QIMEN-PIKE

算法技术规范

(版本 1.0)

- Yu Yu, Shanghai Jiao Tong University, yyuu@sjtu.edu.cn
- Wouter Castryck, COSIC, KU Leuven, wouter.castryck@esat.kuleuven.be
- Mingjie Chen, COSIC, KU Leuven, mjchennn555@gmail.com
- Arthur Herlédan Le Merdy, COSIC, KU Leuven, arthur.herledanlemerdy@esat.kuleuven.be
- Zhi Hu, Central South University, huzhi_math@csu.edu.cn
- Zhuo Huang, Shanghai Jiao Tong University, sh1kaku@sjtu.edu.cn
- Riccardo Invernizzi, COSIC, KU Leuven, riccardo.invernizzi@esat.kuleuven.be
- Yi-Fu Lai, Shanghai Jiao Tong University, yifu.lai@sjtu.edu.cn
- Dania Lazzarini, Université Libre de Bruxelles, dania.lazzarini@ulb.be
- Kaizhan Lin, Fudan University, linkzh@fudan.edu.cn
- Xuzhe Liu, Central South University, luuuxuuz@gmail.com
- Luciano Maino, University of Birmingham, mainoluciano.96@gmail.com
- Krijn Reijnders, COSIC, KU Leuven, reijnders.krijn@gmail.com
- Frederik Vercauteren, COSIC, KU Leuven, frederik.vercauteren@esat.kuleuven.be
- Yunqi Wen, Central South University, 232103006@csu.edu.cn

July 10, 2026

译者说明

本中文版本系对英文原版之译文。因时间所限，译校工作未尽完善；部分专业术语之中文译名尚未形成统一标准，仅供参考。后续我们计划在 QIMEN 主页 (qimen.team) 上持续完善中文译本。

若中英文两版本在语义或解释上存在歧异，**应以英文原版之表述为最终依据**。

Contents

CHAPTER 1

引言

QIMEN 加密套件（**Q**uaternion and **I**sogeny **M**achinery over **EN**domorphism Rings，QIMEN）是一套面向后量子安全的综合密码框架，旨在为数字基础设施提供抵御经典攻击和量子攻击的能力。本套件包含两个核心密码原语：密钥封装机制 QIMEN-PIKE 和数字签名方案 QIMEN-PRISM。两者的安全性均建立在超奇异同源路径问题和自同态环问题的困难性之上。这两类数学问题为后量子密码方案提供了不同于格密码和编码密码的安全基础。由于 QIMEN-PIKE 与 QIMEN-PRISM 共享有限域、椭圆曲线、四元数及同源计算等通用构造模块，两份规范中的部分章节采用了相同的基础内容。

本文档描述 密钥封装机制 QIMEN-PIKE。

1.1 设计理由

QIMEN-PIKE 建立在基于同源的 ElGamal 型公钥加密方案 PIKE [CCLL26] 之上 (该论文发表于 PKC'26)，后者本身建立在 POKÉ [BM25] 之上。QIMEN-PIKE 遵循相同的数学困难假设和协议体系。

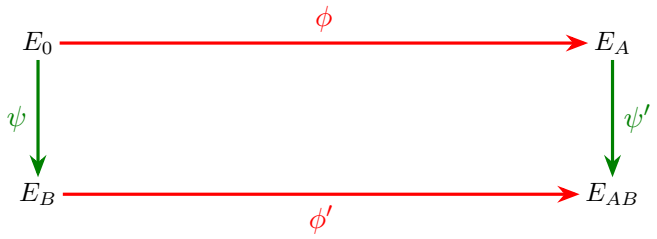

Figure 1.1: 基于同源的 ElGamal 型公钥加密。接收者私密同源 ϕ 定义了公钥 E_A。发送者利用 E_A 附带的充分辅助信息，推导出一条兼容的绿色同源路径 ψ'，并将 E_{AB} 作为加密传递的目标曲线。

数学问题： QIMEN-PIKE 的核心困难问题是带挠点信息的超奇异同源问题的一个掩蔽变体。设 E 和 E' 为超奇异椭圆曲线，设 $\phi : E \to E'$ 为一个秘密同源，且 ϕ 在 $E[N]$ 上的

限制为同构，设 (P,Q) 为 $E[N]$ 的一组基。指定一个掩蔽群 $\Gamma \subseteq \mathrm{GL}_2(\mathbb{Z}/N\mathbb{Z})$，掩蔽挠同源问题 定义为：给定

$$E, \qquad E', \qquad (P,Q), \qquad M\begin{pmatrix}\phi(P)\\ \phi(Q)\end{pmatrix}$$

对于某个未知矩阵 $M \in \Gamma$，恢复 ϕ。未掩蔽情况 $M = I_2$ 是带挠点信息的超奇异同源问题，作为 SIDH 和 SIKE [JD11, JAC+22] 的基础。

SIDH/SIKE 中公开的 未掩蔽挠像导致了多项式时间密钥恢复攻击 [CD23, MMP+23, Rob23]。这些攻击本质上依赖 $\phi(P)$ 和 $\phi(Q)$ 暴露的代数关系。当挠像经未知掩蔽矩阵 M 隐藏后，攻击便无法直接适用。自 SIDH/SIKE 遭破解以来，掩蔽同源问题已成为基于同源的密码学中的核心困难假设，已应用于众多协议设计中，如 M-SIDH [FMP23]、FESTA [BMP23]、QFESTA [NO24]、LIT-SiGamal [MS25] 等。学术界对该同源问题变体的密码分析工作也已展开，包括 [CV23] 和 [DFP24]。然而，对于 QIMEN-PIKE 及上述方案中使用的掩蔽挠实例，目前尚无已知的多项式时间恢复算法，已知的最佳适用算法仍为指数复杂度。

核心设计思路： QIMEN-PIKE 遵循 Fig. 1.1 所示的基于同源的 ElGamal 型体系。接收者采样一个秘密同源 $\phi : E_0 \to E_A$，并以 E_A 连同 ϕ 下若干挠点的掩蔽像作为公钥发布。发送者采样一个临时光滑次数同源 $\psi : E_0 \to E_B$，再利用公开的挠信息构造其前推 $\psi' : E_A \to E_{AB}$。由此得到一个满足 $\phi' \circ \psi = \psi' \circ \phi$ 的交换图表，而掩蔽操作阻止了发送者恢复接收者的秘密同源。

交换图表建立后，QIMEN-PIKE 以配对值作为共享秘密。发送者传递 E_B 上的一个掩蔽 D-挠点以及 E_{AB} 上的一个掩蔽 D-挠点。接收者利用秘密掩蔽信息，计算单个二维同源以获得中间曲线上的对应点，再计算其配对。借助配对与同源的相容性以及图表的交换性，接收者恢复了发送者计算出的同一配对导出值。

FO 变换： 按照标准的 Fujisaki-Okamoto 变换 [FO99]，上述 IND-CPA 安全的公钥加密方案转化为 IND-CCA2 安全的密钥封装机制。封装算法采样随机消息，将消息与公钥一起哈希，导出密钥材料和确定性的加密随机数，然后使用 QIMEN-PIKE.PKE 加密消息。解封装过程中，接收者解密密文并重新加密恢复出的消息以完成验证。有效密文产生相同的共享秘密，而无效密文则触发隐式拒绝 [Per12]，以秘密回退值替代。所得 KEM 的 IND-CCA2 安全性在随机预言机模型下由 QIMEN-PIKE.PKE 的 IND-CPA 安全性导出。

参数选择与失败分析： QIMEN-PIKE 的三组参数集，分别记为 NGCC-1、NGCC-2 和 NGCC-3，其目标经典安全强度分别为 128、256 和 512 比特，目标量子安全强度分别为 80、128 和 256 比特。对于所有三组参数集，总体失败概率的上界为 2^{-128}。在 Section 8.4 中，各安全级别的参数选择按照 NGCC 提交要求进行论证。失败概率的详细分析见 Chapter 9。

1.1.1 相对于 PIKE 的改进

尽管基于 PIKE [CCLL26] 的核心设计，QIMEN-PIKE 引入了若干理论和工程创新，与先前的学术原型形成区分。这些差异总结如下。

- **新参数与优化流程：** QIMEN-PIKE 实例化了一组新的安全参数，专门针对 NGCC 提交中多样化的安全级别需求而设计。此外，方案部署了针对性的椭圆曲线算术优化，包括针对所选参数调优的加速标量乘法和改进的三点阶梯算法。这些底层算法精化显著减少了底层域操作的总数，确保 QIMEN-PIKE 在满足 NGCC 严格安全标准的同时，在所有目标类别中均达到最新的计算效率。
- **定制化公钥/密文压缩：** QIMEN-PIKE 采用两种优化技术来缩减公钥和密文的大小。第一种技术通过点组合来减少存储需求。第二种技术将坐标点编码为相对于固定挠基的标量表示，实现严格更高的压缩率，以一定的计算效率换取更小的表示尺寸。基于此，方案提供了两种不同的实现：未压缩变体和压缩变体。虽然压缩实现的效率低于未压缩实现，但它显著缩小了公钥（约 $\approx 37\%$）和密文（约 $\approx 25\%$）。
- **汇编级域算术内核：** 方案实现融合了手动优化的汇编过程，利用 x86-64 BMI2 和 ADX 指令。通过精炼非饱和基数肢表示并最小化模乘期间的进位传播开销，加密套件的运行时间相比通用多精度基准缩减了超过 10%。

1.2 特性陈述

1.2.1 创新性

大多数在 SIDH 被攻破后提出的同源公钥加密方案都依赖高维同源和辅助挠点信息，例如 [BMP23, NO24, BM25, MS25]。在这些方案中，POKÉ [BM25] 是 PIKE [CCLL26] 和 QIMEN-PIKE 的主要技术基础，其共享秘密由共享同源对挠基的像导出。

不同于 POKÉ，PIKE 及由其衍生的 QIMEN-PIKE 采用了一种新型使用配对的加密架构。其主要创新在于该范式中特有的共享秘密关系和解密机制。为简化表述，下文统一使用 QIMEN-PIKE 指代这一框架。需要强调的是，以下协议层面的创新均源自 PIKE。

- **配对导出共享秘密关系：** QIMEN-PIKE 是首个从配对值，而非椭圆曲线的 j-不变量或完整挠基的像中导出共享秘密的同源公钥加密方案 [BMP23, NO24, BM25, MS25]。发送者对交换同源图中相互对应的 D-挠点计算配对，并利用所得配对值掩蔽消息。借助这一机制，解密速度提高了约 24%。
- **利用配对放松对有理挠的光滑性要求：** 据现有文献，QIMEN-PIKE 是首个利用配对做共享秘密恢、进而放宽对 $\mathbb{F}_{p^2}$-有理 D-挠点阶数光滑性要求的同源公钥加密方案。由于接收者可以直接通过配对恢复共享值，解密过程无需在阶为 D 的群中求解离散对数。因此，D 无需为光滑数，从而放宽了对域特征的限制，并允许选用更小的素数 p。较小的素数进一步将密钥生成、封装和解封装的速度提高至原来的 1.24 至 1.3 倍。

综合而言，这些特性使 QIMEN-PIKE 区别于现有方案，并形成了一套独立的配对辅助同源加密范式，而非对既有工作的简单变体或参数调整。

1.2.2 简洁性

QIMEN-PIKE 的一个核心设计优势在于其解密流程十分简洁。早期的 POKÉ 系列方案 [BM25] 通过高维同源下的挠点像导出共享秘密，接收者需要利用 Kani 引理重构交换

图，并求解光滑阶群上的离散对数，才能恢复消息。相比之下，QIMEN-PIKE 完全省去了这些步骤：共享秘密通过配对值而非挠基的像进行传递，因此解密仅需计算少量配对并执行一次求逆运算，接收者既无需应用 Kani 引理两次，也无需计算光滑离散对数。

与此同时，该协议保留了清晰的 ElGamal 型结构：公钥由一条秘密同源确定，发送者则通过计算相应的同源生成密文（见 Figure 1.1）。这种简洁的结构使技术描述和安全性论证都更加紧凑。此外，上述简化还放宽了对底层素数的算术限制，使方案可以在更小的有限域上实现，从而进一步降低实现复杂度。

1.2.3 灵活性与兼容性

QIMEN-PIKE 采用模块化设计，能够灵活适配不同的工程环境，并兼容标准的 KEM 变换：

- **标准 KEM 变换：** QIMEN-PIKE.KEM 以底层公钥加密方案 QIMEN-PIKE.PKE 为基础，通过标准的 Fujisaki–Okamoto 型变换构造。在（量子）随机预言机模型下，其 IND-CCA2 安全性可由 PKE 的 IND-CPA 安全性导出。该变换以独立封装层的形式实现，因此不同安全级别采用相同的核心加解密流程。
- **可配置的哈希/XOF 后端：** 协议将上层哈希与密钥派生过程同底层可扩展输出函数（XOF）的具体实现相分离。在通用部署场景中，可以采用 SHAKE256；在需要符合国内商用密码标准的场景中，则可以采用基于 SM3 的构造。因此，同一套协议实现只需切换编译选项，即可适配不同的合规要求。
- **灵活的点压缩模式：** 公钥和密文的表示方式可以根据目标平台的带宽与计算资源进行选择。QIMEN-PIKE 支持未压缩的坐标表示，以降低计算开销；同时也支持压缩的基系数表示，将点编码为相对于固定挠基的标量，从而以少量额外计算为代价，显著减小传输尺寸。

1.2.4 可扩展性

QIMEN-PIKE 采用清晰的 ElGamal 型结构，因而可以方便地作为基础构件集成到更复杂的密码协议中。其共享秘密以公开交换图上的配对值为载体，而不依赖方案特定的辅助信息，因此相同的密钥生成和封装接口可以自然支持以下功能：

- **良构性证明：** QIMEN-PIKE 支持对公钥和密文进行高效的良构性证明 [LM26]。借助此类证明，用户可以在不泄露底层秘密信息的前提下，证明给定的公钥或密文满足协议规定的结构。这使得 QIMEN-PIKE 更适合作为基础构件用于高层密码协议，因为在这些协议中，恶意构造的公钥或密文可能破坏安全性论证。
- **公钥加密与密钥封装：**同一底层构造既可实例化为公钥加密方案 QIMEN-PIKE.PKE，也可通过标准变换得到密钥封装机制 QIMEN-PIKE.KEM。因此，需要上述任一原语的应用都可以复用同一套经过分析和实现验证的核心组件。
- **静态与临时密钥部署：** 协议将接收者的长期秘密同源与发送者在每次封装时生成的临时同源明确分离。因此，在不改变协议核心流程的情况下，既可以面向固定接收者重复执行封装，也可以用于构建支持前向安全的临时会话。

- **可组合的压缩层：**点压缩层与协议核心相互独立，因此可以在不影响协议正确性和安全归约的前提下，引入新的压缩方法或替换现有的椭圆曲线点编码方式。

1.2.5 性能

- **紧凑的公钥和密文：**对于通信带宽紧张、内存受限的应用场景，QIMEN-PIKE 的公钥和密文尺寸特别有吸引力。此类场景包括：通过低速网络通信的受限物联网设备、传输容量有限的智能卡和非接触式设备、间歇连接的嵌入式系统，以及卫星或远程遥测链路。在这些场景中，减小公钥和密文大小有助于减少传输时间、分组分片、存储需求和重传开销。在 NIST 安全级别 5 下，使用 NGCC-2 的 QIMEN-PIKE 的公钥和密文分别仅需 595 B 和 882 B。如 Table 1.1 所示，这些尺寸相较于 ML-KEM-1024（Kyber）[SAB+20] 分别缩小了 2.6 倍和 1.8 倍，相较于 HQC-256 [AAB+22] 分别缩小了 12.2 倍和 16.4 倍。

Table 1.1: NIST 安全级别 5（与使用 NGCC-2 的 QIMEN-PIKE 相同）下，NIST 已标准化或已入选标准化的后量子 KEM 的公钥和密文大小对比，单位为字节。

方案	公钥	密文
QIMEN-PIKE (NGCC-2)	**595 B**	**882 B**
ML-KEM-1024 (Kyber)	1568 B (2.6×)	1568 B (1.8×)
HQC-256	7245 B (12.2×)	14 469 B (16.4×)

- **合理的封装和解封装效率：**尽管 QIMEN-PIKE 慢于 ML-KEM 和 HQC，其封装和解封装效率仍保持在数十毫秒量级。例如，在 NGCC-1 安全级别下，优化后的 64 位实现在 2.7 GHz 的 Intel Ultra 9 CPU 上，封装约需 18 ms，解封装约需 29 ms（见 Table 7.4）。这一结果表明，通信开销的大幅降低并未带来难以接受的在线计算成本。

1.2.6 假设的多样性

QIMEN-PIKE 的安全性建立在同源相关困难问题之上，与格密码和编码密码所依赖的问题具有本质区别，因此有助于增强后量子密码基础设施的技术多样性。自同态环问题具有简洁的最坏情况到平均情况自归约，这一性质尤其有利于安全归约。此外，QIMEN-PIKE 公钥的分布与均匀分布之间仅存在很小的统计距离，因此其安全性可以建立在接近均匀分布、通常被认为最难求解的自同态环问题实例之上。

