 作为椭圆曲线密码的一个分支, 基于同源的密码是密码学中相对较新的领域, 近十年来备受关注. 不同于椭圆曲线密码中基于离散对数问题的密码方案 (如 ECDSA^[6]) 和其他分支, 基于同源的密码被认为具有量子抵抗性, 因此受到了密码学界的关注. 基于超奇异椭圆曲线同源计算的公钥密码体制最早见于 Jao 和 De Feo 于 2011 年提出的 SIDH 方案^[48], 其安全性依赖于寻找两条定义在有限域 $\mathbb{F}_q$ 上超奇异椭圆曲线间同源映射的困难性.

SIKE 是由 Jao 等人^[49] 于 2017 年提出的基于同源的 KEM 方案, 是基于 SIDH 通过将 Hofheinz、Hövelmanns 和 Kiltz^[50] 提出的变换应用于文献 [48] 中描述的超奇异同源公钥加密方案来构建的, 解决了由于主动攻击^[51] 造成的 SIDH 静态密钥脆弱性问题.

接下来采用与 ECDH (elliptic curve Diffie-Hellman) 相对比的方式, 简要介绍 SIKE 密钥建立协议.

图 5 (a) 展示了 ECDH 协议的工作方式. 其中, ECDH 协议的公开参数是椭圆曲线 E 和基点 G , $[k] \cdot G$ 表示基点 G 的 k 倍点计算. 协议过程如下:

Alice 和 Bob 分别选择秘密 A 和 B , 各自计算 $[A] \cdot G$ 和 $[B] \cdot G$, 经由敌手 Eve 可见的公开信道发送给对方. Alice 和 Bob 分别计算 $[B] \cdot G \cdot [A]$ 和 $[A] \cdot G \cdot B$, 得到共享秘密 $[AB] \cdot G$.

此协议的安全性基于椭圆曲线上离散对数问题, 即 Eve 无法根据 $[A] \cdot G$ 或 $[B] \cdot G$ 分别推导出 A 或 B .

图 5 (b) 展示了 SIKE 协议的工作方式. SIKE 协议使 Alice 和 Bob 能够推导出同构椭圆曲线 (isomorphic elliptic curve) E_{SharedA} 和 E_{SharedB} , 这两条曲线的 j -不变量 (j -invariant) 是他们的共享秘密.

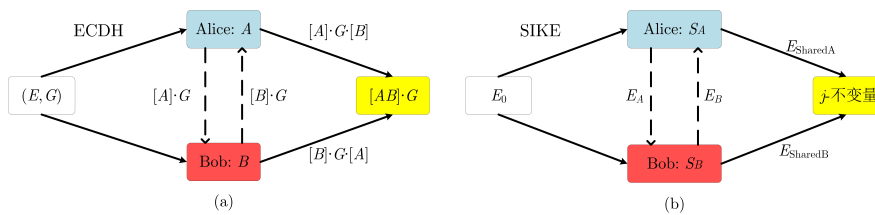


图 5 两种基于椭圆曲线的密钥建立算法^[52]

Figure 5 Two key establishment algorithms based on elliptic curve^[52]

SIKE 和 ECDH 的不同点在于:

- (1) Alice 和 Bob 之间交换且对 Eve 可见的信息是椭圆曲线 E_A 和 E_B . 它们是公开曲线 E_0 分别基于 Alice 和 Bob 的秘密 s_A 和 s_B 得到的同源 ϕ_{Alice} 和 ϕ_{Bob} . 由于椭圆曲线的 j -不变量仅与曲线表达式中一个系数有关. 而该系数可根据椭圆曲线上三个独立的点计算得出, 因此 SIKE 中以给定三个 x 坐标的方式确定或交换椭圆曲线.
- (2) Alice 和 Bob 分别使用 E_{Alice} 和 E_{Bob} 和他们的秘密 s_A 和 s_B 推导出同源 E_{SharedA} 和 E_{SharedB} , 这两条曲线彼此同构. 故他们可通过计算共享曲线的 j -不变量得到共享秘密.

此协议的安全性依赖于同源计算的单向性, 即 Eve 不能从 E_A 和 E_B 推导出得到共享秘密所必需的秘密 s_A 或 s_B 或同源 ϕ_{Alice} 或 ϕ_{Bob} .

SIKE 的主要计算负载分解见图 6, 基本运算是有限域 $\mathbb{F}_p$ 及其二次扩展有限域 $\mathbb{F}_{p^2}$ 中的运算, 其中素数 p 具有形式 $p = 2^{e_2}3^{e_3} - 1$. 而 SIKE 中密钥生成、封装和解封装过程都用到且计算量最大的函数是倍点乘法 (double-point multiplication) 和高次同源 (large-degree isogeny) ^[40] (2-同源、3-同源和 4-同源).

实现中, 已知倍点乘法的最快算法由文献 [53] 提出. 而高次同源是一个复杂计算, 它根据具体实现中预计算策略包括了大量点乘、同源评估 (isogeny evaluation) ^[49] 和同源计算运算. 目前高次同源计算代价最低的方法是 Jao 和 De Feo ^[48] 提出的乘法导向 (multiplication-oriented) 算法与同源导向 (isogeny-oriented) 算法相结合的方法. 针对基于同源的密码方案, 目前研究界的主流优化思路是根据实现基于平台的硬件特性, 优化底层的有限域运算, 包括加法、减法、乘法和模约减运算. SIDH 协议中 Alice 和 Bob 需要分别传输公钥信息给对方, 其中包括椭圆曲线 E 上两个点 P 和 Q 的 x 坐标, 长度为 $4 \log p$ (其中 p 是算法采用的素数) 比特. 在此基础上, 公钥压缩算法通过改变这两个点的表示方式来减少公钥的长度.

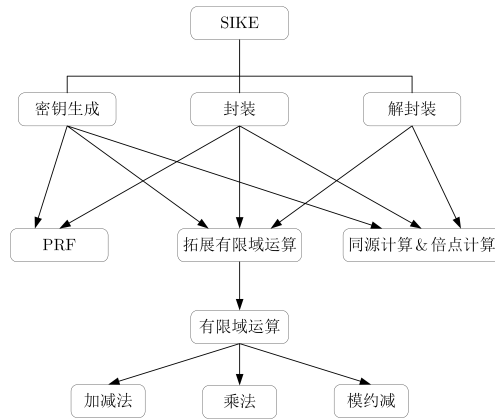


图 6 SIKE 计算负载分解 ^[52]

Figure 6 SIKE computing load decomposition ^[52]

2016 年 Azarderakhsh 等人 ^[8] 首次提出公钥压缩算法, 利用点 P 和 Q 总是扭子群 (torsion subgroup) $E[2^{e_2}]$ 或 $E[3^{e_3}]$ 的元素, 故这两个点可使用 $Z_t \oplus Z_t$ 的元素表示, 其中 $t = 2^{e_2}$ 或 $t = 3^{e_3}$. 而参数满足 $2^{e_2} \approx 3^{e_3}$, 故 $(t_1, t_2) \in Z_t \oplus Z_t$ 长度为 $2 \log p$ 位. 相较于直接表示点 P 和 Q , 此种方法减少了一半长度, 但对计算性能影响较大. 此后, 学术界提出了各种改进方法 ^[54-58], 进一步提高压缩效率、减轻对计算性能的影响.

相较于其他种类的 KEM 方案, 在相同安全等级下, 基于同源的密码方案密钥长度较短 (所有后量子 KEM 方案中 SIKE 公钥最短), 但它需要大量点运算来计算高阶同源, 因此性能较差 ^[59].

2.2.2 基于多变量的后量子密码

众所周知, 方程的线性系统是可高效求解的问题. 然而, 随着多项式数的增加, 变量数也逐渐增多, 对这样的系统要么无解, 要么难以求解. 基于这个前提, 经过密码学界的广泛研究, 确定了求解多变量多项式方程问题的困难性. 在此基础上构建的密码算法安全性基于多变量二次多项式问题, 被证明在任何域上都是非确定多项式时间, 且传统和量子计算机都难以求解 ^[60].

基于多变量的后量子密码的典型代表是 Rainbow 签名算法, 它入围了 NIST 第三轮评选. Rainbow 是一个由 Ding 等人 ^[61] 于 2005 年提出的基于多变量的签名方案. 它是以 Kipnis 等人 ^[62] 提出不平衡的油醋 (unbalanced oil and vinegar, UOV) 签名架构为基础. 算法中形成了油变量和醋变量交叠的两层结构, 类似彩虹, 故得名“彩虹 (rainbow)”签名, 算法简要描述如下:

算法通过两个仿射映射 ($\mathcal{S}: \mathbb{F}^m \rightarrow \mathbb{F}^m$ 和 $\mathcal{T}: \mathbb{F}^n \rightarrow \mathbb{F}^n$) 对油变量和醋变量进行混合, 起到隐藏秘密二次映射 $\mathcal{F}: \mathbb{F}^m \rightarrow \mathbb{F}^n$ 的作用. 而这两个仿射映射 $\mathcal{S}$ 和 $\mathcal{T}$, 实现中采用了两个可逆矩阵: $\mathcal{S} \in \mathbb{F}^{m \times m}$ 和 $\mathcal{T} \in \mathbb{F}^{n \times n}$.

签名者随机生成 $\mathcal{S}$ 、 $\mathcal{F}$ 和 $\mathcal{T}$, 计算它们的复合 $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, 秘密保存私钥 $\text{sk} = (\mathcal{S}, \mathcal{F}, \mathcal{T})$, 公开公钥 $\mathcal{P}: \mathbb{F}^n \rightarrow \mathbb{F}^m$.

如图7所示, 签名时, 签名者计算消息的杂凑值 $\text{digest} \in \mathbb{F}^m$, 随后计算 $x = \mathcal{S}^{-1}(\text{digest})$ 和 $y = \mathcal{F}^{-1}(x)$, 最后计算签名 $\text{sig} = \mathcal{T}^{-1}(y) \in \mathbb{F}^n$. 验证时, 验证者代入签名 $\mathcal{P}(\text{sig}) = \text{digest}'$, 将计算结果 digest' 与消息摘要 digest 相比较, 判断签名是否正确.

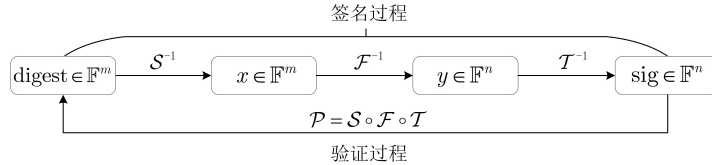


图7 Rainbow 的签名和验证过程

Figure 7 Signature and verification process of Rainbow

Rainbow 的负载分解如图8所示, 基本运算是矩阵运算, 计算量最大的函数是线性系统和变量代入. 其参考实现¹中使用了 Karatsuba 算法^[63] 加速矩阵乘法, 以高斯消去法实现解线性系统.

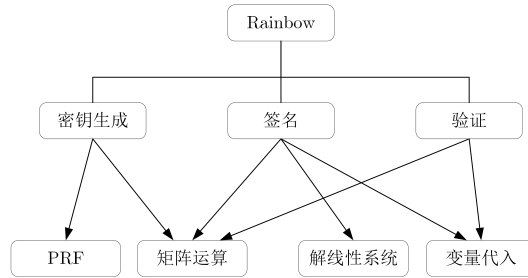


图8 Rainbow 计算负载分解

Figure 8 Rainbow computing load decomposition

在第三轮评选结果出炉的前几个月, 研究者提出了一种对 Rainbow 的有效攻击方法^[64] (下文称为 Beullens 攻击). 根据论文中实验数据, 利用 Beullens 攻击的方法, 敌手可利用一台笔记本电脑以 53 个小时攻破使用安全等级 1 参数集的 Rainbow 签名方案. Beullens 攻击的提出使 Rainbow 原各参数集的安全性显著降低 (算法设计者指出原安全等级 5 的参数集考虑此攻击后的安全性仅为具有安全等级 1 的安全性). 与参与标准化的其他后量子签名算法相比, 经过调整后的新参数集的密钥长度和实现性能均丧失了原有优势, 使得 Rainbow 算法在该轮评比中被淘汰.

不考虑 Beullens 攻击对安全性的影响时, 与其他种类的签名方案相比, Rainbow 签名方案的优势包括速度、计算要求低和签名短, 主要劣势是公钥规模大 (比其他种类的签名算法至少大三个数量级) 和内存占用量大. 考虑 Beullens 攻击对安全性的影响时, 经过安全性校正的 Rainbow 各参数集在密钥、签名长度和实现性能方面性能均没有优势.

2.2.3 基于零知识证明的公钥密码

基于零知识证明的后量子密码的典型代表是 Picnic 签名算法, 它是 NIST 第三轮评选的备选算法.

零知识证明 (zero-knowledge proof, ZKP) 是 Goldwasser 等人^[65] 于 1989 年提出的概念, 此后广泛应用于身份鉴别、电子现金、电子投票、数字签名等场景. 典型的零知识协议是证明者和验证者双方之间的交互式协议. 这些协议中, 证明者可向验证者证明他拥有一个秘密消息, 但不泄露秘密消息本身的信息.

以 Picnic 为例, 介绍基于零知识证明的密码方案. Picnic^[66] 是一个由微软研究团队和其他研究团体设计的新颖签名方案, 它是建立在零知识证明和对称密码原语 (杂凑函数和分组密码) 之上, 且具有后量子安全性的数字签名方案.

¹<https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/Rainbow-Round3.zip>

Picnic 签名方案中, 签名者生成一串随机比特作为私钥 sk , 另一串随机比特作为明文 p , 用私钥 sk 以对称加密算法加密明文 p , 得到密文 c , 公开明文和密文作为公钥 $pk = (p, c)$. Picnic 中使用了 ZKB++ 零知识协议^[66] (基于 ZKBoo 协议^[67] 改进了效率), 此协议通过单向函数 f 以非交互方式将原象 x (即私钥 sk) 的知识证明为公开值 y (即公钥中的密文 c). ZKB++ 协议采用 Ishai 等人^[68] 提出的“MPC-in-the-head”方法. 为了证明对 x 的知识, 证明者 (签名者) 模拟几个参与方 (ZKB++ 中是 3 个) 之间的 MPC (multi-party computation, 多方安全计算), 每个参与方收到 x 的份额, 计算 $f(x) = y$. 然后证明者承诺各个参与方不同内部状态 (又称视图 (view)), 验证者要求公开一部分参与方的承诺来进行挑战, 通过揭示视图来检验承诺的正确性. 重复进行多轮上述过程, 当证明者不知道 x 或进行欺骗时, 不同轮之间参与方的共同视图不会总是一致, 从而确保了 MPC 协议的正确性. 而 MPC 协议的秘密性使得公开参与方的部分视图不会泄露关于 x 的任何信息, 因此进行签名过程不会使得私钥 sk 遭到窃取.

Picnic 签名方案的构建基于 ZKP 协议和对称原语 (分组密码和杂凑函数), 故这种签名方案与其他基于特定数学困难问题的大多数公钥密码方案都有所不同. 而且这种方案具有模块化的特点, 实例化时选择的具体对称原语和 ZPK 协议都对密钥、签名长度和实现效率有很大影响.

如图 9 所示, Picnic 的主要负载集中于杂凑函数和对称加密计算. 签名和验证阶段中需要重复多次利用三个子密钥加密明文 (对安全等级 1/3/5, 分别需重复 219/329/418 次). 具体而言, Picnic 实现中以 LowMC 实例化对称密码. LowMC^[69] 是一个于 2015 年提出的参数化分组密码. 为了提高效率, 此密码的特色是可以表示为具有与门 (AND gate) 和异或门 (XOR gate) 的布尔电路, 而非算术电路. 此外, LowMC 设计使得它以很少的 AND 门实例化, 进一步提高了性能. LowMC 被选为 Picnic 的分组密码主要是因为所有可能的候选方案 (例如 AES) 中, 它在每个安全等级上对 AND 门数的需求数量最小^[70].

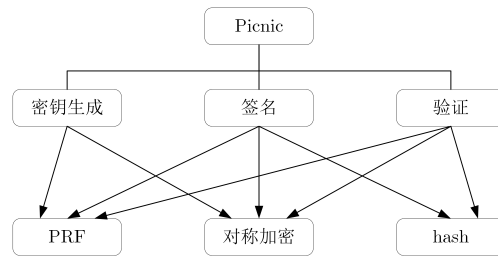


图 9 Picnic 计算负载分解

Figure 9 Picnic computing load decomposition

与其他种类的签名方案相比, 安全等级相同时, 基于零知识证明的签名方案密钥很短 (安全等级 1/3/5 的私钥为 16/24/32 字节, 公钥为 32/48/64 字节), 但签名长度非常长 (安全等级 1/3/5 的签名长度为 32.8/74.1/128 kB), 将会对通信造成很大负担. 此外, 此类签名方案的密钥生成较快, 签名和验证过程消耗时间和内存量都很大.

