针对格密码算法Kyber与Dilithium的能耗侧信道防护技术

李延斌[1),2)] 郭奕康[3)] 张舒琪[1)] 唐 明[4)] 葛春鹏[1),2)] 徐秋亮[1)]

[1)](山东大学软件学院 济南 250100)

[2)](泉城实验室 济南 250103)

[3)](南京农业大学人工智能学院 南京 210095)

[4)](武汉大学国家网络安全学院 武汉 430072)

摘 要 随着量子计算技术的迅猛发展,传统公钥密码体系面临重大安全威胁,后量子密码(PQC)成为新一代密码标准化的核心方向。美国国家标准与技术研究院(NIST)于2024年将基于模块化格上学习问题的Kyber算法和Dilithium算法分别确立为FIPS203密钥封装机制标准和FIPS204数字签名算法标准。在后量子密码的过渡过程中,实际环境中的物理安全性也逐渐被关注。格密码特有的多项式环运算、数论变换、消息编解码、边界检查、FO变换等操作导致其能耗侧信道泄露模式与传统密码存在显著差异,格密码泄露面广、算子种类多、运算复杂等特点对防护实现带来了巨大挑战。本文重点以Kyber和Dilithium为研究对象,系统梳理其侧信道脆弱点与防护目标,并对现有防护技术进行深入分析与评估:(1)揭示格密码侧信道攻击的脆弱性分布,进行泄露机理分析,归纳关键泄露点;(2)提出涵盖不同脆弱点的整体防护架构,对算子(如NTT、编解码等)的防护思路、实现难点进行分析,构建覆盖Kyber与Dilithium的防护全景图;(3)从设计思想、实现代价及优化方向评估现有方案,对防护方案给出实现建议,指出开销瓶颈与未来挑战。本研究为格密码侧信道防护提供理论参考与技术路线支撑,助力后量子密码在实际密码设备中的高效安全部署。

关键词 格密码;后量子密码;侧信道分析;侧信道防护;Kyber;Dilithium

中图法分类号 TP309 **DOI号** 10.11897/SP.J.1016.2026.00952

Countermeasures Against Power Side-Channel Attacks on the Lattice-Based Cryptographic Algorithms Kyber and Dilithium

LI Yan-Bin[1),2)] GUO Yi-Kang[3)] ZHANG Shu-Qi[1)] TANG Ming[4)]

GE Chun-Peng[1),2)] XU Qiu-Liang[1)]

[1)](School of Software, Shandong University, Jinan 250100)

[2)](Quan Cheng Laboratory, Jinan 250103)

[3)](College of Artificial Intelligence, Nanjing Agricultural University, Nanjing 210095)

[4)](School of Cyber Science and Engineering, Wuhan University, Wuhan 430072)

Abstract The rapid advancement of quantum computing has posed significant threats to traditional public-key cryptosystems, making post-quantum cryptography (PQC) a central focus in the standardization of next-generation cryptographic systems. In 2024, the National Institute of Standards and Technology (NIST) formally established Kyber and Dilithium—both based on the

收稿日期:2025-05-15;在线发布日期:2025-12-05。本课题得到国家自然科学基金面上项目(Nos. 62472259, 62572285, 62172258)、泰山学者工程专项经费资助(No. tsqn202408026)、山东省自然科学基金(No. ZR2024QF098)资助。**李延斌**,博士,副教授,中国计算机学会(CCF)高级会员,主要研究领域为密码工程、侧信道分析与防护。E-mail: yanbinli@sdu.edu.cn。**郭奕康**,硕士,主要研究领域为侧信道攻击。**张舒琪**,硕士,主要研究领域为侧信道攻击。**唐 明**(通信作者),博士,教授,主要研究领域为密码芯片安全。E-mail: m.tang@126.com。**葛春鹏**,博士,教授,主要研究领域为密码应用。**徐秋亮**,博士,教授,主要研究领域为应用密码学。

Module Learning with Errors (MLWE) problem—as the FIPS 203 standard for Key Encapsulation Mechanisms (KEM) and the FIPS 204 standard for digital signatures, respectively. As the transition to PQC progresses, increasing attention is being directed toward the physical security of cryptographic implementations in real-world environments. Lattice-based cryptography, which underlies these algorithms, introduces unique algebraic operations, including polynomial ring arithmetic, the Number Theoretic Transform (NTT), message encoding and decoding, boundary checks, and the Fujisaki-Okamoto (FO) transformation. These operations result in distinct power side-channel leakage patterns that differ markedly from those of classical cryptosystems. The characteristics of lattice-based algorithms—including a broad attack surface, diverse computational operators, and high operational complexity—present substantial challenges for implementing effective countermeasures.

This paper focuses on Kyber and Dilithium as key representatives of lattice-based PQC and systematically examines their vulnerabilities to side-channel attacks along with protection strategies. The contributions of this study are organized as follows: (1) Vulnerability Distribution and Leakage Mechanism Analysis: We reveal the distribution of side-channel vulnerabilities across lattice-based cryptosystems and analyze the underlying leakage mechanisms. By examining critical operations such as polynomial multiplication, NTT, and the encoding/decoding and sampling routines, we identify key points where sensitive information is at risk. These points are susceptible to various attacks, including Simple Power Analysis (SPA), Differential Power Analysis (DPA), and template attacks. This systematic mapping provides insights into how sensitive information may be exposed during the execution of lattice-based operations. (2) Fine-Grained Protection Framework: We propose an integrated protection architecture aimed at addressing various vulnerabilities in a structured manner. This framework emphasizes operator-specific protection strategies—for example, dedicated masking and hiding techniques for NTT, secure and constant-time implementations for encoding and decoding modules, and designs resistant to side-channel attacks for sampling and NTT steps. This approach enables the development of a comprehensive protection landscape for Kyber and Dilithium, addressing both algorithmic and implementation-level challenges. (3) Evaluation of Existing Solutions and Practical Guidance: We assess state-of-the-art countermeasures regarding their design principles, implementation costs (such as area, performance, and energy overhead), and potential optimization pathways. Based on this evaluation, we offer practical recommendations for deploying these protections in real-world devices. Additionally, we identify key bottlenecks—such as the high cost associated with masking non-linear operations and the challenges of achieving constant-time execution in complex sampling routines. We also outline future research challenges, including the need for efficient high-order masking schemes and the development of evaluation methodologies specifically tailored to lattice-based cryptography.

This study provides theoretical insights and practical guidelines for protecting against side-channel attacks in lattice-based cryptographic systems. Its goal is to support the secure and efficient implementation of post-quantum cryptosystems in real-world devices. By tackling both the theoretical and practical dimensions of side-channel resistance, this work contributes to the broader effort to ensure that the transition to PQC is both cryptographically strong and physically secure.

Keywords lattice-based cryptography; post-quantum cryptography; side-channel analysis; side-channel countermeasures; Kyber; Dilithium

1 引 言

随着量子计算技术的快速发展，基于传统数论难题的公钥密码体系面临严峻安全挑战，业界普遍认为，非对称加密算法将在2029年首次面临来自量子计算的挑战，到2034年，主流密码学技术可能将全面暴露于量子计算攻击的风险下。2030年和2035年将是后量子密码技术发展的两个关键里程碑节点，预计全球后量子密码技术市场规模将在2030年超过100亿美元①。为应对量子计算的威胁，美国国家标准与技术研究院（NIST）于2024年正式发布的后量子密码（Post-Quantum Cryptography，PQC）标准草案FIPS203和FIPS204中，CRYSTALS-Kyber（简称Kyber）算法与CRYSTALS-Dilithium（简称Dilithium）算法凭借其基于模块化格上学习问题（Module-LWE）的数学基础与高效的实现性能，分别被确立为密钥封装机制（KEM）和数字签名领域的标准化算法。这一里程碑事件标志着格基密码学进入新阶段，但同时也带来了实现级的安全挑战。

在密码实现安全领域，能耗侧信道分析（Power Side-Channel Analysis，SCA）通过采集密码器件运行时的动态功耗特征，可有效提取关键中间变量信息。经典密码体制下已有大量防护技术研究，但格密码特有的代数结构（如多项式环运算、NTT变换）和参数集特征导致其侧信道泄露模式呈现显著差异性。特别值得关注的是，格密码运算过程中既包含算术操作，又包含布尔操作，这对传统防护方案提出了新的适应性要求。

现有针对格密码的侧信道安全性相关综述论文中，文献[1]分析了后量子密码在能耗侧信道分析与物理故障注入场景下的安全性问题，但其未针对格密码体系下各个算子构建细粒度脆弱性模型；文献[2]横向对比了格基、编码基、哈希基及多变量基密码的物理攻击面特征，但缺乏对格密码侧信道泄露机理的讨论和防护策略的应对；文献[3]对后量子密码随机数生成器、物理不可克隆函数及简单侧信道防护（掩码、随机延迟）进行了讨论，但在格密码关键模块（如NTT运算单元、编解码等关键算子）的细粒度防护方案并未涉及；Ravi等人综述了针对格密码的侧信道攻击各算子的方法，并给出了相应的防护策略，该工作侧重攻击方法的梳理，防护部分只进行了简短的防护思路是掩码还是乱序等策略，缺少各算子现有防护方案的分析[4]。本文以NIST标准化算法Kyber和Dilithium为对象，从算法侧信道脆弱点出发，整理格密码抵抗侧信道攻击的整体防护目标，对现有格密码的侧信道防护技术进行分析比较，本文的主要贡献有三个方面：

（1）分析了现有针对格密码的侧信道攻击脆弱点，总结归纳了侧信道防护需要考虑的泄露点；

（2）对于格密码的侧信道防护方案按对应脆弱点、防护算子、设计方案等维度进行讨论，给出Kyber和Dilithium的防护总体架构和各算子防护方案，构建格密码标准的防护全景图；

（3）讨论不同防护方案的设计思想、实现代价及实现建议，分析评估现有防护方案的开销瓶颈和现存问题。

本文后续组织如下：第二节对格密码和侧信道攻击、防护的原理进行了介绍；第三节梳理了格密码Kyber和Dilithium的侧信道脆弱点；第四节给出Kyber和Dilithium的整体防护策略；第五节对Kyber和Dilithium各算子的防护典型方案、防护代价和实现建议进行了具体展开介绍；第六节分析了现有防护方案的防护代价分析；第七节讨论了面临问题和挑战；最后对全文进行了总结。

2 背景知识

2.1 格密码

随着量子计算领域取得重大突破，传统密码体系将会受到巨大的冲击，尤其是公钥加密算法很容易被量子计算机破解。2016年，美国国家标准与技术研究院（NIST）发布了后抗量子公钥加密算法（PQC）的标准列表，并于2024年正式公布了三种后量子密码标准草案，分别为：

• FIPS 203（ML-KEM）：基于格密码的密钥封装机制标准，源自CRYSTALS-KYBER算法提案，专为加密目的而设计。

• FIPS 204（ML-DSA）：基于格密码的数字签名标准，源自CRYSTALS-Dilithium算法提案，旨在保护用户在远程签署文档时使用的数字签名。

• FIPS 205（SLH-DSA）：基于哈希算法的无状态数字签名标准，源自SPHINCS＋算法提案，为Dilithium的替代算法之一。

① 后量子密码安全能力构建技术指南（2025版），https://www.aqniu.net/news/details/20250331085406958527?type=5

在正式公布的三种算法中，Kyber和Dilithium都是基于格的密码。大多数基于格的密码都是基于错误学习(LWE)问题构建的。在不同的替代方案中，代数结构变体Ring LWE(R-LWE)和Module LWE(M-LWE)非常具有吸引力，因为它们比普通LWE问题具有性能优势和更小的密钥大小[5-6]。

Lyubashevsky等人介绍了R-LWE问题[7]。与简单的LWE问题不同，R-LWE用小于 n 次的多项式替换 n 维向量。这些多项式是环 $\mathcal{R}=\mathbb{Z}_q/\langle\phi(x)\rangle$ 的元素，具有整数 n 和 q，以及通常选择为 x^n+1 的割圆多项式 $\phi(x)$。这种替换可以提高计算性能并减小密钥和密文大小。R-LWE可以写成 $\boldsymbol{b}=\boldsymbol{a}\cdot\boldsymbol{s}+\boldsymbol{e}$，其中 $\boldsymbol{a}$ 是公共多项式，$\boldsymbol{s}$ 是秘密多项式，$\boldsymbol{e}$ 是误差多项式，所有算术运算都在环 $\mathcal{R}$ 中执行。R-LWE问题的难度基于给定 $\boldsymbol{a}$ 和结果 $\boldsymbol{b}$ 时恢复 $\boldsymbol{s}$ 的难度。除此之外，区分 $(\boldsymbol{a},\ \boldsymbol{b})$ 和真正均匀独立的样本对是一个难题。通常，$\boldsymbol{a}$ 是从均匀分布 U_q 中采样，其结果范围在0到 $q-1$ 之间。多项式 $\boldsymbol{s}$ 和 $\boldsymbol{e}$ 从误差分布 Ψ_k 中采样，该分布通常是二项式或高斯分布。

随后，Langlois等人介绍了M-LWE问题[8]。与R-LWE相比，M-LWE问题将单个环元素替换为同一环上的模元素。它不是使用单个大环元素，而是使用较小环元素的矩阵来构造公共元素，使用较小环元素的向量来构造秘密和错误元素。R-LWE可以被认为是M-LWE的一个特例，其中矩阵 $\boldsymbol{a}$ 在环 $\mathcal{R}$ 上的宽度总是1[9]。为了提高安全级别，在环始终相同的情况下，需要增加矩阵和向量的维数。也就是说，底层环操作不会改变，而是这些操作的执行量会改变。M-LWE的主要优点是，其安全性和效率的权衡是高度可调的[5]。Kyber和Dilithium均为建立在M-LWE基础上的算法。

2.2　CRYSTALS-Kyber

Kyber是基于模误差学习问题的密钥封装机制，包括密钥生成、封装和解封装，具体过程如算法1到算法6所示。其中，算法1-3分别为Kyber的密钥生成、加密和解密，组成了CPA安全下的公钥加密算法。算法4-6在算法1-3的基础上变换为了密钥生成、封装和解封装三个部分，进一步实现了CCA安全下的密钥封装机制。

算法1-3展示的CPA安全下的加密算法与公钥加密算法流程一致。在密钥生成阶段，算法通过采样生成一个公钥 pk 和一个私钥 sk，用于之后的加密和解密过程。

算法1.　KYBER.CPAPKE.KeyGen()

输出：私钥 $sk\in\mathcal{B}^{12\cdot k\cdot n/8}$

输出：公钥 $pk\in\mathcal{B}^{12\cdot k\cdot n/8+32}$

1. $d\leftarrow\mathcal{B}^{32}$
2. $(\rho,\ \sigma)=\mathrm{G}(d)$
3. $N=0$
4. For i from 0 to $k-1$ do
5. 　　For j from 0 to $k-1$ do
6. 　　$\hat{\mathbf{A}}^T[i][j]=\mathrm{Parse}(\mathrm{XOF}(\rho,\ i,\ j))$
7. For i from 0 to $k-1$ do
8. 　　$\boldsymbol{s}[i]=\mathrm{CBD}_{\eta_1}(\mathrm{PRF}(\sigma,\ N))$
9. 　　$N=N+1$
10. For i from 0 to $k-1$ do
11. 　　$\boldsymbol{e}[i]=\mathrm{CBD}_{\eta_1}(\mathrm{PRF}(\sigma,\ N))$
12. 　　$N=N+1$
13. $\hat{\boldsymbol{s}}=\mathrm{NTT}(\boldsymbol{s})$
14. $\hat{\boldsymbol{e}}=\mathrm{NTT}(\boldsymbol{e})$
15. $\hat{\boldsymbol{t}}=\hat{\boldsymbol{A}}\circ\hat{\boldsymbol{s}}+\hat{\boldsymbol{e}}$
16. $pk=(\mathrm{Encode}_{12}(\hat{\boldsymbol{t}}\bmod^+q)\|\rho)$
17. $sk=\mathrm{Encode}_{12}(\hat{\boldsymbol{s}}\bmod^+q)$
18. Return $(pk,\ sk)$

算法2.　KYBER.CPAPKE.Enc(pk,m,r)

输入：公钥 $pk\in\mathcal{B}^{12\cdot k\cdot n/8+32}$

输入：消息 $m\in\mathcal{B}^{32}$

输入：随机数 $r\in\mathcal{B}^{32}$

输出：密文 $c\in\mathcal{B}^{d_u\cdot k\cdot n/8+d_v\cdot n/8}$

1. $N=0$
2. $\hat{\boldsymbol{t}}=\mathrm{Decode}_{12}(pk)$
3. $\rho=pk+12\cdot k\cdot n/8$
4. For i from 0 to $k-1$ do
5. 　　For j from 0 to $k-1$ do
6. 　　$\hat{\boldsymbol{A}}^T[i][j]=\mathrm{Parse}(\mathrm{XOF}(\rho,\ i,\ j))$
7. For i from 0 to $k-1$ do
8. 　　$\boldsymbol{r}[i]=\mathrm{CBD}_{\eta_1}(\mathrm{PRF}(\boldsymbol{r},\ N))$
9. 　　$N=N+1$
10. For i from 0 to $k-1$ do
11. 　　$\boldsymbol{e}_1[i]=\mathrm{CBD}_{\eta_2}(\mathrm{PRF}(\boldsymbol{r},\ N))$
12. 　　$N=N+1$
13. $\boldsymbol{e}_2=\mathrm{CBD}_{\eta_2}(\mathrm{PRF}(\boldsymbol{r},\ N))$
14. $\hat{\boldsymbol{r}}=\mathrm{NTT}(\boldsymbol{r})$
15. $\boldsymbol{u}=\mathrm{NTT}^{-1}(\hat{\boldsymbol{A}}^T\circ\hat{\boldsymbol{r}})+\boldsymbol{e}_1$
16. $v=\mathrm{NTT}^{-1}(\hat{\boldsymbol{t}}^T\circ\hat{\boldsymbol{r}})+\boldsymbol{e}_2+\mathrm{Decompress}_q(m,\ 1)$
17. $c_1=\mathrm{Encode}_{d_u}(\mathrm{Compress}_q(\boldsymbol{u},\ d_u))$
18. $c_2=\mathrm{Encode}_{d_v}(\mathrm{Compress}_q(v,\ d_v))$
19. Return $c=(c_1\|c_2)$