星号上标

同源密码学是一个高度综合的研究领域，许多协议都建立在已有工作的基础之上。因此，有限域、椭圆曲线和四元数等底层算术通常采用标准方法，并在不同方案的实现与技术文档中复用，以保持与所引用工作的定义和记号一致。鉴于 QIMEN-PIKE 的主要贡献集中

在协议层面，本规范复用了 SIKE [JAC+22] 和 SQIsign [AAA+25] 相关技术文档中的部分基础内容。为明确标示这些复用材料，相应章节的标题均以上标 * 标注。

伪代码惯例与异常处理

为了表述清晰，本技术规范在伪代码中使用异常来描述错误处理。当所需的随机化搜索或结构检查失败时，算法可能抛出异常。如果子过程调用抛出了异常，而调用方算法没有将该调用包含在 **try/catch** 块中，则调用方算法抛出同样的异常；等价地说，未被捕获的异常向上传播到下一个调用方算法。当存在 **catch** 块时，异常按该块中指定的方式处理。

Algorithm 1 异常处理约定

```
try
    执行可能抛出异常的指令。
    if 发生错误条件 then
        raise Exception(" 错误描述")
catch
    从异常中恢复，根据算法需要导出隐式拒绝回退密钥、重试随机化流程、返回 false 或拒绝输入。
```

伪代码还遵循标准的循环控制约定：在循环内部，语句 **continue** 跳过当前迭代中的剩余指令，继续下一次迭代。

Table 1.2: QIMEN-PIKE 的符号与参数

	基本对象
$\mathbb{Z}/d\mathbb{Z}$	整数模 d 的环，其中 d 为某个整数
$\mathbb{F}_p$	含 p 个元素的有限域
$\mathbb{F}_q$	含 $q = p^k$ 个元素的有限域
$\mathbb{F}_{p^2}$	含 p^2 个元素的有限域
$\overline{\mathbb{F}_p}$	$\mathbb{F}_p$ 的代数闭包
$\mathrm{GL}_n(q)$	规模为 n、元素取自 $\mathbb{F}_q$ 的可逆矩阵群
	椭圆曲线对象
E	某域 $\mathbb{F}_q$ 上的一条椭圆曲线
$E \times E'$	两条椭圆曲线 E 与 E' 的乘积
A	某域 $\mathbb{F}_q$ 上的一个阿贝尔簇
$\mathrm{Jac}(C)$	超椭圆曲线 C 的 Jacobian 簇
$E_{A,B}$	Montgomery 型椭圆曲线，参数为 $A, B \in \mathbb{F}_q$
0_E	椭圆曲线或阿贝尔簇 E 上的无穷远点
$E(\mathbb{F}_q)$	E 上坐标在 $\mathbb{F}_q$ 中的点
$E[n]$	E 上的 n-挠
$hintgen_i$	生成 $E[2^f]$ 的一组基的提示
$j(E)$	椭圆曲线 E 的 j-不变量
x_P 与 y_P	点 $P \in E$ 的 x 坐标和 y 坐标
$\varphi : E \to E'$	椭圆曲线之间的同源
$\Phi : A \to A'$	高维同源
$\mathrm{End}(E)$	椭圆曲线 E 的自同态环
$t_n(P, Q)$	次数为 n 的 Tate 配对，在 $P, Q \in E[n]$ 处取值
$e_n(P, Q)$	次数为 n 的 Weil 配对，在 $P, Q \in E[n]$ 处取值
	四元代数对象
$\mathcal{B}_{p,\infty}$	一个四元代数，在 p 和 ∞ 处分歧
$\mathcal{B}^\star_{p,\infty}$	线性函数空间 $\mathcal{B}_{p,\infty} \to \mathbb{Q}$
$\mathcal{O}$	$\mathcal{B}_{p,\infty}$ 中的一个序（order）
$\mathcal{O}_0$	满足 $\mathrm{End}(E_0) \cong \mathcal{O}_0$ 的特殊极值极大序
$\mathcal{O}_L(I)$, $\mathcal{O}_R(I)$	理想 $I \subset \mathcal{O}$ 的左序和右序
$\mathrm{tr}(\alpha)$	元素 $\alpha \in \mathcal{B}_{p,\infty}$ 的迹
$\mathrm{nrd}(\alpha)$	元素 $\alpha \in \mathcal{B}_{p,\infty}$ 的范数
$\mathrm{nrd}(I)$	理想 $I \subset \mathcal{O}$ 的范数
$[I]^*$	理想 I 的拉回
$[I]_*$	理想 I 的前推

Table 1.3: QIMEN-PIKE 的符号与参数

	安全参数
λ_c	目标经典安全强度，单位为比特
λ_q	目标量子安全强度，单位为比特
λ	内部安全参数，取为 $2\lambda_q$
	公共参数
pp	公共参数，包括 $p, a, C, C_1, C_2, D, E_0$、挠基以及 w_0
p	域特征
a	2 幂挠参数，满足 $p+1 = 2^a C_1 D$
C_1, C_2, C	奇挠参数，满足 $C = C_1 C_2$、$C_1 \mid p+1$、$C_2 \mid p-1$ 且 $\gcd(C_1, C_2) = 1$
D	用于配对导出共享值的挠阶
(P_0, Q_0)	$E_0[2^a]$ 的一组基
(R_0, S_0)	$E_0[C]$ 的一组基
(X_0, Y_0)	$E_0[D]$ 的一组基
w_0	Tate 配对值 $t_D(X_0, Y_0)$
	CPA-PKE 对象
Π_{CPAPKE}	IND-CPA 公钥加密方案 QIMEN-PIKE.PKE $=$ $(\mathsf{QIMEN-PIKE.PKE.KeyGen}, \mathsf{QIMEN-PIKE.PKE.Encrypt}, \mathsf{QIMEN-PIKE.PKE.Decrypt})$
$\mathsf{QIMEN-PIKE.PKE.KeyGen}$	QIMEN-PIKE.PKE 的密钥生成算法
$\mathsf{QIMEN-PIKE.PKE.Encrypt}$	QIMEN-PIKE.PKE 的加密算法
$\mathsf{QIMEN-PIKE.PKE.Decrypt}$	QIMEN-PIKE.PKE 的解密算法
q	用于定义密钥生成同源次数 $q(2^{a-2} - q)$ 的秘密整数
$\alpha_2, \beta_2, \delta_D$	存储在 PKE 私钥中的秘密掩蔽标量
γ_C	C-挠像的掩蔽标量，从 $(\mathbb{Z}/C\mathbb{Z})^\times$ 中采样；不存储在私钥中
pk	公钥 $(E_A, (P_A, Q_A), (R_A, S_A), X_A)$
sk	QIMEN-PIKE.PKE 的私钥 $(q, \alpha_2, \beta_2, \delta_D)$
	加密与解密对象
r	加密随机数，通过 $R_0 + [r]S_0$ 和 $R_A + [r]S_A$ 定义次数为 C 的同源
ω_2	传输的 2^a-挠像的掩蔽标量
η_D 与 ζ_D	传输的 D-挠像的掩蔽标量
E_B 与 E_{AB}	加密过程中产生的像曲线
E_M	解密过程中用于恢复配对值的中间曲线
pad	由 $\mathsf{H}(\mathsf{KDF} \Vert \mathsf{Tr}(w'))$ 导出并与消息异或的填充
ct	包含 E_B, E_{AB}、掩蔽像点以及 $\mathsf{pad} \oplus \mathsf{msg}$ 的密文
	CCA-KEM 对象
QIMEN-PIKE.KEM	通过 Fujisaki–Okamoto 型变换从 QIMEN-PIKE.PKE 得到的 CCA 安全 KEM
$\mathsf{Enc.Det}$	KEM 重加密校验中使用的确定性加密算法
$\mathsf{H_{pk}}$	公钥的哈希
$\mathsf{H_{ct}}$	密文转录的哈希
G	产生 $(\bar{K}, \rho)$ 的哈希并展开函数
KDF	用于最终共享秘密的密钥派生函数
ρ	用于确定性导出封装硬币的种子
z	KEM 私钥中存储的用于隐式拒绝的回退秘密
K	封装与解封装输出的最终共享秘密

CHAPTER 2

数学基础 *

2.1 有限域

我们沿用 [JAC+22] 和 [AAA+25] 中的记号。有限域是元素的有限集合，其上定义了加法与乘法运算，满足通常的算术规则。具体而言，加法与乘法封闭，存在加法单位元 0 和乘法单位元 1，每个元素都有加法逆元和乘法逆元（元素 0 除外，它不可逆）。

基数为 q 的有限域存在当且仅当 q 是素数幂，即 $q = p^r$，其中 p 为某个素数，r 为正整数。这样的有限域在同构意义下唯一，记为 $\mathbb{F}_q$。我们将其乘法群，即 $\mathbb{F}_q \setminus \{0\}$，记为 $\mathbb{F}_q^\times$。对于 $q = p^r$，称 p 为 $\mathbb{F}_q$ 的**特征**（$\mathrm{char}(\mathbb{F}_q) = p$）。QIMEN-PIKE 使用的域特征满足 $p \equiv 3 \pmod 4$，由此总可将 $\mathbb{F}_{p^2}$ 中的元素表示为 $a + bi$，其中 $a, b \in \mathbb{F}_p$。关于具体素数 p 的更多细节，参见Chapter 5。

所有特征为 p 的有限域的并称为 $\mathbb{F}_p$ 的 **代数闭包**（algebraic closure），记作 $\overline{\mathbb{F}_p}$。该结构本身是一个域，但不再是有限的。设 L 是一个域，K 为其子域。若将 L 的加法与乘法运算限制在 K 上后与 K 原有的运算一致，则称 L 为 K 的一个**域扩张**（field extension）。常见例子包括 $\mathbb{F}_{p^2}$ 作为 $\mathbb{F}_p$ 的扩张，以及更一般地，$\mathbb{F}_q$ 作为 $\mathbb{F}_p$ 的扩张。

2.1.1 有限域 $\mathbb{F}_p$

有限域 $\mathbb{F}_p$ 的元素可用整数 $\{0, \ldots, p-1\}$ 唯一表示，其代数运算定义如下。

加法。对于 $a, b \in \mathbb{F}_p$，其和 $c = a + b$ 为满足 $c \equiv a + b \pmod p$ 的唯一整数 $c \in \{0, \ldots, p-1\}$。

加法逆。对于 $a \in \mathbb{F}_p$，其加法逆 $-a$ 为满足 $a + (-a) \equiv 0 \pmod p$ 的唯一整数 $-a \in \{0, \ldots, p-1\}$。

乘法。对于 $a, b \in \mathbb{F}_p$，其积 $c = a \cdot b$ 为满足 $c \equiv a \cdot b \pmod p$ 的唯一整数 $c \in \{0, \ldots, p-1\}$。

乘法逆。对于 $a \in \mathbb{F}_p^\times$，其乘法逆 a^{-1} 为满足 $a \cdot a^{-1} \equiv 1 \pmod p$ 的唯一整数 $a^{-1} \in \{0, \ldots, p-1\}$。

二次剩余。设 $a \in \mathbb{F}_p^\times$。判断 a 是否为平方元，即是否存在 $b \in \mathbb{F}_p^\times$ 使得 $b^2 = a$。可通过计算 Legendre 符号 $a^{\frac{p-1}{2}}$ 实现：若 a 为平方元，则该值为 1；否则为 -1。

平方根。设 $a \in \mathbb{F}_p$ 为 $\mathbb{F}_p$ 中的平方元。由于我们限定素数 $p \equiv 3 \pmod 4$，a 的规范平方根如下计算：

$$\sqrt{a} = a^{\frac{p+1}{4}} \pmod p. \tag{2.1}$$

此外，我们在 $\mathbb{F}_p$ 的元素上定义一个**字典序**：将元素提升到区间 $[0, p-1]$，再按整数大小比较。

2.1.2 有限域 $\mathbb{F}_{p^2}$

由于我们仅使用特征 $p \equiv 3 \pmod 4$ 的域，可将域扩张 $\mathbb{F}_{p^2}$ 定义为 $\mathbb{F}_p(i)$，其中 $i^2 + 1 = 0$。$\mathbb{F}_{p^2}$ 的元素可唯一表示为 $a = a_0 + a_1 \cdot i$，其中 $a_0, a_1 \in \mathbb{F}_p$。代数运算定义如下：

加法。对于 $a, b \in \mathbb{F}_{p^2}$，其和为 $c = c_0 + c_1 \cdot i$，其中 $c_0 = a_0 + b_0$，$c_1 = a_1 + b_1$，均使用 $\mathbb{F}_p$ 中的加法。

加法逆。对于 $a \in \mathbb{F}_{p^2}$，其加法逆 $-a$ 为 $-a = (-a_0) + (-a_1)i$, 使用 $\mathbb{F}_p$ 中的加法逆。

乘法。对于 $a, b \in \mathbb{F}_{p^2}$，其积为 $c = c_0 + c_1 \cdot i$，其中 $c_0 = a_0 b_0 - a_1 b_1$，$c_1 = a_0 b_1 + a_1 b_0$, 使用 $\mathbb{F}_p$ 中的加法、加法逆和乘法。

乘法逆。对于 $a \in \mathbb{F}_{p^2}^\times$，其乘法逆为 $a^{-1} = (a_0 N^{-1}) + (-a_1 N^{-1})i$, 其中 $N = a_0^2 + a_1^2 \in \mathbb{F}_p$, 使用 $\mathbb{F}_p$ 中的加法、加法逆、乘法和乘法逆。

二次剩余。设 $a \in \mathbb{F}_{p^2}^\times$。判断 a 是否为平方元：a 为平方元当且仅当 $a^{p+1} = a_0^2 + a_1^2 \in \mathbb{F}_p$ 在 $\mathbb{F}_p$ 中为平方元。

平方根。设 $a \in \mathbb{F}_p$，则 a 在 $\mathbb{F}_{p^2}$ 中总是平方元。若 a 在 $\mathbb{F}_p$ 中为平方元，则其在 $\mathbb{F}_{p^2}$ 中的平方根由Eq. (2.1)给出；否则 $-a$ 在 $\mathbb{F}_p$ 中为平方元，此时取 $\sqrt{a} = \sqrt{-a} \cdot i$. 最后，设 $a \in \mathbb{F}_{p^2} \backslash \mathbb{F}_p$ 为 $\mathbb{F}_{p^2}$ 中的平方元，其规范平方根定义为

$$\sqrt{a} = (-i)^{\frac{1-\chi}{2}} S(a + \delta), \text{ 其中} \delta = \sqrt{a_0^2 + a_1^2},\ S = \big(2(a_0 + \delta)\big)^{\frac{p-3}{4}},\ \chi = \big(2(a_0 + \delta)\big)^{\frac{p-1}{2}}. \tag{2.2}$$

此外，我们在 $\mathbb{F}_{p^2}$ 的元素上定义一个**字典序**：

$$a_0 + a_1 \cdot i < b_0 + b_1 \cdot i \quad \text{iff} \quad a_0 < b_0 \text{ 或} (a_0 = b_0 \text{ 且} a_1 < b_1). \tag{2.3}$$

Algorithm 2 NormalizedDlog(a, ζ_0, ζ_1)

Input: 正整数 a 以及 μ_{2^a} 中的两个值 ζ_0, ζ_1，其中 $a \le e$，且 ζ_0 的阶为 2^a
Output: 满足 $\zeta_1 = \zeta_0^k$ 的值 $k \in [0, 2^a]$

1: **if** $a = 1$ **then**
2: **return** 穷举搜索满足 $\zeta_1 = \zeta_0^k$ 的 k
3: $a' \leftarrow \lfloor a/2 \rfloor$
4: $\zeta_0' \leftarrow \zeta_0^{2^{a-a'}}$
5: $\zeta_1' \leftarrow \zeta_1^{2^{a-a'}}$
6: $k' \leftarrow$ NormalizedDlog(a', ζ_0', ζ_1')
7: $\zeta_0'' \leftarrow \zeta_0^{2^{a'}}$
8: $\zeta_1'' \leftarrow \zeta_1 / \zeta_0^{k'}$
9: $k'' \leftarrow$ NormalizedDlog$(a - a', \zeta_0'', \zeta_1'')$
10: **return** $k = k' + 2^{a'} k''$

2.1.3 有限域中的离散对数

给定有限域元素 $\zeta \in \mathbb{F}_{p^2}^{\times}$ 及其幂 ζ^k，其中指数 $k \in \mathbb{Z}$ 未知。**离散对数问题**（discrete logarithm problem，DLP）的目标是求出 k。当 ζ 的阶充分光滑时，高效算法可求解该问题。[1] 记 μ_n 为 n 次单位根群，即

$$\mu_n := \{\zeta \in \overline{\mathbb{F}}_p \mid \zeta^n = 1\}.$$

求解形如 μ_{2^a} 上的离散对数有若干算法，可追溯到 Pohlig–Hellman [PH78]。一般而言，只要阶光滑，离散对数均可求解。在我们的实现中，采用Algorithm 2所示的迭代算法NormalizedDlog，基于 Pohlig–Hellman 算法计算两个元素之间的离散对数。QIMEN-PIKE 仅将该算法应用于配对计算的输出，参见Section 2.3.2 。

2.2 椭圆曲线

以下始终假设 $\mathbb{F}_q$ 是一个满足 $\mathrm{char}(\mathbb{F}_q) > 3$ 的有限域。$\mathbb{F}_q$ 上的每条**椭圆曲线**（elliptic curve）均由一个短 Weierstrass 方程 $y^2 = x^3 + ax + b$ 定义，其中 $a, b \in \mathbb{F}_q$。