2.3 各类后量子密码参数和性能比较

各类后量子 KEM 和签名方案的参数和性能特征分别总结如表 3 和表 4 所示, 其中“密码参数”和“实现速度”两栏中数据均摘自 OQS 基准测试 (x86_64-ref 实现, 入选算法的测试日期是 2023-11-21, 参考算法的测试日期是 2022 年), “主要计算负载数据举例”列中数据取自文献中研究者的实验数据, 时钟周期简称为 CC (clock cycle). 其中, kCC 表示 10^3 CC, mCC 表示 10^6 CC, gCC 表示 10^9 CC.

表 3 中用于举例的参数集的安全等级都为 3 级 (对 SIKE 暂不考虑 Castryck-Decru 攻击). Kyber 的举例参数集是 Kyber768, Classic McEliece 的举例参数集是 Classic-McEliece-348864, 以及 SIKE 的举例参数集是 SIKE-p610.

表 4 中用于举例的参数集的安全等级都为 3 级 (对 Rainbow 暂不考虑 Beullens 攻击). Dilithium 的举例参数集是 Dilithium3, Rainbow 的举例参数集是 Rainbow-III-Classic, SPHINCS+ 的举例参数集是 SPHINCS+-SHAKE-192s-simple 以及 Picnic 的举例参数集是 picnic_L3_FS.

本文总结各类后量子密码方案的性能特点中 x86 平台性能数据取自开放量子安全 (open quantum

表 3 不同类型 KEM 方案参数及性能比较
Table 3 Parameters and performance comparison of different types of KEM schemes

类别	KEM 方案	密码参数	实现速度	内存占用量	除杂凑外主要 计算负载 (基本运算)	主要计算负载 数据举例
格	Kyber	密钥对长度中等, 例如一组参数集的公钥和私钥长度分别为 1.18 kB 和 2.40 kB	快, 例如一组参数集密钥生成、封装和解封装分别耗时 176 kCC、207 kCC 和 239 kCC	较少, 例如一组参数集密钥生成、封装和解封装分别占用最大堆为 5.16 kB、5.67 kB 和 5.70 kB	NTT (多项式运算)	基于 ARM-A 参考实现中密钥生成/封装/解封装分别占总实现耗时的 24.7%/33.5%/41.6% [71]
编码	Classic McEliece	密钥对长度很长, 例如一组参数集的公钥和私钥长度分别为 524 kB 和 13.6 kB	密钥生成尤其慢, 例如一组参数集密钥生成、封装和解封装分别耗时 1.14 gCC、44 kCC 和 120 kCC	很大, 例如一组参数集密钥生成、封装和解封装分别占用最大栈堆为 542 kB、547 kB 和 542 kB	BM 算法, 加法 FFT 和转置加法 FFT (多项式运算)	文献 [41] 中 vec 类实现 mceliece348864, BM 算法/求根的加法 FFT/校正子计算的转置加法 FFT 分别耗时 1.25 m/833 k/630 kCC
同源	SIKE	密钥对长度较短, 例如一组参数集公钥和私钥长度分别为 462 B 和 524 B	慢, 例如一组参数集密钥生成、封装和解封装分别耗时 80.1 mCC/147 mCC/148 mCC	较少, 例如一组参数集密钥生成、封装和解封装分别占用最大堆为 5.16 kB、5.67 kB 和 5.70 kB	同源计算 (主要涉及有限域 F_q 中运算), 另外杂凑计算占实现总耗时不高于 0.1%	Cortex-A55 上 F_q 求逆运算耗时 1.57/4.47 mCC
多变量	/	/	/	/	/	/
杂凑	/	/	/	/	/	/
ZKP	/	/	/	/	/	/

safe, OQS) 项目¹的基准测试数据, ARM 平台性能数据取自 pqm4 后量子密码库²中基准测试数据 (OQS 和 pqm4 基准测试项目将在第 3 节被介绍).

2.4 各类后量子密码算法对随机数的需求情况

软件实现密码算法时, 随机数的长度和数量对实现的安全性和性能也具有很大的影响. 本节根据各算法的说明文档和参考实现, 总结了各类后量子密码算法对随机数的需求情况如表 5. 注意, 这里的随机数是由随机数生成器直接产生的, 不包括由杂凑函数生成的随机序列.

3 基于平台特性的软件优化技术

3.1 实现平台

为了客观比较各后量子密码方案的软件性能, NIST 推荐平台是“运行 Windows 或 Linux 支持 GCC 编译器的 64 位 Intel 处理器” [7]. 对嵌入式平台, 推荐的则是 ARM Cortex-M4 微控制器. 但是由于 Cortex-M4 属于 ARM 为低性能需求设计的 M 系列, 无法支持基于嵌入式平台的高性能实现, 故研究者还提出了大量基于 ARM-A 系列微控制器的实现, 例如 A53 [75]、A57 [76]、A72 [77]、A75 [71], 以及 Apple M1 [78]. 除此之外, 还有一些研究后量子密码方案软件实现的文献针对的是 RSIC-V [79] 和 GPU [80] 平台.

¹<https://openquantumsafe.org/benchmarking/>
²<https://github.com/mupq/pqm4/blob/master/benchmarks.md>

表 4 不同类型签名方案密码参数及性能比较

Table 4 Parameters and performance comparison of different types of signature schemes

类别	签名方案	密码参数	实现速度	内存占用量	除杂凑外主要 计算负载 (基本运算)	主要计算负载 数据举例
格	Dilithium/ FALCON	长度中等, 例如 一组参数集的公 钥、私钥和签名 长度分别为1.95 kB、4.02 kB 和 3.29 kB	中等, 例如一组参 数集密钥生成、签 名和验证分别耗时 476 kCC、1.88 mCC 和 447 kCC	很少, 例如一组参 数集密钥生成、签 名和验证过程分别 占用最大堆为12.7 kB、15.4 kB 和 10.8 kB	NTT 或 Toom-Cook算 法/Karatsuba 算法 (多项式运算)	NTT和 NTT^{-1} 占实现总耗时的 约 3.4% 和 24.5% [72]
杂凑	SPHINCS+	密钥对短签名长, 例如一组参数集 的公钥、私钥和 签名长度分别为 48 B、96 B 和 17.1 kB	签名尤其慢, 例如 一组参数集密钥生 成、签名和验证分 别耗时169 mCC、 1.5 gCC 和 1.6 mCC	少, 例如一组参数 集密钥生成、签名 和验证过程分别占 用最大堆为 8.96 kB、44.7 kB 和 44.7 kB	除杂凑计算外 无其他主要计 算负载	杂凑计算占实 现总耗时 85% 以上
多变量	Rainbow	密钥对长签名短, 例如一组参数集 的公钥、私钥和 签名长度分别为 611 kB、861 kB 和 1312 B	密钥生成尤其慢, 例如一组参数集密 钥生成、签名和验 证分别耗时 805 mCC、45.4 mCC 和 47.1 mCC	大, 例如一组参数 集密钥生成、签名 和验证过程分别占 用最大堆为 1.51 MB、1.51 MB 和 1.51 MB	变量代入和解 线性系统 (矩阵运算)	变量代入和解 线性系统分别 占实现总耗时的 58.12% 和 36.75% [73]
ZKP	Picnic	密钥对短签名长, 例如一组参数集 的公钥、私钥和 签名长度分别为 49 B、73 B 和 27.2 kB	密钥生成快, 签名 和验证慢, 例如一 组参数集密钥生 成、签名和验证分 别耗时 29.2 kCC、 21.1 mCC 和 17.6 mCC	签名和验证过程占 用大量内存, 例如 一组参数集密钥生 成、签名和验证过 程分别占用最大堆 为 8.90 kB、182 kB 和 126 kB	对称加密	签名过程中 LowMC 加密占实现 总耗时的 63.61% [74]
编码	/	/	/	/	/	/
同源	/	/	/	/	/	/

3.2 基准测试项目

除了安全性以外, 密码算法的标准化项目中最重要衡量指标就是性能. 为了客观公平分析各算法性能优劣, 基于各算法设计者提供的代码和文献中提出的改进实现, 密码学界的一些研究者和组织开展了基于各平台后量子密码定量基准测试 (benchmark) 工作, 如开放量子安全 (open quantum safe, OQS)、pqm4 和 eBATS 等.

OQS 是一个旨在发展和规范化后量子密码算法实现达到后量子安全的开源项目 [81]. 该项目支持提交到 NIST 标准化项目且声称具有量子抵抗性的 KEM 和签名方案. OQS 由两部分组成: 纯 C 代码库 liboqs 和已经集合到协议和应用 (例如广泛应用的 OpenSSL 库) 的代码.

liboqs 是一个后量子安全密码算法的 C 语言开源库, 其中后量子算法实现源自 NIST 后量子标准化项目中各方案设计者提供的参考代码, 提供支持 x86 和 ARM 架构的量子安全 KEM 和数字签名算法的开源实现和相应通用 API 以及测试工具和基准测试程序.

pqm4 是一个专门针对 NIST 嵌入式推荐平台 ARM Cortex-M4 的开源后量子密码库. pqm4 库、基准测试和测试框架是欧盟委员会在 H2020 计划中资助的 PQCRYPTO 项目的成果. 值得注意的是, pqm4 基准测试数据中有一项数据是杂凑运算占总实现耗时的比例, 从中可看出: 由于构造方式不同, 即便同种类型的方案之间杂凑运算占比差异很大; 优化后实现杂凑运算耗时占比大幅提高, 由此可见杂凑运算是算法进一步优化的重要因素之一.

eBACS (ECRYPT benchmarking of cryptographic systems) 项目中的 eBATS (ECRYPT benchmarking of asymmetric systems) 也包括了目前各种后量子密码方案的基准测试.

表 5 各类型后量子密码算法所需随机数情况
Table 5 Random numbers required for various types of post-quantum cryptographic algorithms

类别	方案	阶段	需要随机数的长度 (字节)
格	Kyber	KeyGen	64
		Encaps	32
	Dilithium	KeyGen	32
		Sign	确定签名模式: 0; 随机签名模式: 48×4 (合法签名的签名次数期望)
杂凑	SPHINCS+	KeyGen	48 (SPHINCS+-128s/f)/72 (SPHINCS+-192s/f)/96 (SPHINCS+-256s/f)
		Sign	16 (SPHINCS+-128s/f)/24(SPHINCS+-192s/f)/32 (SPHINCS+-256s/f)
编码	Classic McEliece	KeyGen	32
		Encaps	128 或 256 (mceliece348864(f))/192 或 384 (mceliece460896(f))/
			238 或 476 (mceliece6960119(f))/
			256 或 512 (mceliece6688128(f), mceliece8192128(f))
同源	SIKE	KeyGen	44 (SIKEp434)/56 (SIKEp503)/64 (SIKEp610)/80 (SIKEp751)
		Encaps	16 (SIKEp434)/24 (SIKEp503, SIKEp610)/32 (SIKEp751)
多变量	Rainbow	KeyGen	32 (Classic 模式)/64 (Circumzenithal 模式, Compressed 模式)
		Sign	0
零知识	Picnic	KeyGen	32 (picnic-L1-FS/UR)/48 (picnic-L3-FS/UR)/64 (picnic-L5-FS/UR)
		Sign	0 (说明文档中随机种子在参考实现中由杂凑产生)

这些公开基准测试项目对大量算法的多种参数集进行了测试, 提供了大量详细的实现数据可供研究者比较分析.

3.3 SIMD 优化

单指令流多数据流 (single instruction multiple date, SIMD) 是软件实现加速的基本技术, 也是目前后量子密码软件实现的主流技术. 具体而言, SIMD 是指由微处理器根据一条指令同时对一组数据 (data vector, 数据向量) 中的每一个分别执行相同的操作从而实现空间上的并行性的技术.

目前 x86 平台支持的 SIMD 指令集 (又称向量指令集) 有 SSE 系列和 AVX 系列, 目前大部分 x86 平台软件实现针对性能较强的 AVX、AVX2 和 AVX512 指令集. 而 ARM 平台支持的 SIMD 指令集主要是 NEON 指令集.

密码方案软件实现的基础是纯 C 代码, 在此基础上根据特定平台应用 SIMD 汇编指令集利用处理器内部的并行性减少存取等基本指令冗余提高算法的实现效率. 具体而言, x86 平台上是利用 AVX/AVX2/AVX512 指令集对原代码进行优化. 而 ARM 平台上是根据具体微控制器的性能, 选择将多个数据打包到一个寄存器中处理^[82]或利用 NEON 指令集^[77]优化.

然而第三轮各算法的提交文档中只有少数兼有 x86 和 ARM 两个平台 SIMD 优化实现. 随着时间推移, 研究者们不断提出了大量基于各平台的优化实现, 说明目前各候选算法的性能还有较大提升空间.

3.4 扩展指令集优化

在 SIMD 优化的基础上, 目前还有一些优化实现利用了一些扩展指令集.

AES-NI: AES-NI (advanced encryption standard new instructions) 指令集^[83]是一个 x86 指令集的扩展, 从 2010 年 Intel Core 处理器系列开始支持. AES-NI 被设计来加速 AES 的加密和解密, 并提高侧信道攻击抵抗能力. Kyber 设计者在说明文档中也提出了 Kyber 的“90 年代”变体, 此方案使用由

NIST 标准化的对称原语: AES 和 SHA-256, x86 平台的 AES-NI 指令集对此变体实现有显著加速效果. 此外, SPHINCS+ 设计者在说明文档中指出, 当以 Haraka 实例化时, 支持 AES-NI 的测试平台性能更佳.

GF-NI: GF-NI (Galois field new instructions) 指令集^[84]是在第三代 Gen Intel Xeon 可扩展处理器中引入的指令集, 被设计为加速密码和安全应用. 具体而言, GF-NI 指令集加速 GF(2⁸) 中乘法和求逆运算. GF-NI 包括 VGF2P8MULB、VGF2P8AFFINEQB 和 VGF2P8AFFINEINVB (分别简称为 MULB、AFFINEB 和 ADDINEINVB) 指令. 其中 MULB 执行 GF(2⁸) 中向量化乘法, AFFINEB 和 ADDINEINVB 执行 GF(2⁸) 中向量化仿射变换和仿射逆变换. 具体实现中, Drucker 等人^[85]利用 GF-NI 加速了 Rainbow 签名方案.

CLMUL: CLMUL (carry-less multiplication) 指令集^[86]是 Intel 于 2010 年提出的 x86 指令集的扩展. CLMUL 对有限域 GF(2) 或 GF(2^k) 上多项式上进行无进位式乘法. 无进位乘法是在不生成或传播进位值的情况下计算两个操作数的 (无进位) 乘积的数学运算. 在此指令集提出之前, 最佳方法 (根据 OpenSSL 中代码) 计算一次 64×64 位无进位乘法需要消耗约 100 时钟周期^[86]. 无进位乘法是一些密码系统的基本计算组件. 因此, 利用 CLMUL 指令集加速无进位乘法可显著加速用到此计算的密码实现. 例如 Ceria 等人利用 CLMUL 优化了 McEliece 密码系统的密钥生成过程^[87].

3.5 针对嵌入式平台的优化

当以嵌入式设备为基础实现后量子密码方案时, 平台的硬件资源限制了很多面向高性能的优化技术应用. 为了适应这种资源受限的环境, 研究者提出了大量有针对性的创新优化技术.

3.5.1 计算性能优化

指令对齐: ARMv7E-M 中一些可用指令是 16 位 Thumb 指令, 故可能出现单个 Thumb 指令与后续 32 位 ARM 指令没有对齐的情况, 严重影响性能. Botros 等人^[82]指出应使代码尽可能对齐, 具体方法是在函数开始处使用 .align 2 对齐 (.align n 功能是对 2ⁿ 字节对齐), 在单个 Thumb 指令使用 .w 后缀填充为 32 位.

链接时间优化: 链接时间优化是控制优化的一个重要选择, 代价是会增加代码量. 一般来说, 没有链接时间优化时, 每个源文件都以某种优化级别编译以生成不同的目标文件. 然后这些经过优化的目标文件链接到一起组成可执行文件. 虽然此方法对源代码优化得很好, 但是当启用链接时间优化 (-flto) 时, 链接器可达到更高的性能, 一般性能增幅约为 10%. 而这性能增益源于跨模块函数内联. 另一方面, 代码量增加是因为跨源文件内联函数会带来代码重用. 值得注意的是, 确定代码未使用或代码对输出没有影响的时候, 链接时间优化更有效. pqm4 平台没有使用 -flto 作为默认选项是因为它会增加栈使用量或导致某些方案计算更慢. 文献 [88] 中对 Kyber、NewHope 和 NewHope-COMPACT 实现进行 -flto 效果测试, 实验结果显示都有性能提升.

循环展开: 由于嵌入式微控制器寄存器数量有限 (ARM Cortex-M4 仅有 14 个可用通用寄存器), 这意味着在用于存储并行数据的寄存器之外, 每节约一个寄存器都能提高并行性. 展开最外层循环, 节约循环计数器占用的寄存器是提高性能的通行方法^[82,88,89]. 而文献 [88] 不为循环变量分配寄存器, 提出了两种方法. 方法一是仅简单地使用 .rept 预编译器指令, 方法二是将循环计数器溢出到栈中. 第一种方法实现速度较快, 但因为 .rept 仅重复相同代码, 会大量增加代码量, 适用于内存充足的应用. 第二种方法代码量影响不大且仍可在每个循环中加载更多系数, 适用于内存资源受限的应用.