算法3. KYBER. CPAPKE. Dec(sk,c)

输入:私钥 $sk\in\mathcal{B}^{12\cdot k\cdot n/8}$

输入:密文$c\in\mathcal{B}^{d_u\cdot k\cdot n/8+d_v\cdot n/8}$

输出:消息 $m\in\mathcal{B}^{32}$

1. $\boldsymbol{u}=\mathrm{Decopmress}_q(\mathrm{Decode}_{d_u}(c),\ d_u)$
2. $v=\mathrm{Decopmress}_q(\mathrm{Decode}_{d_v}(c+d_u\cdot k\cdot n/8),\ d_v)$
3. $\hat{\boldsymbol{s}}=\mathrm{Decode}_{12}(sk)$
4. $m=\mathrm{Encode}_1(\mathrm{Compress}_q(v-\mathrm{NTT}^{-1}(\hat{\boldsymbol{s}}^T\circ\mathrm{NTT}(\boldsymbol{u})),$ 1))
5. Return m

在加密阶段,Kyber使用公钥pk将消息m编码为密文c。Kyber利用多项式乘法和NTT进行加密,密文c包含两个分量c_1和c_2,消息m仅包含在c_2中,并加入随机数和噪声项进行加密,而c_1则用于解密。

在解密阶段,Kyber使用私钥sk从密文c中恢复原始消息m。解密过程同样使用了多项式乘法和NTT,结合c_1和c_2进行解密。

Kyber在每次加密时,向量$\boldsymbol{r}$, $\boldsymbol{e}_1$, $\boldsymbol{e}_2$都是从分布中重新采样的,即使消息m相同,密文c也会因随机项不同而变化。因此,攻击者无法通过密文区分明文,满足不可区分性,实现了CPA安全。

算法4-6进一步实现了Kyber的CCA安全,在算法1-3的基础上引入了哈希函数和FO变换,扩展为了封装和解封装流程。

算法4. KYBER. CCAKEM. KeyGen()

输出:私钥 $sk\in\mathcal{B}^{24\cdot k\cdot n/8+96}$

输出:公钥 $pk\in\mathcal{B}^{12\cdot k\cdot n/8+32}$

1. $z\leftarrow\mathcal{B}^{32}$
2. (pk, sk')=KYBER. CPAPKE. KeyGen()
3. $sk=(sk'\|pk\|\mathrm{H}(pk)\|z)$
4. Return (pk, sk)

算法5. KYBER. CCAKEM. Enc(pk)

输入:公钥$pk\in\mathcal{B}^{12\cdot k\cdot n/8+32}$

输出:共享密钥 $K\in\mathcal{B}^*$

输出:密文$c\in\mathcal{B}^{d_u\cdot k\cdot n/8+d_v\cdot n/8}$

1. $m\leftarrow\mathcal{B}^{32}$
2. $m\leftarrow\mathrm{H}(m)$
3. ($\overline{K}$, r)=G($m\|\mathrm{H}(pk)$)
4. c=KYBER. CPAPKE. Enc(pk, m, r)
5. $K=\mathrm{KDF}(\overline{K}\|\mathrm{H}(c))$
6. Return (c, K)

算法6. KYBER. CCAKEM. Dec(c, sk)

输入:密文$c\in\mathcal{B}^{d_u\cdot k\cdot n/8+d_v\cdot n/8}$

输入:私钥 $sk\in\mathcal{B}^{24\cdot k\cdot n/8+96}$

输出:共享密钥 $K\in\mathcal{B}^*$

1. $pk=sk+12\cdot k\cdot n/8$
2. $h=sk+24\cdot k\cdot n/8+32\in\mathcal{B}^{32}$
3. $z=sk+24\cdot k\cdot n/8+64$
4. m'=KYBER. CPAPKE. Dec(sk, c)
5. ($\overline{K}'$, r')=G($m'\|h$)
6. c'=KYBER. CPAPKE. Enc(pk, m', r')
7. If $c=c'$ then
8. Return $K=\mathrm{KDF}(\overline{K}'\|\mathrm{H}(c))$
9. else
10. Return $K=\mathrm{KDF}(z\|\mathrm{H}(c))$
11. Return K

哈希函数(算法5第3行)保证加密中所有随机性都可由消息m和公钥pk唯一确定,从而便于后续的FO变换。此外,KDF(算法5第5行)保证了共享密钥不仅由消息m决定,还依赖于密文c,防止攻击者篡改密文。FO变换(算法6第5-10行)实现了重加密过程并验证密文一致性,从而验证密文是否合法,达到了CCA安全。

2.3 CRYSTALS-Dilithium

Dilithium是NIST第三轮选出的两种基于格的数字签名算法之一,提供安全级别2、3、5的保障,密码设计简单,部署方便。Dilithium基于“带中止的Fiat-Shamir”框架的签名方案,核心功能包括用于密钥生成的KeyGen、用于生成签名的Sign以及用于验证签名的Verify,具体过程如算法7-9所示。

密钥生成阶段负责生成签名方案所需的公钥和私钥。使用随机种子生成伪随机矩阵$\boldsymbol{A}$和向量$\boldsymbol{s}_1$和$\boldsymbol{s}_2$,计算公开向量$\boldsymbol{t}$并将其分解。

算法7. Dilithium. KeyGen

输出:pk, sk

1. $\zeta\leftarrow\{0,1\}^{256}$
2. $(\rho,\mathrm{s},K)\in\{0,1\}^{256\times3}:=\mathrm{H}(\zeta)$
3. $(\boldsymbol{s}_1,\boldsymbol{s}_2)\in S_\eta^l\times S_\eta^k:=\mathrm{H}(\mathrm{s})$
4. $\mathrm{A}\in R_q^{k\times l}:=\mathrm{ExpandA}(\rho)$
5. $\boldsymbol{t}:=\boldsymbol{A}\boldsymbol{s}_1+\boldsymbol{s}_2$
6. $(\boldsymbol{t}_1,\boldsymbol{t}_0):=\mathrm{Power2Round}_q(\boldsymbol{t},d)$
7. $tr\in\{0,1\}^{384}:=\mathrm{CRH}(\rho\|\boldsymbol{t}_1)$
8. return$\big(pk=(\rho,\boldsymbol{t}_1)$, $sk=(\rho,K,tr,\boldsymbol{s}_1,\boldsymbol{s}_2,\boldsymbol{t}_0)\big)$

签名阶段使用私钥为给定的消息M生成签名σ。其核心是一个可能需要多次尝试的循环:首先随机采样$\boldsymbol{y}$并计算$\boldsymbol{w}$,基于消息M和$\boldsymbol{w}$的部分生成挑战c,然后用私钥计算$\boldsymbol{z}$并检查其范数是否合格,最后计算提示位$\boldsymbol{h}$并检查其数量。如果任何检查

失败，则使用新的随机性重试循环；成功则输出签名$\sigma=(\boldsymbol{z},\boldsymbol{h},\tilde{c})$。

算法8. Dilithium. Sign

输入：(sk, M)

输出：$\sigma=(\boldsymbol{z},\boldsymbol{h},\tilde{c})$

1. $\boldsymbol{A}\in R_q^{k\times l}:=\mathrm{ExpandA}(\rho)$
2. $\mu\in\{0,1\}^{384}:=\mathrm{CRH}(tr\|M)$
3. $\kappa:=0$, $(\boldsymbol{z},\boldsymbol{h}):=\perp$
4. $\rho'\in\{0,1\}^{384}$
5. While $(\boldsymbol{z},\boldsymbol{h})=\perp$ do
6. $\boldsymbol{y}\in\tilde{S}_{\gamma_1}^{l}:=\mathrm{ExpandMask}(\rho',\kappa)$
7. $\boldsymbol{w}:=\boldsymbol{A}\boldsymbol{y}$
8. $\boldsymbol{w}_1:=\mathrm{HighBits}_q(\boldsymbol{w},2\gamma_2)$
9. $\tilde{c}\in\{0,1\}^{256}:=\mathrm{H}(\mu\|\boldsymbol{w}_1)$
10. $c\in B_\tau:=\mathrm{SamplelnBall}(\tilde{c})$
11. $\boldsymbol{z}:=\boldsymbol{y}+c\boldsymbol{s}_1$
12. $\boldsymbol{r}_0:=\mathrm{LowBits}_q(\boldsymbol{w}-c\boldsymbol{s}_2,2\gamma_2)$
13. If $\|\boldsymbol{z}\|_\infty\geqslant\gamma_1-\beta$ OR $\|\boldsymbol{r}_0\|_\infty\geqslant\gamma_2-\beta$ then $(\boldsymbol{z},\boldsymbol{h}):=\perp$
14. else
15. $\boldsymbol{h}:=\mathrm{MakeHint}_q(-c\boldsymbol{t}_0,\boldsymbol{w}-c\boldsymbol{s}_2+c\boldsymbol{t}_0,2\gamma_2)$
16. If $\|c\boldsymbol{t}_0\|_\infty\geqslant\gamma_2$ OR $\boldsymbol{h}$中1的数量比ω多 then $(\boldsymbol{z},\boldsymbol{h}):=\perp$
17. $\kappa:=\kappa+1$
18. Return $\sigma=(\boldsymbol{z},\boldsymbol{h},\tilde{c})$

签名验证阶段使用公钥来验证一个签名σ对于给定消息M是否有效。验证通过当且仅当：$\boldsymbol{z}$的范数足够小、重构的$\tilde{c}'$与收到的$\tilde{c}$匹配且提示位h的数量未超标。

算法9. Dilithium. Verify

输入：$(pk,M,\sigma=(\boldsymbol{z},\boldsymbol{h},\tilde{c}))$

输出：0或1(拒绝或接受)

1. $\mathrm{A}\in R_q^{k\times\ell}:=\mathrm{ExpandA}(\rho)$
2. $\mu\in\{0,1\}^{384}:=\mathrm{CRH}(\mathrm{CRH}(\rho\|\boldsymbol{t}_1)\|M)$
3. $c:=\mathrm{SamplelnBall}(\tilde{c})$
4. $\boldsymbol{w}_1':=\mathrm{UseHint}_q(\boldsymbol{h},\mathrm{A}\boldsymbol{z}-c\boldsymbol{t}_1\cdot 2^d,2\gamma_2)$
5. Return $[\|\boldsymbol{z}\|_\infty<\gamma_1-\beta]$ AND $[\tilde{c}=H(\mu\|\boldsymbol{w}_1')]$ AND[# of 1's in h is $\leqslant\omega$]

2.4　能耗侧信道攻击

侧信道攻击(Side-Channel Attacks，SCA)可以通过分析设备在执行加密方案时产生的物理泄漏来进行。这些泄漏可以被攻击者利用，用以分析和破坏加密算法的安全性[10]。1999年，Kocher提出了一种基于设备功耗的侧信道攻击[11]，攻击利用了设备本身在对密码算法的秘密信息执行相关操作时会泄漏物理信息的事实，攻击者可以根据目标设备产生的能耗差异区分该设备上执行的秘密信息，此类攻击即为能耗侧信道攻击。

能耗侧信道攻击的关键在于能耗分析(Power Analysis，PA)，需要攻击者随时间测量目标设备的能耗迹线。这些迹线与算法的计算步骤有关，因为算法的每个操作都有其功率特征[12]。对于能耗迹线有不同的分析方式，当前主流的方法有简单能耗分析(SPA)、差分能耗分析(DPA)和相关能耗分析(CPA)等。

能耗侧信道攻击属于被动攻击，攻击者必须了解目标系统，并仅在算法运行时观察设备产生的能耗，而不能对其进行操作。因此，能耗侧信道攻击是更危险的攻击类型，因为它们不会对设备造成可以检测到的损坏，难以被察觉。

2.5　能耗侧信道防护

为了应对能耗侧信道攻击，已经出现了一些用于抵御此类攻击的防护技术。当前的防护手段主要分为四种类别。

(1) 随机延迟技术(Random Delay)

随机延迟的思想是在程序执行过程中随机插入空操作或延迟指令，为每个函数执行前引入随机时长的等待，使得侧信道攻击者难以对齐每条侧信道迹线中的泄露点。

随机延迟的优势在于实现简单，开销较低，能有效增加攻击者采集与分析的难度。然而，随机延迟并未减少泄漏，只是增加攻击难度。因此，其对能容忍时序抖动的攻击(如深度学习攻击)防护效果较弱，攻击者通过增加攻击的迹线数量仍然可能恢复秘密信息。

(2) 乱序技术(Shuffling)

对于没有严格先后顺序的操作，每次执行时可以采用不同的执行顺序，避免固定模式的泄漏，从而使攻击者无法判断侧信道迹线对应的操作。

乱序技术对简单的攻击有效，能够增加攻击者定位泄露点的难度。然而，密码算法中可打乱的操作数量通常有限，保护效果存在上限。此外，一些高级攻击(如基于模式识别的攻击)可通过重排序或聚类技术绕过乱序防护。

(3) 功耗均衡技术(Power Balancing)

通过硬件设计可以使电路在不同数据输入下产生近似相同的功耗。功耗均衡技术让功耗与计算内

容无关，从而抵抗能耗分析攻击。

功耗均衡技术可显著降低能耗与数据之间的相关性，有效抵抗一阶能耗分析攻击。然而，在实现中难以做到完全相同的功耗，且会导致整体功耗较高。此外，对高阶或多点攻击仍存在风险。

(4) 掩码技术(Masking)

掩码技术引入随机数对算法中的敏感变量进行组合，使得泄露的能耗信息与实际变量无关，让攻击值无法利用侧信道泄露。

掩码技术具备理论安全性，已被形式化证明对任意阶攻击有效，是可以从根本上消除侧信道泄露的防护方案。掩码技术的唯一缺点在于实现开销较大，复杂度较高。因此，我们在下文的写作中重点讨论了掩码防护方案，并分析了每种方案在实现时的开销。

2.6 安全模型

在介绍各个算子的掩码防护方案之前，我们首先需要说明抵抗侧信道攻击的安全模型。

*t-probing*探测模型，也称为阈值探测模型，是一种泄漏模型，用于分析算法实现对侧信道攻击的安全性。Ishai等人(2003)在有限域$\mathbb{F}_2$中引入了该模型来证明掩码对策的安全性，然后由Rivain和Prouff(2010)将该模型扩展到任意有限域。该安全模型指出，如果电路中任何一组t根导线所携带的值都独立于任何秘密值，则该掩码算术电路是*t-probing*安全的，参数t称为安全阶数。

为了增强安全性定义，并扩展到可组合安全性，Barthe等人和Cassiers等人又提出了*t-NI*、*t-SNI*和*t-PINI*定义[13-14]，下面将分别介绍这几种安全模型的定义。

定义1. *t-probing*模型

掩码电路可以被分为三部分：(i) 从敏感输入变量生成掩码(即输入掩码)的随机编码器子电路I，(ii) 对输入掩码执行安全计算，以生成输出掩码的实际掩码电路C，(iii) 组合输出掩码的解码器子电路O。假设I和O部分没有发生泄漏。在探测模型中，攻击者可以放置t个探针(t为安全参数)，一个探针在掩码电路C中选择一条导线。当电路执行时，攻击者观察每个探针的值。如果对于攻击者的任何探测，观察结果与敏感输入变量是联合独立分布的，那么这个电路是*t-probing*安全。因此，当且仅当电路C的每一个中间变量的t个元组都独立于任何敏感变量，电路C是*t-probing*安全的。

定义1. *t-NI*模型(t-Non-Interference)

如果任何一组t个探针可以使用每组输入掩码中最多t个掩码进行模拟，则该装置被称为*t-NI*安全。*t-NI*模型通常等价于*t-probing*模型。

定义3. *t-SNI*模型(t-Strong Non-Interference)

对中间掩码的任何t_i个探针和对输出掩码的t_o个探针进行操作，并且满足$t_i+t_o\leqslant t$，如果可以使用每个输入掩码中最多t_i个掩码进行模拟。那么具有一个输出掩码的装置被称为*t-SNI*安全。

定义4. *t-PINI*模型(t-Probe-Isolating Non-Interference)

如果对于中间掩码的任何t_i个探针和集合$S_1\subset[\![0,\ t]\!]$，满足$t_i+|S_1|\leqslant t$，则存在一个集合$S_2\subset[\![0,\ t]\!]$，满足$|S_2|\leqslant t$，使得输出掩码中索引属于S_1的部分可以通过使用每个输入掩码中索引属于$S_1\cup S_2$的部分进行模拟。那么，该装置就被称为*t-PINI*安全。

根据上述定义，如果一个掩码电路是t阶安全的，则满足在该电路任意位置放置t个探针，监测到的信息不足以恢复出秘密值。这种情况下，通常需要t阶掩码实现：将一个秘密值拆分为$t+1$个掩码值，只要$t+1$个掩码相互独立，没有进行交叉运算，那么t处探针得到的信息就无法掌握整个秘密值。然而，*t-NI*安全不具有可组合性，当前后两个模块都是*t-NI*安全时，就不能保证整体的t阶安全性，例如在第一个模块放置一个探针，第二个模块放置一个探针，就有可能恢复出秘密值。在此基础上提出的*t-PINI*安全性具有可组合性，当两个模块都是*t-PINI*安全时，其组合也是*t-PINI*安全的。