我们回顾与 QIMEN-PIKE 实现相关的 Montgomery 形式椭圆曲线的若干关键事实。关于 Montgomery 曲线及其性质的全面综述，参见 [CS18]。

2.2.1 Montgomery 曲线

设 $A, B \in \mathbb{F}_q$ 满足 $B(A^2 - 4) \neq 0$。$\mathbb{F}_q$ 上的 Montgomery 曲线 $E_{A,B}$ 是由方程

$$By^2 = x^3 + Ax^2 + x \tag{2.4}$$

[1] 当 ζ 的阶为大素数时，有限域离散对数问题在经典计算模型下通常被认为难以求解，这也是传统 Diffie–Hellman 密钥交换的安全基础。

定义的椭圆曲线。即，它由满足该曲线方程的点 $P=(x,y)$（其中 $x,y\in\overline{\mathbb{F}_q}$）以及无穷远点 0_E 组成。我们常记作 $E_{A,B}/\mathbb{F}_q$，以强调该曲线定义在域 $\mathbb{F}_q$ 上；同时记 $E_{A,B}(\mathbb{F}_q)$ 表示满足 $x,y\in\mathbb{F}_q$ 的点 (x,y) 的集合，再加上 0_E。当 $B=1$ 时，简写为 E_A；对于一般的 Montgomery 曲线，用 E 表示。我们将系数 A 称为 **Montgomery 系数**（Montgomery coefficient）。

若存在线性坐标变换 $(x,y)\mapsto(Dx+R,Cy)$，$D,C\in\mathbb{F}_q^\times$，$R\in\mathbb{F}_q$，将 E 映至 E'，则称两条 Montgomery 曲线 E 和 E' **在 $\mathbb{F}_q$ 上同构**（isomorphic over $\mathbb{F}_q$）。当 B/B' 在 $\mathbb{F}_q$ 中为平方元时，两条 Montgomery 曲线 $E_{A,B}$ 和 $E_{A,B'}$ 同构。设 N 为 $E(\mathbb{F}_q)$ 的基数，即 Eq. (2.4)的解的个数再加上点 0_E。当 $N\equiv 1 \bmod \mathrm{char}(\mathbb{F}_q)$ 时，称 $E_{A,B}$ 为**超奇异**（supersingular）曲线。我们只关注定义在 $\mathbb{F}_{p^2}$ 上且 $p\equiv 3 \bmod 4$ 的超奇异曲线。此时，任意满足 $B=1$ 的超奇异曲线 $E_A/\mathbb{F}_{p^2}$ 恰好有 $(p+1)^2$ 个点；而其二次扭 $E_{A,\gamma}/\mathbb{F}_{p^2}$（其中 γ 是 $\mathbb{F}_{p^2}$ 中的任意二次非剩余）恰好有 $(p-1)^2$ 个点，且它们彼此均同构。我们称具有 $(p+1)^2$ 个点的曲线为**极大**（maximal）曲线，具有 $(p-1)^2$ 个点的曲线为**极小**（minimal）曲线。

2.2.1.1 Montgomery 曲线之间的同构

对于一条 Montgomery 曲线 $E_{A,B}$，定义其 **j-不变量**（j-invariant）为

$$j(E_{A,B})=\frac{256(A^2-3)^3}{A^2-4}. \tag{2.5}$$

j-不变量刻画了代数闭包上椭圆曲线的同构类，即两条曲线具有相同的 j-不变量当且仅当它们同构或互为扭曲线。对于两条同构的曲线 $E_{A,B}$ 和 $E_{A',B'}$，将 $E_{A,B}$ 映至 $E_{A',B'}$ 的坐标变换 $(x,y)\mapsto(Dx+R,Cy)$ 由下式给出：

$$D=\lambda^2\frac{B'}{B},\quad R=\lambda^2\frac{AB'}{3B}-\frac{A'}{3},\quad C=\lambda^3\frac{B'}{B},\quad \lambda:=\sqrt{\frac{B(2A'^3-9A')(3-A^2)}{B'(2A^3-9A)(3-A'^2)}}. \tag{2.6}$$

这些公式由Chapter C中描述的IsomorphismMontgomeryCurves算法实现。

2.2.2 群律

本节定义 Montgomery 曲线上点集的加法运算，使 $E_A(\mathbb{F}_q)$ 构成一个阿贝尔群。在此加法律下，0_E 为单位元，每个点 $P=(x,y)$ 有唯一的逆元 $-P=(x,-y)$，满足 $P+(-P)=0_E$。

以下对于点 $P\neq 0_E$，记其 x-坐标为 x_P，y-坐标为 y_P，即 $P=(x_P,y_P)$。注意，优化实现通常使用射影坐标 $(X:Y:Z)$，其中 $x=X/Z$，$y=Y/Z$，以避免后续点加法与同源公式中的求逆运算（参见 [CS18]）。此外，在大多数场合我们仅使用 x-坐标算术，此时点表示为 $P=(X_P:Z_P)$。

2.2.2.1 点加

设 $E_{A,B}/\mathbb{F}_q$ 为一条 Montgomery 曲线，$P=(x_P,y_P)$ 和 $Q=(x_Q,y_Q)$ 为 $E_{A,B}$ 上的点，且 $P\neq\pm Q$。则其和 $R=P+Q$ 按如下方式计算，其中 $R=(x_R,y_R)$：

$$x_R=B\lambda^2-(x_P+x_Q)-A, \tag{2.7}$$

以及

$$y_R = \lambda(x_P - x_R) - y_P, \tag{2.8}$$

其中 $\lambda = (y_P - y_Q)/(x_P - x_Q)$。

对点 $P = (x_P, y_P)$ 进行倍点运算时，使用相同的公式，但将 λ 替换为 $(3x_P^2 + 2Ax_P + 1)/(2By_P)$。此外，若 $P = -P$，则 $[2]P = P + P = P + (-P) = 0_E$。无穷远点是该群律的零元，故 $P + 0_E = 0_E + P = P$。

2.2.2.2 标量乘法

利用阿贝尔群律，可对任意 $k \in \mathbb{Z}$ 定义标量乘法 $[k] : E \to E$：对点 $P \in E$，记 $[k]P = P + P + \cdots + P$（$k$ 个 P）。对于负的 k，设 $[k]P = -[|k|]P$。对于 $k = 0$，设 $[0]P = 0_E$。为提高效率，标量乘法通常用一系列倍点和点加运算来实现。若使用 Montgomery 阶梯（参见 [CS18]），椭圆曲线点运算的次数为 $O(\log k)$。对于点 $P \in E$，将满足 $[m]P = 0_E$ 的最小正整数 m 称为 P 的**阶**（order）。

2.2.2.3 点差

给定两个点 $P, Q \in E(\mathbb{F}_{p^2})$ 的 x-坐标 x_P 和 x_Q，可以利用 [RS17, Prop. 3] 中的如下公式确定性地计算集合 $\{r_+, r_-\} = \{x_{P-Q}, x_{P+Q}\}$：

$$r_\pm = \frac{B_{XZ} \pm \sqrt{B_{XZ}^2 - B_{ZZ}B_{XX}}}{B_{ZZ}}, \tag{2.9}$$

其中

$$\begin{aligned} B_{XX} &= (x_P x_Q - 1)^2, \\ B_{XZ} &= (x_P x_Q + 1)(x_P + x_Q) + 2Ax_P x_Q, \\ B_{ZZ} &= (x_P - x_Q)^2. \end{aligned} \tag{2.10}$$

由于 $x_Q = x_{-Q}$，仅凭 x-坐标无法区分 r_+ 和 r_- 中哪一个是 x_{P-Q}、哪一个是 x_{P+Q}。但在实际中，我们可以直接用 $-Q$ 替代 Q，因此只需以确定方式从中任取一个。我们始终选取 r_+ 的公式，并采用Section 2.1.2中描述的平方根选取方式。此程序称为ProjectiveDifference，因其在实现中采用射影坐标，参见Section C.2。

系数恢复： 给定同一曲线 E_A 上三个点 $P, Q, R \in E_A$ 的 x-坐标 x_P、x_Q，以及 $R = P - Q$ 的坐标 x_R，改写Equation (2.10)即可恢复 Montgomery 系数 A。相应的算法参见RecoverCodomain，详见Section C.2。

2.2.3 挠子群与确定性基的计算

在 QIMEN-PIKE 中，我们在公开曲线 E_0 上预计算若干挠基。如Section 3.1.1所述，我们分别生成 $E_0[2^a]$、$E_0[C]$ 和 $E_0[D]$ 的基。由于这些基是预计算的，本技术规范不详细讨论其生成方式。

2.3 配对

本节介绍 n 级 Weil 配对和 Tate–Lichtenbaum 配对。利用配对可高效计算点之间的离散对数，亦可生成挠基：配对将椭圆曲线离散对数问题转化为有限域离散对数问题，只要阶是光滑的，就可用NormalizedDlog等函数高效求解。在 QIMEN-PIKE 中，这种方法尤其实用，因为我们主要处理 $E[2^a]$（其中 $a \le e$）中点的离散对数计算。对于我们所用曲线，$E[2^e] \subseteq E(\mathbb{F}_q)$ 成立，因此此类配对计算是高效的。

2.3.1 椭圆曲线上的配对

配对（pairing）是阿贝尔群 A、B、C 之间的双线性映射 $A \times B \to C$。密码学中最常用的是 n 级 Weil 配对和 Tate–Lichtenbaum 配对：此时 A 与 B 取 $E[n]$ 的子群或商群，C 取 n 次单位根群 $\mu_n \subset \mathbb{F}_{p^2}^{\times}$。

Weil 配对： 设 E 为 $\mathbb{F}_q$ 上的椭圆曲线，$n \in \mathbb{Z}$ 与 $\mathrm{char}(\mathbb{F}_q)$ 互素。Weil [Wei40] 引入了 n 级 Weil 配对，定义为映射

$$e_n : E[n] \times E[n] \to \mu_n,$$

它是双线性的、交错的且非退化的。

Tate–Lichtenbaum 配对： Tate–Lichtenbaum 配对最初由 Tate [Tat62] 在局部域上的阿贝尔簇上定义。Lichtenbaum [Lic69] 给出了其在曲线 Jacobian 上的高效计算方法；Frey 和 Rück [FR94] 则将其引入密码学并给出了有限域上的高效算法。假设 $n \mid q-1$。**未约化 Tate–Lichtenbaum 配对**（unreduced Tate–Lichtenbaum pairing）定义为

$$T_n : E(\mathbb{F}_q)[n] \times E(\mathbb{F}_q)/nE(\mathbb{F}_q) \to \mathbb{F}_q^{\times}/(\mathbb{F}_q^{\times})^n.$$

因此，$T_n(P,Q)$ 在相差 n 次幂的意义下是唯一的。在密码学中，我们需要定义明确的唯一值。将结果求 $(q-1)/n$ 次幂后，最终结果即落在 μ_n 中且唯一确定。由此得到**约化 Tate–Lichtenbaum 配对**（reduced Tate–Lichtenbaum pairing）

$$t_n : E(\mathbb{F}_q)[n] \times E(\mathbb{F}_q)/nE(\mathbb{F}_q) \to \mu_n.$$

约化和未约化 Tate–Lichtenbaum 配对都是双线性的、非退化的，且具有 Galois 不变性。当 $E/\mathbb{F}_{p^2}$ 为极大超奇异曲线且 $n \mid (p+1)$ 时，约化 Tate–Lichtenbaum 配对是交错的。

2.3.2 QIMEN-PIKE 中配对的用途

在 QIMEN-PIKE 中，配对在协议设计中起关键作用：它用于推导加密方与解密方之间的共享秘密。具体做法是：取整数 D 和 $E_0[D]$ 的指定基 (X_0, Y_0)（二者均为Section 5.1中定义的方案参数），将 $w_0 := t_D(X_0, Y_0)$ 定义为协议参数 pp，其中 t_D 是次数为 D 的 Tate 配对。解密时，再计算 $w = t_D(-X_M, Y_M)$，其中 (X_M, Y_M) 是一个类似的基。利用 w_0 与 w 之间的特定关系，双方可恢复出同一个值 w'（即 w_0 或 w 的某次幂），这正是加解密所依赖的共享秘密。

2.4 一维同源

设 E_1 和 E_2 是 $\mathbb{F}_q$ 上的两条椭圆曲线。**同源**（isogeny）是指非常值映射 $\varphi : E_1 \to E_2$，其坐标分量为 $\mathbb{F}_q$ 上的有理函数，且满足 $\varphi(0_{E_1}) = 0_{E_2}$。特别地，$\varphi$ 是一个群同态 $\varphi : E_1 \to E_2$。如果两条曲线 E_1 和 E_2 之间存在同源，则称它们是**同源的**（isogenous）。这一性质可以用群的阶来刻画：$\mathbb{F}_q$ 上的两条曲线 E_1 和 E_2 同源，当且仅当 $\#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q)$。同源的复合以 $\circ$ 表示。

每个同源 φ 有一个有限的**次数**（degree），记作 $\deg(\varphi)$。具体来说，φ 在有理函数域上可诱导拉回映射 $\varphi^* : \mathbb{F}_q(E_2) \hookrightarrow \mathbb{F}_q(E_1)$，而 $\deg(\varphi)$ 即域扩张 $[\mathbb{F}_q(E_1) : \varphi^*\mathbb{F}_q(E_2)]$ 的次数。φ 的核是一个由几何点构成的有限子群：$\ker(\varphi) = \{P \in E_1(\overline{\mathbb{F}}_q) \mid \varphi(P) = 0_{E_2}\}$。若 $\#\ker(\varphi)(\overline{\mathbb{F}}_q) = \deg(\varphi)$，则称同源 φ 为**可分**（separable）同源。在超奇异曲线上，此条件等价于次数与 p 互素。

可分同源几乎完全由其核唯一确定。具体而言，给定基数为 N 的子群 $G \subset E_1$，不计后复合同构，存在唯一的椭圆曲线 E_2 和次数为 N 的同源 $\varphi : E_1 \to E_2$，使得 $\ker(\varphi) = G$。因此，给定 G 的一个生成元 Q，核为 $G = \langle Q \rangle$ 的同源可用单个点来表示。此外，每个同源 $\varphi : E_1 \to E_2$ 都存在唯一的**对偶同源**（dual isogeny）$\widehat{\varphi} : E_2 \to E_1$，其次数也是 N。复合 $\widehat{\varphi} \circ \varphi$ 即 E_1 上的标量乘法映射 $[N]$，而 $\varphi \circ \widehat{\varphi}$ 即 E_2 上的 $[N]$。当两个同源 $\phi : E \to E_1$ 和 $\psi : E \to E_2$ 的次数互素时，可以将其中一个同源的核（例如 $K = \ker\psi$）通过另一个同源做**前推**（pushforward），进而得到第三个同源 $\psi' : E_1 \to E_3$，其核为 $\phi(K)$。这一操作称为 ψ 经 ϕ 的前推，记作 $[\phi_*]\psi$。

要显式计算 $\mathbb{F}_q$ 上的同源 φ，可以用 $\mathbb{F}_q$ 上的一对有理映射 $f(x)$ 和 $g(x)$ 来表示它，即 $\varphi((x, y)) = (f(x), y \cdot g(x))$。这两个函数均可写成 $\mathbb{F}_q$ 上互素多项式的比值，如 $f(x) = f_1(x)/f_2(x)$。在这种表示下，次数为 $\deg(\varphi) = \max\{\deg(f_1), \deg(f_2)\}$。

给定一个 N 阶点 Q，Vélu 公式 [Vél71] 提供了一种计算方法，可通过有理映射计算以 $\langle Q \rangle$ 为核的同源 φ。Vélu 公式及其变体的复杂度为 $\tilde{O}(\sqrt{N})$，仅当 N 较小时切实可行。当 N 较大且为合数时，我们将 φ 分解为较小次数的同源的复合：设 $N = \prod \ell_i^{e_i}$ 为 N 的素因数分解，则 φ 可计算为 ℓ_1-同源（次数 ℓ_1）、ℓ_2-同源（次数 ℓ_2）等的复合。具体而言，按 $\varphi = \varphi_E \circ \cdots \circ \varphi_2 \circ \varphi_1$ 计算 φ，其中 $E = \sum e_i$。每个同源 φ_i 的次数为某个素数 $\ell_i \mid N$，因此只要 N 是**光滑**（smooth）的——即 N 仅含充分小的素因子 ℓ_i——该计算就是可行的。

在 QIMEN-PIKE 中，我们计算 $\ell \in \{3, 5, 7, 11, 13\}$ 的 ℓ-同源链，所用工具是 Vélu 公式在 Montgomery 曲线上的特化版本 [CH17]，适用于次数为素数 $\ell \neq 2$ 的同源 $\phi : E_A \to E_{A'}$。一般而言，此类同源可写成如下形式：

$$\phi : (x, y) \mapsto (f(x), c \cdot y \cdot f'(x)),$$

其中 $c \in \mathbb{F}_{p^2}$ 为某常数，$f(x)$ 为有理函数，$f'(x)$ 为其导数。设 P 为核 $\ker\phi$ 的生成元，则 c 和 $f(x)$ 可通过如下公式从 P 算出：

$$c := \prod_{0<s<\frac{\ell-1}{2}} x_{[s]P}, \qquad f(x) := x \cdot \prod_{0<s<\ell} \frac{x_{[s]P} \cdot x - 1}{x - x_{[s]P}}.$$