乱序执行流水线: ARMv8 高性能内核例如 Cortex-A57、Cortex-A72 和 Cortex-A73 支持乱序超标量流水线, 这种结构支持 A64 和 ASIMD 指令单元的并行执行. Jalali 等人^[76]利用乱序执行流水线提出了一种使用混合 A64/ASIMD 汇编以及纯 A64 汇编实现多精度乘法的新方法.

3.5.2 内存使用优化

后量子密码软件实现的主流优化目标是提高性能. 除此之外, 一些种类的后量子密码 (例如基于编码的密码) 实现需要占用大量内存空间, 因此不适合资源受限平台 (例如 ARM Cortex-M4), 需要进行内存优化.

文献 [90] 提出了两种减少内存使用量的方法:

方法一, 每次从位串中解包出系数计算成本较高. 与之相反, 当位串被解码为允许轻松访问数据的字节序列时, 在对系数执行算术运算之前, 可以有效地从一个字节中扩展出符号, 此过程使用指令 sxtb 且耗

时单个时钟周期. 在执行矩阵向量乘法和向量点乘法时, 该技术将处理秘密所需的堆栈减半.

方法二, 为了进一步减少堆栈使用量, 也可以以更高的循环次数为代价使用即时拆包. 因为秘密的每个系数的位数是处理器字大小的因子, 打包位串中的系数以恒定偏移量访问. 因此, sbfx 指令可用于每时钟周期解包一个系数. 在解包两个连续系数后, 可使用 pkhXX 将它们重打包为半字来保持利用 DSP 指令进行运算. 此技术将 Saber 的三个参数集的堆栈使用量减低为 256、384 和 512 字节.

此外, 常用的减少内存使用量的方法还有杂凑的簿记 (book-keeping for the hashing)、公共矩阵中即时生成多项式和应用算法的就地 (in-place) 版本 (文献 [91] 中使用就地 Karatsuba 算法, 减少了内存占用). 这些优化不引入执行时间额外开销, 故可系统应用到所有实现.

3.5.3 时间和内存之间的权衡

基于微控制器实现 Dilithium 算法, 提高性能的最大难点在于需要多次调用软件实现耗时的 SHAKE256 来计算矩阵 A (Dilithium 算法中一个核心元素), 而矩阵 A 会在签名过程中多次使用. 因此, 对于实现中如何处理矩阵 A 的问题, 文献 [72] 中提出了三种权衡策略:

- 将矩阵 A 整体存于闪存中: 这种策略将节约大量 SRAM, 适用于小型设备;
- 每次签名的时候拓展矩阵 A : 这种策略会增加栈空间使用量, 适用于 SRAM 足够的设备;
- 动态计算 A 和 y (算法中一个多项式) 来节约内存: 这种策略优化了堆栈使用, 但性能较慢.

3.5.4 栈空间优化

资源受限平台上实现密码算法时, 栈空间也是需要考虑的重要制约因素. 文献 [82] 指出 Kyber 实现中, 基于 NTT 的多项式乘法本身是原地的, 无需额外内存空间, 但方案中的其它部分需要进行栈空间优化. 具体而言, 提出了以下几点:

- 对 Kyber.CCAKEM 的改进: Kyber 使用 FO 变换将 CPA 安全的 PKE 转换为 CCA 安全的 KEM. 此过程体现在解封装中先解密密文, 再重加密得到的明文. 如果重加密得到的密文与原密文相同, 则返回共享密钥. 研究者通过以恒定时间方式将此比较过程内联到 CPA 加密过程来避免堆栈上的额外密文. 此改进不仅减少了栈空间使用, 而且提升了少许速度.
- 对 Kyber.CPAPKE 的改进: 密钥生成和加密过程中, 公开矩阵 $\hat{A}$ (矩阵 A 的 NTT 域表示) 中 $k \times k$ 个多项式是先直接在 NTT 域中采样, 存于内存, 再进行后续计算. 研究者合并了采样和计算, 即仅在每次计算需要时, 采样对应部分.
- 对 $\hat{A}$ 的改进: $\hat{A}$ 占用了大部分栈空间, 但 $\hat{A}$ 仅在矩阵-向量逐点乘加重使用一次. 研究者提出了一种减少内存占用的方法. 在此方法中, 输出向量 t 和 u 多项式一次序列化一个, 并将向量操作数 $\hat{s}$ 和 $\hat{r}$ 保留在内存中. 此方法将 $\hat{A}$ 的存储要求降低为一次仅存储 $\hat{A}$ 中一个多项式的可扩展输出函数的状态, 仅生成乘法所需的少量系数.
- 对加噪声的改进: 与 $\hat{A}$ 类似, 噪声多项式 e 、 e_1 和 e_2 仅在算法中使用一次, 可通过在需要用到这些噪音的计算中动态采样多项式中的系数, 避免存储完整多项式.

4 基于主流平台的后量子密码实现

后量子密码软件实现基于的主流架构是 x86 和 ARM. 基于 x86 处理器相对硬件资源丰富性能较强, 且具有大量可用指令集. 而基于 ARM 架构的实现又分为两种, 分别基于 ARM-A 系列架构 (包括基于 ARMv7-A 架构的 Cortex-A8 和 Cortex-A15 等微控制器和 ARMv8-A 架构的 Cortex-A53、Cortex-A55、Cortex-A75 等微控制器和苹果 M1 处理器) 和 ARM-M 系列架构 (包括基于 ARMv7E-M 的 Cortex-M0、Cortex-M3 和 Cortex-M4). 基于 ARM-A 系列架构的处理器一般性能较高且支持 SIMD 指令集 NEON, 而基于 ARM-M 系列架构的处理器硬件资源受限, 可用指令较少. 这三种实现根据硬件条件和支持的指令集不同, 实现方法存在差异. 本节中可移植 C 实现指: 各密码方案设计者提供的、没有使用以任何基于特定平台的 SIMD 指令的 C 语言程序实现; AVX 实现指: 基于 x86 架构处理器, 以 AVX 指令作为 SIMD 技术应用的汇编程序实现, AVX2 实现和 AVX512 实现同理; NEON 实现指: 基于 ARM-A 架构处理器, 以 NEON 指令作为 SIMD 技术应用的汇编程序实现. 由于不同研究者的优化目标存在差异, 所提出的优

化实现参考的基准各不相同. 因此本文的第4节和第5节在介绍各优化实现时, 侧重介绍研究者提出的新方法和新技术, 略去了大部分实验数据. 请有意深入研究的读者, 参阅对应参考文献以获取详细实验数据.

4.1 基于格的密码实现

除杂凑计算以外, 基于格的密码的主要计算负载是多项式运算. 根据加速多项式运算的具体技术, 可将此类密码方案分为两类: 一类是 NTT 类, 利用 NTT 作为基本加速技术, 对 NTT 计算方法的改进技术可推广到此类中其他方案, 典型方案包括 Kyber、NewHope、Dilithium 和 FALCON; 另一类是非 NTT 类, 以 Toom-Cook 算法^[92,93] 和 Karatsuba 算法^[63] 作为基本加速技术, 典型方案包括 NTRU 和 Saber. 由于目前研究界对基于格的密码实现进行了大量研究, 故本节在介绍这些研究成果的时候先以基本加速技术、再以优化对象是 KEM 或签名方案, 最后以实现的基础硬件平台分类.

4.1.1 NTT 类方案实现

NTT 类方案软件实现的主流优化方法是根据基础平台的硬件特点 (寄存器的数量、内存大小和支持的指令集等) 优化 NTT 的实现细节, 包括模约减、蝶形运算和层合并等.

为了更好说明文献中起到加速效果的具体优化技术, 现对 NTT 计算简要介绍如下 (多项式环采用 Kyber 中定义为例): q 是一个素数, 以 R_q 表示环 $\mathbb{Z}_q[X]/(X^n + 1)$, 其中 $n = 2^{n'-1}$ 满足 $X^n + 1$ 是 $2^{n'}$ 次分圆多项式.

令 $a, b \in R_q$, 则

$$a = \sum_{i=0}^{n-1} a_i x^i, a_i \in \mathbb{Z}_q,$$

则 NTT 变换定义如下

$$\begin{aligned} \text{NTT}(a) = \hat{a} &= \sum_{i=0}^{n-1} \hat{a}_i x^i, \hat{a}_i = \sum_{j=0}^{n-1} a_j \omega^{ij} \pmod{q}, \\ \text{NTT}^{-1}(\hat{a}) = a &= \sum_{i=0}^{n-1} a_i x^i, a_i = \frac{1}{n} \sum_{j=0}^{n-1} \hat{a}_j \omega^{-ij} \pmod{q}, \end{aligned}$$

其中 ω 称为旋转因子 (twiddle factor) 是模数 q 的 n 阶原根.

由于 NTT 域中乘法是以逐点乘法的方式进行, 则可通过加速下式加速一般域中的多项式乘法运算.

$$a \cdot b = \text{NTT}^{-1}(\text{NTT}(a) \cdot \text{NTT}(b)).$$

对正向 NTT 变换计算

$$\hat{a}_i = \text{NTT}(a_i) = \sum_{j=0}^{n-1} a_j \omega^{ij} \pmod{q},$$

当 $k = 0, 1, \dots, n/2 - 1$ 时

$$\begin{aligned} \hat{a}_k &= \sum_{r=0}^{\frac{n}{2}-1} a_{2r} \omega^{2rk} + \sum_{r=0}^{\frac{n}{2}-1} a_{2r+1} \omega^{(2r+1)k} = X_k + \omega^k Y_k, \\ \hat{a}_{k+\frac{n}{2}} &= \sum_{r=0}^{\frac{n}{2}-1} a_{2r} \omega^{2r(k+\frac{n}{2})} + \sum_{r=0}^{\frac{n}{2}-1} a_{2r+1} \omega^{(2r+1)(k+\frac{n}{2})} \\ &= \sum_{r=0}^{\frac{n}{2}-1} a_{2r} \omega^{2rk} \cdot \omega^{2nr} + \sum_{r=0}^{\frac{n}{2}-1} a_{2r+1} \omega^{(2r+1)k} \cdot \omega^{\frac{(2r+1)n}{2}} \\ &= X_k + \omega^{k+\frac{n}{2}} Y_k \\ &= X_k - \omega^k Y_k. \end{aligned}$$

计算 $\hat{a}_k$ 和 $\hat{a}_{k+n/2}$ 的这种方法称为蝶形 (butterfly) 运算, 包括一次加法、一次减法和一次与预计算常数 (旋转因子的幂) 的乘法. 蝶形运算是 NTT 计算的主要计算瓶颈^[94]. 当 $n = 8$ 时 NTT 的计算过程如图 10.

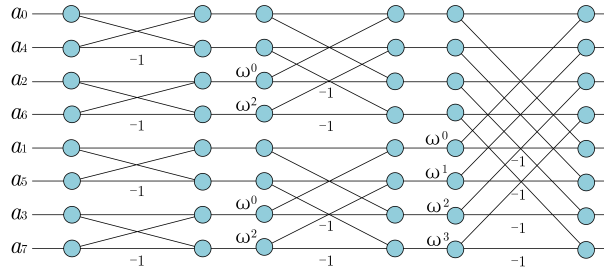


图 10 NTT 计算过程

Figure 10 NTT calculation process

从图 10 中可见, FFT 算法含有一个重排序的过程, 此过程将各个系数按序号的位反转 (bit reverse) 重新排序. 具体而言, 以 Cooley-Tukey 蝶形算法实现 NTT 时, 输入数据是位逆顺序 (bit-reversed order) 的, 变换后数据是正常序 (natural order) 的; 而以 Gentlemen-Sande 蝶形算法实现 NTT 时, 输入数据是正常顺序的, 输出数据是位逆顺序的. 基于这两种蝶形算法的特点, Pöppelmann 等人^[95] 提出以 Cooley-Tukey 蝶形算法实现正向 NTT, 以 Gentlemen-Sande 蝶形算法实现逆向 NTT, 省去了实现中变换前或变换后重排序的步骤.

基于格的密码软件优化实现数量众多且各自优化方法一般都与实现的基础平台关系密切, 故本节主要围绕最为流行的 Kyber KEM 方案, 分别介绍目前研究界提出基于不同硬件平台的优化实现.

基于 x86 架构处理器的实现

2013 年 PQC 会议上, Güneysu 等人^[96] 首次提出将多项式表示为双精度浮点数序列, 显著加速了 NTT 计算 (当模数为 23 位素数时, 仅用了 4480 时钟周期实现了 512 次多项式的 NTT). 此后这种表示方法成为主流, NewHope^[94]、NTRU Prime^[97] 和 Kyber^[20] 等基于格的密码方案在提出时即采用了这种 NTT 加速技术.

2016 年 CANS 上, Longa 和 Naehrig^[98] 提出了一种针对 NTT 所需特殊模数设计的新模约减技术, 这种模约减技术使得系数可增长到 32 位, 从而使得 NTT 计算期间加法后无需进行模约减. 作者使用此技术加速了 NTT 计算, 提出了可移植 C 和 AVX2 的优化实现.

2016 年 USENIX Security 上 Alkim 等人^[94] 提出了后量子密钥交换算法 NewHope, 其中包括了算法的可移植 C 实现和优化的 AVX2 实现: C 实现中考虑简便性和可移植性, 将多项式中系数表示为无符号 16 位整数; 而 AVX2 实现中遵循文献^[96] 的做法将系数表示为浮点数. 此外, NTT 中以 Montgomery 约减^[99] 和 Barrett 约减^[100] 的结合完成蝶形运算中模约减.

2018 年 Inscrypt 上 Zhou 等人^[101] 提出了一种预处理多项式再进行 NTT 的技术, 称为 PtNTT (Preprocess-then-NTT). PtNTT 将 n 次多项式分为几个较低次多项式, 分别对这几个多项式进行 NTT. 他们将此技术应用于 Kyber 和 NewHope, 提出了新参数设置, 以微弱性能下降的代价缩短了公钥长度.

2018 年 Seiler^[102] 提出了 NTT 整数实现, 相较于此前以浮点数表示多项式系数的主流实现了五倍速度提升. 这样的加速效果源于作者将多项式系数表示为 16 位有符号整数, 同时提出了针对 16 位整数的向量化 Montgomery 约减和 Barrett 约减算法. 此工作提出后, 如 NewHope 和 Kyber 等适用 NTT 的基于格的后量子密码算法官方实现中都改用整数表示. 此外, 此工作提出的 FFT 技巧 (FFT trick) 也在后续工作中得到了广泛应用.

2019 年 CHES 上, Lyubashevsky 和 Seiler^[103] 提出了使用 NTT 的基于 NTRU 的 KEM 方案. 他们在文献^[102] 的基础上提出了一种 Montgomery 约减优化方法, 可推广到各种 NTT 计算. 此方法改进了模约减策略并利用更多的预计算常数将一次 $\mathbb{Z}_q$ 乘法所需的 16 位整数乘法数从 4 降为了 3. 另外, 它改进了文献^[102] 中 Montgomery 约减, 且利用了 AVX2 中仅计算乘积的高半部分和低半部分的指令, 提

高了并行性, 进一步提高了 NTT 实现性能.

基于 ARM-A 系列微控制器的实现

2017 年 Streit 和 De Santis^[75] 提出 New Hope 的首个基于 ARM-A 平台实现, 利用 128 位向量 NEON 指令向量化 NTT. 他们对 NTT 计算提出了一种改进约减方法, 此方法类似于 Barrett 约减, 而无需将中间变量扩展为更大的数据类型. 另外, 此工作在 NIST PQC 征集之前, 提出了“合并 NTT 层”结构.

2018 年 Hoang 等人^[104] 基于 ARMv8 架构微控制器, 利用可伸缩向量扩展 (scalable vector extension, SVE) 指令集提出了 Kyber 的优化实现. SVE 架构由 ARM 于 2016 年提出, 在 2018 年时还没有实现, 作者在 ARM 网站上利用编译器和模拟器, 并在 ARMv8-A 上取得了性能优化.

2021 年 Sanal 等人^[71] 提出了基于 ARMv8 的 Kyber 实现, 他们利用了 64 位 ARM 处理器支持更长寄存器 (128 位) 提高了向量化 NTT 和约减并行性取得加速效果.

2021 年 PQC 会议上 Nguyen 等人^[77] 提出了首个基于 ARMv8 利用 NEON 指令集优化实现. 他们指出基于 ARMv8 平台利用 NEON 实现性能略低于基于 x86 平台利用 AVX2 实现的原因是缺少与 AVX2 中 vmulhw (仅计算整数乘积的高位的向量指令) 功能类似的指令.

2022 年 CHES 上, Becker 等人^[78] 基于 ARMv8 架构提出了 Kyber、Saber 和 Dilithium 的优化实现. 其中核心创新点是作者提出了一种基于 ARMv8-A 架构的新约减方法. 此方法通过结合 Montgomery 乘法和 Barrett 约简, 充分利用了 Neon 向量指令的计算能力. 此外作者还提出了“非对称乘法”, 以缓存不完全 NTT 的结果的方式优化多项式矩阵-向量乘法.