3 格密码能耗侧信道脆弱点分析

3.1 Kyber脆弱点分析

Kyber为了抵御侧信道攻击，需要对可能泄露敏感信息的算子进行防护，密钥生成会生成密钥对，因此在密钥生成阶段需要对生成敏感信息的采样器进行防护，防止直接泄露敏感信息，如算法1的第3行；封装阶段将明文消息加密成密文，同时生成一个共享密钥。在封装阶段的加密过程中会用到哈希函数和采样器，此外在编码过程中，解压缩算子会间接地泄露信息，从而使攻击者能够恢复敏感信息，因此加密阶段需要对哈希函数、采样器和解压缩进行防护；解封装阶段用于解密接收到的密文，并从中提取出共享密钥。解封装过程使用接收者的私钥来解

密密文，并生成一个相同的共享密钥，Kyber解封装过程会用到解密和FO变换算法，该过程会额外涉及压缩和比较操作，这些操作也可能成为侧信道攻击者攻击的对象，因此也需要对其进行掩码防护。

Kyber密钥封装机制执行过程中，由于包含了消息与密钥相关的敏感信息，因此会遭受能量侧信道攻击，脆弱点（Kyber Vulnerability，KV）可分为以下几个：

KV1：NTT算法，虽然Kyber算法中有多处调用了NTT，但泄露主要发生在算法2的第14-16行，以及算法3的第4行，NTT和NTT^{-1}会泄露其中计算的信息。针对NTT进行侧信道攻击，通过构建NTT输入和中间值的汉明重量模板，结合BP（置信传播算法）和SASCA（软侧信道攻击）建立因子图进行迭代[15]，从而恢复出私钥，再结合密文算出消息。具体而言，攻击者首先为NTT的输入和中间状态构造汉明重量模板，可以得到概率分布$Pr(X=x|l)$，其中l为测量得到的汉明重量，x为NTT算法中间值。随后，攻击者通过SASCA建立一个因子图，把测得的概率分布$Pr(X=x|l)$作为BP的初始和中间节点输入。BP从第一层（输入层）开始执行因子图更新，直到NTT输出，然后反转此过程从输出更新输入直至收敛，这样就可以更新输入的节点进而得到更准确的概率分布。

KV2：哈希算法，主要的泄露发生在算法5的第3行和算法6的第5行，哈希函数G会泄露敏感信息。哈希算法的核心在于Keccak函数，其中包含五个关键操作，分别为θ, ρ, π, χ和ι。在θ的输入处，SRAM访问（加载和存储）会泄露信息，攻击者可以得到关于输入状态的汉明重量模板。

Kannwischer等人针对封装过程中的哈希函数进行侧信道攻击，通过得到Keccak输入和中间值的汉明重量模板[16]。在构建完模板之后，即可执行加密/解密算法，对哈希函数检测侧信道泄露得到能耗迹线。随后与针对NTT的攻击一样，结合BP和SCACA画出因子图进行迭代得到输入值的概率还原出m，再结合密文得到有关密钥s的信息。

Anvers等人在论文中攻击了FO变换中的哈希函数[17]，其核心思想是构建特殊的$\boldsymbol{u}$和v来推测密钥$\boldsymbol{s}$的值。具体来说，攻击者通过构造特殊的密文$\boldsymbol{u}$和v，可以使得消息$\mathrm{m}[i]$只跟一位密钥$\boldsymbol{s}[i]$有关，$\mathrm{m}[i]$的值可以通过侧信道攻击得到，通过使用多组特殊密文$(\boldsymbol{u},v)$进行攻击可以使得$\mathrm{m}[i]$能够唯一识别密钥$\boldsymbol{s}[i]$，从而建立一个查找表。

KV3：解码器算法，位于算法3的第4行，其中Compress函数会泄露解密过程的信息。Kyber的解码器算法用于将解密后的12位结果映射为1位的消息，并存储为字节的形式输出。该部分中的1位消息m容易通过寄存器泄露其汉明重量，因此容易遭受侧信道攻击。通常，攻击者需要构造特殊的密文$(\boldsymbol{u},v)$，使得消息m只与1位密钥$\boldsymbol{s}$有关，m的值可以通过侧信道攻击得到，通过使用多组特殊密文$(\boldsymbol{u},v)$攻击可以得到一组m，每组m可以唯一识别密钥$\boldsymbol{s}$。

Ravi等人在论文中攻击了解密阶段中的长期密钥$\boldsymbol{s}$[18]。在解码器中，消息m每次输出1位，最终存储为8位（字节）的形式，目前有两种主流存储方式：一种是每更新1位就存1位，称为增量存储；另一种是全部更新完了再存储，称为字节存储。Ravi根据存储方式的不同，利用TVLA选择PoI去建立关于m的模板，如果是增量存储可以直接通过每次迭代的汉明重量得到m的值，如果是字节存储需要结合密钥延展性来进行比较得到m的值。随后构造特殊的$(\boldsymbol{u},v)$以及对应的查找表，并按照查找表搜索对应$\boldsymbol{s}$的值。

Xu等人同样攻击解密中的长期密钥$\boldsymbol{s}$[19]。由于$\boldsymbol{s}$中多项式的系数是从二项分布中采样的，对于Kyber-768，其值的范围仅限于-2到2，因此Xu采用了不同于先前工作的密文构建方法，选择特殊的$\boldsymbol{u}$和v，使得每个消息位充当二分类器，可以区分密钥$\boldsymbol{s}$相应系数的五个可能值。

KV4：比较算法，位于算法6的第7-10行，比较的结果显示了解密成功或是失败，当攻击者了解到发生解密失败时即可执行解密失败攻击。Bhasin等人提出了一种针对掩码比较的侧信道攻击[20]，攻击者通过侧信道泄露可以了解到比较成功或是失败，进而可以知道解密得到的消息m'是否被成功解密。该方法的原理在于，如果将构造的密文用于解密，那么解密得到的m'将有可能是非法的，从而导致重新加密的c'出错，进而导致c和c'的比较会失败。比较失败与比较成功时设备会产生不同的功耗，因此攻击者可以通过侧信道攻击观察到能耗发生变化。

在Bhasin的方法中，攻击者可以首先通过封装得到合法的密文v，通过主动给v添加一个错误项e来构造新的密文。错误项e通常导致新的密文仅在

一个系数上与v不同。如果e足够小，则不会导致解密失败，此时比较的结果仅在一个系数上失败；反之，如果e足够导致解密失败，此时比较的结果中将有大量系数发生失败。因此，攻击者可以间接通过侧信道观察是否发生解密失败，从而推算出导致解密失败的e的临界值。随后，即可采用传统的解密失败攻击恢复密钥$\mathbf{s}$。

3.2 Dilithium脆弱点分析

Dilithium签名算法执行过程中，由于包含了私钥相关的敏感信息，因此可能会遭受能耗侧信道攻击，脆弱点（Dilithium Vulnerability，DV）可分为以下几个：

DV1：采样，首先要在敏感信息$\boldsymbol{s}_1$、$\boldsymbol{s}_2$和$\boldsymbol{y}$的生成阶段需要加以防护，算法7第3行，对于私钥组成部分$\boldsymbol{s}_1$和$\boldsymbol{s}_2$，要避免导致签名伪造的简单密钥恢复攻击；算法8第6行，对于$\boldsymbol{y}$，给定一个有效的签名$\sigma=(\tilde{c},\mathbf{z},\mathbf{h})$，通过已知或部分已知的$\boldsymbol{y}$，可以从$\boldsymbol{z}:=\boldsymbol{y}+c\boldsymbol{s}_1$中恢复向量$\boldsymbol{s}_1$[21]。Wang等人利用私钥解包过程的泄露，通过功耗分析恢复$\boldsymbol{s}_1$和$\boldsymbol{s}_2$一半的系数，再利用$\boldsymbol{t}:=\boldsymbol{A}\boldsymbol{s}_1+\boldsymbol{s}_2$，建立方程组求解剩余的私钥系数，其中$\boldsymbol{A}$和$\boldsymbol{t}$是公钥的组成部分[22]。结果表明，在假设$\boldsymbol{t}_0$已知的攻击中，从不同分析设备捕获的单个轨迹成功恢复完整向量$\boldsymbol{s}_1$的概率是不可忽略的（9%），如果攻击多条轨迹，成功率接近100%。Qiao等人通过两阶段攻击先恢复多项式加法$\boldsymbol{z}:=\boldsymbol{y}+c\boldsymbol{s}_1$中的$\boldsymbol{y}$和$c\boldsymbol{s}_1$的值，再求解含有误差的$c\boldsymbol{s}_1$方程组[23]。结果仅使用两个签名，在五分钟内就能执行攻击，且单个签名在一小时内恢复私钥的概率为60%。Fan等人[24]提出了一种针对Dilithium参考实现中随机多项式生成过程的筛选最小二乘辅助多元模板攻击方法（SLST assisted MTA），通过利用与系数$\boldsymbol{y}_i$、无符号数$\boldsymbol{x}_i$和随机字节串$\boldsymbol{a}_i$相关的泄漏，实现多项式系数恢复，构建针对Dilithium私钥$\boldsymbol{s}_1$的容错方程体系。因此$\boldsymbol{s}_1$、$\boldsymbol{s}_2$和$\boldsymbol{y}$的构成必须保密，以避免通过相关攻击导致签名伪造[25]。

DV2：分解，在签名阶段，算法8第7行，向量$\boldsymbol{w}:=\boldsymbol{A}\boldsymbol{y}$应该进行掩码，如果未进行掩码处理，由于$\boldsymbol{A}$要么是方阵，要么是行数大于列数的矩阵，可以通过$\boldsymbol{w}$高效地恢复变量$\boldsymbol{y}$，从而恢复私钥[26]。另外，算法8的第8行中，利用*Decompose*函数提取$\boldsymbol{w}$的高、低阶位$\boldsymbol{w}_1$和$\boldsymbol{w}_0$，即$(\boldsymbol{w}_0,\boldsymbol{w}_1):=Decompose_q(\boldsymbol{w},2\gamma_2)$，其中对$\boldsymbol{w}_0$的攻击恢复可能导致$\boldsymbol{s}_1$、$\boldsymbol{s}_2$的泄露，因此多项式向量$\boldsymbol{w}$必须受到保护，*Decompose*函数也需要以掩码的形式安全计算。Berzati等人首次在实际攻击场景中利用了$\boldsymbol{w}_0$的泄露信息[27]。在签名过程中，当$\boldsymbol{w}_0$的一个坐标为零时，攻击者可以恢复$\boldsymbol{s}_1$相应坐标的值。该方法虽然效率不高，攻击所需要的签名数量多，但证明了在实际中利用$\boldsymbol{w}_0$这种泄露信息的可行性。Azouaoui等人也指出，前文提到恢复向量$\boldsymbol{s}_1$中的分析思路也适用于$\boldsymbol{w}_0$，可能导致$\boldsymbol{s}_2$的恢复[26]。

DV3：拒绝采样，见算法8第13行，对$\boldsymbol{z}$和$\boldsymbol{r}_0$最大范数的检查。Dilithium签名算法通过拒绝采样循环构建，不断生成新的签名，直到其满足某些安全属性。为了确保签名不会泄露有关密钥的信息，执行一些边界检查，如果此验证失败，则生成新的签名。侧信道攻击可以通过分析拒绝采样期间的泄漏信息来获取敏感数据$\boldsymbol{z}$和$\boldsymbol{r}_0$的相关信息，进而获取密钥。Zhou等人[28]针对拒绝采样环节，将被拒绝的$\boldsymbol{z}$及其对应的c所蕴含的信息，转化为关于私钥$c\boldsymbol{s}_1$的上界或下界不等式约束，从而恢复出私钥$\boldsymbol{s}_1$。Liu等人[29]更是首次同时利用了签名过程中产生的被拒绝签名与有效签名的泄露信息，构建因子图，并应用置信传播算法恢复私钥。为了防止密钥泄露，掩码变量$\boldsymbol{z}$和$\boldsymbol{r}_0$只有在成功完成拒绝采样后才能被组合，这意味着需要一个安全的边界检查算法[26]。成功的边界检查后，它们不会泄露其他敏感值的信息，可以公开。

4 整体防护策略

4.1 Kyber防护目标

为了更清晰地阐述Kyber的防护目标，下面将用算法流程图展示Kyber需要防护的模块。Kyber共分成三部分，第一部分为密钥生成，第二部分为封装，第三部分为解封装。Kyber解封装过程在解密后引入FO变换使其达到抗CCA安全性，由于FO变换较为复杂，因此我们将单独分析其防护目标。下图中，红色的部分为非线性操作，需要设计掩码防护方案；黑色的部分为线性操作，不需要设计掩码防护方案，只需要对每个掩码分别独立计算即可。此外，图中的“+”表示该部分为算术掩码，“⊕”表示该部分为布尔掩码，当涉及掩码类型的转换时就需要A2B（Arithmetic to Boolean）及B2A（Boolean to Arithmetic）算法。

Kyber密钥生成：密钥生成过程中涉及需要设计防护方案的算子如图1所示。其中，G表示哈希

函数，PRF表示伪随机数发生器（通常与哈希函数采用相同的算法），CBD表示采样器。需要注意的是，采样器两端的掩码类型不同，因此在采样器中需要进行B2A计算。除了上述算子之外，密钥生成中涉及的Parse、NTT和多项式乘法等操作均为线性操作，无需设计防护方案。

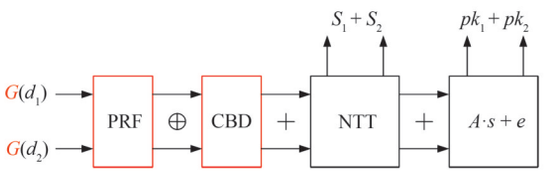

图1　Kyber密钥生成阶段掩码防护设计（红色部分为需要设计防护方案的算子）

Kyber封装：封装过程中涉及需要设计防护方案的算子如图2所示。封装过程的前半部分与密钥生成相似，同样由哈希函数G，伪随机数发生器PRF和采样器CBD组成，它们均为需要设计防护方案的算子。封装的后半部分中，$Decompress(m, 1)$表示编码器，$Compress(\boldsymbol{u}, d_u)$和$Compress(v, d_v)$表示密文压缩，这些部分同样是非线性的。此外，采样器和密文压缩两端的掩码类型不同，因此采样器中需进行B2A转换，密文压缩中需进行A2B转换。

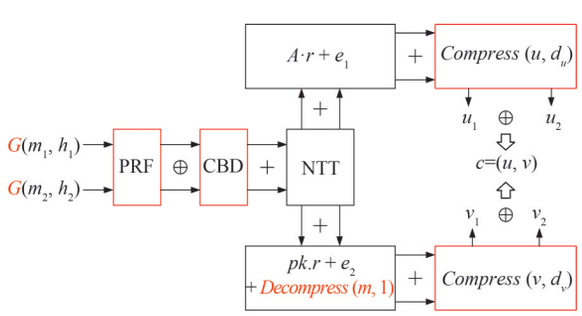

图2　Kyber封装阶段掩码防护设计（红色部分为需要设计防护方案的算子）

Kyber解密（属于解封装）：Kyber的解密过程如图3所示，其组成较为简单，其中的大部分操作都为NTT和多项式乘法，均为线性操作，需要防护的部分只有解码器$Compress(x, 1)$。同样需要注意的是，$Compress(x, 1)$两端的掩码类型发生了变化，需要引入A2B算法。此外，虽然解密过程中同样用到了*Decompress*函数，但这里的操作是仅针对密文进行的。由于密文是公开的，因此解密过程中的*Decompress*即使不进行掩码防护也不会泄露敏感信息。

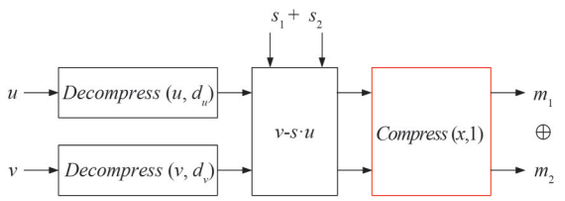

图3　Kyber解密阶段掩码防护设计（红色部分为需要设计防护方案的算子）

Kyber的FO变换（属于解封装）：Kyber的重新加密与比较算法共同组成了FO变换，再结合上先前的解密算法，即可得到完整的解封装过程，如图4所示。Kyber重新加密与封装采用相同的算法，对解密的结果再次执行加密，为比较算法提供输入。因此，重新加密部分将与封装部分采用相同的防护方案。比较算法通过比较封装的密文与重新加密的密文是否相同，进而实现了CCA安全性。图中的$Compare(c', c)$表示比较算法，该部分为非线性算法，同时其中可能会用到A2B、B2A以及密文压缩等其他算法，是较为复杂的算法。

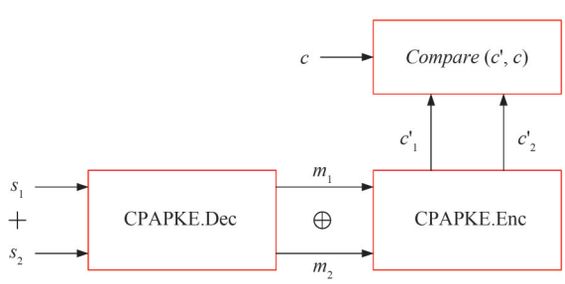

图4　Kyber解封装阶段掩码防护设计（红色部分为需要设计防护方案的算子）

特别需要注意的是，第3.1节中描述的脆弱点需要重点关注。针对3.1节中介绍的Kyber中易受侧信道攻击的四个脆弱点KV1到KV4，我们给出了相应的防护手段（Kyber Protection，KP）。