这些公式在次数和 x-坐标运算上还可进一步优化，因为在 QIMEN-PIKE 的大多数应用中，只需计算点 $Q \in E_A$ 的像 $x_{\phi(Q)}$。具体实现细节将在OddIsogenyChain中给出。

2.5 二维同源

QIMEN-PIKE 使用二维同源来高效计算非光滑次数的椭圆曲线同源。本节介绍所需的必要背景知识。

主极化阿贝尔曲面（principally polarized abelian surfaces, PPAS）是椭圆曲线（参见Section 2.2）到二维的自然推广。具体而言，PPAS 是由多项式方程定义的几何对象，其上的群运算由有理函数（即多项式的分式）给出。在代数闭域（如 $\overline{\mathbb{F}_p}$）上，PPAS 同构于以下二者之一 [Wei57]：

1. 一个亏格-2 超椭圆曲线 C 的 Jacobian $\mathrm{Jac}(C)$，
2. 椭圆曲线的积 $E_1 \times E_2$。

椭圆曲线 E_1, E_2 的积 $E_1 \times E_2$ 上的算术可直接归约为 E_1 和 E_2 各自的算术。$(P_1, P_2), (Q_1, Q_2) \in E_1 \times E_2$ 的加法定义为

$$(P_1, P_2) + (Q_1, Q_2) := (P_1 + Q_1,\ P_2 + Q_2), \tag{2.11}$$

其中 P_1 与 Q_1 在 E_1 上相加，P_2 与 Q_2 在 E_2 上相加。在 $E_1 \times E_2$ 上，单位元为 $(0_{E_1}, 0_{E_2})$。

Jacobian： 以下简要说明 PPAS 的 Jacobian 类型。每个定义在 $\mathbb{F}_{p^2}$ 上的亏格 2 超椭圆曲线均可写为

$$C : y^2 = f(x), \tag{2.12}$$

其中 f 是 $\mathbb{F}_{p^2}$ 上的无平方因子多项式，当 $p > 5$ 时，$\deg(f) = 5$ 或 6。与椭圆曲线不同，曲线 C 在点加下不构成群。我们需要转而构造其 Jacobian $\mathrm{Jac}(C)$ [Mil86]，即关联于曲线 C 的阿贝尔群。注意，对于椭圆曲线 E，有 $\mathrm{Jac}(E) \simeq E$，因此 Jacobian 的构造在椭圆曲线情形中通常可以略去。$\mathrm{Jac}(C)$ 上的群律可用 Cantor 算法 [Can89] 计算。我们将零元记作 0_J，或当讨论一般的 PPAS A 时记作 0_A。此后，$\mathrm{Jac}(C)$ 总是指亏格-2 超椭圆曲线 C 的 Jacobian。

同源： PPAS 之间的同源 $\Phi : A_1 \to A_2$ 是一个满射，逐坐标由有理分式定义，且同时为群同态。根据 PPAS A_1 和 A_2 的类型，我们处理的同源 Φ 有四种类型：

- 同源 $\Phi : E_1 \times E_2 \to \mathrm{Jac}(C)$ 称为**粘合同源**（gluing isogeny），
- 同源 $\Phi : \mathrm{Jac}(C) \to E_1 \times E_2$ 称为**分裂同源**（splitting isogeny），
- 同源 $\Phi : \mathrm{Jac}(C) \to \mathrm{Jac}(C')$ 称为**一般同源**（generic isogeny），
- 否则，形如 $\Phi : E_1 \times E_2 \to E_3 \times E_4$ 的 $\Phi(P, Q) = (\phi_1(P), \phi_2(Q))$ 同源称为**对角同源**（diagonal isogeny），其中 $\phi_1 : E_1 \to E_3$ 且 $\phi_2 : E_2 \to E_4$。

与椭圆曲线同源类似，PPAS 之间的同源具有有限核

$$\ker(\Phi) = \{P \in A_1 \mid \Phi(P) = 0_{A_2}\}$$

并且在相差一个后复合同构的意义下由核唯一确定。若 Φ 的核 $\ker(\Phi)$ 由 A_1 中两个线性无关的 N 阶点 $P, Q \in A_1$ 生成，也即 $\ker(\Phi) \cong \mathbb{Z}/N\mathbb{Z} \times \mathbb{Z}/N\mathbb{Z}$，则称 Φ 为一个 (N, N)-同源。这里，**线性无关**（linearly independent）是指对所有整数 k 和 l，有 $[k]P + [l]Q = 0_{A_1}$ 当且仅当$N \mid k$ 且 $N \mid l$. 记 $\ker(\Phi) = \langle P \rangle \oplus \langle Q \rangle$。该同源 Φ 可由 P 和 Q 表示，且能借助推广的 Vélu 公式在 $O(N^2)$ 时间内从这两个点计算出来 [LR23]。

并非任意一对线性无关的 N-挠点 P, Q 都能作为 PPAS 之间同源的核生成元。P 和 Q 定义一个 PPAS 之间同源的充要条件是它们为 **迷向**（isotropic）的，即 P, Q 的 Weil 配对[2]平凡：$e_N(P, Q) = 1$. 若 N 的素因子分解为 $N = \prod \ell_i^{e_i}$，则直接计算 (N, N)-同源 Φ 需要 $O(N^2)$ 时间。更高效的做法是将 Φ 分解为

$$\Phi = \Phi_r \circ \cdots \circ \Phi_1,$$

其中各 Φ_i 为 (ℓ_i, ℓ_i)-同源，此时计算复杂度为 $O(\sum e_i \ell_i^2)$.

在 QIMEN-PIKE 的使用： 对于 QIMEN-PIKE，我们特化到 $(2^k, 2^k)$-同源的情形

$$\Phi : E_1 \times E_2 \longrightarrow E_3 \times E_4$$

其中 $E_1 \times E_2$ 和 $E_3 \times E_4$ 均为定义在 $\mathbb{F}_{p^2}$ 上的椭圆曲线积。我们将该同源分解为一条长度为 k（k 为某个整数）的 $(2, 2)$-同源链。在 QIMEN-PIKE 中，我们预期同源链的第一步 $\Phi_1 : E_1 \times E_2 \to A_1$ 为粘合同源，最后一步 $\Phi_k : A_{k-1} \to E_3 \times E_4$ 为分裂同源，而中间各步均为一般同源。

Remark 2.5.1. 在 *QIMEN-PIKE* 中，与其他基于同源的方案不同，所计算的 $(2, 2)$-同源的次数几乎等于可用的 2-挠点的阶 2^a。因此，我们以 2^a 表示可用的 2-挠点的阶数，且主要使用 $(2^{a-2}, 2^{a-2})$-同源。这与其它文献中的记法不同：其它文献使用 2^{a+2} 阶点，并称次数为 $(2^a, 2^a)$。

2.5.1 $(2, 2)$-同源的实现细节

Chapter E将给出计算 $(2, 2)$-同源的算法细节。实际中，我们使用 **2 级 theta 坐标**（theta coordinates of level 2）表示 PPAS 上的点，它可视为椭圆曲线上 x-坐标算术的高维推广（参见Section 2.2.2）。在同源密码学中，theta 坐标的使用已相当成熟。由于这些算法技术性较强，我们将细节推迟到Chapter E，这里仅概述主要算法。

- ThetaDBL：给定一个点 P 的 theta 坐标以及常量 `consts`，输出点 $[2]P$ 的 theta 坐标。
- GenericCodomainWith8Torsion：给定原曲面 A 上 8-挠点 T_1'', T_2'' 的 theta 坐标，该算法输出对偶 theta 零点 $(\alpha : \beta : \gamma : \delta)$、其逆 $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$，以及同源 $\Phi : A \to B$（其中 $\ker(\Phi) = \langle [4]T_1'' \rangle \oplus \langle [4]T_2'' \rangle$）的像 B 的 theta 零点 0_B。

[2] 与椭圆曲线类似，可以在所有 PPAS 上定义 Weil 配对。

- GenericCodomainWith4Torsion：给定原曲面 A 上一个 4-挠点 T_1' 的 theta 坐标（满足 $[2]T_1' \in \ker(\Phi)$），以及 A 的 theta 零点 0_A，该算法输出对偶 theta 零点 $(\alpha:\beta:\gamma:\delta)$、其逆 $(\alpha^{-1}:\beta^{-1}:\gamma^{-1}:\delta^{-1})$，以及 $\Phi: A \to B$ 的像 B 的 theta 零点 0_B。
- GenericCodomain：给定原曲面 A 的 theta 零点 0_A，计算 $(2,2)$-同源 $\Phi: A \to B$ 的对偶 theta 零点 $(\alpha:\beta:\gamma:\delta)$、其逆 $(\alpha^{-1}:\beta^{-1}:\gamma^{-1}:\delta^{-1})$，以及像 B 的 theta 零点 0_B。
- GenericEval：给定原曲面 A 上的一个点 P（以 theta 坐标表示）以及点 $(\alpha^{-1}:\beta^{-1}:\gamma^{-1}:\delta^{-1})$，输出 $\Phi(P)$ 的 theta 坐标。
- GluingCodomain：给定原积曲面上 8-挠点 T_1'', T_2'' 的 theta 坐标，该算法输出同源 $\Phi: E_1 \times E_2 \to A$（其中 $\ker(\Phi) = \langle [4]T_1'' \rangle \oplus \langle [4]T_2'' \rangle$）的像曲面 A 的对偶 theta 零点 $(\alpha:\beta:\gamma:0)$、其“逆”$(\alpha^{-1}:\beta^{-1}:\gamma^{-1}:0)$、theta 零点 0_B、theta 点 $\Phi(T_1'')$ 的对偶，以及一个基变换矩阵 N（用于求值复用）。
- GluingEval：给定一个点 $P \in E_1 \times E_2$、满足 $[4]T_1'' \in \ker(\Phi)$ 的 8-挠点 T_1''、A 上对偶 theta 点 $\Phi(T_1'')$，以及在GluingCodomain中计算的基变换矩阵 N，输出 $\Phi(P)$ 的 theta 坐标。
- GluingEvalSpecial：给定一个形如 $(P_1, 0)$ 或 $(0, P_2)$ 的点 $P \in E_1 \times E_2$、A 上对偶 theta 零点的“逆” $(\alpha^{-1}:\beta^{-1}:\gamma^{-1}:0)$，以及在GluingCodomain计算的基变换矩阵 N，输出 $\Phi(P)$ 的 theta 坐标。
- SplittingIsomorphism：给定曲面 $A \cong E_1 \times E_2$ 上的 theta 零点 0_A，计算一个同构，该同构将 0_A 映为与积 theta 结构关联的 theta 零点。

2.5.2 计算椭圆曲线积之间的 $(2,2)$-同源链

借助前一节的核心算法，可描述 $(2,2)$-同源链的计算过程。考虑椭圆曲线积之间的一个 $(2^a, 2^a)$-同源 $\Phi: E_1 \times E_2 \to E_3 \times E_4$。给定两个满足 $\ker(\Phi) = \langle [4]P, [4]Q \rangle$ 的点 P 和 Q，下面说明如何将 Φ 作为 $(2,2)$-同源链计算：

$$E_1 \times E_2 \xrightarrow{\Phi_1} A_1 \xrightarrow{\Phi_2} A_2 \cdots \xrightarrow{\Phi_{a-1}} A_{a-1} \xrightarrow{\Phi_a} E_3 \times E_4.$$

计算链中每个同源只需确定其像的 theta 零点。事实上，一旦知道了 theta 零点，即可对同源求值。按如下朴素方法可计算一条 $(2,2)$-同源链：

1. 对于 Φ_1：8-挠点 $[2^a]P$ 和 $[2^a]Q$ 分别是核生成元 $[2^{a+1}]P$ 和 $[2^{a+1}]Q$ 的二倍原像。利用这两个点，通过 GluingCodomain计算粘合同源 $\Phi_1: E_1 \times E_2 \to A_1$，其核为 $\ker(\Phi_1) = \langle [2^{a+1}]P, [2^{a+1}]Q \rangle$。随后，通过GluingEval计算 $\Phi_1(P)$ 和 $\Phi_1(Q)$。

2. 对于 $2 \le i \le a$ 的 Φ_i：8-挠点 $[2^{a-i}](\Phi_{i-1} \circ \cdots \circ \Phi_1(P))$ 和 $[2^{a-i}](\Phi_{i-1} \circ \cdots \circ \Phi_1(Q))$ 位于 $\ker(\Phi_i)$ 之上，用它们通过GenericCodomainWith8Torsion 计算一般同源 $\Phi_i: A_{i-1} \to A_i$。注意，最后一步得到的是 $A_a = E_3 \times E_4$ 的一个 theta 零点，而非这两条曲线本身。

3. 对于 $E_3 \times E_4$：用SplittingIsomorphism将从 Φ_a 获得的 $E_3 \times E_4$ 上的 theta 坐标系统变换为积 theta 坐标系统。这样就能将 $E_3 \times E_4$ 上的像点分别在分量 E_3 和 E_4 上表示为 $(X : Z)$-Montgomery 坐标。

取 P 和 Q 为 2^{a+2} 阶的点（满足 $\ker\Phi = \langle [4]P, [4]Q\rangle$），可避免昂贵的平方根计算：对于 $2 \le i \le a$ 中的每个 Φ_i（包括最后两步），只需利用 8-挠点 $[2^{a-i}](\Phi_{i-1} \circ \cdots \circ \Phi_1(P))$ 和 $[2^{a-i}](\Phi_{i-1} \circ \cdots \circ \Phi_1(Q))$ 即可通过 GenericCodomainWith8Torsion计算。该优化要求 2^{a+2}-挠点定义在 $\mathbb{F}_{p^2}$ 上，这并非总成立。在 QIMEN-PIKE 中，次数 a 的选择满足 $a + 2 \le e$，其中 2^e 是最大可用的 $2^\bullet$-挠，因此我们始终采用此方法。

策略： 上述计算 Φ 的方法称为**朴素策略**（naive strategy），并非最优。替代方案是**均衡策略**（balanced strategy）：将倍点过程中得到的中间点存储下来，并通过每个同源前推，从而将ThetaDBL倍点算法的执行次数降至拟线性量级 $O(a\log(a))$。[3]

2.5.3 同源链的实现细节

Chapter E给出完整计算 $(2^a, 2^a)$-同源的算法细节，该算法涵盖了Section 2.5.2中的所有步骤。由于始终假设 $a + 2 \le e$，这里仅给出一种方法：

- Isogeny22Chain：输入迷向点 $P, Q \in E_1 \times E_2$（阶为 2^{a+2}，且满足 $a + 2 \le e$），以及包含 $E_1 \times E_2$ 上点的数组 `pts`。输出一条 $(2, 2)$-同源链 $\Phi = \Phi_a \circ \cdots \circ \Phi_1$，满足 $\ker(\Phi) = \langle [4]P, [4]Q\rangle$，以及求值点 $\{\Phi(R) : R \in \texttt{pts}\}$。使用均衡策略计算。

2.6 四元数

2.6.1 大整数

QIMEN-PIKE 需要表示可变规模的大整数。整数的最大规模取决于系统参数，但难以估计，中间结果的规模尤其难以界定。因此，建议使用 GMP 等动态多精度整数库。本技术规范的后续版本可能确定可表示整数的精确上界，届时即可采用固定精度的大整数。

QIMEN-PIKE 需要对大整数执行的操作，在大多数大整数库中均有现成实现，故此处仅列出，不做详细说明：

- 整数的基本算术（加法、乘法等）；
- 从区间中均匀采样整数；
- 近似和精确的整数平方根；
- 使用 Miller–Rabin 检验的伪素性测试；
- 扩展最大公约数 XGCD：给定 (a, b)，求整数 (g, u, v) 满足 $ua + bv = g = \gcd(a, b) > 0$。若 $a \ne 0$ 且 $b \ne 0$，则要求 $1 \le au \le |ab|/g$ 且 $-|ab|/g < bv \le 0$。注意，GMP 的 XGCD 算法并不强制输出 $u \ne 0$，而 QIMEN-PIKE 需要这一保证；

[3] 我们不使用 SIDH/SIKE [JD11, § 4.2.2] 中的**最优策略**（optimal strategies），原因是它们仅带来微小的效率提升，却需要适中的内存来存储（预计算的）策略。

- 整数模运算；
- Legendre 符号；
- 整数的 2 进赋值；
- 模素数平方根。

除了模平方根（其伪代码见ModularSQRT）以外，其余算法均在 GMP 中有现成实现；GMP 也正是 QIMEN-PIKE 参考实现所采用的大整数库。

Algorithm 3 ModularSQRT(n, m)