基于 ARM-M 系列微控制器的实现

2016 年 Alkim 等人^[89] 基于 NewHope 的参考实现^[94] 将 NewHope 移植到 ARM-M 系列微控制器上. 针对 NTT 计算, 利用可用的指令集减少了 NTT 中的内存访问并汇编优化了 NTT 的三个子程序: 逐点乘法、位逆序化多项式的参数和 NTT 本身.

2019 年 Africacrypt 上, Botros 等人^[82] 基于 Cortex-M4 提出了 NTT 优化和减少 RAM 使用量而不明显影响性能的技术, 利用了 Cortex-M4 平台上 DSP 扩展和桶式移位器. ARMv7E-M 架构提供了 DSP (digital signal processing, 数字信号处理) 指令, 此指令可对并行对两个 16 位半字进行算术运算, 在之前的工作^[105,106] 中作为重要的加速技术. 而桶式移位器使得移位或轮换算术指令中参数不额外消耗时钟周期. 根据文献 [102] 提出的方法, 作者以有符号 16 位整数表示多项式的系数, 一次在寄存器中载入四个系数, 执行四次蝴蝶运算, 通过仅在需要时载入约减算法常数和展开最外层循环的方法, 节省寄存器资源, 提高了并行性.

2020 年 CHES 上, Alkim 等人^[88] 提出了 ARM Cortex-M4 上对 Kyber、NewHope 和 NewHope-COMPACT 的优化实现, 性能提升源于三点创新: 更高效的模约减、优化的低次多项式乘法和更积极的层合并. 这三点具体实现方法分别为, 将文献 [102] 中三周期 Montgomery 约减算法改进为二周期; 低次多项式乘法运算时应用懒约减; 通过不将循环计数器存储于寄存器将节省下来的寄存器资源用于存储更多系数来促进 NTT 层合并.

2022 年 ACNS 上, Abdulrahman 等人^[107] 提出了基于 Cortex-M4 的 Kyber 和 Dilithium 的优化实现. 作者首先研究了文献 [78, 102, 108, 109] 中提出对 Saber、NTRU 和 NTRU Prime 的优化技术, 并将它们应用到 Kyber 和 Dilithium 的实现中. Kyber 方面, 作者改进了文献 [82] 中 Barrett 约减, 将每次批量约减需要 8 个时钟周期改进为 6 个. Dilithium 方面, 作者提出一种切换到更小模数方法来计算算法中间乘积, 有效提高了计算效率. 值得注意的是, 作者开源了他们的实现代码, 此代码随后被 pqm4 采纳为基准实现代码.

另一方面, 以 Dilithium 为例, 介绍对基于格的签名方案优化实现.

2020 年 CARDIS 上, 基于 x86 和 Cortex-M4 平台, Ravi 等人^[110] 针对 Dilithium 的签名生成过程, 通过对条件检查 (conditional check) 的早代入 (early-evaluation) 来尽可能早地中断会被拒绝的采样迭代, 节约计算资源. 此外, 他们还利用预计算加速了签名生成过程中几种特定情况.

2021 年 CHES 上 Greconici 等人^[72] 提出了基于 Cortex-M3 和 Cortex-M4 的 Dilithium 优化实现. 作者优化了 Dilithium 中 NTT 计算: Cortex-M4 平台上, 作者在文献 [111] 的基础上将多项式系数的表

示由无符号整型改为有符号整型, 免去了无符号整型减法后为避免溢出而加上模数的倍数; 在上述基础上, 他们提出了基于 Cortex-M3 的恒定时间实现. 此外, 根据 Dilithium 签名算法的迭代特性, 需要在速度和栈使用量之间作出权衡, 对此作者给出了三种不同具体策略.

2022 年 Kim 等人^[112] 提出了首个基于 ARMv8 嵌入式微控制器的 Dilithium 优化实现. 该实现主要关注 Dilithium 中 NTT 环节的性能. 在节约内存的方面, 作者使用合并层的方法最小化了 NTT 中 0-2 层的内存访问, 利用 NEON 向量寄存器来减少 NTT 中 2-7 层的内存访问. 另一方面, 他们根据 NEON 引擎和 ARM 处理器的硬件特点设计了蝶形计算高效方法.

4.1.2 非 NTT 类方案的实现

与 NTT 类方案类似, 非 NTT 类方案软件实现的主流优化方法针对的是 Toom-Cook 算法和 Karatsuba 算法^[90,91,113].

值得注意的是, 2021 年 Chung 等人^[114] 提出之前普遍认为不适用 NTT 的环结构也可以应用 NTT 加速, 实现了 NTT 加速 Saber 和 NTRU 方案实现. 从此以后, 这些被算法设计者认为不适用 NTT 的算法, 也可以选择使用 NTT 完成多项式乘法.

首先, 介绍基于 x86 架构处理器的优化实现.

NTRU 算法的实现方面, Danba^[113] 提出了 AVX2 优化实现, 将 NTRU 中基本运算—512 系数 (511 次) 多项式乘法利用 Toom-Cook 算法、Karatsuba 算法层层分解为可并行的 4 系数 (3 次) 多项式, 如图 11 所示. 最终, 经过分解后得到的低次多项式以教科书乘法 (schoolbook multiplication) 通过 AVX2 向量指令并行计算. NTRU 中一个系数是 16 位的, AVX2 指令长为 256 位, 故一条指令可进行 16 次乘法运算.

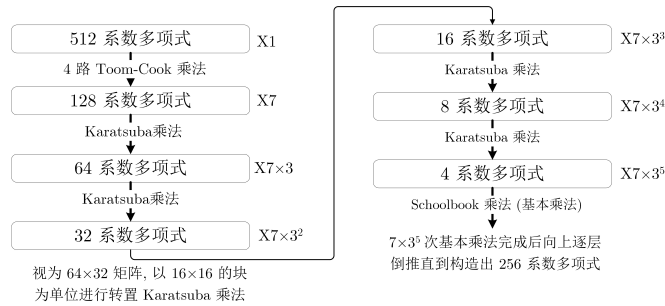


图 11 NTRU 中多项式分解过程^[113]

Figure 11 Polynomial decomposition process in NTRU^[113]

Saber 算法的实现方面, 2018 年 D'Anvers 等人^[19] 提出时即包括 AVX2 实现. 2019 年 Roy^[115] 为提高服务器场景下实现的吞吐量, 提出了同时进行 4 个 Saber KEM 的 SaberX4 实现. 此实现使用向量批处理指令批处理 4 个 Saber KEM, 通过使用 AVX2 优化 SHAKE-128x4 (四次 SHAKE-128 调用的批处理) 产生随机数、并行 4 次杂凑运算, 多项式乘法以既有 AVX2 实现交替实现的方式, 以增加单个 KEM 运行时间为代价, 提高实现总体吞吐量.

2020 年 CHES 上, Mera 等人^[90] 利用 Toom-Cook 的乘法预处理和后处理可表示为线性变换, 提出了两种新加速技术, 分别称为“预计算”和“懒插值” (lazy interpolation). 他们将此两种技术应用于 Saber, 加速了 AVX2 和 C 实现中矩阵-向量乘法. 此外, 他们还针对基于 Cortex-M4 微控制器实现提出了几种减少内存占用的方法, 详见 3.5.2 节中“内存使用优化”.

接下来, 介绍基于 ARM-A 系列微控制器的优化实现.

2021 年 PQC 会议上 Nguyen 等人^[77] 根据文献^[114] 的研究, 基于 ARMv8 利用 NEON 指令集在传统 Toom-Cook 算法之外, 还使用了 NTT 两种技术加速多项式乘法. 根据论文作者实验得到的性能数据, NTT 实现和 Toom-Cook 实现速度相近, 作者建议基于 ARMv8 继续使用 Toom-Cook 作为主要加速技术.

最后, 介绍基于 ARM-M 系列微控制器的优化实现.

2018 年 CHES 上 Karmakar 等人^[91]提出了 Saber 的首个资源受限平台实现,从速度和内存两个方面进行了优化.速度优化方面,他们利用 DSP 指令优化频繁调用的教科书乘法,利用 ALU (arithmetic and logic unit, 算术逻辑单元) 尽可能多地存储系数以减少内存读取.内存优化方面,他们使用内存高效的 Karatsuba 算法和需要时生成公开矩阵的策略减少实现的内存占用.

除了针对多项式乘法的优化之外,采样也是一部分基于格的密码方案的重要计算过程,故对采样过程的优化技术具有一定的通用性.2016 年 Gueron 和 Schlieker^[116]利用 AVX2 进一步向量化了采样过程,提出了一种降低伪随机数拒绝率的技术.

4.1.3 小结

后量子密码研究领域中,基于格的密码近些年一直是研究热点,研究者提出了大量优化实现.本节介绍的实现简要总结如表 6 所示.

表 6 基于主流平台的基于格的密码的软件优化实现总结
Table 6 Summary of software optimization and implementation of lattice-based cryptography based on mainstream platforms

x86 架构	ARM 架构	
	ARM-A	ARM-M
2013 年 Güneysu 等人 ^[96] NTT 优化实现, 首个双精度浮点数实现	2017 年 Streit 等人 ^[75] NewHope 的 NEON 优化实现改进约减技术	2016 年 Alkim 等人 ^[89] NewHope 的资源受限实现
2016 年 Longa 等人 ^[98] NTT 优化实现, 改进约减技术	2018 年 Hoang 等人 ^[104] Kyber 的 SVE 优化实现, 利用 SVE 架构优化 NTT	2018 年 Karmakar 等人 ^[91] Saber 的性能和内存优化实现
2016 年 Alkim 等人 ^[94] NewHope 的 C 和 AVX2 实现, 改进约减技术	2021 年 Sanal 等人 ^[71] Kyber 的 64 位平台优化实现, 利用 128 位寄存器加速了 NTT、噪声采样和对称函数	2019 年 Botros 等人 ^[82] Kyber 优化实现, 利用 DSP 指令加速了 NTT
2018 年 Zhou 等人 ^[101] NTT 优化实现, 缩短公钥技术	2021 年 Nguyen 等人 ^[77] Kyber, NTRU 和 Saber 的 NEON 优化实现, 三算法恒定时间实现和基准测试	2020 年 Ravi 等人 ^[110] Dilithium 优化实现优化拒绝迭代
2018 年 Seiler ^[102] NTT 优化实现, 首个高性能整数实现	2022 年 Kim 等人 ^[112] Dilithium 优化实现, 基于 ARMv8 架构特点改进 NTT 计算方法	2020 年 Alkim 等人 ^[88] Kyber 和 NewHope 的优化实现, 改进约减、乘法和 NTT 技术
2019 年 Lyubashevsky 等人 ^[103] NTT 优化实现, 改进约减技术	2022 年 Becker 等人 ^[78] Kyber, Saber 和 Dilithium 的优化实现改进约减技术	2020 年 Mera 等人 ^[90] Dilithium 优化实现时间-内存权衡技术
2019 年 Danba 等人 ^[113] NTRU 的 AVX2 优化实现多项式乘法拆分技术		2021 年 Greconici 等人 ^[72] Dilithium 优化实现采用有符号多项式
2019 年 Roy 等人 ^[115] Saber 的 AVX2 高吞吐量实现		2022 年 Abdulrahman 等人 ^[107] Kyber 和 Dilithium 的优化实现移植现有技术改进约减技术

4.2 基于编码的密码实现

基于编码的密码方案 Classic McEliece 实现时性能的特点之一是密钥生成过程非常耗时.针对这个问题, Ceria 等人^[87]将密钥生成过程中计算负载进行拆解, 选择了几个重点运算: 多项式乘法、高斯消去法、线性映射计算和线性映射应用进行优化, 取得了整体加速效果.

另一方面, 由于 Classic McEliece 的公钥长度非常大 (安全等级 1/3/5 时, 公钥长度为 261/524/1044 kB), 通常被认为不适合嵌入式设备.2021 年, Chen 等人^[41]提出了基于 ARM Cortex-M4 的恒定时间实现, 将公钥存储在闪存中处理, 解决内存容量不足的问题.

4.3 基于同源的密码实现

相较于其他种类的后量子密码, 基于同源的密码速度较慢 (耗时比基于格的密码多三个数量级), 故此类后量子密码更需要速度优化. 此类密码实现的主要优化对象是有限域运算, 如表 7, 包括有限域中的乘法、约减、加法和减法运算.

表 7 SIKE 中各种有限域运算的数量 [76]
Table 7 Number of various finite field operations in SIKE [76]

参数集	安全等级	乘法	约减	加法	减法
SIKEp503	2	195 889	149 138	56 978	83 142
SIKEp751	3	307 946	234 253	88 764	131 618
SIKEp964	5	408 786	310 707	117 666	172 910

由于 NIST 征集对不同平台的实现评估以及对平台特定的优化¹, 研究者提出了基于不同平台的大量优化实现, 包括 x86 架构处理器、32 位 ARMv7 Cortex-A 处理器 [117, 118]、64 位 ARMv8 Cortex-A [87, 118] 和 32 位 ARMv7E-M Cortex-M4 [106, 119, 120]. 本节根据公开发表顺序, 对提出优化实现的文献简要介绍如下:

2016 年美密会上 Costello 等人 [121] 基于 x86 平台使用投影坐标公式提出了 SIDH 的首个恒定时间实现. 此实现仿射坐标公式表示改为投影坐标表示, 不仅使得实现时间恒定, 还使得同源计算机中无需求有限域逆元, 消除了 SIDH 实现的部分性能阻碍.

2016 年 CANS 上 Koziel 等人 [122] 针对 ARMv7-A 架构利用 NEON 指令集优化了有限域算术, 将所有有限域运算都在 Montgomery 域中进行以利用极为高效的 Montgomery 约减. 作者比较了 SIDH 设计者使用的仿射同源公式和文献 [121] 中提出的投影同源公式, 实验数据显示对非恒定时间实现而言, 在 ARMv7 平台上仿射同源公式速度更快.

2017 年 Jalali 等人 [123] 提出了基于 ARMv8-A 架构的 SIDH 高效实现. 他们根据逆乘比 (inversion-to-multiplication ratio) 对 SIDH 实现中仿射和投影公式的使用进行了分析, 指出在不考虑实现安全的情况下, 仿射 SIDH 速度显著更快; 投影 SIDH 对密钥交换协议整体性能更佳. 在更大的有限域上, 与仿射公式相比, 使用优化的域乘法会使得投影的性能更好. 他们对 SIDH 的两个参数集提出了基于 64 位 ARMv8 的优化实现, 缩小了与基于 Intel 处理器实现之间的性能差距.

2018 年 Koppermann 等人 [119] 研究了基于资源受限平台的 SIDH 实现. 作者提出了基于 32 位 Cortex-M4 和 16 位 MSP430 的 SIDH 优化实现并进行了性能基准测试. 此外, 作者指出即使是经过优化的实现, 完成一次临时密钥交换也需耗时 18 秒 (基于 Cortex-M4), 需要更进一步的性能优化, 否则此方案对小型嵌入式设备是不现实的.

2018 年 SPACE 上 Jalali 等人 [117] 提出基于 32 位 ARMv7-A 架构的恒定时间 SIKE 优化实现. 主要性能提升源于处理器 Cortex-A8 和 Cortex-A15 具有 32 个 128 位宽的 NEON 寄存器, 可通过 d 或 q 标识来分别处理 64 或 128 位的数据, 减少内存读写次数. 在上述基础上, 作者通过汇编代码加速了有限域运算. 此外, 由于 Cortex-A8 流水线顺序执行指令, 性能较低, 故其相应实现侧重降低能耗. 而 Cortex-A15 的内核是乱序变长流水线, 其相应实现侧重性能提升.