KP1：掩码NTT。对于NTT变换部分，由于该部分为线性运算，因此容易通过掩码进行防护。实现时仅需要将运算中的每个变量拆分为多个部分分别独立运算，即可抵御侧信道攻击。NTT的另一个实现难点在于如何降低其实现开销。Fritzmann等人实现了一种高效的NTT方案，同时给出了其硬件实现版本[30]。

KP2：掩码哈希。哈希函数的核心部分f变换分为线性部分和非线性部分，其中θ, ρ, π, ι都是线性部分，而χ是非线性部分。线性部分无需设计特殊的防护方案，仅需要将原始值拆分为多个掩码分别

计算即可。非线性部分χ不能直接拆分为多个掩码分别计算，必须定制掩码防护方案。Arribas等人在对哈希算法的一阶掩码防护中使用了DOM方案[31]，这种方案相比于传统的TI掩码方案使用了更少的掩码数量，实现了更高的性能。

KP3：掩码解码器。在对解码器实现掩码防护时，需要注意掩码取整时产生的误差可能导致解码错误。Oder等人[32]提出的掩码解码器方案首次解决了此类误差，随后Bos等人[33]实现了一种效率更高的方案。

KP4：掩码比较。为了判断是否发生了解密失败，攻击者需要观察比较的结果中存在多少不匹配的系数。因此，对比较算法的防护关键点在于，需要保证比较的结果是成功或是失败不能被攻击者观察到。传统的比较算法会逐系数进行比较，因此攻击者可以轻松地看出有多少系数不匹配，从而判断是否发生解密失败。D′Anvers等人[34]首次提出了安全的比较方案，仅需执行一次对所有系数的比较操作，让攻击者无法确定比较失败的系数个数。

4.2 Dilithium防护目标

为了抵御侧信道攻击，掩码是一种通用且可证明的对策。最常见的有布尔掩码和算术掩码，在Dilithium中随机采样和拒绝采样是复杂的操作，更适合布尔掩码，而涉及秘钥的签名算法主要部分是模某个素数p的线性运算，适合算术掩码，因此保护整个算法也需要在算术和布尔值掩码之间进行转换，Dilithium的A2B、B2A模块与Kyber部分重合，都是实现模q的掩码转换。同时我们主要关注密钥生成和签名阶段，因为Verify不操纵敏感数据，也不需要适当的掩码。密钥生成阶段涉及一些随机数的采样，并将其与密钥结合，需要对采样进行防护，签名阶段涉及Dilithium特有的操作：边界检查（拒绝采样）和*Decompose*分解函数，都需要特殊防护。由上，可以据此对Dilithium的防护实现设计特别的模块，分为采样、边界检查（拒绝采样）和*Decompose*三个模块分别阐述。

图5是Dilithium密钥生成阶段的图形表示。其中输出为$pk=(\rho,\boldsymbol{t}_1)$，$sk=(\rho,K,tr,\boldsymbol{s}_1,\boldsymbol{s}_2,\boldsymbol{t}_0)$。红色表示需要加防护的变量或操作，黑色表示不需要防护。密钥生成阶段基本上可以分为三个阶段：$\boldsymbol{A}$、$\boldsymbol{s}_1$和$\boldsymbol{s}_2$的采样；计算$\boldsymbol{t}:=\boldsymbol{A}\boldsymbol{s}_1+\boldsymbol{s}_2$；以及使用*PowerToRound*函数计算$t$的高位比特。$\boldsymbol{s}_1$和$\boldsymbol{s}_2$是密钥的一部分，变量$\boldsymbol{t}:=\boldsymbol{A}\boldsymbol{s}_1+\boldsymbol{s}_2$是公钥的一部分。因此在密钥生成阶段，$\boldsymbol{s}_1$和$\boldsymbol{s}_2$的采样、计算$\boldsymbol{t}:=\boldsymbol{A}\boldsymbol{s}_1+\boldsymbol{s}_2$时对$\boldsymbol{s}_1$和$\boldsymbol{s}_2$的使用要加以防护。

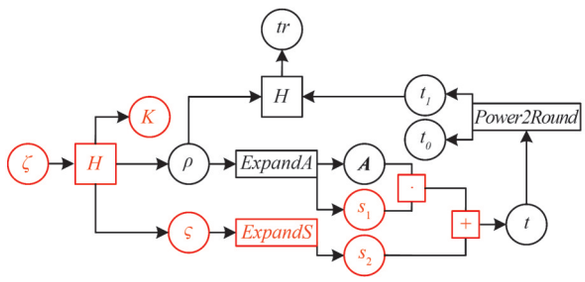

图5 Dilithium密钥生成阶段掩码防护设计（红色部分为需要设计防护方案的算子[26]）

图6为Dilithium签名生成过程。其中，输入为sk和M，输出为$\sigma=(\boldsymbol{z},\boldsymbol{h},\tilde{c})$。曲线箭头表示边界检查。签名阶段对变量的泄漏分析如下：

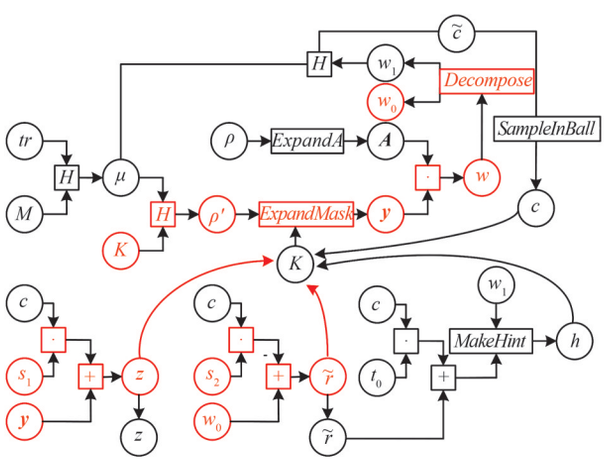

图6 Dilithium签名生成阶段掩码防护设计（红色部分为需要设计防护方案的算子[26]）

需要防护的变量：$\boldsymbol{s}_1$、$\boldsymbol{s}_2$，密钥构成必须保密，以避免通过简单的密钥恢复攻击导致签名伪造；$\boldsymbol{y}$，若攻击者知道签名$\sigma=(\boldsymbol{z},\boldsymbol{h},\tilde{c})$，并有$\boldsymbol{y}$的部分或全部信息，可以通过$\boldsymbol{z}:=\boldsymbol{y}+c\boldsymbol{s}_1$恢复出$\boldsymbol{s}_1$；$\boldsymbol{w}_0$，在已知$\boldsymbol{A}$和$\boldsymbol{w}$的情况下，求解$\boldsymbol{A}\boldsymbol{y}=\boldsymbol{w}$即可恢复出$\boldsymbol{y}$，因此$\boldsymbol{w}$必须加以防护，分解函数也要安全实现；$\rho'$，作为种子生成$\boldsymbol{y}$，也需要防护；在确定性签名的情况下，$K$也需要受到保护。

可公开变量：tr和$\boldsymbol{t}_0$、消息M、种子ρ、哈希μ和矩阵$\boldsymbol{A}$，这些都是公开变量；$\boldsymbol{w}_1$，当边界检查通过时，$\boldsymbol{w}_1$不会泄露敏感信息，当边界检查未通过时，推测$\boldsymbol{w}_1$不需要侧信道对策，但目前没有相关结论；挑战c，可以公开，是由$\boldsymbol{w}_1$和公开输入的一次性哈希生成的；边界检查后的$\boldsymbol{z}$和$\tilde{r}$，边界检查前的$\boldsymbol{z}$和$\tilde{r}$必须加以防护，这意味着需要一个安全实现的边界检查算法。成功通过边界检查后，不再会有泄露。

同样，针对3.2节中介绍的Dilithium中易受侧信道攻击的三个脆弱点DV1到DV3，我们也给出了相应的防护手段（Dilithium Protection，DP）。

DP1：掩码生成$\boldsymbol{s}_1$、$\boldsymbol{s}_2$以及$\boldsymbol{y}$。掩码向量$\boldsymbol{s}_1$、$\boldsymbol{s}_2$和$\boldsymbol{y}$的每个系数都在某个特定区间内均匀分布，根据Barthe等人[35]的研究，可通过首先生成一个随机的μ位布尔掩码系数，然后执行模q的布尔到算术转换来对此操作进行掩码处理[36]。

DP2：掩码实现分解函数。$(\boldsymbol{w}_1,\boldsymbol{w}_0)\leftarrow Decompose(\boldsymbol{w})$中只有$\boldsymbol{w}_0$需要掩码。Azouaoui等人[26]在论文中描述了一种高效的*Decompose*掩码方法，之后Coron等人[37]继续改进了掩码分解方法。

DP3：掩码实现边界检查。Barthe等人的方法是使用整数比较，利用算术到布尔掩码的转换进行符号位提取[35]。Azouaoui采用了相同的方法，并改进了算法，操作数更少。Coron等人又提出基于多项式系数算术移位和快速零检验的新方法，检查相对较小的被拒绝区间[36]。

5　算子防护方案

5.1　通用算子

在Kyber和Dilithium中，存在部分可以共用的防护算子，这些算子可以采用相同的防护方案，对模块复用起到了积极作用。在本节中，我们整理了Kyber和Dilithium中可以共用的算子，包含A2B算法（Arithmetic to Boolean）、B2A算法（Boolean to Arithmetic）和哈希函数。对于Kyber和Dilithium特有的算子，我们将分别在第5.2节和第5.3节中详细介绍。

5.1.1　A2B

（1）防护目标

在后量子密码算法中，不同算子可能需要用到不同的数据类型，有的部分需要算术值，有的部分需要布尔值。在掩码防护方案中，算术操作部分必须相应地使用与算术操作兼容的掩码方案，也就是算术掩码。例如，如果要安全地计算$x_3=x_1+x_2$，可以首先将x_1和x_2分别拆分为两个算术掩码，即将x_1和x_2拆分为$x_1=A_1+r_1$和$x_2=A_2+r_2$，其中r_1和r_2是随机数。然后，在计算求和结果时可以分别添加这两个掩码部分，这将再次产生两个新的算术掩码A_3和r_3，使得$x_3=(A_1+A_2)+(r_1+r_2)=A_3+r_3$。算术掩码可以保证算术计算时的安全性，但当算法的下一阶段需要布尔操作时，就需要将算术掩码转换为布尔掩码，A2B指的是将算术掩码转换为布尔掩码的算法。为保证A2B算法安全性，在转换过程中不可以暴露原始秘密信息，即不可以将掩码组合为原始信息后再转换，而是要为每个掩码单独转换。由于该转换过程是非线性的，因此需要设计防护方案。

（2）典型防护方案

最早的A2B算法已经可以追溯到2001年Goubin提出的算法，实现了一种高效的A2B转换，但其局限在于只能用于一阶掩码的转换[38]。Coron等人首次提出可以应用于任意阶掩码的A2B方案[39]，该方案采用递归的思想，将具有n阶掩码的算术值递归地进行拆分，直到最后每一部分只剩下一个掩码值，即将n阶的算术掩码拆分为$n+1$个算术值。此时，由于每部分均可视为一个无掩码的算术值，因此其等价于无掩码的布尔值。随后，再将这$n+1$个布尔值递归地进行布尔安全加法SecAdd后即可得到n阶的布尔掩码。

Coron提出的布尔加法SecAdd需要引入额外的进位值。具体来说，两个k位变量x和y的模加法可以通过添加一个进位值c递归地定义为$(x+y)^{(i)}=x^{(i)}\oplus y^{(i)}\oplus c^{(i)}$，其中：

$$\begin{cases} c^{(0)}=0 \\ \forall i\geqslant 1,(x^{(i-1)}\wedge y^{(i-1)})\oplus(x^{(i-1)}\wedge c^{(i-1)}) \\ \oplus(c^{(i-1)}\wedge y^{(i-1)}) \end{cases} \tag{1}$$

虽然这种递归思想在后来的其他A2B方案中也被广泛沿用，但是该SecAdd算法的开销相对较大，因此后续其他工作改进的核心是提高算法执行的效率。

Bronchain等人尝试将位切片（Bitslicing）技术应用于A2B上，提高其执行效率[40]。位切片作为一种适用于布尔电路的优化技术，通过将数据拆分成多个比特位的切片，可以同时对这些切片进行并行处理，从而大幅提高计算速度。由于SecAdd算法是布尔加法，可以使用位切片技术进行加速，因此获得了更高的执行效率。

由于布尔安全加法中存在进位，因此以往的A2B算法中需要多次计算进位值，这会造成很大的开销。Liu等人针对进位值计算的问题提出了解决方案，他们引入进位保存加法器（Carry-Save Adder，CSA）用于布尔加法[41]。CSA同时计算两个输入数的进位值以及和，而不是像其他加法器那样先计算进位再计算求和。这使得进位值变成每次CSA的输出，而不需要额外的计算。进位值与求得

的和可以作为下一级CSA的输入，形成级联结构，避免了额外的计算开销，如图7所示。通过消除进位值计算带来的额外开销，他们的算法在执行效率上获得了巨大的提升。

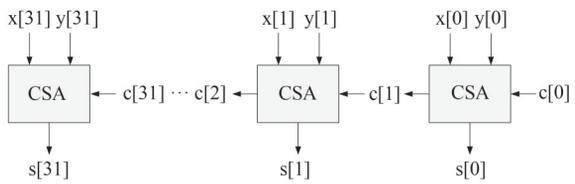

图7 CSA级联结构

（3）防护方案对比分析

对于软件实现版本，Liu等人在相同的平台和编译选项下比较了基于CSA的方法与Bronchain基于位切片方法。对于32位的A2B转换，所有的测试在Intel(R)Core(TM)i5-1135G7@2.40 GHz CPU上进行，测试其时钟周期数如表1所示。Liu等人的A2B转换的时钟周期消耗比Bronchain等人的方案少约30%。

表1 A2B软件实现开销对比[41]

相关工作	编译选项	安全阶数		
		2	3	4
Bronchain et al. 2022[40]	-O0	13,124	23,049	42,212
Liu et al. 2024[41]	-O0	2,725	4,950	8,593

Liu等人也在ASIC标准单元技术和FPGA下对A2B硬件实现进行了性能评估。对于ASIC的实现，他们在TSMC 28 nm(ssg0p81v125c)工艺节点下进行了评估，使用了SS角和0.9V电源，结果如表2所示。Liu等人的方法在时钟周期延迟(CLK)上大幅领先于Bronchain等人的方法，同时没有引入额外的面积消耗。

对于FPGA的实现，Liu等人在Artix-7 FPGA(XC7A200TFFG1156-3)上评估了不同设计的资源利用率和性能。表3比较了延迟、最大频率、时间延迟、资源利用率和随机性方面的设计。设计期间确定的时钟周期延迟(CLK)和随机数消耗(bits)与ASIC性能一致。

（4）实现建议

从实现的角度考虑，Coron的方法不适合直接使用，其相比于其他方案执行效率不高。Bronchain的改进主要在于引入了位切片技术，该技术可以作为一种优化的手段，移植在各种算法中。值得一提的是，如果算法中涉及的位级操作较少，或者无法充分利用并行性，位切片的优势会降低。Liu提出的方案则是当前在时延方面最优的解决方案，且不仅仅提供了软件实现，还进一步证明了该方法在硬件上的有效性，是当前更推荐的一种方案。

5.1.2 B2A

（1）防护目标

A2B算法将算术掩码转换为布尔掩码，与之对应的，B2A算法将布尔掩码转换为算术掩码。与A2B相同，B2A同样是非线性组件，需要设计特殊的防护方案，从而将每个掩码单独转换。此外，在Kyber和Dilithium的密钥生成阶段和加密阶段都会用到采样器，采样器会直接泄露秘密信息，而构成采样器的主要算法就是B2A，因此要对其进行特别防护。B2A的基本转换公式为($b_1 \oplus b_2 = a_1 +$

表2 A2B在TSMC 28 nm实现的开销对比[41]

安全阶数	相关工作	时延/CLK	频率/GHz	延迟/ns	面积/kGE	随机数/bits
2	Bronchain et al. 2022[40]	124	5.02	24.70	197.20	124
	Liu et al. 2024[41]	14	4.87	2.87	157.42	1,056
3	Bronchain et al. 2022[40]	124	4.44	27.93	294.85	248
	Liu et al. 2024[41]	16	4.31	3.25	287.34	2,208
4	Bronchain et al. 2022[40]	186	3.95	47.09	496.76	465
	Liu et al. 2024[41]	18	3.84	4.96	456.32	3,808

表3 A2B在Artix-7 FPGA实现的开销对比[41]

安全阶数	相关工作	频率/MHz	延迟/ns	LUTs	FFs	Slices
2	Bronchain et al. 2022[40]	512.82	241.80	2,234	20,423	4,352
	Liu et al. 2024[41]	370.37	37.80	11,196	14,550	3,715
3	Bronchain et al. 2022[40]	444.44	279.00	3,818	30,555	6,753
	Liu et al. 2024[41]	300.30	53.28	22,189	26,093	7,275
4	Bronchain et al. 2022[40]	370.37	502.20	6,642	51,567	10,852
	Liu et al. 2024[41]	219.78	81.90	36,885	41,331	10,879

$a_2 \bmod p$)。

(2) 典型防护方案

Coron等人引入了快速高阶模 q 的布尔到算术掩码转换，其复杂度与 μ 和 q 的大小无关[37]。该技术基于模数切换：首先使用Bettale等人提出的方法执行快速的模 2^k 的B2A[42]，然后进行模数切换，最后，为了消除模数切换引入的误差，执行一系列算术移位。Bettale的转换方法优点是算法复杂度与 k 无关，尽管渐近复杂度为 $O(2^n)$，但对于较小的 n 来说优势很大。然而，上述转换算法需要相对较大的寄存器大小 k，对于安全级别为2的Dilithium中，相应 $k \geqslant 41$，在Bettale的转换算法中，必须使用64位寄存器。