Input: 奇素数 m 和整数 n，满足 n 是模 m 的平方剩余
Output: n 的模平方根 x，即一个整数 x 满足 $x^2 \equiv n \pmod m$

if $n \equiv 0 \pmod m$ **then return** 0
if $m \equiv 3 \pmod 4$ **then return** $n^{(m+1)/4} \bmod m$
if $m \equiv 5 \pmod 8$ **then**
    if $n^{(m-1)/4} \equiv 1 \pmod m$ **then return** $n^{(m+3)/8} \bmod m$
    elsereturn $2n(4n)^{(m-5)/8} \bmod m$
$w \leftarrow 2$ 且 $e \leftarrow 2$ 进赋值$(m-1)$
$q \leftarrow (m-1)/2^e$
while w 是模 m 的平方剩余 **do**
    $w \leftarrow w + 1$
$z \leftarrow w^q \bmod m$ 且 $r \leftarrow e$ 且 $y \leftarrow n^q \bmod m$
$x \leftarrow n^{(q+1)/2} \bmod m$ 且 $f \leftarrow 2^{e-2}$
for i 从 0 到 $e-1$ **do**
    $b \leftarrow y^f \bmod m$
    if $b = m - 1$ **then**
        $x \leftarrow xz \bmod m$
        $y \leftarrow yz^2 \bmod m$
        $z \leftarrow z^2 \bmod m$
        $f \leftarrow f/2$
    return x

2.6.2 四元数代数的基本定义

设 $p \equiv 3 \bmod 4$ 为一个素数。**四元数代数**（quaternion algebra）$\mathcal{B}_{p,\infty}$ 是由 $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ 张成的 4 维 $\mathbb{Q}$ 向量空间，其基满足

$$\mathbf{i}^2 = -1, \qquad \mathbf{j}^2 = -p, \qquad \mathbf{ij} = -\mathbf{ji} = \mathbf{k}. \tag{2.13}$$

该向量空间上的 $\mathbb{Q}$-代数结构如下：

- $x \cdot (a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}) := (xa) + (xb)\mathbf{i} + (xc)\mathbf{j} + (xd)\mathbf{k}$，其中 $x, a, b, c, d \in \mathbb{Q}$；

- $\mathcal{B}_{p,\infty}$ 中两个元素的乘法 $(a+b\mathbf{i}+c\mathbf{j}+d\mathbf{k})(a'+b'\mathbf{i}+c'\mathbf{j}+d'\mathbf{k})$ 按分配律展开，再利用Eq. (2.13)即得。

$\mathcal{B}_{p,\infty}$ 的元素 α 用一个有理 4-元组 $(a,b,c,d)\in\mathbb{Q}^4$ 表示，代表

$$a+b\mathbf{i}+c\mathbf{j}+d\mathbf{k}.$$

我们混用 α 既指该四元数元素本身，也指其向量表示 $(a,b,c,d)^t\in\mathbb{Q}^4$。实际实现中用 $\mathbb{Z}^5$ 中的 5-元组存储，约去公分母即得规范表示。

$\mathcal{B}_{p,\infty}$ 中元素的**加法**和**乘法**如上所述。我们进一步定义 $\mathcal{B}_{p,\infty}$ 上的其他运算如下：对 $\alpha=a+b\mathbf{i}+c\mathbf{j}+d\mathbf{k}\in\mathcal{B}_{p,\infty}$，

共轭： α 的共轭 $\bar{\alpha}$ 定义为 $\bar{\alpha}:=a-b\mathbf{i}-c\mathbf{j}-d\mathbf{k}$。

约化迹： α 的约化迹（reduced trace）定义为 $\mathrm{tr}(\alpha):=\alpha+\bar{\alpha}=2a$。

约化范数： α 的约化范数（reduced norm）定义为 $\mathrm{nrd}(\alpha):=\alpha\bar{\alpha}=a^2+b^2+p(c^2+d^2)$。

序（order）是 $\mathcal{B}_{p,\infty}$ 中的格，同时也是子环（即对乘法封闭）。序 $\mathcal{O}$ 中的元素称为整（integral）的，因为其约化迹与约化范数均落在 $\mathbb{Z}$ 中。若一个序不被任何更大的序所包含，则称其为极大序（maximal order）。QIMEN-PIKE 使用的四元数代数含有一个极大序，基为 $(1,\mathbf{i},\frac{\mathbf{i}+\mathbf{j}}{2},\frac{1+\mathbf{k}}{2})$；本节余下部分将该序记为 $\mathcal{O}_0$。$\mathcal{O}_0$ 包含一个（非极大的）子序，基为 $(1,\mathbf{i},\mathbf{j},\mathbf{k})$。

给定一条 $\mathbb{F}_{p^2}$ 上的超奇异椭圆曲线 E，E 的所有自同态的集合称为 E 的**自同态环**（endomorphism ring），记作 $\mathrm{End}(E)$。$\mathrm{End}(E)$ 同构于四元数代数 $\mathcal{B}_{p,\infty}$ 中的一个极大序 $\mathcal{O}$。固定同构 $\mathcal{O}\simeq\mathrm{End}(E)$ 后，元素 $\alpha\in\mathcal{O}$ 对应于 E 的一个自同态。为简便起见，对 $P\in E$，我们用 $\alpha(P)$ 表示 α（在此同构下）在 $P\in E$ 处求值的像；用 $\ker\alpha$ 表示该同态的核。

2.6.3 范数方程

在 RandFixNormIdeal 中，一个重要的子程序是寻找具有给定范数的四元数元素；为此我们引入算法GeneralizedRepresentInteger。本节通篇使用的四元数序 $\mathcal{O}_0$ 定义为：

$$\mathcal{O}_0:=\mathbb{Z}+\mathbf{i}\mathbb{Z}+\frac{\mathbf{i}+\mathbf{j}}{2}\mathbb{Z}+\frac{1+\mathbf{k}}{2}\mathbb{Z},$$

其中 p 为满足 $p\equiv 3 \bmod 4$ 的素数。

2.6.3.1 Cornacchia 算法

Cornacchia 算法 [Cor08] 可高效求解方程 $x^2+qy^2=m$，其中 q,m 为正整数。该算法需要 m 的因子分解信息足够多。在下述程序中，我们只需要 $q=1$ 的情形，即把整数表为两平方和。对于素数 m，Cornacchia采用 [MN90] 的算法。此外，还提供了一个面向合数输入的封装CornacchiaGeneral：它先剥离预先算好的固定小素数列表中各素数的幂，再尽力对剩下的辅因子调用Cornacchia求解。

Algorithm 4 Cornacchia(m)

Input: 一个素数 m
Output: $(x, y) \in \mathbb{Z}^2$ 满足 $x^2 + y^2 = m$，若未找到解则返回 $\perp$

if $m = 2$ **then**
  return $(1, 1)$
if $m \not\equiv 1 \pmod 4$ **then** ▷ -1 不是模 m 的平方剩余
  return $\perp$
$r \leftarrow$ ModularSQRT$(-1, m)$
$s \leftarrow m$
while $r^2 \geq m$ **do**
  $(r, s) \leftarrow (s \bmod r, r)$
$x \leftarrow r$
$Y \leftarrow m - r^2$
if Y 在 $\mathbb{Z}$ 中不是完全平方 **then**
  return $\perp$
$y \leftarrow \sqrt{Y}$
return (x, y)

以下算法使用一个预先算好的小素数列表

$$\mathcal{P} = (p_0, \ldots, p_{s-1})$$

对 QIMEN-PIKE 而言，该列表包含 2 以及前 100 个满足 $\ell \equiv 1 \pmod 4$ 的奇素数，共 101 项。该列表由参数集确定，在 QIMEN-PIKE 中记为 `cornacchia_prime_list`，即 $\mathcal{P}$ 取作 `cornacchia_prime_list`。在CornacchiaGeneral内部，变量 z 是一个 Gauss 整数 $z = u + v\sqrt{-1} \in \mathbb{Z}[\sqrt{-1}]$；相应地，$\mathrm{Re}(z) = u$ 和 $\mathrm{Im}(z) = v$ 分别表示它的两个整数系数。

Algorithm 5 CornacchiaGeneral$(m, \mathcal{P})$

Input: 正整数 m 和预计算列表 $\mathcal{P} = (p_0, \ldots, p_{s-1})$
Output: $(x, y) \in \mathbb{Z}^2$ 满足 $x^2 + y^2 = m$，若未找到解则返回 $\perp$

$m' \leftarrow m$ 且对每个 $p_i \in \mathcal{P}$ 设 $e_i \leftarrow 0$
for i 从 0 到 $s-1$ **do**
  while $p_i \mid m'$ **do**
    $m' \leftarrow m'/p_i$
    $e_i \leftarrow e_i + 1$
if $m' = 1$ **then**
  $z \leftarrow 1$
else
  if PrimalityTest$(m') =$ False 或 $m' \not\equiv 1 \pmod 4$ **then**
    return $\perp$
  $(x, y) \leftarrow$ Cornacchia(m')
  if $(x, y) = \perp$ **then**
    return $\perp$
  $z \leftarrow x + y\sqrt{-1}$
for i 从 0 到 $s-1$ **do**
  if $e_i > 0$ **then**
    $(a_i, b_i) \leftarrow$ Cornacchia(p_i)
    if $(a_i, b_i) = \perp$ **then**
      return $\perp$
    $z \leftarrow z(a_i + b_i\sqrt{-1})^{e_i}$
return $(\mathrm{Re}(z), \mathrm{Im}(z))$

2.6.3.2 用特殊极序表示整数

给定一个固定的整数 M（通常需大于 p），目标是找到 $x, y, z, t \in \mathbb{Z}$，使得四元数元素 $\gamma := x + y\mathbf{i} + z\frac{\mathbf{i}+\mathbf{j}}{2} + t\frac{1+\mathbf{k}}{2}$ 的范数等于 M。

观察可知，求解方程 $\mathrm{nrd}(\gamma) = (x + t/2)^2 + (y + z/2)^2 + p((t/2)^2 + (z/2)^2) = M$ 等价于求解

$$x^2 + y^2 + p(z^2 + t^2) = 4M \tag{2.14}$$

的整数解 (x, y, z, t)，且满足 $x \equiv t \bmod 2$ 和 $y \equiv z \bmod 2$。此时该元素可写成 $\gamma = \frac{x-t}{2} + \frac{y-z}{2}\mathbf{i} + z\frac{\mathbf{i}+\mathbf{j}}{2} + t\frac{1+\mathbf{k}}{2} = \frac{x+y\mathbf{i}+z\mathbf{j}+t\mathbf{k}}{2}$。

因此，找到Eq. (2.14)的一个合适解即可。算法GeneralizedRepresentInteger的执行过程如下：首先在算法规定的适当范围内采样 z, t，然后计算 $M' := 4M - p(z^2 + t^2)$。随后用预计算列表 `cornacchia_prime_list` 调用CornacchiaGeneral求解二元二次方程 $x^2 + y^2 = M'$。若该调用返回 $\perp$，则丢弃当前候选，继续采样新的 z, t 值。

Algorithm 6 GeneralizedRepresentInteger(M)

Input: 奇数 $M \in \mathbb{Z}$ 满足 $M > p$
Output: $\gamma \in \mathcal{O}_0$ 满足 $\mathrm{nrd}(\gamma)$ 等于 M，或引发异常

counter $\leftarrow 0$，bound $\leftarrow \left\lceil \frac{4M}{p} \right\rceil$，且 found $\leftarrow$ false
while (found $=$ false) 且 (counter $<$ bound) **do**
    counter $\leftarrow$ counter $+ 1$
    从 $\left[1, \ldots, \left\lfloor \sqrt{\frac{4M}{p}} \right\rfloor\right]$ 中均匀采样 z
    从 $[-m', \ldots, m']$ 中均匀采样 t，其中 $m' = \left\lfloor \sqrt{\frac{4M - pz^2}{p}} \right\rfloor$
    设 $M' \leftarrow 4M - p(z^2 + t^2)$
    if $M' > 0$ **then**
        $(x, y) \leftarrow$ CornacchiaGeneral(M', cornacchia_prime_list)
        if $(x, y) \neq \perp$ **then**
            found $\leftarrow$ true
            if $x \neq t \bmod 2$ 或 $y \neq z \bmod 2$ **then**
                $(x, y) \leftarrow (y, x)$
            $\gamma \leftarrow \frac{x + y\mathbf{i} + z\mathbf{j} + t\mathbf{k}}{2}$
if found **then**
    return γ
else
    raise Exception("GeneralizedRepresentInteger failed")

CHAPTER 3

协议

3.1 协议参数

本节规定 QIMEN-PIKE 的协议参数。QIMEN-PIKE 为 **P**airing-assisted, **I**sogeny-based **K**ey **E**ncryption（配对辅助的同源密钥封装）方案。关于如何高效表示交换数据，请参见Chapter 4。

与 POKÉ [BM25] 类似，QIMEN-PIKE 遵循 ElGamal 范式：加密时，先派生一个共享秘密，再将其与消息组合，得到密文。协议概览见Figure 3.1。QIMEN-PIKE.PKE 和 QIMEN-PIKE.KEM 的完整规范将在后续各节中给出。

3.1.1 参数需求

QIMEN-PIKE 算法需要若干参数和预计算值。所有算法均隐式地以一个参数集作为输入。关于这些参数的具体数值，请参见Chapter 5。下面列出这些参数，并在部分情况下说明其如何由其他参数计算得出。

3.1.2 方案参数

参数和预计算值包括：

- 内部安全参数 λ，设为 $2\lambda_q$。
- 正整数 a, C_1, C_2, D，满足 $p+1 = 2^a C_1 D$、$C_2 \mid p-1$ 以及 $\gcd(C_1, C_2) = 1$。其具体值在Section 5.1中定义。此外，定义 $C = C_1 C_2$。
- E_0 是方程为 $y^2 = x^3 + x$ 的曲线。
- (P_0, Q_0) 是 $E_0[2^a]$ 的一组基。
- (R_0, S_0) 是 $E_0[C]$ 的一组基。
- (X_0, Y_0) 是 $E_0[D]$ 的一组基。
- w_0 表示 Tate 配对值 $t_D(X_0, Y_0)$。
- $\mathsf{H} : \{0,1\}^\star \to \{0,1\}^{2\lambda}$ 是一个哈希函数（hash function）。为进行域分离（domain separation），密钥派生的查询写为 $\mathsf{H}(\mathsf{KDF} \mid \cdot)$。

3.1.3 协议框架

下面给出协议框架，以便直观理解上文定义的部分参数。

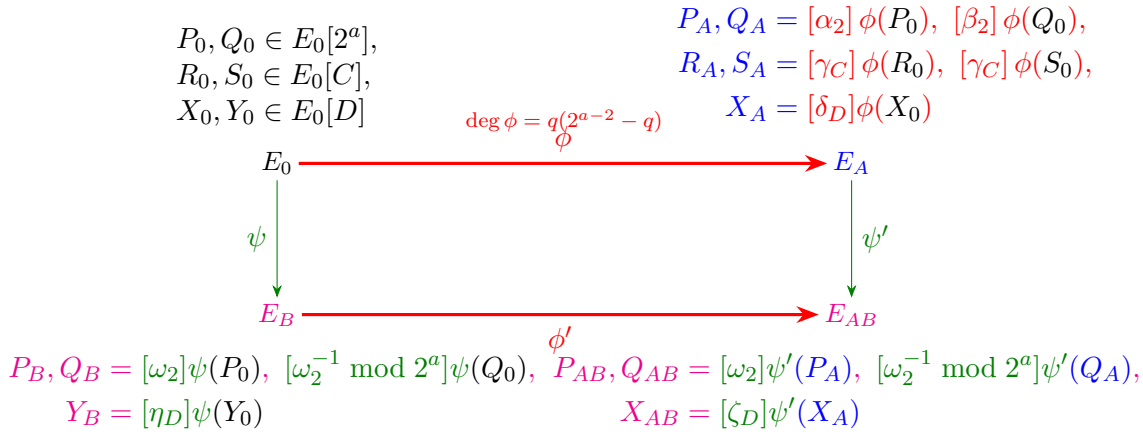

Figure 3.1: QIMEN-PIKE 框架图。黑色：公开参数；红色：私钥；蓝色：公钥；深绿色：加密 nonce；品红色：密文（ciphertext）。

3.2 PIKE.PKE 规范

本节以三个算法QIMEN-PIKE.PKE.KeyGen、QIMEN-PIKE.PKE.Encrypt和QIMEN-PIKE.PKE.Decrypt的集合，给出 IND-CPA 安全的公钥加密（public-key encryption）方案 QIMEN-PIKE.PKE。

3.2.1 密钥生成

密钥生成算法分为两个子节。Section 3.2.1.2描述各子程序，随后在Section 3.2.1.3中介绍密钥生成算法。

3.2.1.1 预计算数据

密钥生成子程序使用Section 2.6.2中描述的固定同构 $\mathcal{O}_0 \simeq \mathrm{End}(E_0)$。回顾

$$\mathcal{O}_0 = \mathbb{Z} \oplus \mathbf{i}\mathbb{Z} \oplus \frac{\mathbf{i}+\mathbf{j}}{2}\mathbb{Z} \oplus \frac{1+\mathbf{k}}{2}\mathbb{Z}$$

具有整基 $\left(1, \mathbf{i}, \frac{\mathbf{i}+\mathbf{j}}{2}, \frac{1+\mathbf{k}}{2}\right)$。对 QIMEN-PIKE，以下数据已预计算：

- $E_0[C]$ 的一组基 (R_0, S_0)，以及矩阵 $M_1, \ldots, M_4 \in \mathsf{M}_2(\mathbb{Z}/C\mathbb{Z})$，它们表示 $1, \mathbf{i}, \frac{\mathbf{i}+\mathbf{j}}{2}, \frac{1+\mathbf{k}}{2}$ 在 $E_0[C]$ 上关于 (R_0, S_0) 的作用。