2018 年 CHES 上 Seo 等人 [118] 研究了 SIDH 和 SIKE 在 32 位 ARMv7-A 和 64 位 ARMv8-A 平台上的实现. 基于 32 位 ARM-A 处理器及其支持的 NEON 向量扩展, 他们使用 Karatsuba (用于多精度乘法) 或教科书乘法 (用于 Montgomery 约简) 将运算拆分为更小的子乘积, 然后分别以 ARM 和 NEON 指令交错处理, 最大化指令集并行 (instruction level parallelism, ILP) 从而减少流水线停顿数和内存访问次数. 此外他们结合了 UMAAL 指令和一种混合扫描 (hybrid-scanning) 方法, 提出了一种新 Montgomery 约减技术. 基于 64 位 ARMv8-A 处理器, 乘法指令耗时和简单指令 (例如加法) 耗时相较于低性能处理器

¹<https://groups.google.com/a/list.nist.gov/forum/#!topic/pqc-forum/nteDiyV66U8>

更大, 为了充分利用硬件性能, 作者设计了一种广泛利用 Karatsuba 乘法, 最小化使用乘法指令的多精度乘法。

以上优化 SIKE 的工作或是使用纯 SIMD 指令, 或是基于 32 位 ARM 平台混合利用 128 位 NEON 向量和 32 位通用寄存器, 结合使用 SIMD 和通用寄存器的资源。此外, 这研究中实现域乘法采用的具体方法是基于 Comba^[124] 和两层 Karatsuba 乘法

2019 年 Jalali 等人^[76] 基于 64 位 ARMv8 系列处理器, 利用硬件的乱序执行流水线特性, 提出了混合 A64/ASIMD 汇编以及纯 A64 汇编实现来实现多精度乘法的新方法。此外, 他们对 SIKE 的 $\mathbb{F}_{p^{503}}$ 和 $\mathbb{F}_{p^{751}}$ 参数提出了一层 Karatsuba 乘法实现, 对 $\mathbb{F}_{p^{964}}$ 参数采用文献 [123] 中提出的性能更高的两层实现。

Seo 等人于 2019 和 2020 年于三篇论文里提出了分别基于嵌入式低端^[120,125] 和高端平台^[126] 的 SIKE 优化实现:

基于 Cortex-M4, 他们优化了 SIKE 实现的多个关键运算:

- (1) 针对多精度乘法运算, 提出了改进版操作数缓存 (refined operand caching, R-OC) 方法, 优化可用寄存器, 增加了操作数规模; 使用 UMUALL 和 UMAAL 指令重新设计了乘积扫描的前部, 优化了显式寄存器初始化; 重排乘法累加程序, 消除指令之间的数据延迟。
- (2) 针对多精度平方运算, 利用 SIKE 参数中模数小于 32 位字的倍数的特点, 将一次平方运算分为三部分, 分别是一次使用 R-OC 方法的子乘法运算和两次使用 SBD (sliding block doubling, 滑动块加倍, 在文献 [127] 中提出) 方法的子平方运算。
- (3) 针对模约减运算, Montgomery 乘法包括乘法和 Montgomery 约减, 此前实现中这两者由不同函数实现, 作者将其整合到一个函数中并优化中间结果的内存访问。

基于 64 位 ARMv8 平台^[126], 他们提出了针对 NIST 第二轮中新出现的参数集 SIKEp434 和 SIKEp610 的优化实现。他们对有限域加减法运算, 使用隐藏的 (masked) 模约减方法在不进行条件判断的情况下进行模约减; 对乘法运算, 根据较短的模数 (434 位) 使用与此前实现不同的逐行乘法 (row-wise multiplication); 对模约减运算, 将尽可能多的中间结果存储在寄存器中并以交错的方式实现 Montgomery 乘法, 减少内存访问。此外, 他们还利用模数移位 (shifted modulus) 减少部分乘积数, 从而减少约减运算次数。

2021 年 Anastasova 等人^[106] 提出了基于 Cortex-M4 的 SIKE 优化实现。针对有限域中加减法运算, 他们通过加法和减法的交替序列 (alternating sequence), 减少对进位/借位捕捉器 (carry/borrow catcher) 操作的需求; 采取“操作块翻转 (operation block flipping)”的策略节约寄存器资源。此外, ARM Cortex-M4 执行内存访问指令需要 2 周期, 执行其他指令只需要 1 周期。根据这一点, 作者提出以浮点寄存器模拟 1 级缓存, 利用访问浮点寄存器仅需 1 时钟周期减少指令延迟。基于对浮点寄存器的创新开发, 作者优化了有限域中的乘法、约减和平方运算。例如, 针对乘法运算根据参数集中素数的不同, 提供了两种基于浮点寄存器的策略: 算法开始时, 以将操作数值载入到浮点寄存器中; 将浮点寄存器用作 1 级缓存, 存储乘法的中间结果。这些细节优化减少了内存访问次数, 取得了实现性能的提升。

本节中介绍的基于同源的密码实现众多, 小结如表 8。

4.4 基于多变量的密码实现

多项式乘法是基于多变量的密码的基本运算, 也是大多数软件实现主要优化对象。基于 x86 平台, Drucker 等人^[85] 利用 Intel 的新指令 AVX512-GFNI 指令提升了计算效率。基于相对性能较高的嵌入式平台 ARMv8, Kwon 等人^[128] 利用预计算的乘法表实现了相较于 Karatsuba 算法实现最多 167.2 倍的加速。除此之外, 基于资源受限的 Cortex-M4, Chou 等人^[129] 利用位切片 (bitslice) 技术实现了恒定时间 $\mathcal{F}_{16}$ 乘法。

基于多变量的密码中主要计算负载集中于变量代入和解线性系统操作。对此, Shim 等人^[73] 提出以基于舒尔补 (Schur complement) 的三角分解作为解线性系统新方法, 相较于传统高斯消去法速度更快。他们还指出变量代入和解线性系统过程中预计算可极大提高计算效率。

最后, 文献 [129] 中还提出了优化验证的新方法: Rainbow 的验证过程是一个变量代入方程组的过程, 通过将乘法替换为 256 元素查找表并跳过其中一个乘数为 0 的乘法运算取得加速效果。

表 8 基于主流平台 SIDH 和 SIKE 软件优化实现总结
Table 8 Summary of software optimization based on mainstream platforms SIDH and SIKE

年	实现平台	文献	简介
2016	x86	[121]	首个 SIDH 恒定时间实现
2016	ARM-A	[122]	SIDH 的 NEON 优化实现, 改进模约减
2017	ARM-A	[123]	基于 ARMv8 的 SIDH 优化实现, 分析和比较两种 SIDH 计算方法
2018	ARM-M	[119]	基于 Cortext-M4 的 SIDH 基准测试, 指出 SIDH 在嵌入式设备上实现性能问题突出
2018	ARM-A	[117]	SIKE 的 NEON 优化实现, 基于 Cortex-A8 和 Cortex-A15 基准测试
2018	ARM-A	[118]	SIDH 和 SIKE 的 NEON 优化实现, 优化多精度和模运算减少流水线停顿和内存访问
2019	ARM-A	[76]	基于 ARMv8 处理器的 SIKE 优化实现, 并行采用 A64 和高级 SIMD 处理单元
2019	ARM-M	[125]	基于 Cortext-M4 的 SIKE 优化实现, 利用汇编改进了模运算
2020	ARM-M	[120]	基于 Cortext-M4 的 SIKE 优化实现, 同作者基于 19 年的工作进一步提升了性能
2020	ARM-A	[126]	基于 ARMv8 处理器的 SIKE 优化实现, 减少了有限域运算总数
2021	ARM-M	[106]	基于 Cortext-M4 的 SIKE 优化实现, 改进了低层有限域算术和拓展域操作

4.5 基于杂凑的密码实现

基于杂凑的密码算法具有核心组件 (杂凑算法) 可灵活替换的特点, 因此以不同的杂凑算法实例化时, 基于杂凑的密码实现性能不同. Kölbl 等人^[130] 基于 ARM Cortex A57 比较了以不同杂凑算法实例化的性能差异, 指出根据实现平台对选定实例化的杂凑算法是否具有硬件支持对性能影响很大.

基于杂凑的密码的特点是签名长和签名时计算量大, 而一些资源受限平台没有足够内存容纳全部基于杂凑的签名. 针对这种情况, 研究者提出了对 SPHINCS+ (以及前身方案 SPHINCS) 应用流式签名的技术: Hülsing 等人^[131] 基于 16 kB RAM 的 Cortex-M3 处理器实现了 SPHINCS-256 的 41 kB 签名计算, 表明通过将 SPHINCS-256 签名分解几个相对较大的部分, 分别流式传输, 可以在内存资源不足的情况下生成密钥、生成签名和验证签名. Gonzalez 等人^[132] 基于仅 8 kB RAM 研究包括 SPHINCS+ 的几种 PQC 签名算法的签名验证实现. 在此基础上, Niederhagen 等人^[133] 基于 ARM Cortex-M4 将签名流方法扩展到密钥生成和生成签名.

4.6 基于零知识证明的密码实现

基于零知识证明的签名方案 Picnic 特点是仅使用对称密钥原语和结构高度模块化. 这就使得 Picnic 构建模块和参数选择非常多样. 针对 Picnic 的这一特点, Kales 等人^[74] 对各种参数选择进行了基准测试并提出了具有更短签名和签名时间的改进 MPC 协议. 针对 Picnic 实现占用大量内存, 难以适应资源受限平台的问题, Winkler 等人^[134] 提出了生成签名过程的内存优化技术:

按需生成种子和盐值: 种子和盐值是以杂凑计算生成的, 作者以 AES 代替推荐实现中 SHAKE128/SHAKE256, 充分利用部署于大多数 IoT 设备或智能卡的 AES 加速器.

不缓存承诺的改进挑战生成过程: 根据杂凑函数的输入值不同顺序不影响安全性的特点, 改变输入值的拼接顺序, 避免缓存全部 MPC 轮中的承诺值.

签名生成时将部分签名流式传输到客户端: 如 2.2.3 节所述, Picnic 签名方案缺点是签名长度很长, 为了节约内存占用, 签名生成时即输出部分签名, 从而可以释放一部分内存占用.

5 基于其它平台的后量子密码实现

5.1 基于 GPU 的实现

GPU 是一种为了处理图形渲染的大规模并行工作负载而设计大规模并行计算架构设备, 包含数以千计的处理器核心, 因此具有并行处理能力强和吞吐量大的特点. 尽管 GPU 被设计于专门处理图形, 它特别的并行架构也可以用于加速更一般的计算任务, 这就是 GPGPU (general-purpose computing on graphics processing units) 的概念.

GPGPU 早在 2000 年左右, 研究人员就开始尝试使用 GPU 进行非图形计算. 然而, 由于当时的 GPU 主要针对图形处理进行优化, 其编程模型和接口并不适合通用计算. 直到 2007 年, NVIDIA 为 GPU 增加了一个易用的编程接口, 即 CUDA 编程模型. CUDA 允许编程人员忽略底层图形概念, 专注于高性能计算技术, 大大降低了 GPGPU 的技术门槛. 此后, 各种依赖高算力的技术 (如蓬勃发展的区块链和神经网络技术) 对于算力的旺盛需求进一步吸引技术人员研究如何将 GPU 的强大计算能力来处理更广泛的计算负载, 这其中就包括应用到后量子密码领域. 接下来介绍各个基于 GPU 的后量子密码实现研究工作.

韩国嘉泉大学的 Lee 团队长期研究基于 GPU 的后量子密码算法实现, 发表了多篇论文:

- 2021 年该团队基于 GPU 研究并比较了 NTT 和 Nussbaumer 算法这两种实现多项式乘法的方法, 提出一种可避免 GPU 中 warp (一组连续线程) 之间差异 (关键性能瓶颈) 的 NTT 实现方法^[135]. 他们将所提出的技术应用于优化 qTESLA (一个基于格的签名方案), 并指出该技术也适用于其它基于格的密码系统.
- 同年该团队针对此前工作^[135] 存在的一些不足: 无法优化次数 $n < 1024$ 的多项式和不同的 NTT 层之间存在差异, 提出了 Kyber 基于 GPU 的优化实现^[136], 应用了四种优化技术: 合并层并行 NTT、并行拒绝采样、合并内存访问的中心二项式分布和并行化细粒度 AES-256.
- 2022 年该团队提出了利用点乘指令加速矩阵乘法和多项式卷积计算的技术^[80]. 作者注意到 NVIDIA 在它的 Pascal 架构 GPU 中引入 DP4A 和 DP2A 点乘指令, 而点乘指令可在一个时钟周期内完成乘加计算, 显著快于传统 32 位整数单元. 而后量子密钥封装方案 FrodoKEM 和 Saber 中需要进行大量点乘, 基于此特性作者利用点乘指令提出了这两种算法的 GPU 优化实现.
- 同年, 该团队还针对 NTT 类的基于格的密码算法, 提出应用 Tensor Core 的 GPU 优化实现^[137]. 此外, 中国科学院大学的 Zheng 团队也在此研究领域内开拓了基于 GPU 的 AI 加速器的性能潜力.

- 2021 年 SecureComm 上, 该团队提出了首个利用 AI (artificial intelligence) 加速器的基于格的密码优化实现^[138]. 利用 Tensor Core (NVIDIA 推出的 AI 加速器) 的强大计算能力, 作者没有选择主流的蝶形计算方式, 采用 Tensor Core 更擅长的矩阵乘法来计算 NTT. 他们将这种方法应用于 LAC (一种基于格的 KEM 方案), 取得了比 LAC AVX2 参考实现快两个数量级的性能.
- 2022 年 ESORICS 上该团队提出了基于 Tensor Core 的 Kyber 优化实现^[139]. 作者研究发现 Tensor Core 支持的不同精度中 INT8 具有最佳性能. 而 Kyber 采用的模数是 $q = 3329 < 2^{12}$, 故在计算 NTT 时将 12 位的 Kyber 多项式的系数和对应旋转因子的高 6 位和低 6 位分别存于两个 INT8 变量, 即可利用 Tensor Core 的强大矩阵运算能力, 实现高效 NTT 计算. 采用上述优化方法, 他们优化实现取得了相较于 Kyber-1024 AVX2 参考实现的 26-35 倍性能.
- 2024 年 CHES 上该团队进一步提出了基于 Tensor Core 的 Kyber 优化实现^[140]. 作者提出了以矩阵实现 NTT 的新框架, 相较于此前方案, 新框架的计算复杂度降为 $1/4$. 此外, 他们还通过研究 Tensor Core 计算单元与寄存器的数据访问关系, 提出了在寄存器中连续进行两次矩阵乘法的方法, 避免了额外的内存占用和访存开销.

还有一些研究者也在此领域里作出了贡献.

2020 年 Gupta 等人^[141] 探索了以 GPU 实现后量子密钥交换算法, 提出了三种算法 (FrodoKEM-976、NewHope-1024 和 Kyber-1024) 的首个 GPU 实现. 作者为不同场景的应用设计了两种实现模式: 简单模式, 利用算法的内在并行性, 适用于追求算法实现更短延迟的场景; 批处理模式, 同时执行一定数量的算法独立实例, 适用于需求高吞吐量的应用场景, 例如服务器. 此外, 作者注意到线程束中的线程总是访

问连续的内存空间, 以及线程束中的线程以锁步的方式执行, 提出了内存合并的 NTT 实现方法. 这种新实现避免内存访问跨越, 实现了性能提升.

2020 年 Sun 等人^[142]研究了基于多核平台 (基于 x86/64 的多核 CPU 和 GPU) 的 SPHINCS 优化方法. 作者基于多核平台设计了 SPHINCS 及其内部结构 WOTS+ 和 HORST 的并行化版本. 通过应用了循环展开、内联函数或宏、统一数据表示、内联 PTX 汇编 (仅 GPU 适用)、合并访问 (仅 GPU 适用) 及常量和共享内存 (仅 GPU 适用) 等优化技术, 提出了以三种不同杂凑函数 (ChaCha、Haraka 和 Keccak) 分别实例化的 SPHINCS 优化实现.

2021 年 Gao 等^[143]基于 GPU, 提出并比较了三种类型的实现: 基准实现、细粒度实现和多流实现.

- **基准实现:** 基准实现是以朴素方式将参考实现移植到 GPU 平台上的实现. 这种实现以最直接的方式做到了数据并行化, 为接下来两种优化实现提供了性能基准.
- **细粒度实现:** 细粒度实现的设计目标是降低整体延迟、提高吞吐量. 作者通过对不同模块分配一个 warp 里的不同线程数 (8, 16, 32), 探究了各模块不同细粒度下最佳配置. 此种实现相较于 Gupta 等人提出的实现取得了 51.5% 性能提升.
- **多流实现:** 多流实现的设计目标是通过调度多个 CUDA 流来覆盖数据在 CPU 和 GPU 之间拷贝的延迟. 作者指出细粒度实现基于内核级并发, 仅从内核角度关注了内核执行的延迟. 由于主机设备数据传输和主机计算在内核执行之外, 故从用户角度衡量还包括了主机和 GPU 之间的延迟. 于是作者利用多流技术开发了网格级并发的细粒度实现, 相较于 Gupta 等人提出的实现, 该实现取得了 45.5% 性能提升.

此外, 作者还提出了一种新度量标准, 用于评估任何并行软件实现的并行化相对于非并行软件实现的并行化的性能损失.

5.2 基于 RISC-V 的实现

RISC-V 是一种开源且免版税的指令集架构 (ISA), 具有模块化和可扩展性的特点, 并明确支持特定领域的自定义扩展. RISC-V ISA 是一种流行的嵌入式处理 ISA, 它由基本指令集和模块化 ISA 扩展组成, 并且还明确支持特定领域的自定义扩展. 其中 RISC-V 基本指令集仅包括用于一般整数操作的基本整数指令集. 而 RISC-V ISA 扩展包括一组模块化扩展指令来满足不同计算需求. RISC-V 标准组织不断引入新指令集扩展来提高 RISC-V 的功能, 例如用于向量计算的 V 扩展和用于虚拟化的 H 扩展.

文献 [144] 总结了优化处理器的三个常用方法:

- (1) 向系统添加协处理器来执行特定操作.
- (2) 拓展计算机架构的基本指令.
- (3) 设计一个混合系统, 根据需要添加协处理器和特定指令.

2020 年 CHES 上 Alkim 等人^[79]提出并测试了针对有限域运算的 RISC-V 指令集的自定义扩展, 为小型嵌入式设备提供了软硬件协同设计的紧凑实现. 此扩展中的指令在执行有限域运算后即进行模约减, 可直接为包含有限域运算的算法 (文献中作者选择了 Kyber 和 NewHope) 提供软件加速. 同时, 作者还提出了基于标准 RISC-V 指令集的 Kyber 和 NewHope 优化实现, 作为衡量上述优化实现的基准. 相较于基准实现, 作者基于扩展 RISC-V 的优化实现不仅取得了 85% 的性能提升, 而且在不引入延迟惩罚的前提下, 以略增加电路规模的代价大大降低了实现的代码和数据量.