Coron等人又在2024年对此做出改进[36]，令 $k=\mu+\lceil\log_2 n\rceil$，而不是 $k=\mu+\lceil\log_2 q\rceil+\lceil\log_2 n\rceil$，其对应的算术掩码为 $z=x_1+\cdots+x_n=x+\delta\cdot 2^k \pmod{2^k q}$。通过求 δ 模 q 的算术掩码，得出 x 模 q 的算术掩码，如算法10所示。

算法10. $\mathrm{BtoA}_q\mathrm{Belta}$

输入：模数 q，μ 位布尔掩码 $u_1,\cdots,u_n, u_1\oplus\cdots\oplus u_n=x$

输出：算术掩码 $v_1,\cdots,v_n, v_1+\cdots+v_n=x \pmod q$

1. $\alpha \leftarrow \lceil\log_2 n\rceil$
2. $k \leftarrow \mu+\alpha$
3. $x_1,\cdots,x_n \leftarrow \mathrm{BtoAExp}_{\mu,2^k}(u_1,\cdots,u_n)$
4. For $i=1$ to n do $y_i \leftarrow \lfloor x_i/2^\alpha \rfloor$
5. $y_1 \leftarrow y_1+n-1$
6. $(\delta_1,\cdots,\delta_n) \leftarrow \mathrm{ShiftMod}_\alpha(2^\alpha\cdot q,(y_1,\cdots,y_n))$
7. $(\delta_1,\cdots,\delta_n) \leftarrow \mathrm{RefreshMasks}(\delta_1,\cdots,\delta_n)$
8. For $i=1$ to n do $v_i \leftarrow x_i-2^k\cdot\delta_i \bmod q$
9. Return $(v_1,\cdots,v_n)$

(3) 防护方案对比分析

表4中提供了各种方法之间的操作计数的比较，假设使用32位处理器，因此Coron 23年的方法[37]复杂度从 $T_{\mathrm{BtoAExp}}(n)=10\cdot 2^n-6n-13$ 变为 $2\cdot T_{\mathrm{BtoAExp}}(n)$，这反映了需要适配64位整数的必要性。可以看出，Coron 24年的方法[36]在所有安全阶数上比23年的方法都要好，同时在低阶防护下比Barthe和Schneider的方法效果都要好。

(4) 实现建议

在实际掩码防护实现时，考虑到可行性，通常为低阶掩码，因此Coron在2024年提出的适用于32位处理器的转换方案[36]优势明显。

5.1.3　掩码哈希

(1) 防护目标

哈希算法的核心为Keccak算法，使用了如图8所示的海绵结构，输入为字符串 N，输出为字符串 Z。海绵结构中存储的数据被称为状态，总长度为 $b=r+c$，其中 r 为比特率，c 为容量，状态数据被分为外部状态(前 r 位)和内部状态(后 c 位)。每次海绵结构执行时，状态经过一次 f 变换得到下一个状态。Keccak算法通常可以分为两个阶段：吸收(Absorbing)阶段和挤压(Squeezing)阶段。

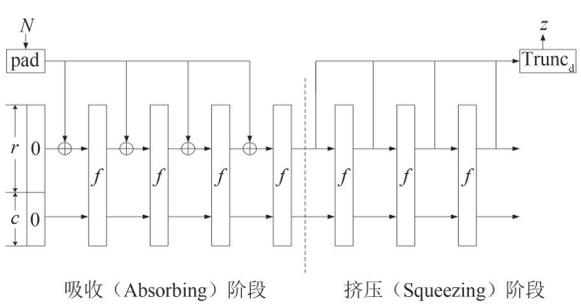

图8　Keccak算法的海绵结构

吸收阶段：假设长度为 l 的输入 N 经过填充得到的字符串长度为 n，n 是 r 的整数倍，将填充后字符串以 r 位为一组分为 n/r 组。当吸收阶段开始时，初始状态的值为全0，此时将第一组字符串与初始状态的前 r 位进行异或，得到的值与初始状态的后 c 位组合并执行一次 f 变换，得到的状态前 r 位再

表4　对于素数 $q=2^{23}-2^{13}+1$，18位模 q B2A算法的操作次数，安全阶数 t 取2到12，对应 $n=t+1$ 个份额[36]

B→A mod q	安全阶数 t							
	2	3	4	5	6	8	10	12
Barthe et al. 2018[35] 18→mod q	2,841	5,215	8,782	12,897	17,825	30,235	46,012	64,776
Schneider et al. 2019[43] 18→mod q	804	1,414	2,186	3,120	4,216	6,894	10,220	14,194
Coron et al. 2023[37] 18→mod q	194	396	857	1,587	2,969	11,132	42,240	165,572
Coron et al. 2024[36]	146	280	596	1,024	1,787	6,161	21,972	83,939

和第二组字符串相异或，反复执行直到n/r次输入异或和f变换全部执行完成。

挤压阶段：假设函数需要k次输出，则海绵结构先输出一次外部状态，再将整个状态执行一次f变换，得到结果后再输出一次外部状态，这个得到的结果同时也是f变换下一次的输入，如此直到n次输出完成。每次外部状态需要经过一次Trunc_d函数，因此海绵结构的总输出长度为$k\cdot d$位。

哈希模块的掩码防护主要在于f变换。f变换的基础为轮函数f_i，每轮f_i的输出作为下一轮f_{i+1}的输入。所有f_i的结构完全相同，其无防护实现基于如下公式：

$$\begin{aligned}
&A[x,y] && = In[x,y],\\
\theta:\ &B[x] && = A[x,0]\oplus A[x,1]\oplus\cdots\oplus A[x,4],\\
&C[x] && = B[x-1]\oplus\big(B[x+1]\ll 1\big),\\
&D[x,y] && = A[x,y]\oplus C[x],\\
\rho:\ &E[x,y] && = D[x,y]\ll S[x,y],\\
\pi:\ &F[y,2x+3y] && = E[x,y],\\
\chi:\ &G[x,y] && = F[x,y]\oplus\big(\overline{F}[x+1,y]\\
& && \quad F[x+2,y]\big),\\
\iota:\ &G[0,0] && = G[0,0]\oplus RC(i),\\
&Out[x,y] && = G[x,y]
\end{aligned}\tag{2}$$

其中，In表示f_i的输入状态，Out表示f_i的输出状态。$S[x,y]$是与x，y有关的常数数组，$RC(i)$是与执行轮次i有关的常数数组。

（2）典型防护方案

DOM方案于2016年由Hannes等人提出，这种方案相比于传统的TI（Threshold implementation，门限实现）掩码方案使用了更少的掩码数量。为了实现一阶掩码防护，TI需要使用3个掩码，而DOM只需要使用2个掩码。2018年，Arribas等人在对Keccak算法的一阶掩码防护中使用了该方案[31]。

Arribas将轮函数f_i分为线性部分和非线性部分，其中θ,ρ,π,ι都是线性部分，而χ是非线性部分。线性部分无需设计特殊的防护方案，仅需要将原始值拆分为多个掩码分别计算即可。非线性部分χ不能直接拆分为多个掩码分别计算，必须定制掩码防护方案。因此，Arribas对χ部分设计的算法如下所示：

$$\begin{aligned}
R_{00}[x,y] &= F_0[x,y]\oplus\big(\overline{F_0}[x+1,y]F_0[x+2,y]\big),\\
R_{01}[x,y] &= \big(F_0[x+1,y]F_1[x+2,y]\big)\oplus r[x,y],\\
R_{10}[x,y] &= \big(F_1[x+1,y]F_0[x+2,y]\big)\oplus r[x,y],\\
R_{11}[x,y] &= F_1[x,y]\oplus\big(\overline{F_1}[x+1,y]F_1[x+2,y]\big),\\
G_0 &= R_{00}\oplus R_{01},\\
G_1 &= R_{10}\oplus R_{11}
\end{aligned}\tag{3}$$

（3）防护方案对比分析

Arribas对DOM防护方案和TI防护方案进行了对比，使用了Synopsys Design Compiler版本I-2013.12和NanGate 45 nm开放单元库，在ASIC平台上来进行硬件资源的对比测试，表5中显示了测试的结果。请注意，表中呈现的随机位数是针对每一轮的，即每两个时钟周期。

表5 DOM方案与TI方案的开销对比[31]

方案	面积				随机数	时钟周期	频率
	χ	θ	状态	总计			
无防护	542.63	638.4	1,333	2,759	-	18	1,136
DOM一阶	9,881	1,600	2,667	17,105	200	36	1,087
TI一阶	2,613	1,600	5,334	17,493	200	36	1,300
DOM二阶	23,177	2,400	4,000	33,535	600	36	1,111
TI二阶	6,200	2,400	12,001	35,223	600	36	1,205

（4）实现建议

Arribas提出的是非常经典的使用DOM进行防护的方案，这种方案兼顾了线性部分和非线性部分，在使用了较少掩码数量的情况下也能保持较少的资源占用。TI以及其他方案或许使用的资源更少，但是这些方案需要使用更多的掩码数量。在格密码掩码实现只使用了两个掩码的情况下，DOM方案无疑是最佳的选择。

5.2 Kyber算子防护方案

5.2.1 Kyber采样器

（1）防护目标

Kyber在密钥生成阶段和加密阶段中都会用到采样器，采样器负责生成敏感变量$\mathbf{r}$、$\mathbf{e}_1$和e_2。由于采样器在生成敏感信息时会产生明显的侧信道泄露，很容易被攻击者利用，因此需要对采样器进行掩码防护。

（2）典型防护方案

Oder等人于2016年设计了一种掩码二项采样器，来保护FO变换的重加密过程[32]，但其方案只能抵抗一阶攻击，并且不易被扩展到高阶。

2019年，Schneider等人提出了第一个任意阶掩

码采样器[43]，并且适用于任意模数，成为了格密码掩码设计中的主流方案。因此，我们主要介绍Schneider提出的采样算法。

在Schneider等人的方法中，假设给定两个变量(x, y)，这两个变量的长度为κ比特，并且被布尔编码为(x, y)。其随机性依赖于PRNG(为布尔掩码)来产生均匀的伪随机值。采样器需要在这些编码上以安全的方式计算$HW(x)-HW(y)$，并基于给定的模数q产生$\sum_i A_i=HW(x)-HW(y)$的算术掩码A，以适应后续的运算。此转换中用到的B2A可以使用任何$B2A_q$方案来完成。可根据以下公式进行算法描述：

$$\begin{aligned}\sum_i A_i &= \mathrm{HW}(x)-\mathrm{HW}(y)=B2A_q(SecConstAdd\\ &(SecBitSub(HW(x),y),\kappa))-\kappa=B2A_q\\ &\Big(SecConstAdd\big(HW(x)-HW(y),\kappa\big)\Big)-\\ &\kappa=B2A_q\big(HW(x)-HW(y)+\kappa\big)-\kappa=\\ &HW(x)-HW(y)\bmod q\end{aligned}\tag{4}$$

算法11.　SecSampler

输入：$\boldsymbol{x}=(x_i)_{1\leqslant i\leqslant n}\in\mathbb{F}_{2^r}$，$\mathbf{y}=(y_i)_{1\leqslant i\leqslant n}\in\mathbb{F}_{2^r}$，$\kappa$，其中$\bigoplus_i x_i=x$，$\bigoplus_i y_i=y$

输出：$\boldsymbol{A}=(A_i)_{1\leqslant i\leqslant n}\in\mathbb{F}_q$，其中$\sum_i A_i=\mathrm{Hw}(x)-\mathrm{Hw}(y)\bmod q$

1. $\mathbf{z}\leftarrow\mathrm{SecBitAdd}(\mathbf{x})$
2. $\mathbf{z}\leftarrow\mathrm{SecBitSub}(\mathbf{z},\mathbf{y})$
3. $\mathbf{z}\leftarrow\mathrm{SecConstAdd}(\mathbf{z},\kappa)$
4. $\boldsymbol{A}\leftarrow\mathrm{B2A}_q(\mathbf{z})$
5. $A_1\leftarrow A_1-\kappa\bmod q$

(3) 防护方案对比分析

表6中分别对比了两个采样器，$\mathrm{SecSampler}_1$是Oder等人文章中的采样器方法，$\mathrm{SecSampler}_2$是Schneider等人文章中提出的采样器。其中特别需要注意的是，两个采样器使用的B2A算法不同，$\mathrm{SecSampler}_2$中使用的B2A算法是Schneider等人提出的B2A算法，拥有更高的执行效率。由表6可知，采用Schneider新提出的采样器结合新提出的B2A算法效果最好。

(4) 实现建议

通过搜集近几年的采样器防护方案，发现Schneider提出的采样器方案还在被广泛使用，此外Fritzmann等人在文章中[30]，提出了对Schneider的采样器进行加速(DOM/TI防护方案)，其中DOM效率更高，但防护能力没有TI好，因此具体采取哪种方案需要看防护方案的优先级。

5.2.2　掩码压缩

(1) 防护目标

在Kyber规范中，会将加密后得到的密文压缩到较短的长度以减小存储开销。对于Kyber-512和Kyber-768，会将最终的密文$\boldsymbol{u}$压缩到10位，将密文v压缩到4位；对于Kyber-1024，则会将密文$\boldsymbol{u}$压缩到11位，将密文v压缩到5位。算法2展示了Kyber的封装过程，压缩算法应用于封装的第17行和第18行。掩码压缩的核心难点在于如何保证掩码情况下的精度问题。

(2) 典型防护方案

Fritzmann等人提出了一种掩码压缩技术[30]，从掩码压缩时产生的误差项入手，通过增加除法运算时的小数位来增加压缩的精度，保证了掩码压缩时将密文压缩到多位长度情况下的正确性。Fritzmann观察到，压缩时必然会产生误差，这是因为压缩的本质是将对密文影响不大的低位舍弃。具体来说，压缩公式为：

$$Compress_q(x, d)=\lfloor(2^d/q)\cdot x+e\rfloor\bmod 2^d\tag{5}$$

因此，可以仅使用有限精度算法，从$x=x^0+x^1\bmod q$的掩码中计算出一个近似商。例如，使用整数除法，可以计算出

$$\begin{aligned}&\lfloor(2^d/q)\cdot x^0\rfloor_f+\lfloor(2^d/q)\cdot x^1\rfloor_f=\\ &((2^d/q)\cdot x+e)\bmod 2^d\end{aligned}\tag{6}$$

在上述情况下，通过增加小数位数f可以减小舍入误差e。因此，现在就需要选择适合的f使不等式$-1/(2q)\leqslant e<1/(2q)$成立。当上述不等式成

表6　使用SecB2A和SecBoolArithModp对二项分布系数进行掩码采样的循环计数[43]

	B2A方案	安全阶数			
		1	2	3	4
$\mathrm{SecSampler}_1$	SecBoolArithModp	271,423	638,315	1,076,155	1,758,184
	SecB2A	6,145	13,913	24,397	37,880
$\mathrm{SecSampler}_2$	SecBoolArithModp	17,564	40,977	68,914	112,402
	SecB2A	2,649	5,573	9,462	14,587

立时，即可用于正确获取掩码压缩的输出。

算法仅使用了整数运算和向下取整除法，对于一阶掩码来说，需要保证小数位数 $f \geqslant 13$ 即可。但为了实现更高阶的安全性，必须选择更大的 f 以容忍来自更多的掩码的舍入误差。算法12展示了Fritzmann等人提出的掩码压缩算法。

算法12. 一阶MaskedCompress算法[30]

输入：$\mathbf{x}^{\{0:1\}}=(x^0, x^1)$ such that $x=x^0+x^1$, d, $f>13$

输出：$\mathbf{z}^{\{0:1\}}=(z^0, z^1)$ such that $z=z^0 \oplus z^1 \in \{0,1\}^d$

1. $x'^0 \leftarrow \lfloor (2^{d+f} \cdot x^0)/q \rfloor \bmod 2^{d+f}$
2. $x'^1 \leftarrow (\lfloor (2^{d+f} \cdot x^1)/q \rfloor + 2^f \cdot 0.5) \bmod 2^{d+f}$
3. $\mathbf{z}^{\{0:1\}} \leftarrow \mathrm{A2B}(\mathbf{x}'^{\{0:1\}})$
4. $\mathbf{z}^{\{0:1\}} \leftarrow \mathbf{z}^{\{0:1\}} \gg f$

虽然Fritzmann提到他们的方法可以简单地通过扩大 f 的值来扩展到高阶，但没有给出各阶掩码下对应 f 的具体位数。为了将Fritzmann的方法扩展到高阶掩码中，Coron等人[44]给出了各阶掩码下的 f 值。

Coron证明，在 $2^f > q \cdot n$ 的条件下，有 $Compress_{q,d}(x)=y_1 \oplus y_2 \oplus \cdots \oplus y_n$。这个条件决定了精度位数的 f 对于掩码数 n 的函数。完整的算法流程如算法13所示。

算法13. MaskedCompress算法[44]

输入：$x_1, x_2, \cdots, x_n \in \mathbb{Z}_q$

输出：$y_1, y_2, \cdots, y_n \in \{0,1\}^d$ 使得 $y_1 \oplus \cdots \oplus y_n = Compress_{q,d}(x_1+\cdots+x_n)$

1. $f \leftarrow \lceil \log_2(q \cdot n) \rceil$
2. $z_1 = \lfloor (x_1 \cdot 2^{d+f+1}+q)/(2q) \rfloor + 2^{f-1} \bmod 2^{d+f}$
3. For i from 2 to n do
4. $\quad z_i \leftarrow \lfloor (x_i \cdot 2^{d+f+1}+q)/(2q) \rfloor \bmod 2^{d+f}$
5. $(c_1, \cdots, c_n) \leftarrow \mathrm{A2B}(d+f, (z_1, \cdots, z_n))$
6. For i from 1 to n do
7. $\quad y_i \leftarrow c_i \gg f$
8. Return $y_1, y_2, \cdots, y_n$