给定任意**自同态**（endomorphism）$\alpha \in \mathcal{O}_0$，将其在上述整基下的坐标模 C 约化后，EndomorphismToKernel便可利用这些矩阵求值 α 在 $E_0[C]$ 上的作用。

3.2.1.2 子程序

本子节规定密钥生成所需的辅助过程。这些过程描述密钥生成中同源的计算及求值方法，但私钥和公钥的定义不在此列。

主过程QIMEN-PIKE.CoreKeyGenIso首 先计算一个次数为 $q(2^{a-2}-q)$ 的同源 $\varphi : E_0 \to E_A$，然后求值该同源在 $E_0[2^a]$ 的基上以及 $E_0[CD]$ 的指定点上的像。该过程遵循 POKÉ [BM25] 中启发式非光滑同源 $\varphi : E_0 \to E_A$ 的生成方法，并包含了实现级的优化。首先，调用GeneralizedRepresentInteger采样一个次数为 $q(2^{a-2}-q)C$ 的自同态 θ。然后提取其次数为 $q(2^{a-2}-q)$ 的分量：先用EndomorphismToKernel计算同源 $\psi : E_0 \to E_A$——其核为 $\ker(\overline{\theta}) \cap E_0[C]$——再令 φ 等于 $[1/C] \circ \psi \circ \theta$。

Algorithm 7 QIMEN-PIKE.CoreKeyGenIso(λ)

Input: 内部安全参数 λ

Output: $(q, E_A, (P_A, Q_A), (R_A, S_A), X_A)$，其中 $\varphi : E_0 \to E_A$ 是某个随机同源，其次数为 $q(2^{a-2}-q)$（q 为随机值），且

$$(P_A, Q_A, R_A, S_A, X_A) = (\varphi(P_0), \varphi(Q_0), \varphi(R_0), \varphi(S_0), \varphi(X_0))$$

repeat
  $q \xleftarrow{\$} \{0, \ldots, 2^{a-2}-1\}$
until $\gcd(q(2^{a-2}-q),\ 2 \cdot C \cdot D) = 1$
$\theta \leftarrow$ GeneralizedRepresentInteger$(q(2^{a-2}-q)C)$
$(P_0^\tau, Q_0^\tau) \leftarrow (\theta(P_0), \theta(Q_0))$
$K \leftarrow$ EndomorphismToKernel$(\overline{\theta})$
$\widetilde{E_A}, (\widetilde{P_A}, \widetilde{Q_A}) \leftarrow$ OddIsogenyChain$(E_0, K, C, (P_0^\theta, Q_0^\theta))$
$s \leftarrow Cq \bmod 2^a$
$\_, ((\widetilde{P_A}, \widetilde{P'_A}), (\widetilde{Q_A}, \widetilde{Q'_A}), (\widetilde{R_A}, \_), (\widetilde{S_A}, \_), (\widetilde{X_A}, \_)) \leftarrow$
  Isogeny22Chain$(([s]P_0, \widetilde{P_A}), ([s]Q_0, \widetilde{Q_A}), [(P_0, 0), (Q_0, 0), (R_0, 0), (S_0, 0), (X_0, 0)])$
$(E_A, \_), ((P_A, \_), (Q_A, \_), (R_A, \_), (S_A, \_), (X_A, \_)) \leftarrow$
  Isogeny22Chain$((\widetilde{P_A}, \widetilde{P'_A}), (\widetilde{Q_A}, \widetilde{Q'_A}), [(\widetilde{P_A}, 0), (\widetilde{Q_A}, 0), (\widetilde{R_A}, 0), (\widetilde{S_A}, 0), (\widetilde{X_A}, 0)])$
return $(q, E_A, (P_A, Q_A), (R_A, S_A), X_A)$

EndomorphismToKernel EndomorphismToKernel给出了从自同态 $\alpha \in \mathrm{End}(E_0)$ 中提取其限制在 C 挠子群上的核的详细过程。给定满足 $C \mid \deg(\alpha)$ 的自同态 α，该算法求值 α 在 $E_0[C]$ 的固定 C 挠基 (R_0, S_0) 上的作用。

将 α 用 $\mathcal{O}_0$ 的固定整基表示，即

$$\alpha = x_1 + x_2\mathbf{i} + x_3\frac{\mathbf{i}+\mathbf{j}}{2} + x_4\frac{1+\mathbf{k}}{2}, \quad x_i \in \mathbb{Z}.$$

利用Section 3.2.1.1中预计算的矩阵 $M_1, \ldots, M_4 \in \mathsf{M}_2(\mathbb{Z}/C\mathbb{Z})$，可以显式恢复作用矩阵 $\mathbf{M}_\alpha \in \mathsf{M}_2(\mathbb{Z}/C\mathbb{Z})$ 为

$$\mathbf{M}_\alpha = x_1M_1 + x_2M_2 + x_3M_3 + x_4M_4.$$

然后计算向量 $[v_1, v_2]^T \in \ker \mathbf{M}_\alpha$，它直接给出对应次数为 C 的同源的核生成元 $K = [v_1]R_0 + [v_2]S_0$。

Algorithm 8 EndomorphismToKernel(α)

Input: 一个自同态 $\alpha \in \mathcal{O}_0 \simeq \mathrm{End}(E_0)$，满足 $C \mid \deg(\alpha)$
Output: 子群 $\{K \in E_0[C] \mid \alpha(K) = 0_{E_0}\}$ 的核生成元 K

1: 令 $\mathbf{A} = [1, \mathbf{i}, \frac{\mathbf{i}+\mathbf{j}}{2}, \frac{1+\mathbf{k}}{2}]$ 表示 $\mathcal{O}_0$ 的固定整基
2: 计算 $\mathbf{v}_\alpha = [x_1, x_2, x_3, x_4]^T \in \mathbb{Z}^4$，使得 $\mathbf{A}\mathbf{v}_\alpha = \alpha$
3: $\mathbf{M}_\alpha = x_1M_1 + x_2M_2 + x_3M_3 + x_4M_4$
4: 计算 $[v_1, v_2]^T \in \ker(\mathbf{M}_\alpha)$
5: $K \leftarrow [v_1]R_0 + [v_2]S_0$
6: **return** K

3.2.1.3 密钥生成

QIMEN-PIKE.PKE.KeyGen规定的密钥生成过程包括：采样接收方秘密标量，调用上述子程序计算所需的同源像，施加公钥掩蔽，最后输出密钥对。具体构成如下：

- 保存的私钥为 $\mathsf{sk} = (q, \alpha_2, \beta_2, \delta_D)$；
- 公钥为 $\mathsf{pk} = (E_A, P_A, Q_A, R_A, S_A, X_A)$.

标量 γ_C（在以下伪代码中记作 γ_1, γ_2）在密钥生成期间采样，仅用于掩蔽公开发布的 C 挠像。该值不作为长期私钥的一部分存储。

3.2.2 加密

加密过程在QIMEN-PIKE.PKE.Encrypt中规定。它计算以下同源

$$\psi : E_0 \to E_B \qquad 和 \qquad \psi' : E_A \to E_{AB},$$

满足

$$\ker(\psi) = \langle R_0 + [r_3]S_0 \rangle \qquad 和 \qquad \ker(\psi') = \langle R_A + [r_3]S_A \rangle,$$

其中 $r_3 \in \mathbb{Z}/C\mathbb{Z}$ 为某个值。

Algorithm 9 QIMEN-PIKE.PKE.KeyGen(λ)

Input: 内部安全参数 λ
Output: 私钥和公钥 $(\mathsf{sk}, \mathsf{pk})$

$(q, E_A, (P'_A, Q'_A), (R'_A, S'_A), X'_A) \leftarrow$ QIMEN-PIKE.CoreKeyGenIso(pp)
$\alpha_2, \beta_2 \xleftarrow{\$} (\mathbb{Z}/2^{a-2}\mathbb{Z})^\times$, $\gamma \xleftarrow{\$} (\mathbb{Z}/C\mathbb{Z})^\times$, $\delta_D \xleftarrow{\$} (\mathbb{Z}/D\mathbb{Z})^\times$
$P_A, Q_A \leftarrow [\alpha_2]P'_A, [\beta_2]Q'_A$
$R_A, S_A \leftarrow [\gamma]R'_A, [\gamma]S'_A$
$X_A \leftarrow [\delta_D]X'_A$
$\mathsf{sk} \leftarrow (q, \alpha_2, \beta_2, \delta_D)$
$\mathsf{pk} \leftarrow (E_A, (P_A, Q_A), (R_A, S_A), X_A)$
return $(\mathsf{sk}, \mathsf{pk})$

Algorithm 10 QIMEN-PIKE.PKE.Encrypt($\mathsf{pk}, \mathsf{msg}$)

Input: 公钥 $\mathsf{pk} = (E_A, P_A, Q_A, R_A, S_A, X_A)$ 和消息 $\mathsf{msg} \in \{0,1\}^{2\lambda}$
Output: 密文 ct

$r \xleftarrow{\$} \mathbb{Z}/C\mathbb{Z}$, $\omega_2 \xleftarrow{\$} (\mathbb{Z}/2^a\mathbb{Z})^\times$, $\eta_D, \zeta_D \xleftarrow{\$} (\mathbb{Z}/D\mathbb{Z})^\times$
$w' \leftarrow w_0^{\eta_D \zeta_D}$
$\mathsf{pad} \leftarrow \mathsf{H}(\mathsf{KDF} \| \mathrm{tr}(w'))$
$E_B, (P'_B, Q'_B, Y'_B) \leftarrow$ OddIsogenyChain$(E_0, R_0 + [r]S_0, C, (P_0, Q_0, Y_0))$
$E_{AB}, (P'_{AB}, Q'_{AB}, X'_{AB}) \leftarrow$ OddIsogenyChain$(E_A, R_A + [r]S_A, C, (P_A, Q_A, X_A))$
$P_B, Q_B, Y_B \leftarrow [\omega_2]P'_B,\ [\omega_2^{-1} \bmod 2^a]Q'_B,\ [\eta_D]Y'_B$
$P_{AB}, Q_{AB}, X_{AB} \leftarrow [\omega_2]P'_{AB},\ [\omega_2^{-1} \bmod 2^a]Q'_{AB},\ [\zeta_D]X'_{AB}$
$c \leftarrow \mathsf{pad} \oplus \mathsf{msg}$
$\mathsf{ct} \leftarrow (E_B,\ P_B,\ Q_B,\ Y_B,\ E_{AB},\ P_{AB},\ Q_{AB},\ X_{AB},\ c)$
return ct

然后，加密过程计算这些同源在公开挠数据上的掩蔽求值结果，即

$$P_B, Q_B, Y_B \in E_B(\overline{\mathbb{F}}_p) \qquad \text{和} \qquad P_{AB}, Q_{AB}, X_{AB} \in E_{AB}(\overline{\mathbb{F}}_p),$$

如Figure 3.1所示。

密文由像曲线 E_B 和 E_{AB}、上述六个掩蔽点像以及掩蔽消息 $\mathsf{pad} \oplus \mathsf{msg}$ 组成。填充值 pad 由掩蔽标量 η_D 和 ζ_D 确定的共享 D 挠值派生而来。

在上述步骤 3 中，填充值 pad 取自 $w_0^{\eta_D\zeta_D}$ 的迹，而不是直接使用 $w_0^{\eta_D\zeta_D}$ 本身。这一选择与密文编码有关。若仅保留所传输挠点的 x 坐标，则解密时恢复出的配对值在正负逆意义上存在歧义：其具体值可能为其本身，也可能为其逆。取迹可消除这一歧义，因为 $\mathrm{tr}(w_0^{\eta_D\zeta_D}) = \mathrm{tr}(w_0^{-\eta_D\zeta_D}) = w_0^{\eta_D\zeta_D} + w_0^{-\eta_D\zeta_D}$. 因此，密钥派生步骤中使用的值落在 $\mathbb{F}_p$ 而非 $\mathbb{F}_{p^2}$ 中。

3.2.3 解密

解密过程需要推算出与QIMEN-PIKE.PKE.Encrypt步骤 3相同的填充值 pad。为此，接收方基于密文点 $Y_B \in E_B[D]$ 和 $X_{AB} \in E_{AB}[D]$，结合私钥 $(q, \alpha_2, \beta_2, \delta_D)$，恢复相应的配对值。

更具体地说，解密首先从 2^a 挠数据 $(P_B, Q_B) \in E_B[2^a]$ 和 $(P_{AB}, Q_{AB}) \in E_{AB}[2^a]$ 重建一个二维同源。将该同源作用于 Y_B 和 X_{AB}，得到公共中间曲线上的两个点，在 QIMEN-PIKE.PKE.Decrypt中记作 Y_M 和 X_M。其配对给出加密所用共享值的某已知幂次。该指数与期望指数相差因子 $q(2^{a-2} - q)C\delta_D$，而接收方知道该因子。因此，取配对输出的 t 次幂——其中 t 为该因子模 D 的逆——即可恢复同一共享值。

在 QIMEN-PIKE 中，共享值在解密过程恢复的中间曲线上计算。这避免了在 E_{AB} 上显式重建 $\phi(Y_B)$，并与上述密钥生成和加密的格式保持一致。完整的解密过程见QIMEN-PIKE.PKE.Decrypt。

Algorithm 11 QIMEN-PIKE.PKE.Decrypt(sk, ct)

Input: 私钥 $\mathsf{sk} = (q, \alpha_2, \beta_2, \delta_D)$ 和密文 $\mathsf{ct} = (E_B, P_B, Q_B, Y_B, E_{AB}, P_{AB}, Q_{AB}, X_{AB}, c)$
Output: 消息 msg

$\widetilde{P_B} \leftarrow [-q]P_B, \widetilde{Q_B} \leftarrow [-q]Q_B$
$\widetilde{P_{AB}} \leftarrow [\alpha_2^{-1} \mod 2^a]P_{AB}, \widetilde{Q_{AB}} \leftarrow [\beta_2^{-1} \mod 2^a]Q_{AB}$
$E_M, \_, (Y_M, \_), (X_M, \_)$
$\quad \leftarrow$ Isogeny22Chain$((\widetilde{P_B}, \widetilde{P_{AB}}), (\widetilde{Q_B}, \widetilde{Q_{AB}}), [(Y_B, 0_{E_B}), (0_{E_{AB}}, X_{AB})])$
$t \leftarrow \left(q(2^{a-2} - q)C\delta_D\right)^{-1} \bmod D$
$w \leftarrow$ TateOdd$(E_M, D, X_M, Y_M, X_M - Y_M)$
$w' \leftarrow w^t$
$\widetilde{\mathsf{pad}} \leftarrow \mathsf{H}(\mathsf{KDF} \| \mathrm{tr}(w'))$
return $\widetilde{\mathsf{pad}} \oplus c$

3.3 PIKE.KEM 规范

以 IND-CPA 安全的公钥加密方案 QIMEN-PIKE.PKE 为基础，通过 **Fujisaki–Okamoto 转换**（Fujisaki–Okamoto transform），构造出 IND-CCA2 安全的**密钥封装机制**（key encapsulation mechanism）QIMEN-PIKE.KEM。QIMEN-PIKE.KEM 由以下三个算法组成：QIMEN-PIKE.KeyGen、QIMEN-PIKE.Encaps和 QIMEN-PIKE.Decaps。

KEM 私钥包含以下四个部分：底层 PKE 私钥、公钥、公钥的哈希值，以及用于**隐式拒绝**（implicit rejection）的后备秘密。

3.3.1 KEM 层的辅助函数

KEM 层使用以下域分离函数。

Algorithm 12 QIMEN-PIKE.Enc.Det($\mathsf{pk}, \mathsf{msg}, \rho$)

Input: 公钥 pk、消息 $\mathsf{msg} \in \{0,1\}^{2\lambda}$ 和种子 $\rho \in \{0,1\}^{\mu}$
Output: 密文 ct

1: $(r_3, \omega_2, \eta_D, \zeta_D) \leftarrow \mathsf{DeriveCoins}(\rho)$
2: 运行 QIMEN-PIKE.PKE.Encrypt，以其替换步骤 1 中的随机采样
3: **return** 所得密文 ct

Algorithm 13 QIMEN-PIKE.KeyGen(λ)

Input: 内部安全参数 λ
Output: 公钥 pk 和私钥 sk

1: $z \leftarrow \{0,1\}^{2\lambda}$
2: $(\mathsf{pk}, \mathsf{sk}_0) \leftarrow$ QIMEN-PIKE.PKE.KeyGen(λ)
3: $h_{\mathsf{pk}} \leftarrow \mathsf{H}_{\mathsf{pk}}(\mathsf{pk})$
4: $\mathsf{sk} \leftarrow (\mathsf{sk}_0 \| \mathsf{pk} \| h_{\mathsf{pk}} \| z)$
5: **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{H}_{\mathsf{pk}} : \{0,1\}^{\star} \to \{0,1\}^{2\lambda}$ 公钥的哈希值。在封装中使用，并作为 KEM 私钥的一部分存储。

$\mathsf{H}_{\mathsf{ct}} : \{0,1\}^{\star} \to \{0,1\}^{2\lambda}$ 密文的哈希值。它将最终共享秘密绑定到完整的密文记录（transcript）。

$\mathsf{G} : \{0,1\}^{\star} \to \{0,1\}^{2\lambda} \times \{0,1\}^{\mu}$ 哈希扩展函数。输入 $m \| h_{\mathsf{pk}}$，输出预密钥 $\bar{K} \in \{0,1\}^{2\lambda}$ 和用于确定性封装的种子 $\rho \in \{0,1\}^{\mu}$。