同年 CHES 上 Fritzmann 等人^[145]提出 RISQ-V—通过整合一组高效紧耦合加速器来加速基于格的 PQC 的 RISC-V 增强架构. 作者总结了前人松耦合加速器实现的三个主要劣势—高数据传输开销、占用大量硬件资源和低灵活性, 采用紧耦合方式设计 RISQ-V. 此外, RISQ-V 的高性能源于它高效复用处理器资源减少了大量内存访问, 在显著提升性能同时没有额外增加元件的硅面积. 具体而言:

- (1) 为并行的两个蝶形运算、动态旋转因子生成、量化模算术和随机多项式的高效生成设计了一组硬件加速器, 并深度整合到了 RISC-V 流水线.
- (2) 为高效实现基于格的密码方案设计了 29 个新指令, 执行打包的模运算, 其中包括加法、减法、乘法和乘加.

FPL 会议上 Karabulut 等人^[146]针对此前 NTT 实现无法同时兼顾高效和灵活的问题,提出了基于 RISC-V 的特定域架构的优化 NTT 实现. 作者指出 Alkim 等人提出的自定义指令是针对特定用例定制的,并且必须进行调整以支持多种算法或适应算术结构的变化. 以及即使对于特定的用例,自定义指令也必须被硬编码到编译器中,并且需要手动汇编编码. 他们针对这一点,设计了可直接将 C 代码翻译为 NTT 的优化实现,其中应用了特定于应用的动态指令调度、内存依赖性预测和数据路径优化技术.

Xin 等人^[147]研究了基于格的密码算法中向量化问题,基于 RISC-V 架构提出了特定域向量处理器 VPQC. 其中包括了可并行计算不同维度 (64–2048) NTT 的向量 NTT 单元和支持均匀采样和二项分布采样的向量采样器. 作者以台积电的 28 nm CMOS 技术实现 VPQC,相较于 2020 年的 pqm4 基准数据取得了两个数量级的性能增益.

2022 年 DAC 会议上 Li 等人^[148]为 Kyber 设计了基于硬软件协同设计的可扩展 SIMD RISC-V 处理器. 作者提出了寄存器池和自动生成索引两种硬件/软件接口优化技术. 其中寄存器池表示执行相同工作的多个寄存器. 作者观察到同一个寄存器池中的数据独立运行,可以同时读写多个数据. 因此他们通过应用寄存器池来增加 NTT、INTT 和逐系数乘法中每个循环中的读写吞吐量,并减少与 RAM 之间进行数据交换的性能损失,为上述三种计算过程提出了三种类型的寄存器池来促进并行计算. 而自动生成索引是指作者在设计保留了 NTT 和 INTT 外层循环结构并解除了内层两个循环结构. 通过设计的向量扩展控制最外层的循环数,使寄存器池中多项式索引根据循环次数自动变化. 此外作者还为提高 Kyber 中乘法和有限域运算计算性能分别设计了 12 和 4 个向量扩展.

同年 Zhang 等人^[149]基于资源受限 RISC-V 平台提出了 Saber 变体的软件优化实现. 首先作者提出了 Saber 的内存优化修改版“Saber+”,此变体修改了公开矩阵 A 和秘密向量 s 的生成方式. 实现中作者主要研究时间和内存之间的权衡,基于 SaberCHES18 提出了动态生成 A 和 s 的改进技术. 此外,作者还提出了在时间和内存用量之间做出三种不同权衡的计算矩阵-向量乘法的策略.

Zhao 等人^[150]针对文献 [147] 实现中缺乏对 MLWE 方案优化的不足,基于 MLWE 的基于格的密码方案 (Kyber 和 Dilithium) 提出了异构双核优化实现. 该实现中设计了一个运行软件的主机内核和一个可加速基于矩阵运算的密码内核. 硬件部分包括基于 RISC-V 架构矩阵扩展的特定域处理器,其中封装了基于矩阵的环运算. 其中具有可配置功能的二维脉动阵列可执行基于矩阵的 NTT 和其它算术运算,不仅支持多项式矩阵和向量结构大小可变,而且实现了数据的高度并行. 此外,作者还开发了一种乱序执行机制,通过提高多项式矩阵和向量中不同元素进行环运算之间的指令级并行来降低整体延迟.

2022 年 Kuo 等人^[144]研究发现 AES、ECC 和 Classic McEliece 等传统和后量子密码算法都包括 $GF(2^m)$ 算术 (纠错码中基本运算),为轻量级通用处理器提供了通用 RISC-V Galois 域 ISA 扩展. 一些密码算法 (例如 Kyber 和 NewHope) 使用 $GF(q)$ 算术,另一些 (例如 Classic McEliece) 使用 $GF(2^m)$ 算术,因此此工作与文献 [79] 中 $GF(q)$ ISA 扩展形成了互补,加速不同类别的有限域运算. 作者对传统密码 AES、非二进制纠错码和后量子算法 Classic McEliece 分别测试了他们提出的 ISA 扩展,仅以逻辑元件略微增加的代价实现了 5 倍性能提升.

6 后量子密码软件实现的侧信道防御技术

侧信道攻击是一种利用系统的物理实现或实际运行时的非直接信息泄漏渠道 (如输出能量轨迹、电磁放射情况和执行时间等) 来获取敏感信息的攻击方式. 与传统的直接攻击方式不同,侧信道攻击不是通过直接攻击系统的软件或硬件漏洞来获取信息. 根据获取侧信道数据的不同,侧信道攻击可分为功耗攻击、计时攻击、故障攻击、电磁攻击、光学攻击、声学攻击等.

随着后量子密码逐渐从理论设计转向实际应用,研究者不断提出新的侧信道攻击,响应防御措施需求也日益迫切. 侧信道攻击的防御措施设计思路是通过综合采用掩码、隐藏或乱序等基本方法,或是使侧信道泄露的信息与敏感值无关,或是减少侧信道泄露的数据量.

基本的侧信道防御策略分为两类:

掩码策略: 掩码通常将敏感秘密数据分割为随机份额,使侧信道泄露的单个份额不足以泄露敏感数据. 然而对于测量并结合多个份额的侧信道泄露的攻击形式 (例如差分能量攻击) 而言,掩码策略可能无效.

隐藏策略: 隐藏是在整个加密方案的时间常数过程中隐藏信息的泄露, 从而使输出功耗保持恒定, 实现策略包含在时间和能量幅度两个维度上进行操作。

这些防御策略中每一个都可应用于硬件实现、软件实现或软硬件结合实现。仅对基于软件的防御技术而言, 还包括分片、比特拆分和其它阻止敌手获取信息的算法更改。

接下来以 Kyber 为例, 本节简要介绍基于软件的侧信道分析防御措施的进展。

2021 年 Bos 等人^[151]提出了首个可抵抗一阶和高阶攻击的 Kyber 完全掩码实现方案。此实现基于两个创新点: 单比特压缩的高阶算法和一种可用来比较未经压缩的掩码多项式和经过压缩的公开多项式的技术。其中前者基于可应用于素模数的掩码的比特切片二分搜索, 后者避免了计算量大的密文压缩掩码, 同时能够在掩码任意阶进行实例化。

2021 年 Hamburg 等人^[152]提出了对掩码 CCA2 安全 Kyber 的选择密文 k 轨迹攻击方法。作者提出了一种构造稀疏形式的密文多项式的方法。该方法构造的密文进行解密时会在逆 NTT 计算的输入处产生稀疏多项式, 则敌手就能利用这种稀疏性实施有效的侧信道攻击。文中作者还讨论了与该工作相应的防御措施, 给出两种策略: 对逆 NTT 的输入进行掩码和在 NTT 内插入随机虚拟操作。这两种方法可以有效地降低密钥系数被恢复的成功率。

防御措施可分为两类, 对各种攻击通用的防御措施和对某种攻击专门设计的防御措施。学界已有多个工作提出了对基于格的方案的通用掩码策略。然而, 采用通用防御措施也有一些短板。首先, 研究者已经提出了对已掩码实现的成功攻击^[153], 以及理论上安全掩码方案中的缺陷也可被用于密钥恢复^[154]。其次, 掩码很影响实现效率。

针对现有攻击和防御措施存在的问题, 2023 年 Ravi 等人^[155]总结了学界对 Kyber 和 Dilithium 提出的大量侧信道攻击、故障注入攻击及相应防御措施, 并分别在开源软件库 pqm4 和 liboqs 上进行了实现, 确保了防御措施不会对性能造成严重影响。论文分析 Kyber 的篇幅里, 作者根据攻击的具体场景将对 Kyber 的攻击分为 4 类: 对密钥生成过程的攻击、对封装过程的攻击、对解封装过程的攻击和对解密 (公钥密码部分) 过程的攻击, 列述了各类现有攻击的分析方法, 随后或是介绍响应的防御措施, 或是利用乱序和掩码等基本技术设计有针对性的新措施。

这里仅作了一些简要介绍, 详细且全面的内容请读者参阅综述性文献 [155, 156]。

7 总结与展望

随着量子计算机逐渐从理论走向现实, 量子算法对传统公钥密码颠覆性影响日益凸显, 为了保证当下的机密信息在量子时代保持机密性, 越来越多的密码学家投身于设计具有抗量子计算特性的密码方案。后量子密码算法标准化成为非常急迫的任务, 因为只有对目前纷繁复杂的理论密码提案进行标准后才能集中精力于密码工程的任务, 即性能优化、实现安全和应用方案等。

本文以 NIST PQC 标准化项目中优秀算法为代表, 介绍了六类主流的后量子密码算法的结构和实现特征, 关注各类算法的实现优化技术, 比较了不同种类之间的实现差异, 总结了文献中提出的具有平台通用性的优化技术。

目前 NIST PQC 标准化项目取得阶段性成果, 开始对最终选定的算法进行标准化。一方面, 需要密切关注 NIST 标准化最终成果 (可能针对不同的应用场景标准化多个方案)。另一方面, 现有研究成果启示我们需结合我国实际情况, 推动后量子国家密码标准的制订工作。

参考文献

- [1] GRUSKA J. Quantum Computing[M]. London: McGraw-Hill, 1999.
- [2] SHOR P W. Algorithms for quantum computation: Discrete logarithms and factoring[C]. In: Proceedings of IEEE Symposium on Foundations of Computer Science. IEEE, 1994: 124–134. [DOI: 10.1109/SFCS.1994.365700]
- [3] LOVIC V. Quantum key distribution: Advantages, challenges and policy[J]. Cambridge Journal of Science & Policy, 2020, 1(2). [DOI: 10.17863/CAM.58622]
- [4] BARKER E, CHEN L, KELLER S, et al. Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography[R]. National Institute of Standards and Technology, 2017. [DOI: 10.6028/NIST.SP.800-

- 56Ar3]
- [5] BARKER E, CHEN L, ROGINSKY A, et al. Recommendation for pair-wise key-establishment using integer factorization cryptography[R]. National Institute of Standards and Technology, 2018. [DOI: 10.6028/NIST.SP.800-56Br2]
 - [6] KERRY C F, GALLAGHER P D. Digital Signature Standard (DSS): FIPS PUB 186-4[S]. 2013. [DOI: 10.6028/NIST.FIPS.186-4]
 - [7] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process[EB/OL]. National Institute of Standards and Technology, 2017.
<https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>
 - [8] AZARDERAKHSH R, JAO D, KALACH K, et al. Key compression for isogeny-based cryptosystems[C]. In: Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography (AsiaPKC '16). ACM, 2016: 1–10. [DOI: 10.1145/2898420.2898421]
 - [9] QIN Y, CHENG C, DING J T. A complete and optimized key mismatch attack on NIST candidate NewHope[C]. In: Computer Security—ESORICS 2019, Part II. Springer Cham, 2019: 504–520. [DOI: 10.1007/978-3-030-29962-0_24]
 - [10] BAUER A, GILBERT H, RENAULT G, et al. Assessment of the key-reuse resilience of NewHope[C]. In: Topics in Cryptology—CT-RSA 2019. Springer Cham, 2019: 272–292. [DOI: 10.1007/978-3-030-12612-4_14]
 - [11] DING J T, CHENG C, QIN Y. A simple key reuse attack on LWE and ring LWE encryption schemes as key encapsulation mechanisms (KEMs)[J/OL]. IACR Cryptology ePrint Archive, 2019: 2019/271.
<https://eprint.iacr.org/2019/271.pdf>
 - [12] FUJISAKI E, OKAMOTO T. Secure integration of asymmetric and symmetric encryption schemes[C]. In: Advances in Cryptology—CRYPTO '99. Springer Berlin Heidelberg, 1999: 537–554. [DOI: 10.1007/3-540-48405-1_34]
 - [13] RAVI P, ROY S S, CHATTOPADHYAY A, et al. Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020(3): 307–335. [DOI: 10.13154/tches.v2020.i3.307-335]
 - [14] GULLY S, GOPAL V, YAP K, et al. New instructions supporting the secure hash algorithm on Intel architecture processors[R/OL]. 2013.
<https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sha-extensions.html>
 - [15] REGEV O. On lattices, learning with errors, random linear codes, and cryptography[J]. Journal of the ACM (JACM), 2009, 56(6): 1–40. [DOI: 10.1145/1568318.1568324]
 - [16] BANERJEE A, PEIKERT C, ROSEN A. Pseudorandom functions and lattices[C]. In: Advances in Cryptology—EUROCRYPT 2012. Springer Berlin Heidelberg, 2012: 719–737. [DOI: 10.1007/978-3-642-29011-4_42]
 - [17] LYUBASHEVSKY V, PERKERT C, REGEV O. On ideal lattices and learning with errors over rings[C]. In: Advances in Cryptology—EUROCRYPT 2010. Springer Berlin Heidelberg, 2010: 1–23. [DOI: 10.1007/978-3-642-13190-5_1]
 - [18] LANGLOIS A, STEHLÉ D. Worst-case to average-case reductions for module lattices[J]. Designs, Codes and Cryptography, 2015, 75(3): 565–599. [DOI: 10.1007/s10623-014-9938-4]
 - [19] D'ANVERS J P, KARMAKAR A, ROY S S, et al. Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM[C]. In: Progress in Cryptology—AFRICACRYPT 2018. Springer Cham, 2018: 282–305. [DOI: 10.1007/978-3-319-89339-6_16]
 - [20] BOS J, DUCAS L, KILTZ E, et al. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM[C]. In: Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P 2018). IEEE, 2018: 353–367. [DOI: 10.1109/EuroSP.2018.00032]
 - [21] ROY S S, BASSO A. High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020: 2020(4): 443–466. [DOI: 10.13154/tches.v2020.i4.443-466]
 - [22] COOLEY J W, TUKEY J W. An algorithm for the machine calculation of complex Fourier series[J]. Mathematics of Computation, 1965, 19(90): 297–301. [DOI: 10.2307/2003354]
 - [23] GENTLEMAN W M, SANDE G. Fast Fourier transforms: For fun and profit[C]. In: Proceedings of Fall Joint Computer Conference (AFIPS '66 (Fall)). ACM, 1966: 563–578. [DOI: 10.1145/1464291.1464352]
 - [24] TAO Y T, KONG F Y, YU J, et al. Survey of number theoretic transform algorithms for quantum-resistant lattice-based cryptography[J]. Netinfo Security, 2021, 21(9): 46–51. [DOI: 10.3969/j.issn.1671-1122.2021.09.007]