(3) 防护方案对比分析

Coron等人测试了Kyber-768下掩码压缩的开销，所有均采用C语言实现，测试平台为Intel(R) Core(TM) i7-1065G7。对于Kyber-768，密文压缩阶段需要将两个密文分量 $\boldsymbol{u}$ 和 v 分别压缩到 $d_u=10$ 和 $d_v=4$。因此，作者在表7中分别对这两种情况进行了测试。

基于测量结果，可以看到对于 $d_u=10$ 的压缩开销要略大于对于 $d_v=4$ 的压缩开销，这主要是因为A2B的位数增多导致的。

表7 不同阶数下密文压缩的循环计数[44]

	安全阶数						
	1	2	3	4	5	6	7
$d_u=10$	131	868	1,579	3,181	4,898	6,764	8,809
$d_v=4$	101	658	1,195	2,431	3,740	5,162	6,721

(4) 实现建议

Fritzmann和Coron在方案理论上是一致的，他们都采用保留更多小数位的操作来保证算法的正确性，但Coron进一步对于高阶掩码压缩进行了阐述。该方案已是目前主流的密文压缩方案。

5.2.3 掩码编码器

(1) 防护目标

Kyber编码器指的是将1位的消息 m 编码为12位的过程，以便后续执行加密。算法2给出了Kyber封装的流程。其中，算法2第16行的 $\mathrm{Decompress}_q$ 即为编码器。编码时会涉及对消息 m 的操作，该操作虽然简单，但在掩码实现中会出现误差，如何避免该误差是设计防护方案的关键之处。

(2) 典型防护方案

在Kyber的一阶掩码实现中，密文分量 $\boldsymbol{u}$ 和 v 的计算是在掩码中进行的。对于 $\boldsymbol{u}$ 来说，它只需要进行简单的线性操作，因此可以分别在每个输入掩码上执行

$$\boldsymbol{u}' = \boldsymbol{A} \cdot \boldsymbol{r}' + \boldsymbol{e}'_1 \tag{7}$$

$$\boldsymbol{u}'' = \boldsymbol{A} \cdot \boldsymbol{r}'' + \boldsymbol{e}''_1 \tag{8}$$

其中，$\boldsymbol{u}'$ 和 $\boldsymbol{u}''$ 分别表示 $\boldsymbol{u}$ 的第一个掩码和第二个掩码，$\boldsymbol{A} \cdot \boldsymbol{r}'$ 和 $\boldsymbol{A} \cdot \boldsymbol{r}''$ 分别为它们对应的多项式乘法，$\boldsymbol{e}'_1$ 和 $\boldsymbol{e}''_1$ 为噪声项。

然而，对于 v 来说，其中涉及对消息 m 的编码操作，必须考虑从编码到获取正确结果的舍入误差，即仅通过以下公式去计算是不够的。

$$v' = \boldsymbol{t} \cdot \boldsymbol{r}' + e'_2 + Decompress(m') = \boldsymbol{t} \cdot \boldsymbol{r}' + e'_2 + m' \cdot \left\lfloor \frac{q}{2} \right\rfloor \tag{9}$$

$$v'' = \boldsymbol{t} \cdot \boldsymbol{r}'' + e''_2 + Decompress(m'') = \boldsymbol{t} \cdot \boldsymbol{r}'' + e''_2 + m'' \cdot \left\lfloor \frac{q}{2} \right\rfloor \tag{10}$$

式中，$m' \oplus m'' = m$。但由于Kyber的模数 q 是奇数，因此 $2 \cdot \lfloor q/2 \rfloor \neq q$，必须调整这个操作以计算出正确的结果，即使得重新加密的结果必须与原始加密的结果完全相同。

Oder等人解决了这个误差问题[32]。具体来说，可以仅在m'和m''这两个掩码值都为1的情况下返回False。在这种情况下，向下取整操作会使每个掩码都少了1/2，因此结果会偏差1。为了获得正确的结果，必须添加m'和m''相与的结果。

但是直接计算掩码相与的结果会导致泄露，因此需要将掩码进一步拆分为子掩码。

$$m'=m'_1+m'_2 \tag{11}$$

$$m''=m''_1+m''_2 \tag{12}$$

对于这个计算，m'和m''被视为多项式，而不是位向量。由于掩码拆分，因此需要计算$(m'_1+m'_2)\circ(m''_1+m''_2)$，而不是$(m'\circ m'')$。其中$\circ$表示逐点相乘运算，即将两个向量或矩阵上的对应位置元素相乘，得到新的向量或矩阵。为了获得正确的结果，最终v'的计算公式如下：

$$v'=\boldsymbol{t}\cdot\boldsymbol{r}'+e'_2+Decompress(m')+m'_1m''_1+m'_1m''_2+m'_2m''_1+m'_2 \tag{13}$$

而v''依然采用原先的计算公式即可。

(3) 防护方案对比分析

Oder提出的编码器方案是当前的主流方案，该方案很好地解决了掩码中的误差问题，在后续的实现中被广泛使用。Bos等人和Heinz等人均采用了Oder的编码器方案实现了Kyber-768，论文中报告了编码器在全部768个系数上的开销如表8所示。其中，Bos等人在FRDM-KL82Z(Cortex-M0+)和STM32F407G(Cortex-M4F)平台上进行了性能基准测试，Heinz在STM32F407VG(Cortex-M4)平台上进行了性能基准测试[45]。

表8　不同平台上编码器开销对比

	Bos et al. 2021[33]		Heinz et al. 2022[45]
	FRDM-KL82Z	STM32F407G	STM32F407VG
整体开销	644,000	113,000	85,866
单个系数开销	838	147	112

(4) 实现建议

总体而言，编码器的掩码实现难度不大，只需要对每个消息掩码值进行编码即可，需要注意的主要是误差问题。Oder真正解决了误差问题，保证了算法正确性，因此该方案在今天依然被沿用。

5.2.4　掩码解码器

(1) 防护目标

作为编码器的逆运算，Kyber解码器的作用是将12位的解密结果恢复为1位消息m的过程。算法3给出了Kyber解密的流程。其中，算法3第4行的$\mathrm{Compress}_q$即为解码器。解码过程同样是涉及对m的运算，因此需要确保其不会泄露。由于Kyber的密文压缩操作，因此得到的密文存在精度上的误差，解码的核心问题就是如何保证掩码实现下的正确性。

(2) 典型防护方案

Reparaz等人提出了一种相当复杂的解码器来处理算术掩码，且解码失败的概率较高[46-47]。为了提高算法效率，Oder等人提出了新的解码方法MDecode，该方法首先将模q的算术掩码转换为模2^k的布尔掩码，然后进行解码[32]。通过该方法，可以避免使用代价昂贵的算术掩码解码器，并减少Reparaz方案中出现的额外误差。

图9中展示了Oder等人提出的方法的解码流程。每个圆圈表示经过每个步骤后，解密结果对应不同m值的分布情况($m=0$为空白部分，$m=1$为阴影部分)。图中左上角的第一个圆圈显示了解密后结果的分布，其中$m=0$对应的解密结果聚集在0附近，$m=1$对应的解密结果聚集在$q/2$附近。

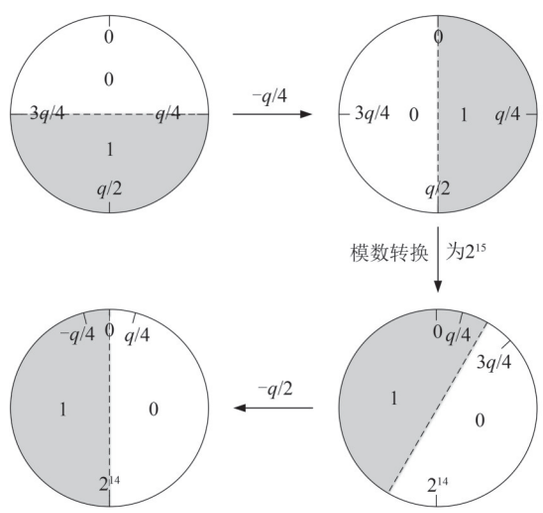

图9　Kyber解码流程

首先从(x_1, x_2)中减去$q/4$，这样就不会出现分布跨越模运算边界的情况。随后，调用Transform算法将原本模q的算术掩码转换为模2^{15}的布尔掩码。完成转换后，从结果中减去$q/2$，以创造新的掩码(y_1, y_2)构成以下关系：

$$MSB\left(y_1+y_2 \bmod 2^{bits}\right)=\begin{cases}0 & m=0\\ 1 & m=1\end{cases} \tag{14}$$

由于分布与零的距离相等，防止了解码错误概率的增加。在最后一步中，由于分别提取两个算术

掩码的最高有效位(MSB)会存在进位问题,因此执行A2B变换A2B(y_1, y_2)=(y'_1, y'_2),以提取布尔掩码m的最高有效位,其中

$$MSB(y'_1)\oplus MSB(y'_2)=m_1\oplus m_2=m \quad (15)$$

然而,此方法仅局限于一阶掩码,Bos等人采用了二分查找的方法优化了解码器方案[33],他们通过二分查找快速找到密文所在的区间,然后映射到对应的消息比特,提升了算法效率,并可以用于任意阶掩码。

Bos首先同样从将偏移量$q/4$添加到算术掩码中,并进行后续的A2B转换,以创建12位布尔掩码。给定这些布尔掩码,然后只需要安全地计算掩码值是否小于$q/2$。因此,将解码函数记为$Compress_q^s(x)$,目标就是使得$Compress_q(x, 1)=Compress_q^s(x+\lfloor q/4\rfloor \bmod q)$,其中

$$Compress_q^s(x)=\begin{cases}0, & \text{if } x<\dfrac{q}{2}\\ 1, & \text{otherwise}\end{cases} \quad (16)$$

为了以掩码方式计算$Compress_q^s(x)$,需要从最高有效位开始对系数的布尔掩码进行掩码二分搜索。例如,如果MSB设置为1,可以忽略所有后续位的值,并将系数压缩为1,因为$2^{MSB}=2^{k-1}=2^{11}>q/2$。如果MSB设置为0,则还需要考虑更多剩余的位。这个过程会重复进行,直到所有可能的系数值都映射到单个比特值为止。对于Kyber来说,$\lfloor q/2\rfloor=1664$,因此只需要考虑第11位到第7位的值。最终,$Compress_q^s(x)$的计算如下所示:

$$Compress_q^s(x)=x_{11}\oplus(\neg x_{11}\cdot x_{10}\cdot x_9\cdot (x_8\oplus(\neg x_8\cdot x_7)) \quad (17)$$

针对Bos的解码器算法,Özeren等人[48]提出了一种可能的替代方案。这是基于Bos的算法使用了很多逻辑运算,并调用了A2B算法,这可能造成较大的开销。为了尝试降低开销,Özeren在Bos的基础上进行了一些改进,提出了双重校验算法,在某些情况下可能优于Bos的方案。

双重校验算法提供了一种处理算术掩码的方法,引入了一定程度的混淆。在这种方法中,直到包括移位过程在内的所有内容都与Bos的方法保持一致。首先,每个掩码都通过添加$q/4$来进行移位,这使区间对称地围绕$q/2$。这允许该算法产生一个布尔输出,指示原始值是否超过了$q/2$。

在实现对称性之后,该算法将每个移位后的掩码加到自身,即每个掩码乘2,随后计算模q。这种方法测量移位后的掩码是否在求模q时溢出,从而间接判断原始掩码与$q/2$的大小关系。如果溢出了,这意味着原始掩码大于$q/2$。通过检查求模q后的掩码是否与求模q之前不同,可以检测到这种溢出的情况。如果这两个值不相同,则说明发生了溢出,即原始掩码大于$q/2$。

(3) 防护方案对比分析

Bos等人测试了他们的算法的开销,如表9所示。作者采用了C语言实现了一阶掩码解码器,在FRDM-KL82Z上进行了测试。对于解码器中调用的A2B,作者采用了Debraize的实现[49]。同时,作者对比了两种不同的解码器实现,一种是基于查找表解码器,另一种是Bos的基于二分法的解码器。

表9 基于查找表与基于二分法的解码器开销对比[33]

实现方案	时钟周期		
	初始化	解码器	总计
查找表	766	66	832
二分法	181	145	326

每次Kyber调用都会生成新的查找表,该部分开销包含在初始化中。由于初始化和解码器在每次解封装时只调用一次,因此解码器总成本计算为初始化和解码器算法的总和。

(4) 实现建议

Bos提出的解码器相比于Oder的解码器效率更高,因为它解决了模数转换所造成了额外开销。此外,Bos首次将解码器扩展到高阶,因此其方案也是当前的主流方案。

Özeren的方法对Bos的方法进行了一些可能的改进,将其中的A2B算法省去,为了通过将原始掩码值乘为2倍,并与原始掩码值进行比较。这种方案虽然将开销较大的A2B省略了,但引入的加法和比较并不是开销小的操作,因此该方案与Bos的方法孰优孰劣还需要更进一步的讨论。

5.2.5 掩码比较

(1) 防护目标

Fujisaki-Okamoto变换(即FO变换)是一种标准的转换算法,Kyber使用它将CPA安全的方案转换为CCA安全的方案。FO变换的核心思想是,在解密过程中,重新加密解密完的输出以检测输入密文是否有效。重新加密后的密文将与原始输入(封装得到的密文)进行比较,如果两者的结果相等则输

出解密结果，反之则算法输出一个随机值。FO变换包含了重新加密和比较两个部分，重新加密过程与封装过程一致，采用相同的防护方案，本章节仅介绍比较算法的防护方案。图10描述了FO变换如何应用于Kyber规范，图中Compare部分执行了比较算法。

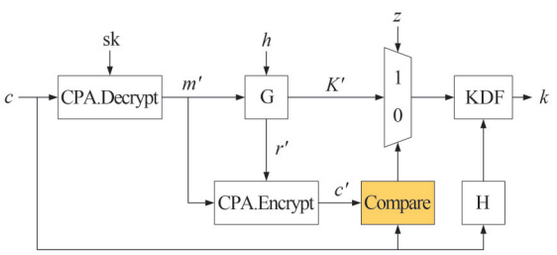

图10　FO变换流程

虽然比较算法可以保护解密过程免受黑盒攻击，但它无法抵御使用侧信道攻击来获取敏感中间值的攻击者。这是因为比较算法的安全性需要依赖于一个事实，即当输入了一个恶意的密文时，攻击者无法获取中间计算的敏感值。具体而言，比较算法的目标是验证重新加密的密文c'是否与输入的密文c匹配。虽然比较的最终结果并不敏感，但侧信道攻击者必须不能在比较失败的情况下了解c和c'的差异[20]。否则攻击者可以很容易地执行解密失败攻击。因此，如果没有适当的保护措施，侧信道攻击者可以很容易地破坏比较算法。

（2）典型防护方案

Bache等人首次提出高阶掩码的比较算法，并提出了“随机求和”算法降低了算法的开销[50]。其核心思想在于对多项式系数进行分组求和后再参与比较计算，而不是为每个系数单独计算。该算法可以通过简单调整每组系数的个数，来达到安全性与执行效率之间的平衡。

系数集合选择的要求是，攻击者只有极小的概率能让构造的密文通过比较算法。具体来说，k个系数被分配到x个长度为$l=k/x$的集合中，每个集合用索引集合I_m表示（为了简单起见，假设k能被x整除）。集合I_m的构造方式如下：

$$0\leqslant m<x:\ I_m=\left\{m\cdot\frac{k}{x},\cdots,(m+1)\cdot\frac{k}{x}-1\right\} \tag{18}$$

在构造了集合I_m之后，分别计算以下x个集合的和B_m^j，它们是在系数子集上进行的（所有操作都是模q进行的，并且按集合进行）：

$$0\leqslant j<n,\ 0\leqslant m<x:\ B_m^j=\sum_{i\in I_m}(A_i^j+r_{1,i})\cdot r_{2,i} \tag{19}$$

其中，$r_{1,i}$和$r_{2,i}$是均匀采样的随机数。然后将每个掩码相加：

$$B_m=\sum_{j=0}^{n-1}B_m^j \tag{20}$$

通过整理并计算和，可以得到每个集合的和：

$$\begin{aligned}B_m&=\sum_{j=0}^{n-1}\left(\sum_{i\in I_m}(A_i^j+r_{1,i})\cdot r_{2,i}\right)=\\&\sum_{i\in I_m}((\sum_{j=0}^{n-1}A_i^j+n\cdot r_{1,i})\cdot r_{2,i})=\\&\sum_{i\in I_m}((A_i^j+n\cdot r_{1,i})\cdot r_{2,i})\end{aligned} \tag{21}$$

因此，现在可以计算封装得到的原始密文多项式中相同索引集系数的和$\tilde{B}_m$。然后将其与B_m进行比较。比较过程是不需要掩码的，可以使用任何方法，例如将B_m与$\tilde{B}_m$相减并检查是否为零。如果对于每个集合，都有$\tilde{B}_m=B_m$，则比较算法返回成功，否则返回失败。

Bache等人证明了攻击者只有$(1/q)^x$的概率能让构造的密文通过比较算法，然而Bhasin等人表明该方案存在安全性问题，攻击者可以设法将构造的密文通过比较算法的概率提高到$(1/q)$[20]。Anver等人改进了Bache的方案，修复了安全性漏洞[34]。新的算法包括五个步骤：第1步将输入从模数q转换为一个更大的模数$p\cdot 2^{s-1}$，其中s是安全系数，它决定了碰撞概率（等于2^{-s}）；第2步在每个系数上分别执行原始密文与重新加密密文的相减；第3步中执行了A2B转换，转换后将$\boldsymbol{u}$和v的长度统一，随后执行B2A转换；在第4步中，执行了随机求和算法，但不同于Bache的版本，这里将计算所有系数的求和；第5步仅对一个求和结果进行零检查。