$\mathsf{KDF} : \{0,1\}^{\star} \to \{0,1\}^{N_{\mathsf{ss}}}$ 密钥派生函数（key derivation function），用于派生最终的共享秘密。

$\mathsf{DeriveCoins} : \{0,1\}^{\mu} \to \mathbb{Z}/C\mathbb{Z} \times (\mathbb{Z}/2^{a}\mathbb{Z})^{\times} \times (\mathbb{Z}/D\mathbb{Z})^{\times} \times (\mathbb{Z}/D\mathbb{Z})^{\times}$ 确定性解析函数，将 ρ 扩展并派生出封装所用的元组 $(r_3, \omega_2, \eta_D, \zeta_D)$。

函数 DeriveCoins 的定义如下：将 ρ 扩展为字节流，再以拒绝采样（rejection sampling）方式解析该字节流，直至获得以下各域中的值——$r_3 \in \mathbb{Z}/C\mathbb{Z}$、$\omega_2 \in (\mathbb{Z}/2^{a}\mathbb{Z})^{\times}$、$\eta_D \in (\mathbb{Z}/D\mathbb{Z})^{\times}$ 和 $\zeta_D \in (\mathbb{Z}/D\mathbb{Z})^{\times}$。

3.3.2 算法

QIMEN-PIKE.KEM 的完整描述见以下各算法。

Algorithm 14 QIMEN-PIKE.ENCAPS(pk)

Input: 公钥 pk
Output: 密文 ct 和共享秘密 K

$m \leftarrow \{0,1\}^{2\lambda}$
$h_{\mathsf{pk}} \leftarrow \mathsf{H}_{\mathsf{pk}}(\mathsf{pk})$
$(\bar{K}, \rho) \leftarrow \mathsf{G}(m \| h_{\mathsf{pk}})$
$\mathsf{ct} \leftarrow$ QIMEN-PIKE.ENC.DET(pk, m, ρ)
$K \leftarrow \mathsf{KDF}(\bar{K} \| \mathsf{H}_{\mathsf{ct}}(\mathsf{ct}))$
return (ct, K)

Algorithm 15 QIMEN-PIKE.DECAPS(ct, sk)

Input: 密文 ct 和私钥 $\mathsf{sk} = (\mathsf{sk}_0 \| \mathsf{pk} \| h_{\mathsf{pk}} \| z)$
Output: 共享秘密 K

try
    $m' \leftarrow$ QIMEN-PIKE.PKE.DECRYPT$(\mathsf{ct}, \mathsf{sk}_0)$
catch
    return $K \leftarrow \mathsf{KDF}(z \| \mathsf{H}_{\mathsf{ct}}(\mathsf{ct}))$
$(\bar{K}', \rho') \leftarrow \mathsf{G}(m' \| h_{\mathsf{pk}})$
$\mathsf{ct}' \leftarrow$ QIMEN-PIKE.ENC.DET(pk, m', ρ')
if $\mathsf{ct} = \mathsf{ct}'$ **then**
    return $K \leftarrow \mathsf{KDF}(\bar{K}' \| \mathsf{H}_{\mathsf{ct}}(\mathsf{ct}))$
else
    return $K \leftarrow \mathsf{KDF}(z \| \mathsf{H}_{\mathsf{ct}}(\mathsf{ct}))$

CHAPTER 4

尺寸压缩

本节介绍 QIMEN-PIKE 中公钥和密文尺寸的优化技术。具体提出两种方法：第一种方法通过点组合减少需存储点的数量；第二种方法将点信息存储为其相对于固定挠基的标量表示，从而实现进一步压缩。与第一种方法相比，第二种方法以计算效率换取更紧凑的表示。由此衍生出两种实现：非压缩实现和压缩实现。

4.1 非压缩实现

回顾Chapter 3，公钥形式为 $\mathsf{pk} = \{E_A, (P_A, Q_A), (R_A, S_A), X_A\}$，其中 (P_A, Q_A) 是 2^a-挠基，(R_A, S_A) 是 C-挠基，X_A 是阶为 D 的点。所有有理点均定义在 E_A 上。以下首先讨论如何通过组合这些点来缩减公钥尺寸。

注意到 $C = C_1C_2$，其中 $C_1 \mid p+1$ 且 $C_2 \mid p-1$。给定一个阶为 C 的点 R，定义 $R^1 = [C_2]R$ 和 $R^2 = [C_1]R$。出于效率考虑，在设置阶段在 E_0 上定义 $E_0[C_1] = \langle R_0^1, S_0^1\rangle \subset E_0(\mathbb{F}_{p^2})$ 和 $E_0[C_2] = \langle R_0^2, S_0^2\rangle \subset E_0(\mathbb{F}_{p^4})$，以替代 $E_0[C] = \langle R_0, S_0\rangle \subset E_0(\mathbb{F}_{p^4})$。虽然 $E_0[C_2] = \langle P_0^2, Q_0^2\rangle \subset E_0(\mathbb{F}_{p^4})$，但 $E_0[C_2]$ 中点的 x-坐标定义在 $\mathbb{F}_{p^2}$ 上，因此可用 x-only 算术在 $\mathbb{F}_{p^2}$ 上执行计算。

因此，在公开参数中添加以下值：

- P_0^+：以 x-坐标表示的点 P_0^+，满足 $[C_1D]P_0^+ = P_0$，$[2^aD]P_0^+ = R_0^1$ 和 $[2^aC_1]P_0^+ = X_0$；
- Q_0^+：以 x-坐标表示的点 Q_0^+，满足 $[C_1]Q_0^+ = Q_A$ 和 $[2^a]Q_0^+ = S_A^1$；
- R_0^-：以 x-坐标表示的点 $R_0^- = R_0^2$；
- S_0^-：以 x-坐标表示的点 $S_0^- = S_0^2$；
- T_0^+：以 x-坐标表示的点 $T_0^+ = [D]P_0^+ - Q_0^+$；
- T_0^-：以 x-坐标表示的点 $T_0^- = R_0^- - S_0^-$。

4.1.1 公钥

为优化公钥尺寸，非压缩实现将公钥格式化为

$$(P_A^+, Q_A^+, R_A^-, S_A^-, T_A^-, \mathsf{label}_{\mathsf{pk}}).$$

Algorithm 16 QIMEN-PIKE.ModifiedCoreKeyGenIso(λ)

Input: 内部安全参数 λ

Output: $(q, E_A, P_A^+, Q_A^+, T_A^+, R_A^-, S_A^-, T_A^-)$，使得 $q(2^{a-2}-q)$ 是随机同源 $\varphi : E_0 \to E_A$ 的次数，且

$$(P_A^+, Q_A^+, T_A^+, R_A^-, S_A^-, T_A^-) = (\varphi(P_0^+), \varphi(Q_0^+), \varphi(T_0^+), \varphi(R_0^-), \varphi(S_0^-), \varphi(T_0^-))$$

repeat
  $q \xleftarrow{\$} \{0, \ldots, 2^{a-2}-1\}$
until $\gcd(q(2^{a-2}-q),\ 2 \cdot C \cdot D) = 1$
$\theta \leftarrow$ GeneralizedRepresentInteger$(q(2^{a-2}-q)C)$
$(P_0^\theta, Q_0^\theta) \leftarrow (\theta(P_0), \theta(Q_0))$
$K_1 \leftarrow$ EndomorphismToKernel$(E_0, (R_0^1, S_0^1), C_1, \overline{\theta})$
$K_2 \leftarrow$ EndomorphismToKernel$(E_0, (R_0^2, S_0^2), C_2, \overline{\theta})$
$\widetilde{E_A}, (\widetilde{P_A}, \widetilde{Q_A}, \widetilde{K_2}) \leftarrow$ OddIsogenyChain$(E_0, K_1, C_1, (P_0^\theta, Q_0^\theta, K_2))$
$\widetilde{E_A}, (\widetilde{P_A}, \widetilde{Q_A}) \leftarrow$ OddIsogenyChain$(\widetilde{E_A}, \widetilde{K_2}, C_2, (\widetilde{P_A}, \widetilde{Q_A}))$
$\texttt{pts} \leftarrow [(P_0, 0), (Q_0, 0), (P_0^+, 0), (Q_0^+, 0), (T_0^+, 0), (R_0^-, 0), (S_0^-, 0), (T_0^-, 0)]$
$\_, \texttt{pts} \leftarrow$ Isogeny22Chain$(([Cq]P_0, \widetilde{P_A}), ([Cq]Q_0, \widetilde{Q_A}), \texttt{pts})$
将 pts 解析为 $((\widetilde{P_A}, \widetilde{P'_A}), (\widetilde{Q_A}, \widetilde{Q'_A}), (\widetilde{P_A^+}, \_), (\widetilde{Q_A^+}, \_), (\widetilde{T_A^+}, \_), (\widetilde{R_A^-}, \_), (\widetilde{S_A^-}, \_), (\widetilde{T_A^-}, \_))$
$\texttt{pts} \leftarrow [(\widetilde{P_A^+}, 0), (\widetilde{Q_A^+}, 0), (\widetilde{T_A^+}, 0)), (\widetilde{R_A^-}, 0), (\widetilde{S_A^-}, 0), (\widetilde{T_A^-}, 0)]$
$(E_A, \_), ((P_A^+, \_), (Q_A^+, \_), (T_A^+, \_), (R_A^-, \_), (S_A^-, \_), (T_A^-, \_)) \leftarrow$ Isogeny22Chain$((\widetilde{P_A}, \widetilde{P'_A}), (\widetilde{Q_A}, \widetilde{Q'_A}), \texttt{pts})$
return $(q, E_A, P_A^+, Q_A^+, T_A^+, R_A^-, S_A^-, T_A^-)$

其中

- P_A^+：以 x-坐标表示的点 P_A^+，满足 $[C_1 D]P_A^+ = P_A$，$[2^a D]P_A^+ = R_A^1$ 和 $[2^a C_1]P_A^+ = X_A$；
- Q_A^+：以 x-坐标表示的点 Q_A^+，满足 $[C_1]Q_A^+ = Q_A$ 和 $[2^a]Q_A^+ = S_A^1$；
- R_A^-：以 x-坐标表示的点 $R_A^- = R_A^2$；
- S_A^-：以 x-坐标表示的点 $S_A^- = S_A^2$；
- T_A^-：以 x-坐标表示的点 $T_A^- = R_A^2 - S_A^2$；
- $\mathsf{label}_{\mathsf{pk}}$：一个布尔标志，用于恢复点 $[D]P_A^+ - Q_A^+$ 的 x-坐标，满足 $[2^a]([D]P_A^+ - Q_A^+) = R_A^1 - S_A^1$ 和 $[C_1]([D]P_A^+ - Q_A^+) = P_A - Q_A$。

由于 P_A, Q_A, R_A^1, S_A^1 和 X_A 定义在 $E_A(\mathbb{F}_{p^2})$ 上，点 P_A^+ 和 Q_A^+ 也定义在 $E_A(\mathbb{F}_{p^2})$ 上。公钥总共包含五个 $\mathbb{F}_{p^2}$ 元素和一个布尔比特。

综上所述，Algorithm 17给出了非压缩实现的密钥生成过程。子算法QIMEN-PIKE.CoreKeyGenIso也做了相应修改，Algorithm 16所示。

Algorithm 17 QIMEN-PIKE.KeyGenUncompressed(λ)

Input: 内部安全参数 λ
Output: 私钥和公钥 $(\mathsf{sk}, \mathsf{pk})$

$(q, E_A, P_A^+, Q_A^+, T_A^+, R_A^-, S_A^-, T_A^-) \leftarrow$ QIMEN-PIKE.ModifiedCoreKeyGenIso()
$\alpha_2, \beta_2 \xleftarrow{\$} (\mathbb{Z}/2^{a-2}\mathbb{Z})^\times$, $\gamma_1 \xleftarrow{\$} (\mathbb{Z}/C_1\mathbb{Z})^\times$, $\gamma_2 \xleftarrow{\$} (\mathbb{Z}/C_2\mathbb{Z})^\times$, 以及 $\delta_D \xleftarrow{\$} (\mathbb{Z}/D\mathbb{Z})^\times$
计算 t_1 使得 $t_1 \bmod 2^a \equiv \alpha_2$, $t_1 \bmod C_1 \equiv \gamma_1$ 且 $t_1 \bmod D \equiv \delta_D$
计算 t_2 使得 $t_2 \bmod 2^a \equiv \beta_2$ 且 $t_2 \bmod C_1 \equiv \gamma_1$
$T_3 \leftarrow$ LadderBiscalar$([D]P_A^+, Q_A^+, T_A^+, E_A, t_1, t_2)$
$P_A^+ \leftarrow [t_1]P_A^+, Q_A^+ \leftarrow [t_2]Q_A^+$
$x_{P_A^+} \leftarrow x(P_A^+)/z(P_A^+)$, $x_{Q_A^+} \leftarrow x(Q_A^+)/z(Q_A^+)$
$(x_{[D]P_A^+} : z_{[D]P_A^+}) \leftarrow$ Ladder$((x_{P_A^+} : 1), E_A, D)$
$T_1 \leftarrow$ ProjectiveDifference$((x_{[D]P_A^+} : z_{[D]P_A^+}), (x_{Q_A^+} : 1), E_A)$
$\mathsf{label}_{\mathsf{pk}} \leftarrow (T_1 = T_3)$
$R_A^- \leftarrow [\gamma_2]R_A^-$, $S_A^- \leftarrow [\gamma_2]S_A^-$, $T_A^- \leftarrow [\gamma_2]T_A^-$
$\mathsf{sk} \leftarrow (q, \alpha_2, \beta_2, \delta_D)$
$\mathsf{pk} \leftarrow (x_{P_A^+}, x_{Q_A^+}, x(R_A^-)/z(R_A^-), x(S_A^-)/z(S_A^-), x(T_A^-)/z(T_A^-), \mathsf{label}_{\mathsf{pk}})$
return $(\mathsf{sk}, \mathsf{pk})$

Remark 4.1.1. 注意到在*QIMEN-PIKE.ModifiedCoreKeyGenIso*步 骤 *11*和 *14*中，二维同源不仅计算了四个点 P_0^+、Q_0^+、R_0^- 和 S_0^-，还计算了 $T_0^+ = [D]P_A^+ - Q_0^+$ 和 T_0^-。一个自然的问题是：既然 T_A^+（相应地 T_A^-）可以从 P_{AB}^+、Q_{AB}^+（相应地 R_A^-、S_A^-）推导出来，为什么还要额外计算这些差？原因在于实现使用了 *x-only* 算术。虽然计算点加法 $[D]P_A^+ - Q_0^+$ 是可行的，但在仅有 x-坐标的情况下，区分 $[D]P_A^+ - Q_0^+$ 和 $[D]P_A^+ + Q_0^+$ 效率低下。算法在 T_0^- 处计算二维同源也是出于同样的原因。

4.1.2 密文

接下来考虑密文内点信息的组合方式。密文涉及两条椭圆曲线 E_B 和 E_{AB}，2^a-挠基 $(P_B, Q_B) \in E_B[2^a]$ 和 $(P_{AB}, Q_{AB}) \in E_{AB}[2^a]$，以及两个点 $Y_B \in E_B[D]$ 和 $X_{AB} \in E_{AB}[D]$。在非压缩实现中，密文形式为

$$(E_B, E_{AB}, P_B^+, Q_B^+, P_{AB}^+, Q_{AB}^+, c), \qquad c = \mathsf{pad} \oplus \mathsf{msg},$$

其中：

- E_B：以 Montgomery 系数表示的椭圆曲线 E_B；
- E_{AB}：以 Montgomery 系数表示的椭圆曲线 E_{AB}；
- P_B^+：以 x-坐标表示的点 $P_B^+ = P_B$；
- Q_B^+：以 x-坐标表示的点 Q_B^+，满足 $[D]Q_B^+ = Q_B$ 和 $[2^a]Q_B^+ = Y_B$；
- P_{AB}^+：以 x-坐标表示的点 P_{AB}^+，满足 $[D]P_{AB}^+ = P_{AB}$ 和 $[2^a]P_{AB}^+ = X_{AB}$；

- Q_{AB}^{+}：以 x-坐标表示的点 $Q_{AB}^{+} = Q_{AB}$.