- 陶云亭, 孔凡玉, 于佳, 等. 抗量子格密码体制的快速数论变换算法研究综述 [J]. 信息安全, 2021, 21(9): 46–51. [DOI: 10.3969/j.issn.1671-1122.2021.09.007]
- [25] LAMPORT L. Constructing digital signatures from a one way function[R/OL]. SRI International, 1979: 1–7. <https://www.microsoft.com/en-us/research/publication/2016/12/Constructing-Digital-Signatures-from-a-One-Way-Function.pdf>
- [26] CASTELNOVI L, MARTINELLI A, PREST T. Grafting trees: A fault attack against the SPHINCS framework[C]. In: Post-Quantum Cryptography—PQC 2018. Springer Cham, 2018: 165–184. [DOI: 10.1007/978-3-319-79063-3_8]
- [27] MERKLE R C. A certified digital signature[C]. In: Advances in Cryptology—ASIACRYPT '89. Springer New York, NY, 1990: 218–238. [DOI: 10.1007/0-387-34805-0_21]
- [28] GOLDBREICH O. Two remarks concerning the Goldwasser-Micali-Rivest signature scheme[C]. In: Advances in Cryptology—EUROCRYPT '86. Springer Berlin Heidelberg, 1987: 104–110. [DOI: 10.1007/3-540-47721-7_8]
- [29] BERNSTEIN D J, HOPWOOD D, HÜLSING A, et al. SPHINCS: Practical stateless hash-based signatures[C]. In: Advances in Cryptology—EUROCRYPT 2015, Part I. Springer Berlin Heidelberg, 2015: 368–397. [DOI: 10.1007/978-3-662-46800-5_15]
- [30] BERNSTEIN D J, HÜLSING A, KÖLBL S, et al. The SPHINCS+ signature framework[C]. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS 2019). ACM, 2019: 2129–2146. [DOI: 10.1145/3319535.3363229]
- [31] COOPER D A, APON D C, DANG Q H, et al. Recommendation for stateful hash-based signature schemes[R]. NIST Special Publication, 800-208. 2020. [DOI: 10.6028/NIST.SP.800-208]
- [32] HÜLSING A, BERNSTEIN D J, DOBRAUNIG C, et al. Round 3 submissions: SPHINCS+[EB/OL]. National Institute of Standards and Technology, 2020. <https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/SPHINCS-Round3.zip>
- [33] BALAMURUGAN C, SINGH K, GANESAN G, et al. Post-quantum and code-based cryptography—Some prospective research directions[J]. Cryptography, 2021, 5(4): 38. [DOI: 10.3390/cryptography5040038]
- [34] BERLEKAMP E, MCELIECE R, VAN TILBORG H. On the inherent intractability of certain coding problems (corresp.)[J]. IEEE Transactions on Information Theory, 1978, 24(3): 384–386. [DOI: 10.1109/TIT.1978.1055873]
- [35] MCELIECE R J. A public-key cryptosystem based on algebraic coding theory[R/OL]. Deep Space Network Progress Report, 1978. https://tda.jpl.nasa.gov/progress_report/42-44/44N.PDF
- [36] NIEDERREITER H. Knapsack-type cryptosystems and algebraic coding theory[J]. Problems of Control and Information Theory, 1986, 15(2): 159–166.
- [37] ALBRECHT M R, BERNSTEIN D J, CHOU T, et al. Round 3 submissions: Classic McEliece[EB/OL]. National Institute of Standards and Technology, 2020. <https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/Classic-McEliece-Round3.zip>
- [38] BERLEKAMP E. GOPPA codes[J]. IEEE Transactions on Information Theory, 1973, 19(5): 590–592. [DOI: 10.1109/TIT.1973.1055088]
- [39] GAO S, MATEER T. Additive fast Fourier transforms over finite fields[J]. IEEE Transactions on Information Theory, 2010, 56(12): 6265–6272. [DOI: 10.1109/TIT.2010.2079016]
- [40] KOZIEL B, KERMANI M M, AZARDERAKHSH R. Post-Quantum Cryptographic Hardware and Embedded Systems[M]. In: Emerging Topics in Hardware Security. Springer Cham, 2021: 229–255. [DOI: 10.1007/978-3-030-64448-2_9]
- [41] CHEN M S, CHOU T. Classic McEliece on the ARM Cortex-M4[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021(3): 125–148. [DOI: 10.46586/tches.v2021.i3.125-148]
- [42] BERLEKAMP E R. Algebraic Coding Theory[M]. Revised Edition. World Scientific, 2015. [DOI: 10.1142/9407]
- [43] MASSEY J. Shift-register synthesis and BCH decoding[J]. IEEE Transactions on Information Theory, 1969, 15(1): 122–127. [DOI: 10.1109/TIT.1969.1054260]
- [44] XU Y Z. Implementation of Berlekamp–Massey algorithm without inversion[J]. IEE Proceedings I (Communications, Speech and Vision), 1991, 138(3): 138–140. [DOI: 10.1049/ip-i-2.1991.0018]
- [45] CASTRYCK W, DECRU T. An efficient key recovery attack on SIDH[J/OL]. IACR Cryptology ePrint Archive, 2022: 2022/975. <https://eprint.iacr.org/2022/975.pdf>
- [46] CASTRYCK W, LANGE T, MARTINDALE C, et al. CSIDH: An efficient post-quantum commutative group action[C]. In: Advances in Cryptology—ASIACRYPT 2018, Part III. Springer Cham, 2018: 395–427. [DOI: 10.1007/978-3-030-03332-3_15]

- [47] DE FEO L, KOHEL D, LEROUX A, et al. SQISign: Compact post-quantum signatures from quaternions and isogenies[C]. In: *Advances in Cryptology—ASIACRYPT 2020, Part I*. Springer Cham, 2020: 64–93. [DOI: 10.1007/978-3-030-64837-4_3]
- [48] JAO D, DE FEO L. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies[C]. In: *Post-Quantum Cryptography—PQC 2011*. Springer Berlin Heidelberg, 2011: 19–34. [DOI: 10.1007/978-3-642-25405-5_2]
- [49] AZARDERAKHSH R, CAMPAGNA M, COSTELLO C, et al. Supersingular isogeny key encapsulation[EB]. Submission to the NIST Post-Quantum Standardization Project, 2017.
- [50] HOFHEINZ D, HÖVELMANN K, KILTZ E. A modular analysis of the Fujisaki-Okamoto transformation[C]. In: *Theory of Cryptography—TCC 2017, Part I*. Springer Cham, 2017: 341–371. [DOI: 10.1007/978-3-319-70500-2_12]
- [51] GALBRAITH S D, PETIT C, SHANI B, et al. On the security of supersingular isogeny cryptosystems[C]. In: *Advances in Cryptology—ASIACRYPT 2016, Part I*. Springer Berlin Heidelberg, 2016: 63–91. [DOI: 10.1007/978-3-662-53887-6_3]
- [52] KUMAR M, PATTNAIK P. Post quantum cryptography (PQC)—An overview[C]. In: *Proceedings of IEEE High Performance Extreme Computing Conference (HPEC 2020)*. IEEE, 2020: 1–9. [DOI: 10.1109/HPEC43674.2020.9286147]
- [53] FAZ-HERNÁNDEZ A, LÓPEZ J, OCHOA-JIMÉNEZ E, et al. A faster software implementation of the supersingular isogeny Diffie-Hellman key exchange protocol[J]. *IEEE Transactions on Computers*, 2017, 67(11): 1622–1636. [DOI: 10.1109/TC.2017.2771535]
- [54] COSTELLO C, JAO D, LONGA P, et al. Efficient compression of SIDH public keys[C]. In: *Advances in Cryptology—EUROCRYPT 2017, Part I*. Springer Cham, 2017: 679–706. [DOI: 10.1007/978-3-319-56620-7_24]
- [55] ZANON G H M, SIMPLICIO M A, PEREIRA G C C F, et al. Faster key compression for isogeny-based cryptosystems[J]. *IEEE Transactions on Computers*, 2018, 68(5): 688–701. [DOI: 10.1109/TC.2018.2878829]
- [56] ZANON G H M, SIMPLICIO M A, PEREIRA G C C F, et al. Faster isogeny-based compressed key agreement[C]. In: *Post-Quantum Cryptography—PQC 2018*. Springer Cham, 2018: 248–268. [DOI: 10.1007/978-3-319-79063-3_12]
- [57] NAEHRIG M, RENES J. Dual isogenies and their application to public-key compression for isogeny-based cryptography[C]. In: *Advances in Cryptology—ASIACRYPT 2019, Part II*. Springer Cham, 2019: 243–272. [DOI: 10.1007/978-3-030-34621-8_9]
- [58] PEREIRA G, DOLISKANI J, JAO D. x-only point addition formula and faster compressed SIKE[J]. *Journal of Cryptographic Engineering*, 2021, 11(1): 57–69. [DOI: 10.1007/s13389-020-00245-4]
- [59] SEO S C. SIKE on GPU: Accelerating supersingular isogeny-based key encapsulation mechanism on graphic processing units[J]. *IEEE Access*, 2021, 9: 116731–116744. [DOI: 10.1109/access.2021.3106551]
- [60] DING J T, PETZOLDT A. Current state of multivariate cryptography[J]. *IEEE Security & Privacy*, 2017, 15(4): 28–36. [DOI: 10.1109/MSP.2017.3151328]
- [61] DING J T, SCHMIDT D. Rainbow, a new multivariable polynomial signature scheme[C]. In: *Applied Cryptography and Network Security—ACNS 2005*. Springer Berlin Heidelberg, 2005: 164–175. [DOI: 10.1007/11496137_12]
- [62] KIPNIS A, PATARIN J, GOUBIN L. Unbalanced oil and vinegar signature schemes[C]. In: *Advances in Cryptology—EUROCRYPT '99*. Springer Berlin Heidelberg, 1999: 206–222. [DOI: 10.1007/3-540-48910-X_15]
- [63] KARATSUBA A A, OFMAN Y P. Multiplication of many-digital numbers by automatic computers[J]. *Doklady Akademii Nauk Russian Academy of Sciences*, 1962, 145(2): 293–294.
- [64] BEULLENS W. Breaking Rainbow takes a weekend on a laptop[J/OL]. *IACR Cryptology ePrint Archive*, 2022: 2022/214. <https://eprint.iacr.org/2022/214.pdf>
- [65] GOLDWASSER S, MICALI S, RACKOFF C. The knowledge complexity of interactive proof systems[J]. *SIAM Journal on Computing*, 1989, 18(1): 186–208. [DOI: 10.1137/0218012]
- [66] CHASE M, DERLER D, GOLDFEDER S, et al. Post-quantum zero-knowledge and signatures from symmetric-key primitives[C]. In: *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, 2017: 1825–1842. [DOI: 10.1145/3133956.3133997]
- [67] GIACOMELLI I, MADSEN J, ORLANDI C. ZKBoo: Faster zero-knowledge for Boolean circuits[C]. In: *Proceedings of the 25th USENIX Conference on Security Symposium (SEC '16)*. USENIX, 2016: 1069–1083. [DOI: 10.5555/3241094.3241177]
- [68] ISHAI Y, KUSHILEVITZ E, OSTROVSKY R, et al. Zero-knowledge from secure multiparty computation[C]. In: *Proceedings of ACM Symposium on Theory of Computing (STOC '07)*. ACM, 2007: 21–30. [DOI: 10.1145/

- 1250790.1250794]
- [69] ALBRECHT M R, RECHBERGER C, SCHNEIDER T, et al. Ciphers for MPC and FHE[C]. In: *Advances in Cryptology—EUROCRYPT 2015, Part I*. Springer Berlin Heidelberg, 2015: 430–454. [DOI: 10.1007/978-3-662-46800-5_17]
 - [70] ZAVERUCHA G, CHASE M, DERLER D, et al. Round 3 submissions: Picnic[EB/OL]. National Institute of Standards and Technology, 2020.
<https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/Picnic-Round3.zip>
 - [71] SANAL P, KARAGOZ E, SEO H, et al. Kyber on ARM64: Compact implementations of Kyber on 64-bit ARM Cortex-A processors[C]. In: *Security and Privacy in Communication Systems—SecureComm 2021*. Springer Cham, 2021: 424–440. [DOI: 10.1007/978-3-030-90022-9_23]
 - [72] GRECONICI D O C, KANNWISCHER M J, SPRENKELS D. Compact Dilithium implementations on Cortex-M3 and Cortex-M4[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021, 2021(1): 1–24. [DOI: 10.46586/tches.v2021.i1.1-24]
 - [73] SHIM K A, LEE S, KOO N. Efficient implementations of Rainbow and UOV using AVX2[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022, 2022(1): 245–269. [DOI: 10.46586/tches.v2022.i1.245-269]
 - [74] KALES D, ZAVERUCHA G. Improving the performance of the Picnic signature scheme[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020, 2020(4): 154–188. [DOI: 10.13154/tches.v2020.i4.154-188]
 - [75] STREIT S, DE SANTIS F. Post-quantum key exchange on ARMv8-A: A new hope for NEON made simple[J]. *IEEE Transactions on Computers*, 2017, 67(11): 1651–1662. [DOI: 10.1109/TC.2017.2773524]
 - [76] JALALI A, AZARDEKRAKSH R, KERMANI M M, et al. ARMv8 SIKE: Optimized supersingular isogeny key encapsulation on ARMv8 processors[J]. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2019, 66(11): 4209–4218. [DOI: 10.1109/TCSI.2019.2920869]
 - [77] NGUYEN D T, GAJ K. Fast NEON-based multiplication for lattice-based NIST Post-Quantum Cryptography finalists[C]. In: *Post-Quantum Cryptography—PQC 2021*. Springer Cham, 2021: 234–254. [DOI: 10.1007/978-3-030-81293-5_13]
 - [78] BECKER H, HWANG V, KANNWISCHER M J, et al. Neon NTT: Faster Dilithium, Kyber, and Saber on Cortex-A72 and Apple M1[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022, 2022(1): 221–244. [DOI: 10.46586/tches.v2022.i1.221-244]
 - [79] ALKIM E, EVKAN H, LAHR N, et al. ISA extensions for finite field arithmetic: Accelerating Kyber and NewHope on RISC-V[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020, 2020(3): 219–242. [DOI: 10.13154/tches.v2020.i3.219-242]
 - [80] LEE W K, SEO H, HWANG S O, et al. DPCrypto: Acceleration of post-quantum cryptography using dot-product instructions on GPUs[J]. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022, 69(9): 3591–3604. [DOI: 10.1109/TCSI.2022.3176966]
 - [81] STEBILA D, MOSCA M. Post-quantum key exchange for the Internet and the open quantum safe project[C]. In: *Selected Areas in Cryptography—SAC 2016*. Springer Cham, 2016: 14–37. [DOI: 10.1007/978-3-319-69453-5_2]
 - [82] BOTROS L, KANNWISCHER M J, SCHWABE P. Memory-efficient high-speed implementation of Kyber on Cortex-M4[C]. In: *Progress in Cryptology—AFRICACRYPT 2019*. Springer Cham, 2019: 209–228. [DOI: 10.1007/978-3-030-23696-0_11]
 - [83] GUERON S. Intel® advanced encryption standard (AES) new instructions set[R/OL]. Intel White Paper, Reversion 3.0, 2010: 1–81.
<https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf>
 - [84] TOWNER D, KINSELLA R. Galois field new instructions (GFNI)[R/OL]. Intel, 2021.
<https://networkbuilders.intel.com/solutionslibrary/galois-field-new-instructions-gfni-technology-guide>
 - [85] DRUCKER N, GUERON S. Speed up over the Rainbow[C]. In: *Advances in Intelligent Systems and Computing—ITNG 2021*. Springer Cham, 2021: 131–136. [DOI: 10.1007/978-3-030-70416-2_17]
 - [86] GUERON S, KOUNAVIS M E. Intel® carry-less multiplication instruction and its usage for computing the GCM mode[R/OL]. White Paper, 2010: 10.
<https://www.intel.com/content/dam/develop/public/us/en/documents/carry-less-multiplication-instruction.pdf>
 - [87] CERIA M, DE PICCOLI A, TIZIANI M, et al. Optimizing the key-pair generation phase of McEliece cryptosystem[C]. In: *Proceedings of 4th International Conference on Wireless, Intelligent and Distributed Environment*