与以前的随机求和方法一样，Anver的方法同样可能出现碰撞问题，但其概率大约为2^{-s}，这也就意味着攻击者平均需要提交2^s个无效的密文才能触发一次碰撞。在安全性证明中，作者证明了这种碰撞概率与攻击者的输入无关，因此攻击者无法增加触发碰撞的概率。也就是说，必须触发多次碰撞才能显著降低方案的安全性，这对攻击者而言攻击代价巨大。

Anver的方案保证了掩码比较的安全性，但其在算法效率上并不高。Coron等人引入了一种混合方法来执行比较操作。由于Kyber的封装阶段最终会对密文执行压缩，因此先前的比较算法同样需要

对重新加密的密文执行压缩后才能进行比较，但压缩往往开销较大，因为该算法需要在掩码的情况下执行。混合方法的思想是使用不同的方法处理密文分量$\boldsymbol{u}$和v。其原因在于：密文分量$\boldsymbol{u}$只经历了较小的压缩（如由12位压缩到10位），而密文分量v通常经历了更强的压缩（如由12位压缩到4位）。解压技术只适用于较小的压缩，因此只用于处理$\boldsymbol{u}$，而v则采用传统的压缩方法进行处理。由于原始密文是无掩码状态，因此对其执行解压所需的开销相较于掩码压缩要小得多。

Anver等人[51]于22年提出了对21年算术比较方法的改进版本伽罗瓦域比较，以及对混合比较的改进版本简化混合比较。伽罗瓦域比较与算术比较方法的区别在于，当执行完算术比较中的A2B后不再执行B2A，而是直接在布尔域上继续计算。这种方法的主要优势在于，在特征为2的伽罗瓦域中，加法运算等价于对输入的异或操作，这在布尔表示中非常有效。因此，最终的求和操作可以直接在布尔掩码上执行，无需进行昂贵的B2A转换。混合比较算法的一个缺点是与其他比较方法相比较复杂，因为它使用了大量子函数。简化混合比较的目标是减少算法中的子函数，提高其效率。在简化的混合比较中，作者仍然使用了对u的解压，因为它们提供了显著的加速。在执行完这些操作之后，就恢复到了标准的算术比较过程。对于密文分量v，他们将其转换到布尔域，并对布尔掩码系数执行OR操作。根据推算，使用这种实现技巧可以在实际实现中显著加速OR操作。虽然从理论上讲，这些操作的扩展性不如混合方法中的某些替代方法，但它们在实际实现中却胜过这些函数。

（3）防护方案对比分析

Anvers等人实现了上述的算法并进行基准测试，实验结果见表10。所有的基准测试在STM32F407开发板上进行，该板使用ARM-Cortex M4F，编译器为arm-none-eabi-gcc版本9.2.1，并使用了-O3优化选项。系统时钟设置为24 MHz，TRNG时钟设置为48 MHz，符合当前流行的基准测试框架PQM4。随机数从片上TRNG中采样，其采样成本已包含在周期计数中。

（4）实现建议

首先，由于Bache的方案并不安全，因此不会采用这种方案。在剩下的方案中，Anver的算术比较方案和伽罗瓦域比较方案是相似的，Coron的混合比较方案和伽罗瓦域的简化混合比较方案是相似的。

表10 不同掩码比较方案开销对比[51]

实现方案	时钟周期		随机数	
	一阶	二阶	一阶	二阶
算术比较	5.3 M	9.7 M	111 K	251 K
GF比较	4.2 M	6.7 M	47 K	95 K
混合比较	3.3 M	4.5 M	80 K	94 K
简化混合比较	2.5 M	3.6 M	44 K	62 K

Anver的算术比较方案的优点在于通用性强，它没有针对特殊情况做出优化，因此可以适用于任何场景，但它的缺点是执行效率不高。伽罗瓦域比较方案相比于算术比较省去了B2A的开销，因此该方案更适合硬件实现。但它的缺点是需要实现额外伽罗瓦域乘法，这提高了算法实现的复杂性。

Coron的混合比较方案在理论上最优，拥有最好的算法复杂度，但是其涉及大量的函数，在实际应用时未必能达到理论最优。伽罗瓦域的简化混合比较方案简化了Coron的方案，省略了其中一些函数。这样做虽然在理论上会增加复杂度，但在实际应用时可以达到更高的效率。

5.3 Dilithium算子防护方案

5.3.1 Dilithium采样器

（1）防护目标

在不泄露关于$\boldsymbol{s}_1$、$\boldsymbol{s}_2$和$\boldsymbol{y}$的信息的情况下生成相应算术掩码。在Dilithium签名方案中，采样器模块的泄漏点主要集中在生成和处理随机多项式的过程中。具体泄漏点包括：密钥构成$\boldsymbol{s}_1$和$\boldsymbol{s}_2$的生成，在密钥生成过程中，如果生成过程中的中间值未能充分保护，会成为潜在的泄漏点。随机向量$\boldsymbol{y}$的生成，在签名过程中，$\boldsymbol{y}$的生成如果未能进行有效掩码，可能会被侧信道攻击者利用，从而恢复部分秘密信息。

（2）典型防护方案

算法14在$-\phi_0 \leqslant x \leqslant \phi_1 \bmod p$范围内对$x$进行均匀采样，在提供掩码的均匀随机流$(x_0^{B,k}, x_1^{B,k}, \cdots)$时，输出算术掩码。将其均匀分布的输入位视为布尔掩码$x_i^{B,k}$，通过SecLeq检查得到x严格大于$\phi_0+\phi_1$，反之，x均匀分布在$0 \leqslant x \leqslant \phi_0+\phi_1$的范围内。然后将$x$的布尔掩码转换为算术掩码$x^{A_q}$。最后，在掩码中加入$-\phi_0 \bmod p$，使得$x$均匀分布在$-\phi_0 \leqslant x \leqslant \phi_1 \bmod p$上。

算法14. $\mathrm{SecSample}\mathit{Mod}\mathrm{p}^{d}_{q,\phi_0,\phi_1}(\mathbf{x}_0^{B,k}, \mathbf{x}_1^{B,k}, \cdots)$

输入：边界ϕ_0和ϕ_1，其中$\phi_0 \geqslant 0, \phi_1 \geqslant 0$且$\phi_0+\phi_1<q$

输出：算术掩码$\mathbf{x}^{A_q}$，满足$-\phi_0 \leqslant x \leqslant \phi_1 \bmod q$

1. $k \leftarrow \lceil \log_2(\phi_0+\phi_1+1) \rceil$
2. $i \leftarrow 0$
3. While $\neg \mathrm{SecLeq}^{d}_{\phi_0+\phi_1}(\mathbf{x}_i^{B,k})$ do
4. $\quad i \leftarrow i+1$
5. $\mathbf{x}^{A_q} \leftarrow SecB2AModp^{d}_{q}(\mathbf{x}_i^{B,k})$
6. $\mathbf{x}^{A_q}[0] \leftarrow \mathbf{x}^{A_q}[0]-\phi_0 \bmod q$

（3）实现建议

在实现采样器模块时，需要结合实际应用需求进行优化。Azouaoui等人[26]的方法是Migliore等人[25]方法的改进，并且减少了一个SecAdd操作，因此可以优先选择使用Azouaoui的方案。

5.3.2　掩码分解函数

（1）防护目标

已知$\boldsymbol{w}$掩码，在不进行掩码组合的前提下对$\boldsymbol{w}$进行分解，得到$\boldsymbol{w}_0$的掩码和$\boldsymbol{w}_1$，即

$$w_1, (w_{0,1}, \cdots, w_{0,n}) = \mathrm{SecDecompose}((W_1, \cdots, W_n), 2\gamma_2) \quad (22)$$

（2）典型防护方案

SecDecomposeComp方法不但适用于不同安全级别，弥补了其他方案的不足，而且在实际应用中表现优于其他方案。算法重点部分是第二行的$r_1 = \lfloor r \cdot \delta/q \rfloor \bmod^{+} \delta$，$\lfloor x \cdot p/q \rfloor$的高阶计算可以表示为两个高阶计算的组合：

$$\left\lfloor \frac{x \cdot p}{q} \right\rfloor = \left\lfloor \left(\left\lfloor \frac{x \cdot p \cdot 2^{\rho}}{q} \right\rfloor + 2^{\rho-1} + e \right) / 2^{\rho} \right\rfloor \pmod p \quad (23)$$

算法15.　SecDecomposeComp掩码分解

输入：$x_1, \cdots, x_n \in \mathbb{Z}_q$，其中$r = x_1 + \cdots + x_n \bmod q$

输出：r_1和$y_1, \cdots, y_n \in \mathbb{Z}_q$，其中$(r_1, r_0) = \mathrm{Decompose}_q(r, \alpha)$，$y_0 = y_1 + \cdots + y_n \pmod q$

1. $\rho \leftarrow \lceil \log_2(q \cdot n) \rceil + 1$
2. For $i = 1$ to n do $z_i := \lfloor x_i \cdot \delta \cdot 2^{\rho}/q \rfloor \bmod (\delta 2^{\rho})$
3. $z_1 \leftarrow z_1 + (n-1) + 2^{\rho-1} \bmod (\delta 2^{\rho})$
4. For $i = 0$ to $\rho - 1$ do $z_1, \cdots, z_n \leftarrow \mathrm{ShiftMod}(2^{\rho-i} \cdot \delta, (z_1, \cdots, z_n))$
5. $(z_1, \cdots, z_n) \leftarrow \mathrm{RefreshMasks}_{\delta}(z_1, \cdots, z_n)$
6. $r_1 := \sum_i z_i \bmod^{+} \delta$
7. $(y_1, \cdots, y_n) := (x_1, \cdots, x_n)$
8. $y_1 := y_1 - \alpha \cdot r_1 \bmod q$
9. Return $r_1, (y_1, \cdots, y_n)$

（3）防护方案对比分析

表11列出了不同掩码分解算法运行时的循环计数，从表中可以看出SecDecomposeComp方法比SecDecomposeMod更高效，因为它仅需要利用ShiftMod操作进行连续的算术移位，而无需进行完整的模q运算到布尔值的转换。然而，当安全等级为3和5时，两种方法的效率差异会略微缩小，因为此时SecDecomposeMod可以避免布尔值到算术值的转换过程。

（4）实现建议

Coron等人[37]对Azouaoui等人[26]的两个方法进行了扩展，使得每种方法都可以覆盖所有安全级别，对于需要较高效率和适用广泛的实现，建议采用Coron提出的SecDecomposeComp方法。这种方法不仅在实际实现的表现中效率更快，而且适用于所有安全级别。

5.3.3　掩码边界检查

（1）防护目标

通过$\boldsymbol{z}$、$\boldsymbol{r}_0$的掩码判断$\boldsymbol{z}$和$\boldsymbol{r}_0$的无穷范数是否在某个特定范围内，不泄露$\boldsymbol{z}$、$\boldsymbol{r}_0$的信息。侧信道攻击可以通过分析拒绝采样期间的泄漏信息来获取敏感数据，本章节针对防止信息泄露，抵御侧信道攻击和确保签名正确的目标，防护签名阶段的边界检查步骤。

（2）典型防护方案

SecBoundCheck算法提供了一种有效的方法来确保输入的算术掩码符合特定的范围要求，从而有助于保护Dilithium签名方案免受信息泄露的影响。$\mathrm{SecBoundCheck}^{d}_{q,\lambda_0,\lambda_1}(x^{A_q})$，如果输入的算术掩码$x^{A_q}$满足性质$-\lambda_0 \leqslant x \leqslant \lambda_1 \bmod q$，则返回一位$b$。算法16的第一步是将$\lambda_0$加到$x$的算术掩码中，得到$x'$的算术掩码。因此，只有当$0 \leqslant x' \leqslant \lambda_0 + \lambda_1 \bmod q$时，输出比特$b$被设为1。第二步是通过SecLeq检查该条件。在Dilithium中，SecBoundCheck可用于执行拒绝检查$\|\boldsymbol{z}\|_{\infty} \geqslant \gamma_1 - \beta$和$\|\boldsymbol{r}_0\|_{\infty} \geqslant \gamma_2 - \beta$，其中$\gamma_1$、$\gamma_2$和$\beta$由Dilithium规格定义。

表11　掩码分解的循环计数（i9-13900H CPU，2.60 GHz）（其中$n=t+1$，$q=2^{23}-2^{13}+1$）

	t	1	2	3	4	5	6
Level 2	SecDecomposeComp	2,021	4,715	8,508	13,743	17,660	20,322
	SecDecomposeMod	2,357	5,456	10,063	14,548	18,989	27,299
Level 3 & 5	SecDecomposeComp	3,823	3,419	7,231	8,687	14,273	24,674
	SecDecomposeMod	2,171	3,540	8,529	17,074	16,385	31,169

算法16. $\mathrm{SecBoundCheck}^{d}_{q,\lambda_0,\lambda_1}\left(x^{A_q}\right)$边界检查

输入:算术掩码x^{A_q},$\lambda_0+\lambda_1<q$,$\lambda_0\geqslant 0$,$\lambda_1\geqslant 0$

输出:b,若$-\lambda_0\leqslant x\leqslant\lambda_1$,$b=1$,否则$b=0$

1. $x'^{A_q}_0\leftarrow x^{A_q}_0+\lambda_0 \bmod q$

2. $x'^{B,k}\leftarrow \mathrm{SecA2BModp}^{d}_{q}\left(x'^{A_q}\right)$

3. $b\leftarrow \mathrm{SecLeq}^{d}_{\lambda_0+\lambda_1}\left(x'^{B,k}\right)$

(3) 实现建议

针对Dilithium拒绝采样的边界检查条件,Azouaoui等人[26]提出的SecBoundCheck简化了Migliore等人[25]提出的bool::rejection方法,用算术掩码加法代替了其中一个SecAdd,性能提升了约1.1倍,实现效果更好。

5.4 防护方案总结

基于上述对Kyber和Dilithium中各算子掩码防护方案的评估,本节中将总结Kyber和Dilithium中各算子的脆弱点和开销最优的掩码防护方案。

5.4.1 Kyber各算子防护方案总结

表12总结了无防护Kyber中各算子的侧信道脆弱点,并给出了对应的开销最优的掩码防护方案。需要注意的是,解封装中包含重新加密,由于该过程与封装阶段一致,表12中仅统计了解封装中新增的算子。此外,虽然A2B和B2A算法在无防护Kyber中不需要实现,但掩码采样器中会使用B2A算法,掩码压缩和解码器中会使用A2B算法,因此我们将A2B和B2A添加在了对应算子后面。

5.4.2 Dilithium各算子防护方案总结

表13总结了无防护Dilithium中各算子的侧信道脆弱点,并给出了对应的开销最优的掩码防护方案。其中,采样器用到B2A算法,边界检查用到A2B算法,我们同样将A2B和B2A添加在了对应的算子后面。

表12 Kyber脆弱点及建议防护方案总结

算子名称	脆弱点	建议防护方案
密钥生成阶段		
哈希函数	算法1第2行	Arribas等人[31]
采样器	算法1第8和11行	Fritzmann等人[30]
B2A	算法1第8和11行	Coron等人[36]
封装阶段		
哈希函数	算法5第3行	Arribas等人[31]
采样器	算法2第8、11和13行	Fritzmann等人[30]
B2A	算法1第8和11行	Coron等人[36]
编码器	算法2第16行	Oder等人[32]
压缩	算法2第17和18行	Fritzmann等人[30]
A2B	算法2第17和18行	Liu等人[41]
解封装阶段		
解码器	算法3第4行	Bos等人[33]
A2B	算法2第17和18行	Liu等人[41]
比较	算法6第7-10行	D'Anvers等人[51]

表13 Dilithium脆弱点及建议防护方案总结

算子名称	脆弱点	建议防护方案
加密阶段		
采样器	算法7第3行	Azouaoui等人[26]
B2A	算法7第3行	Coron等人[36]
分解函数	算法7第6行	Coron等人[37]
签名阶段		
采样器	算法8第6行	Azouaoui等人[26]
B2A	算法8第6行	Coron等人[36]
分解函数	算法8第8行	Coron等人[37]
边界检查	算法8第13行	Azouaoui等人[26]
A2B	算法8第13行	Liu等人[41]

6 防护代价分析

6.1 软件防护实现中各防护算子的占比

格密码的侧信道防护方案相比无防护方案的代价是目前防护方向发展重点关注的挑战问题。由于算子的运算特点和操作流程,各个算子的防护代价存在差异。一般地,线性操作更容易设计防护方案,非线性操作需要额外的特殊设计处理。格密码包含的算子种类繁多,给防护带来了较大开销。

表14是对FRDM-KL82Z(Cortex-M0+)平台上Kyber-768不同算子的掩码实现进行的性能基准测试。表中统计的循环计数以千为单位,并四舍五入到最近的10^3个周期,结果中为一次完整的解封装操作,即包括一次加密、一次解密和一次FO变换操作,不包括密钥生成阶段。其中,Hash H、Parse和XOF算子不需要掩码防护。

表15按照不同算子展示了在一阶掩码防护下level-3 Dilithium实现的循环计数,以及无掩码实现的整体循环计数,以千为单位[37]。算法大部分时间都花在分解和拒绝中,近期研究和未来攻关的研究点大多集中在这两个算子的防护方案设计上。

6.2 硬件实现中无防护与有防护的代价对比

现有防护方案在硬件上的实现工作较少,表16展示了Kyber无防护与防护方案的硬件实现代价对比。相比无防护的硬件实现,采用随机延迟类的防护实现可以以较低代价取得一定安全性的提升,但是无

表 14　Kyber防护代价

算子名称	无防护PQClean	一阶防护	参考文献	防护算子时间占比
Hash G	61	361	Bos et al. 2021[33]	3.0%
Hash H	530	535	Bos et al. 2021[33]	4.5%
Parse和XOF	1,755	1,723	Bos et al. 2021[33]	14.4%
CBD和PRF	494	3,729	Bos et al. 2021[33]	31.2%
NTT和多项式乘法	2,321	3,853	Bos et al. 2021[33]	32.2%
Encoder	21	287	Bos et al. 2021[33]	2.4%
Decoder	22	143	Bos et al. 2021[33]	1.2%
Compress_{du}	/	101	Coron et al. 2021[44]	0.8%
Compress_{dv}	/	26	Coron et al. 2021[44]	0.2%
Comparison	5	1,206	Bos et al. 2021[33]	10.1%

表 15　level-3 Dilithium一阶掩码的防护代价[37]

算子名称	Cycle counts
NTTs	304
Sample y	3,034
Compute Ay	616
Decompose	13,088
z=y+cs1	355
Reject	18,956
$w-c\cdot s_2$	262
Dilithium3	853
Masked Dilithium3	36,986 (×43)

表 16　Kyber无防护与防护方案的硬件实现代价

方案	防护	LUT	FF	BRAM	DSP	CLOCK
Xing et al. 2021[52]	无防护	7,412	4,644	3	2	6,668
Moraitis et al. 2023[53]	随机时钟	14,340	9,190	6	/	10,030
Jati et al. 2024[54]	随机延迟&乱序	7,157	3,730	5.5	2	57,200
Kamucheka et al. 2022[55]	掩码	152,680	92,977	489.5	76	137,700

法从根源去除掉泄露；当前采用掩码实现的硬件实现代价相比于无防护实现代价较高，如何优化格密码掩码方案的硬件实现是当前急需的关键技术之一。

7　总　结

格密码在后量子密码标准中占据重要角色，当在物联网、金融等领域大规模应用时，不可避免地需要考虑实际实现的安全性，因此，面向格密码的能耗侧信道防护技术是格密码抵抗侧信道攻击的重要保证。

现有格密码的能耗侧信道防护技术的研究已经取得了较大发展，在不同的格密码种类上均提出了相应防护方案，将可证明安全框架适配到格密码中保证理论安全性，针对性地对复杂掩码操作进行了改进优化。另外，仍然存在以下几个不足：

(1) 防护方案代价仍然过高

格密码的核心操作在掩码后需要分解为多个共享值分别计算，为了满足安全模型还需要引入大量随机数以及掩码刷新操作，导致运算时间和计算负载显著增加，难以满足一些低延迟场景要求。

(2) 防护方案面临新型攻击威胁

随着攻击技术的发展，开始逐渐出现针对掩码方案的攻击工作。这些攻击不是证明掩码方案的本身安全性存在问题，而是在实际应用场景中以更高效的攻击效率达到恢复密钥效果。目前这些攻击可以分成两类。第一类攻击是利用深度学习的侧信道攻击技术对掩码方案进行高阶攻击。Wang等人在2024年ACNS会议上提出通过MLP模型对掩码解压缩进行攻击[56]，其攻击原理是利用了MLP网络对两个共享因子的泄露进行联立，从而达到高阶攻击的效果；第二类攻击是针对特定实现提出的高效攻击。Hermelink1等人在2024年CCS会议上提出了一种针对掩码比较方案的攻击[57]，其攻击原理利用了实现过程中两个共享因子比特同乘一个64比特的随机数，导致当共享因子为0和1时，乘法结果的汉明重分布差异巨大，从而可以让攻击者较容易恢复出来。该攻击同样没有挑战方案本身安全性，而是利用实现过程中的泄露特征提出高效的攻击手段。

(3) 缺少防护方案的安全测评标准

现有针对防护方案的测评方法在研究中更多的还是采用TVLA，通过观察是否存在异常值粗略判断方案是否安全。目前，尚未形成针对格密码甚至后量子密码的侧信道检测标准，因此，对于防护方案的检测和安全性评估尚不完善。另外，对于防护方案的泄露定位和形式化验证同样是未来的发展趋势。

格密码的能耗侧信道防护是后量子密码从理论标准迈向实际应用的关键环节。本文以Kyber和Dilithium为研究对象，系统分析了抵抗能耗侧信道攻击的防护目标整体架构和各算子的防护策略。在可证明安全框架下，针对不同防护方案分析其设计思路、防护代价，给出防护建议。本文通过系统性分析揭示了其脆弱性本质与防护优化路径，为学术界与工业界提供了理论参考与技术指南。未来研究需聚焦于跨算子协同防护、新型攻击防御及标准化评估体系的构建，推动格密码在金融、物联网等领域的可信落地，保障后量子时代的信息安全根基。

参 考 文 献

[1] Taha M, Eisenbarth T. Implementation attacks on post-quantum cryptographic schemes. Cryptology ePrint Archive, 2015, 11: 1-14

[2] Wu W., Liu Z., Yang H., et al. (2021). A survey on side-channel attacks and defenses for post-quantum cryptographic algorithms. Journal of Software, 32(4), 1165-1185 (in Chinese) (吴伟彬，刘哲，杨昊，等. 后量子密码算法的侧信道攻击与防御综述. 软件学报，2021, 32(4): 1165-1185)

[3] Chowdhury S, Covic A, Acharya R Y, et al. Physical security in the post-quantum era: A survey on side-channel analysis, random number generators, and physically unclonable functions. Journal of Cryptographic Engineering, 2022, 12(3): 267-303

[4] 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. ACM Transactions on Embedded Computing Systems, 2024, 23(2): 1-54

[5] Albrecht M R, Deo A. Large modulus ring-LWE≥module-LWE//Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security. San Francisco, USA, 2017: 267-296

[6] Banerjee A, Peikert C, Rosen A. Pseudorandom functions and lattices//Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin, Germany, 2012: 719-737

[7] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings//Advances in Cryptology-EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques. French, 2010: 1-23

[8] Langlois A, Stehlé D. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography, 2015, 75(3): 565-599

[9] Avanzi R, Bos J, Ducas L, et al. CRYSTALS-Kyber algorithm specifications and supporting documentation. NIST PQC Round, 2019, 2(4): 1-43

[10] Peeters E. Side-channel cryptanalysis: A brief survey. Advanced DPA Theory and Practice: Towards the Security Limits of Secure Embedded Circuits, 2013: 11-19

[11] Kocher P, Jaffe J, Jun B. Differential power analysis//Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference. Santa Barbara, USA, 1999: 388-397

[12] Sendrier N. Post-quantum cryptography//Proceedings of the Third International Workshop, PQCrypto. Darmstadt, Germany. 2010

[13] Barthe G, Belaïd S, Dupressoir F, et al. Strong non-interference and type-directed higher-order masking//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna, Austria, 2016: 116-129

[14] Cassiers G, Standaert F X. Trivially and efficiently composing masked gadgets with probe isolating non-interference. IEEE Transactions on Information Forensics and Security, 2020, 15: 2542-2555

[15] Pessl P, Primas R. More practical single-trace attacks on the number theoretic transform//Progress in Cryptology-LATINCRYPT 2019: 6th International Conference on Cryptology and Information Security in Latin America. Santiago de Chile, Chile, 2019: 130-149

[16] Kannwischer M J, Pessl P, Primas R. Single-trace attacks on keccak. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020: 243-268

[17] D'Anvers J P, Tiepelt M, Vercauteren F, et al. Timing attacks on error correcting codes in post-quantum schemes//Proceedings of ACM Workshop on Theory of Implementation Security Workshop. London, UK, 2019: 2-9

[18] Ravi P, Bhasin S, Roy S S, et al. On exploiting message leakage in (few) NIST PQC candidates for practical message recovery attacks. IEEE Transactions on Information Forensics and Security, 2021, 17: 684-699

[19] Xu Z, Pemberton O, Roy S S, et al. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. IEEE Transactions on Computers, 2021, 71(9): 2163-2176

[20] Bhasin S, D'Anvers J P, Heinz D, et al. Attacking and defending masked polynomial comparison for lattice-based cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021: 334-359

[21] Ulitzsch V Q, Marzougui S, Tibouchi M, et al. Profiling side-channel attacks on dilithium. Nature, 2024, 13742: 3

[22] Wang R, Ngo K, Gärtner J, et al. Single-trace side-channel attacks on CRYSTALS-dilithium: Myth or reality?. Cryptology ePrint Archive, 2023, 12: 1-28

[23] Qiao Z, Liu Y, Zhou Y, et al. Single trace is all it takes: Efficient side-channel attack on dilithium. Cryptology ePrint Archive, 2024, 4: 1-24

[24] Fan H, Zhang H, Wang Y, et al. Screening Least Square Technique assisted Multivariate Template Attack against the Random Polynomial Generation of Dilithium. IEEE Transactions on Information Forensics and Security, 2024, 19: 7118-7132

[25] Migliore V, Gérard B, Tibouchi M, et al. Masking dilithium:

Efficient implementation and side-channel evaluation//Applied Cryptography and Network Security: 17th International Conference, ACNS 2019. Bogota, Colombia, 2019: 344-362

[26] Azouaoui M, Bronchain O, Cassiers G, et al. Protecting dilithium against leakage: Revisited sensitivity analysis and improved implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2023,(4): 58-79

[27] Berzati A, Viera A C, Chartouny M, et al. Exploiting intermediate value leakage in dilithium: a template-based approach. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2023,(4): 188-210

[28] Zhou Y, Wang W, Sun Y, et al. Rejected Challenges Pose New Challenges: Key Recovery of CRYSTALS-Dilithium via Side-Channel Attacks. Cryptology ePrint Archive, 2025, 10: 1-31

[29] Liu Z, Wang A, Wei C, et al. Release the Power of Rejected Signatures: An Efficient Side-Channel Attack on Dilithium. Cryptology ePrint Archive, 2025, 10: 1-14

[30] Fritzmann T, Van Beirendonck M, Roy D B, et al. Masked accelerators and instruction set extensions for post-quantum cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022: 414-460

[31] Arribas V, Bilgin B, Petrides G, et al. Rhythmic keccak: SCA security and low latency in HW. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018: 269-290

[32] Oder T, Schneider T, Pöppelmann T, et al. Practical CCA2-secure and masked ring-LWE implementation. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018: 142-174

[33] Bos J W, Gourjon M, Renes J, et al. Masking kyber: First-and higher-order implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021: 173-214

[34] D'Anvers J P, Heinz D, Pessl P, et al. Higher-order masked ciphertext comparison for lattice-based cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021: 624-650

[35] Barthe G, Belaïd S, Espitau T, et al. Masking the GLP lattice-based signature scheme at any order//Advances in Cryptology-EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II 37. AvivTel, Israel, 2018: 354-384

[36] Coron J S, Gérard F, Lepoint T, et al. Improved High-Order Masked Generation of Masking Vector and Rejection Sampling in Dilithium. Cryptology ePrint Archive, 2024, 11: 1-25

[37] Coron J S, Gérard F, Trannoy M, et al. Improved gadgets for the high-order masking of dilithium. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2023, (4): 110-145

[38] Goubin L. A sound method for switching between boolean and arithmetic masking//International Workshop on Cryptographic Hardware and Embedded Systems. Berlin, Germany, 2001: 3-15

[39] Coron J S, Großschädl J, Vadnala P K. Secure conversion between boolean and arithmetic masking of any order//International Workshop on Cryptographic Hardware and Embedded Systems. Berlin, Germany, 2014: 188-205

[40] Bronchain O, Cassiers G. Bitslicing arithmetic/boolean masking conversions for fun and profit: with application to lattice-based kems. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022: 553-588

[41] Liu J, Zhao C, Peng S, et al. A low-latency high-order arithmetic to boolean masking conversion. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2024, 2024(2): 630-653

[42] Bettale L, Coron J S, Zeitoun R. Improved high-order conversion from Boolean to arithmetic masking. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018: 22-45

[43] Schneider T, Paglialonga C, Oder T, et al. Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto//IACR International Workshop on Public Key Cryptography. Cham: Springer International Publishing. Beijing, China, 2019: 534-564

[44] Coron J S, Gérard F, Montoya S, et al. High-order polynomial comparison and masking lattice-based encryption. Cryptology ePrint Archive, 2023: 1-48

[45] Heinz D, Kannwischer M J, Land G, et al. First-order masked Kyber on ARM Cortex-M4[J]. Cryptology ePrint Archive, 2023, 12: 1-11

[46] Reparaz O, Sinha Roy S, Vercauteren F, et al. A masked ring-LWE implementation//International Workshop on Cryptographic Hardware and Embedded Systems. Berlin, Germany, 2015: 683-702

[47] Reparaz O, Roy S S, De Clercq R, et al. Masking ring-LWE. Journal of Cryptographic Engineering, 2016, 6(2): 139-153

[48] Özeren S, Yayla O. Methods for masking crystals-kyber against side-channel attacks//2023 16th International Conference on Information Security and Cryptology (ISCTürkiye). Ankara, Turkiye, 2023: 1-6

[49] Debraize B. Efficient and provably secure methods for switching from arithmetic to Boolean masking//International Workshop on Cryptographic Hardware and Embedded Systems. Berlin, Germany, 2012: 107-121

[50] Bache F, Paglialonga C, Oder T, et al. High-speed masking for polynomial comparison in lattice-based KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020: 483-507

[51] D'Anvers J P, Van Beirendonck M, Verbauwhede I. Revisiting higher-order masked comparison for lattice-based cryptography: Algorithms and bit-sliced implementations. IEEE Transactions on Computers, 2022, 72(2): 321-332

[52] Xing Y, Li S. A compact hardware implementation of CCA-secure key exchange mechanism Crystals-kyber on FPGA. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021: 328-356

[53] Moraitis M, Ji Y, Brisfors M, et al. Securing CRYSTALS-Kyber in FPGA using duplication and clock randomization. IEEE Design & Test, 2023, 41(5): 7-16

[54] Jati A, Gupta N, Chattopadhyay A, et al. A configurable crystals-kyber hardware implementation with side-channel

protection. ACM Transactions on Embedded Computing Systems, 2024, 23(2): 1-25
[55] Kamucheka T, Nelson A, Andrews D, et al. A masked pure-hardware implementation of kyber cryptographic algorithm// Proceedings of the 2022 International Conference on Field-Programmable Technology (ICFPT). Hong Kong, China, 2022: 1-1
[56] Wang R, Brisfors M, Dubrova E. A Side-Channel Attack on a Higher-Order Masked CRYSTALS-Kyber Implementation// Proceedings of the International Conference on Applied Cryptography and Network Security. Abu Dhabi, United Arab Emirates, 2024: 301-324
[57] Hermelink J, Ning K C, Petri R, et al. The Insecurity of Masked Comparisons: SCAs on ML-KEM's FO-Transform// Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security. Salt Lake City, USA, 2024: 2430-2444

LI Yan-Bin, Ph. D., associate professor. His main research interests include side-channel attack, cryptography and cryptographic chip.

GUO Yi-Kang, M. S. candidate. Hisresearch interests include side-channel analysis and cryptanalysis.

ZHANG Shu-Qi, M. S. candidate. Her research interests include side-channel analysis and pqc.

TANG Ming, Ph. D., professor, Ph. D. supervisor. Her main research interests include cryptography, hardware security, side-channel attacks and countermeasures.

GE Chun-Peng, Ph. D., professor, Ph. D. supervisor. His main reaearch interests include privacy protection in cloud computing, blockchain, AI security and privacy.

XU Qiu-Liang, Ph. D., professor, Ph. D. supervisor. His main research interests include applied cryptography, data security in clouds, and privacy protection.

Background

The impending realization of scalable quantum computers poses a significant threat to public-key cryptosystems. To resist attacks from quantum computers, post-quantum cryptography (PQC) has become the core direction for quantum-resistant cryptography standardization. In particular, NIST established the Kyber algorithm based on the Module-LWE problem as the FIPS 203 Key Encapsulation Mechanism standard and the Dilithium algorithm as the FIPS 204 Digital Signature Algorithm standard in 2024. These algorithms are now being progressively adopted and integrated into practical systems.

During the ongoing transition to post-quantum cryptography, physical security in real-world environments has become an important issue. This survey aims to provide a comprehensive view of the vulnerabilities of Kyber and Dilithium to side-channel attacks, analyze the defense strategies and overhead bottlenecks of various masking schemes, and build a protection framework. Lattice-based cryptography, which underlies these algorithms, introduces unique algebraic operations, including polynomial ring arithmetic, the Number Theoretic Transform (NTT), message encoding and decoding, boundary checks, and the Fujisaki-Okamoto (FO) transformation. Instead of addressing single vulnerabilities, this paper presents an overall protection architecture for lattice-based cryptography. This structured approach with tailored strategies for critical operations like NTT and polynomial multiplication, is crucial for building a comprehensive defense. This aligns with the global research trend of moving beyond theoretical security to build practical, efficient, and secure implementations. By evaluating the implementation costs (area, performance, security) of existing countermeasures and providing practical deployment guidance, this paper analyzes the major bottlenecks, such as the high cost of masking non-linear operations.

This paper focuses on Kyber and Dilithium as key representatives of lattice-based PQC and systematically examines their vulnerabilities to side-channel attacks along with protection strategies. Our research team has researched on the field of power, electro-magnetic analysis for 10 years. We completed a prototype systems for side-channel analysis, protection, and testing. This research was supported by the National Natural Science Foundation of China (62472259, 62572285, 62172258), the Program of Taishan Young Scholars of Shandong Province (tsqn202408026), project ZR2024QF098 supported by Shandong Provincial Natural Science Foundation. The identification of future challenges, like the need for efficient high-order masking, provides a valuable roadmap for research in the cryptographic engineering field.