- for Communication—WIDECOM 2022. Springer Cham, 2022: 111–122. [DOI: 10.1007/978-3-030-89776-5_8]
- [88] ALKIM E, BILGIN Y A, CENK M, et al. Cortex-M4 optimizations for {R, M} LWE schemes[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020(3): 336–357. [DOI: 10.13154/tches.v2020.i3.336-357]
- [89] ALKIM E, JAKUBEIT P, SCHWABE P. NewHope on ARM Cortex-M[C]. In: Security, Privacy, and Applied Cryptography Engineering—SPACE 2016. Springer Cham, 2016: 332–349. [DOI: 10.1007/978-3-319-49445-6_19]
- [90] MERA J M B, KARMAKAR A, VERBAUWHEDE I. Time-memory trade-off in Toom-Cook multiplication: An application to module-lattice based cryptography[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020(2): 222–244. [DOI: 10.13154/tches.v2020.i2.222-244]
- [91] KARMAKAR A, MERA J M B, ROY S S, et al. Saber on ARM: CCA-secure module lattice-based key encapsulation on ARM[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018, 2018(3): 243–266. [DOI: 10.13154/tches.v2018.i3.243-266]
- [92] TOOM A L. The complexity of a scheme of functional elements realizing the multiplication of integers[J]. Soviet Mathematics Doklady, 1963, 3(4): 714–716.
- [93] COOK S A, AANDERAA S O. On the minimum computation time of functions[J]. Transactions of the American Mathematical Society, 1969, 142: 291–314. [DOI: 10.2307/1995359]
- [94] ALKIM E, DUCAS L, PÖPPELMANN T, et al. Post-quantum key exchange—A new hope[C]. In: Proceedings of the 25th USENIX Conference on Security Symposium (SEC '16). USENIX, 2016: 327–343. [DOI: 10.5555/3241094.3241120]
- [95] PÖPPELMANN T, ODER T, GÜNEYSU T. High-performance ideal lattice-based cryptography on 8-bit ATxmega microcontrollers[C]. In: Progress in Cryptology—LATINCRYPT 2015. Springer Cham, 2015: 346–365. [DOI: 10.1007/978-3-319-22174-8_19]
- [96] GÜNEYSU T, ODER T, PÖPPELMANN T, et al. Software speed records for lattice-based signatures[C]. In: Post-Quantum Cryptography—PQC 2013. Springer Berlin Heidelberg, 2013: 67–82. [DOI: 10.1007/978-3-642-38616-9_5]
- [97] BERNSTEIN D J, CHUENGSAITANSUP C, LANGE T, et al. NTRU prime: Reducing attack surface at low cost[C]. In: Selected Areas in Cryptography—SAC 2017. Springer Cham, 2017: 235–260. [DOI: 10.1007/978-3-319-72565-9_12]
- [98] LONGA P, NAEHRIG M. Speeding up the number theoretic transform for faster ideal lattice-based cryptography[C]. In: Cryptology and Network Security—CANS 2016. Springer Cham, 2016: 124–139. [DOI: 10.1007/978-3-319-48965-0_8]
- [99] MONTGOMERY P L. Modular multiplication without trial division[J]. Mathematics of Computation, 1985, 44(170): 519–521. [DOI: 10.1090/S0025-5718-1985-0777282-X]
- [100] BARRETT P. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor[C]. In: Advances in Cryptology—CRYPTO '86. Springer Berlin Heidelberg, 1987: 311–323. [DOI: 10.1007/3-540-47721-7_24]
- [101] ZHOU S, XUE H Y, ZHANG D D, et al. Preprocess-then-NTT technique and its applications to Kyber and NewHope[C]. In: Information Security and Cryptology—Inscrypt 2018. Springer Cham, 2019: 117–137. [DOI: 10.1007/978-3-030-14234-6_7]
- [102] SEILER G. Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography[J/OL]. IACR Cryptology ePrint Archive, 2018: 2018/039. <https://eprint.iacr.org/2018/039.pdf>
- [103] LYUBASHEVSKY V, SEILER G. NTTRU: Truly fast NTRU using NTT[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019, 2019(3): 180–201. [DOI: 10.13154/tches.v2019.i3.180-201]
- [104] HOANG G L, SCHWABE P, BATINA L. Optimization of the NTT Function on ARMv8-A SVE[D/OL]. Nijmegen, the Netherlands: Radboud University, 2018. https://www.cs.ru.nl/bachelors-theses/2018/Hoang_Gia_Linh_4553519_Optimization_of_the_NTT_function_on_ARMv8-A_SVE.pdf
- [105] KANNWISCHER M J, RIJNEVELD J, SCHWABE P. Faster multiplication in $\mathbb{Z}_{2^m}[x]$ on Cortex-M4 to speed up NIST PQC candidates[C]. In: Applied Cryptography and Network Security—ACNS 2019. Springer Cham, 2019: 281–301. [DOI: 10.1007/978-3-030-21568-2_14]
- [106] ANASTASOVA M, AZARDERAKHSH R, KERMANI M M. Fast strategies for the implementation of SIKE Round 3 on ARM Cortex-M4[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(10): 4129–4141. [DOI: 10.1109/TCSI.2021.3096916]
- [107] ABDULRAHMAN A, HWANG V, KANNIWSCHER M J, et al. Faster Kyber and Dilithium on the Cortex-M4[C]. In: Applied Cryptography and Network Security—ACNS 2022. Springer Cham, 2022: 853–871. [DOI:

- 10.1007/978-3-031-09234-3_42]
- [108] ABDULRAHMAN A, CHEN J P, CHEN Y J, et al. Multi-moduli NTTs for Saber on Cortex-M3 and Cortex-M4[J/OL]. IACR Cryptology ePrint Archive. 2021: 2021/995. <https://eprint.iacr.org/2021/995.pdf>
 - [109] ALKIM E, CHENG D Y, CHUNG C M, et al. Polynomial multiplication in NTRU prime: Comparison of optimization strategies on Cortex-M4[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021(1): 217–238. [DOI: 10.46586/tches.v2021.i1.217-238]
 - [110] RAVI P, GUPTA S S, CHATTOPADHYAY A, et al. Improving speed of Dilithium's signing procedure[C]. In: Smart Card Research and Advanced Applications—CARDIS 2020. Springer Cham, 2020: 57–73. [DOI: 10.1007/978-3-030-42068-0_4]
 - [111] GÜNEYSU T, KRAUSZ M, ODER T, et al. Evaluation of lattice-based signature schemes in embedded systems[C]. In: Proceedings of International Conference on Electronics, Circuits and Systems (ICECS2018). IEEE, 2018: 385–388. [DOI: 10.1109/ICECS.2018.8617969]
 - [112] KIM Y, SONG J, YOUN T Y, et al. Crystals-Dilithium on ARMv8[J]. Security and Communication Networks, 2022, 2022: 5226390. [DOI: 10.1155/2022/5226390]
 - [113] DANDA O, SCHWABE P, BATINA L. Optimizing NTRU Using AVX2[D/OL]. Nijmegen, the Netherlands: Radboud University, 2019. https://www.ru.nl/publish/pages/769526/y_oussama_danba.pdf
 - [114] CHUNG C M M, HWANG V, KANNWISCHER M J, et al. NTT multiplication for NTT-unfriendly rings: New speed records for Saber and NTRU on Cortex-M4 and AVX2[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021(2): 159–188. [DOI: 10.46586/tches.v2021.i2.159-188]
 - [115] ROY S S. SaberX4: High-throughput software implementation of Saber key encapsulation mechanism[C]. In: Proceedings of IEEE International Conference on Computer Design (ICCD 2019). IEEE, 2019: 321–324. [DOI: 10.1109/ICCD46524.2019.00050]
 - [116] GUERON S, SCHLIEKER F. Speeding up R-LWE post-quantum key exchange[C]. In: Secure IT Systems—NordSec 2016. Springer Berlin Heidelberg, 2016: 187–198. [DOI: 10.1007/978-3-319-47560-8_12]
 - [117] JALALI A, AZARDERAKHSH R, KERMANI M M. NEON SIKE: Supersingular isogeny key encapsulation on ARMv7[C]. In: Security, Privacy, and Applied Cryptography Engineering—SPACE 2018. Springer Cham, 2018: 37–51. [DOI: 10.1007/978-3-030-05072-6_3]
 - [118] SEO H, LIU Z, LONGA P, et al. SIDH on ARM: Faster modular multiplications for faster post-quantum supersingular isogeny key exchange[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018, 2018(3): 1–20. [DOI: 10.13154/tches.v2018.i3.1-20]
 - [119] KOPPERMANN P, POP E, HEYSZL J, et al. 18 seconds to key exchange: Limitations of supersingular isogeny Diffie-Hellman on embedded devices[J/OL]. IACR Cryptology ePrint Archive, 2018: 2018/032. <https://eprint.iacr.org/2018/932.pdf>
 - [120] SEO H, ANASTASOVA M, JALALI A, et al. Supersingular isogeny key encapsulation (SIKE) Round 2 on ARM Cortex-M4[J]. IEEE Transactions on Computers, 2020, 70(10): 1705–1718. [DOI: 10.1109/TC.2020.3023045]
 - [121] COSTELLO C, LONGA P, NAEHRIG M. Efficient algorithms for supersingular isogeny Diffie-Hellman[C]. In: Advances in Cryptology—CRYPTO 2016, Part I. Springer Berlin Heidelberg, 2016: 572–601. [DOI: 10.1007/978-3-662-53018-4_21]
 - [122] KOZIEL B, JALALI A, AZARDERAKHSH R, et al. NEON-SIDH: Efficient implementation of supersingular isogeny Diffie-Hellman key exchange protocol on ARM[C]. In: Cryptology and Network Security—CANS 2016. Springer Cham, 2016: 88–103. [DOI: 10.1007/978-3-319-48965-0_6]
 - [123] JALALI A, AZARDERAKHSH R, KERMANI M M, et al. Supersingular isogeny Diffie-Hellman key exchange on 64-bit ARM[J]. IEEE Transactions on Dependable and Secure Computing, 2017, 16(5): 902–912. [DOI: 10.1109/TDSC.2017.2723891]
 - [124] COMBA P G. Exponentiation cryptosystems on the IBM PC[J]. IBM Systems Journal, 1990, 29(4): 526–538. [DOI: 10.1147/sj.294.0526]
 - [125] SEO H, JALALI A, AZARDERAKHSH R. SIKE Round 2 speed record on ARM Cortex-M4[C]. In: Cryptology and Network Security—CANS 2019. Springer Cham, 2019: 36–60. [DOI: 10.1007/978-3-030-31578-8_3]
 - [126] SEO H, SANAL P, JALALI A, et al. Optimized implementation of SIKE Round 2 on 64-bit ARM Cortex-A processors[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 67(8): 2659–2671. [DOI: 10.1109/TCSI.2020.2979410]
 - [127] FUJII H, ARANHA D F. Curve25519 for the Cortex-M4 and beyond[C]. In: Progress in Cryptology—LATINCRYPT 2017. Springer Cham, 2017: 109–127. [DOI: 10.1007/978-3-030-25283-0_6]
 - [128] KWON H, KIM H, SIM M, et al. Look-up the Rainbow: Efficient table-based parallel implementation of Rainbow signature on 64-bit ARMv8 processors[J/OL]. IACR Cryptology ePrint Archive, 2021: 2021/1015.

<https://eprint.iacr.org/2021/1015.pdf>

- [129] CHOU T, KANNWISCHER M J, YANG B Y. Rainbow on Cortex-M4[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021, 2021(4): 650–675. [DOI: 10.46586/tches.v2021.i4.650-675]
- [130] KÖLBL S. Putting wings on SPHINCS[C]. In: *Post-Quantum Cryptography—PQC 2018*. Springer Cham, 2018: 205–226. [DOI: 10.1007/978-3-319-79063-3_10]
- [131] HÜLSING A T, RIJNEVELD J, SCHWABE P. ARMed SPHINCS—Computing a 41 KB signature in 16 KB of RAM[C]. In: *Public-Key Cryptography—PKC 2016, Part I*. Springer Berlin Heidelberg, 2016: 446–470. [DOI: 10.1007/978-3-662-49384-7_17]
- [132] GONZALEZ R, HÜLSING A, KANNWISCHER M J, et al. Verifying post-quantum signatures in 8 kB of RAM[C]. In: *Post-Quantum Cryptography—PQC 2021*. Springer Cham, 2021: 215–233. [DOI: 10.1007/978-3-030-81293-5_12]
- [133] NIEDERHAGEN R, ROTH J, WÄLDE J. Streaming SPHINCS+ for embedded devices using the example of TPMs[C]. In: *Progress in Cryptology—AFRICACRYPT 2022*. Springer Cham, 2022: 269–291. [DOI: 10.1007/978-3-031-17433-9_12]
- [134] WINKLER J, HÖLLER A, STEGER C. Optimizing Picnic for limited memory resources[C]. In: *Proceedings of Euromicro Conference on Digital System Design (DSD 2020)*. IEEE, 2020: 200–204. [DOI: 10.1109/DSD51259.2020.00041]
- [135] LEE W K, AKLEYLEK S, WONG D C K, et al. Parallel implementation of Nussbaumer algorithm and number theoretic transform on a GPU platform: Application to qTESLA[J]. *The Journal of Supercomputing*, 2021, 77: 3289–3314. [DOI: 10.1007/s11227-020-03392-x]
- [136] LEE W K, HWANG S O. High throughput implementation of post-quantum key encapsulation and decapsulation on GPU for Internet of Things applications[J]. *IEEE Transactions on Services Computing*, 2021, 15(6): 3275–3288. [DOI: 10.1109/TSC.2021.3103956]
- [137] LEE W K, SEO H, ZHANG Z F, et al. Tensorcrypto: High throughput acceleration of lattice-based cryptography using tensor core on GPU[J]. *IEEE Access*, 2022, 10: 20616–20632. [DOI: 10.1109/ACCESS.2022.3152217]
- [138] WAN L P, ZHENG F Y, LIN J Q. TESLAC: Accelerating lattice-based cryptography with AI accelerator[C]. In: *Security and Privacy in Communication Networks—SecureComm 2021*. Springer Cham, 2021: 249–269. [DOI: 10.1007/978-3-030-90019-9_13]
- [139] WAN L P, ZHENG F Y, FAN G, et al. A novel high-performance implementation of CRYSTALS-Kyber with AI accelerator[C]. In: *Computer Security—ESORICS 2022, Part III*. Springer Cham, 2022: 514–534. [DOI: 10.1007/978-3-031-17143-7_25]
- [140] ZHOU T, ZHENG F Y, FAN G, et al. ConvKyber: Unleashing the power of AI accelerators for faster Kyber with novel iteration-based approaches[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024, 2024(2): 25–63. [DOI: 10.46586/tches.v2024.i2.25-63]
- [141] GUPTA N, JATI A, CHAUHAN A K, et al. PQC acceleration using GPUs: Frodokem, NewHope, and Kyber[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2020, 32(3): 575–586. [DOI: 10.1109/TPDS.2020.3025691]
- [142] SUN S Z, ZHANG R, MA H. Efficient parallelism of post-quantum signature scheme SPHINCS[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2020, 31(11): 2542–2555. [DOI: 10.1109/TPDS.2020.2995562]
- [143] GAO Y W, XU J, WANG H B. cuNH: Efficient GPU implementations of post-quantum KEM NewHope[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2021, 33(3): 551–568. [DOI: 10.1109/TPDS.2021.3097277]
- [144] KUO Y M, GARCÍA-HERRERO F, RUANO O, et al. RISC-V Galois field ISA extension for non-binary error-correction codes and classical and post-quantum cryptography[J]. *IEEE Transactions on Computers*, 2022, 72(3): 682–692. [DOI: 10.1109/TC.2022.3174587]
- [145] FRITZMANN T, SIGL G, SEPÚLVEDA J. RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020, 2020(4): 239–280. [DOI: 10.13154/tches.v2020.i4.239-280]
- [146] KARABULUT E, AYSU A. RANTT: A RISC-V architecture extension for the number theoretic transform[C]. In: *Proceedings of 30th International Conference on Field-Programmable Logic and Applications*. IEEE, 2020: 26–32. [DOI: 10.1109/FPL50879.2020.00016]
- [147] XIN G Z, HAN J, YIN T Y, et al. VPQC: A domain-specific vector processor for post-quantum cryptography based on RISC-V architecture[J]. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020, 67(8): 2672–2684. [DOI: 10.1109/TCSI.2020.2983185]
- [148] LI H M, MENTENS N, PICEK S. A scalable SIMD RISC-V based processor with customized vector extensions for CRYSTALS-Kyber[C]. In: *Proceedings of 59th ACM/IEEE Design Automation Conference (DAC '22)*. ACM,

2022: 733–738. [DOI: 10.1145/3489517.3530552]

[149] ZHANG J P, HUANG J H, LIU Z, et al. Time-memory trade-offs for Saber+ on memory-constrained RISC-V platform[J]. IEEE Transactions on Computers, 2022, 71(11): 2996–3007. [DOI: 10.1109/TC.2022.3143441]

[150] ZHAO Y F, XIE R Q, XIN G Z, et al. A high-performance domain-specific processor with matrix extension of RISC-V for module-LWE applications[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(7): 2871–2884. [DOI: 10.1109/TCSI.2022.3162593]

[151] BOS J W, GOURJON M O, RENES J, et al. Masking Kyber: First-and higher-order implementations[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021(4): 173–214. [DOI: 10.46586/tches.v2021.i4.173-214]

[152] HAMBURG M, HERMELINK J, PRIMAS R, et al. Chosen ciphertext k -trace attacks on masked CCA2 secure Kyber[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021(4): 88–113. [DOI: 10.46586/tches.v2021.i4.88-113]

[153] NGO K, DUBROVA E, GUO Q, et al. A side-channel attack on a masked IND-CCA secure Saber KEM implementation[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021(4): 676–707. [DOI: 10.46586/tches.v2021.i4.676-707]

[154] BHASIN S, D’ANVERS J P, HEINZ D, et al. Attacking and defending masked polynomial comparison for lattice-based cryptography[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021(3): 334–359. [DOI: 10.46586/tches.v2021.i3.334-359]

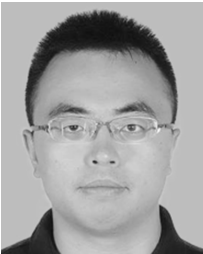
[155] RAVI P, CHATTOPADHYAY A, D’ANVERS J P, et al. Side-channel and fault-injection attacks over lattice-based post-quantum schemes (Kyber, Dilithium): Survey and new results[J]. ACM Transactions on Embedded Computing Systems, 2024, 23(2): 35. [DOI: 10.1145/3603170]

[156] SHALLER A, ZAMIR L, NOJOUMIAN M. Roadmap of post-quantum cryptography standardization: Side-channel attacks and countermeasures[J]. Information and Computation, 2023, 295, Part B: 105112. [DOI: 10.1016/j.ic.2023.105112]

作者信息



周天 (1996–), 安徽合肥人, 博士研究生. 主要研究领域为应用密码学.
weekdayzt@mail.ustc.edu.cn



郑昉昱 (1988–), 福建三明人, 博士, 副教授. 主要研究领域为应用密码学、高性能计算、计算机算术.
zhengfangyu@ucas.ac.cn



林璟锵 (1978–), 福建泉州人, 博士, 研究员、教授. 主要研究领域为密码应用、系统与网络安全.
linjq@ustc.edu.cn



魏荣 (1992–), 甘肃白银人, 博士. 主要研究领域为应用密码学.
nearbyssbit@163.com



唐文煦 (2000–), 江苏徐州人, 硕士研究生. 主要研究领域为应用密码学.
citrus@mail.ustc.edu.cn