由于所有考虑的点都定义在 $E_B(\mathbb{F}_{p^2})$ 或 $E_{AB}(\mathbb{F}_{p^2})$ 上，它们的 x-坐标定义在 $\mathbb{F}_{p^2}$ 上。因此，密文包含六个 $\mathbb{F}_{p^2}$-元素。

Algorithm 18和Algorithm 19分别给出了加密和解密过程。

Remark 4.1.2. 在非压缩实现中，公钥不显式包含椭圆曲线 E_A。然而，加密过程中标量乘法和阶梯算法仍然需要曲线系数。例如，发送方必须执行 *Ladder3pt* 来计算同源 $\psi' : E_A \to E_{AB}$ 的核，或调用*ProjectiveDifference*来 确定 $[D]P_A^{+} - Q_A^{+}$ 的 x-坐标。为此，可在*Algorithm 18*的步骤 *1* 处调用*RecoverCodomain*， 以点 R_A^{-}、S_A^{-} 及其差 T_A^{-} 为输入，唯一地恢复椭圆曲线。

4.2 压缩实现

除 D-挠点外，所有其他点都具有光滑阶，便于进一步压缩。对于给定阶为光滑数 N 的点 $P \in E$，可以生成同阶的确定性基 (U, V)，并将 P 表示为该基的线性组合：

$$P = [s]U + [t]V. \tag{4.1}$$

通过存储标量 (s, t) 而非 P 的坐标，存储需求从 $2\log p$ 比特显著降低到 $2\log N$ 比特，其中 $s, t \in \mathbb{Z}/N\mathbb{Z}$。此外，由于 P、U 和 V 均具有满阶 N，标量 s 或 t 中至少有一个模 N 可逆。在只需传输子群 $\langle P \rangle$ 的生成元而非 P 本身的场景中，可以传输 $[s^{-1}]P$ （或 $[t^{-1}]P$，若 s 不可逆）。因此，可将标量对 (s, t) 规范化，并进一步压缩为 $(1, s^{-1}t)$ 或 $(s^{-1}t, 1)$。

按照Eq. (4.1)，从 P 恢复标量 s 和 t 需求解二维椭圆曲线离散对数。虽然 N 的光滑性保证这可在多项式时间内求解，但由于需要大量的标量乘法和点加法，该过程计算量仍然很大。为缓解这一开销，可以利用密码学配对。观察到

$$\begin{aligned} h_0 &= t_N(U, V), \\ h_1 &= t_N(U, P) = t_N(U, [s]U + [t]V) = t_N(U, V)^t, \\ h_2 &= t_N(V, P) = t_N(V, [s]U + [t]V) = t_N(U, V)^{-s}, \end{aligned} \tag{4.2}$$

配对计算可将问题映射到乘法群 $\mu_N = \{x \in \mathbb{F}_{p^2} \mid x^N = 1\}$ 中。由此，可在 μ_N 中通过离散对数高效提取标量 s 和 t，从而避免直接计算二维椭圆曲线离散对数。

将类似方法应用于整个 N-挠基 (P, Q)，可将该基压缩为 $(s, t, u, v) \in (\mathbb{Z}/N\mathbb{Z})^4$。由于 (P, Q) 构成有效基，对应的变换矩阵可逆，故存在有效的标量逆。因此，可将此表示规范化，并进一步压缩为恰好三个标量。

4.2.1 公钥

回顾公钥由椭圆曲线 E_A、一个 2^a-挠基 (P_A, Q_A)、一个 C_1-挠基 (R_A^1, S_A^1)、一个 C_2-挠基 (R_A^2, S_A^2) 和一个阶为 D 的点 X_A 组成。由于 D 非光滑，X_A 无法通过离散对数计算高效

压缩。因此，公钥必须至少包含一个 x-only 坐标点来表示 X_A。另一方面，由于 2^a、C_1 和 C_2 的光滑性，可将以上基表示如下：

$$\begin{pmatrix} P_A \\ Q_A \end{pmatrix} = \begin{pmatrix} s_{A,2^a} & t_{A,2^a} \\ u_{A,2^a} & v_{A,2^a} \end{pmatrix} \cdot \begin{pmatrix} U_{A,2^a} \\ V_{A,2^a} \end{pmatrix},$$
$$\begin{pmatrix} R_A^1 \\ S_A^1 \end{pmatrix} = \begin{pmatrix} s_{A,C_1} & t_{A,C_1} \\ u_{A,C_1} & v_{A,C_1} \end{pmatrix} \cdot \begin{pmatrix} U_{A,C_1} \\ V_{A,C_1} \end{pmatrix},$$
$$\begin{pmatrix} R_A^2 \\ S_A^2 \end{pmatrix} = \begin{pmatrix} s_{A,C_2} & t_{A,C_2} \\ u_{A,C_2} & v_{A,C_2} \end{pmatrix} \cdot \begin{pmatrix} U_{A,C_2} \\ V_{A,C_2} \end{pmatrix},$$

其中 $(U_{A,2^a}, V_{A,2^a})$、(U_{A,C_1}, V_{A,C_1}) 和 (U_{A,C_2}, V_{A,C_2}) 分别是 E_A 上的确定性 2^a-、C_1-和 C_2-挠基。

所有挠基的标量表示均可在不影响加密过程的情况下分别压缩为三个标量。粗略地，此表示需 $2\log p$ 比特存储曲线 E_A，$2\log p$ 比特存储点 X_A，以及分别需 $3a$、$3\log C_1$ 和 $3\log C_2$ 比特存储 (P_A, Q_A)、(R_A^1, S_A^1) 和 (R_A^2, S_A^2)。设 $C_1 = \prod_{i=1}^{n} \ell_i^{n_i}$。为充分利用 x-only 射影坐标的容量，我们改为存储一个单一的复合点 P_{AB}^+，其阶为 $2^a C_1 D$，使得

$$[C_1 D]P_A^+ = [m_{A,2^a}]P_A, [2^a D]P_A^+ = [m_{A,C_1}]R_A^1 \text{ 且} [2^a C_1]P_A^+ = X_A,$$

其中

$$m_{A,2^a} = \begin{cases} v_{A,2^a}^{-1} \bmod 2^a, & \text{若 } v_{A,2^a} \text{ 在 } \mathbb{Z}/2^a\mathbb{Z} \text{ 中可逆}, \\ u_{A,2^a}^{-1} \bmod 2^a, & \text{否则}, \end{cases}$$

且

$$m_{A,C_1} = \begin{cases} u_{A,C_1}^{-1} \bmod \ell_i^{n_i}, & \text{若 } u_{A,C_1} \text{ 在 } \mathbb{Z}/\ell_i^{n_i}\mathbb{Z} \text{ 中可逆}, \\ v_{A,C_1}^{-1} \bmod \ell_i^{n_i}, & \text{否则}, \end{cases}$$

其中 $i = 1, 2, \ldots, n$。

公钥中包含该复合点 P_A^+ 的 x-坐标后，即完全消除了显式存储与 P_A 和 R_A^1 相关的标量表示的必要性。接收方获得 $x(P_A^+)$ 后，可独立恢复 $[C_1 D]P_A^+$ 和 $[2^a D]P_A^+$ 的 x-坐标。因此，P_A 和 R_A^1 的信息以隐式方式传输，无需任何额外比特。此优化分摊了以非压缩 x-only 坐标存储 X_A 的高昂代价，使得其余基点（如 Q_A 和 S_A^1）可用极少量的标量表示，从而得到一个高度紧凑的公钥。

为进一步缩减压缩公钥的尺寸，在设置中选择 Q_0 作为满足 $[2^{a-1}]Q_0 = (0,0)$ 的 2^a-挠点，即 Montgomery 曲线上阶为 2 的点。由于同源求值算法（OddIsogenyChain和Isogeny22Chain）都将原曲线上的 $(0,0)$ 映射到像曲线上的 $(0,0)$，且同源次数均为奇数（即它们在 2-挠群上同构作用），所有考虑的同源将 P_0 映射到远离 $(0,0)$ 的点，而将 Q_0 映射到 $(0,0)$ 上的点。通过确保 E_A 上的确定性 2^a-挠基 $\{U_{A,2^a}, V_{A,2^a}\}$ 具有相同的结构性质，即 $[2^{a-1}]V_{A,2^a} = (0,0)$，则系数 $v_{A,2^a}$ 保证在 $\mathbb{Z}/2^a\mathbb{Z}$ 中可逆。由此可确定地设 $m_{A,2^a} = (v_{A,2^a})^{-1} \bmod 2^a$，从而无需在公钥中使用布尔标志区分 $m_{A,2^a} = (v_{A,2^a})^{-1} \bmod 2^a$ 与 $m_{A,2^a} = (u_{A,2^a})^{-1} \bmod 2^a$。此技术同样适用于密文压缩。此外，限制 C_2 为素数幂，确保压缩的 C_2-挠信息不需要额外的标识符，如 label_{C_1}。

总之，压缩公钥的形式为

$$\mathsf{pk} = (E_A, P_A^+, \mathsf{hint}_{C_1}, \mathsf{hint}_{C_2}, s_{A,2^a}, s_{C_1}, \mathsf{label}_{C_1}, \mathsf{label}_+, s_{C_2}^1, s_{C_2}^2, s_{C_2}^3, \mathsf{label}_{C_2}),$$

其中：

- E_A：以 Montgomery 系数表示的椭圆曲线 E_A；
- P_A^+：以 x-坐标表示的点 P_A^+，满足 $[D]P_A^+ = [m_{2^a}]P_A$，$[2^a D]P_A^+ = [m_{C_1}]R_A^1$ 和 $[2^a C_1]P_A^+ = X_A$；
- $\mathsf{hint}_{2^a C_1}$：用于高效计算 E_A 上确定性 $2^a C_1$-挠基的提示值；
- hint_{C_2}：用于高效计算 E_A 上确定性 C_2-挠基的提示值；
- $s_{A,2^a}$：用于恢复点 $[m_{2^a}]Q_A$ 的标量；
- s_{C_1} 和 label_{C_1}：用于恢复点 $[m_{C_1}]S_A^1$ 的标量和整数；
- label_+：用于恢复 $2^a C_1$-挠点差值的布尔标志；
- $s_{C_2}^1$、$s_{C_2}^2$、$s_{C_2}^3$ 和 label_{C_2}：用于恢复点 $[m_{C_2}]R_A^2$ 和 $[m_{C_2}]S_A^2$ 的标量和布尔标志，其中

$$m_{C_2} = \begin{cases} u_{A,C_2}^{-1} \bmod C_2, & \text{若 } u_{A,C_2} \text{ 在 } \mathbb{Z}/C_1\mathbb{Z} \text{ 中可逆}, \\ v_{A,C_2}^{-1} \bmod C_2, & \text{否则}. \end{cases}$$

Algorithm 20给出了生成压缩公钥的过程。

4.2.2 密文

密文涉及两条椭圆曲线 E_B 和 E_{AB}，以及 ϕ' 在 2^a-挠上的赋值，即 (P_B, Q_B) 和 (P_{AB}, Q_{AB})。此外，还包含两个阶为 D 的点 Y_B 和 X_{AB}。

与压缩公钥类似，可用 E_B 和 E_{AB} 上的确定性基来计算 2^a-挠基的线性表示：

$$\begin{pmatrix} P_B \\ Q_B \end{pmatrix} = \begin{pmatrix} s_{B,2^a} & t_{B,2^a} \\ u_{B,2^a} & v_{B,2^a} \end{pmatrix} \cdot \begin{pmatrix} U_{B,2^a} \\ V_{B,2^a} \end{pmatrix},$$
$$\begin{pmatrix} P_{AB} \\ Q_{AB} \end{pmatrix} = \begin{pmatrix} s_{AB,2^a} & t_{AB,2^a} \\ u_{AB,2^a} & v_{AB,2^a} \end{pmatrix} \cdot \begin{pmatrix} U_{AB,2^a} \\ V_{AB,2^a} \end{pmatrix},$$

其中 $(U_{B,2^a}, V_{B,2^a})$ 和 $(U_{AB,2^a}, V_{AB,2^a})$ 分别是 E_B 和 E_{AB} 上的确定性 2^a-挠基。

为充分利用 x-only 射影坐标的容量并最小化密文尺寸，存储以下点：

- Q_B^+：以 x-坐标表示的点 Q_B^+，满足 $[D]Q_B^+ = [m_{B,2^a}]Q_B$ 和 $[2^a]Q_B^+ = Y_B$；
- P_{AB}^+：以 x-坐标表示的点 P_{AB}^+，满足 $[D]P_{AB}^+ = [m_{AB,2^a}]P_{AB}$ 和 $[2^a]P_{AB}^+ = X_{AB}$；

其中

$$m_{B,2^a} = v_{B,2^a}^{-1} \bmod 2^a,$$
$$m_{AB,2^a} = u_{AB,2^a}^{-1} \bmod 2^a.$$

则点 $[m_{B,2^a}]P_B$ 和 $[m_{AB,2^a}]Q_{AB}$ 可分别用 $\mathbb{Z}/2^a\mathbb{Z}$ 中的一个标量存储。

总之，压缩密文的形式为

$$\mathsf{ct} = (E_B, E_{AB}, Q_B^+, P_{AB}^+, \mathsf{hint}_B, \mathsf{hint}_{AB}, s_{B,2^a}, s_{AB,2^a}, c),$$

其中：

- E_B：以 Montgomery 系数表示的椭圆曲线 E_B；
- E_{AB}：以 Montgomery 系数表示的椭圆曲线 E_{AB}；
- Q_B^+：以 x-坐标表示的点 Q_B^+，满足 $[D]Q_B^+ = [m_{B,2^a}]Q_B$ 和 $[2^a]Q_B^+ = Y_B$；
- P_{AB}^+：以 x-坐标表示的点 P_{AB}^+，满足 $[D]P_{AB}^+ = [m_{AB,2^a}]P_{AB}$ 和 $[2^a]P_{AB}^+ = X_{AB}$；
- hint_B：用于高效计算 E_B 上确定性 2^a-挠基的提示值；
- hint_{AB}：用于高效计算 E_{AB} 上确定性 2^a-挠基的提示值；
- $s_{B,2^a}$：用于恢复点 $[m_{B,2^a}]P_B$ 的标量；
- $s_{AB,2^a}$：用于恢复点 $[m_{AB,2^a}]Q_{AB}$ 的标量；
- c：掩码消息 $\mathsf{pad} \oplus \mathsf{msg}$。

Algorithm 21和Algorithm 22给出了生成压缩密文并解密以恢复明文的过程。

Algorithm 18 QIMEN-PIKE.KeyEncUncompressed($\mathsf{pk}, \mathsf{msg}$)

Input: 公钥 $\mathsf{pk} = (x_{P_A^+}, x_{Q_A^+}, x_{R_A^-}, x_{S_A^-}, x_{T_A^-}, \mathsf{label}_{\mathsf{pk}})$ 和消息 $\mathsf{msg} \in \{0,1\}^{2\lambda}$

Output: 密文 ct

$E_A \leftarrow \textsc{RecoverCodomain}((x_{R_A^-} : 1), (x_{S_A^-} : 1), (x_{T_A^-} : 1))$

$r_1 \xleftarrow{\$} \mathbb{Z}/C_1\mathbb{Z}$, $r_2 \xleftarrow{\$} \mathbb{Z}/C_2\mathbb{Z}$, $\omega_2 \xleftarrow{\$} (\mathbb{Z}/2^a\mathbb{Z})^\times$, $\eta_D, \zeta_D \xleftarrow{\$} (\mathbb{Z}/D\mathbb{Z})^\times$

$w' \leftarrow w_0^{\eta_D \zeta_D}$

$K_1 \leftarrow \textsc{Ladder3pt}(R_0^1, S_0^1, R_0^1 - S_0^1, E_0, r_1)$

$K_2 \leftarrow \textsc{Ladder3pt}(R_0^2, S_0^2, R_0^2 - S_0^2, E_0, r_2)$

$\mathsf{pad} \leftarrow \mathsf{H}(\mathsf{KDF} \| \mathrm{tr}(w'))$

$E'_B, (T_1, T_2, K'_2) \leftarrow \textsc{OddIsogenyChain}(E_0, K_1, C_1, (P_0, Q_0^+, K_2))$

$E_B, (T_1, T_2) \leftarrow \textsc{OddIsogenyChain}(E'_B, K'_2, C_2, (T_1, T_2))$

计算 s 使得 $s \bmod 2^a \equiv \omega_2^{-1}$ 且 $s \bmod D \equiv \eta_D$

$P_B^+ \leftarrow [\omega_2]T_1$, $Q_B^+ \leftarrow [s]T_2$

$(x_{[D]P_A^+} : z_{[D]P_A^+}) \leftarrow \textsc{Ladder}((x_{P_A^+} : 1), E_A, D)$

$T_3 \leftarrow \textsc{ProjectiveDifference}((x_{[D]P_A^+} : z_{[D]P_A^+}), (x_{Q_A^+} : 1), E_A)$

if $\mathsf{label}_{\mathsf{pk}} = 0$ **then**

    $T_3 \leftarrow \mathrm{xADD}((x_{[D]P_A^+} : z_{[D]P_A^+}), (x_{Q_A^+} : 1), T_3)$

$K_1 \leftarrow \textsc{Ladder3pt}([D]P_A^+, Q_A^+, T_3, E_A, r_1)$

$K_1 \leftarrow [2^a]K_1$

$K_2 \leftarrow \textsc{Ladder3pt}((x_{R_A^-} : 1), (x_{S_A^-} : 1), (x_{T_A^-} : 1), E_A, r_2)$

$E'_{AB}, (T_1, T_2, K'_2) \leftarrow \textsc{OddIsogenyChain}(E_A, K_1, C_1, (P_A^+, Q_A^+, K_2))$

$E_{AB}, (T_1, T_2) \leftarrow \textsc{OddIsogenyChain}(E'_{AB}, K'_2, C_2, (T_1, T_2))$

计算 s 使得 $s \bmod 2^a \equiv \omega_2$ 且 $s \bmod D \equiv \zeta_D$

$P_{AB}^+ \leftarrow [s]T_1$, $Q_{AB}^+ \leftarrow [\omega_2^{-1} \bmod 2^a]T_2$

$c \leftarrow \mathsf{pad} \oplus \mathsf{msg}$

$\mathsf{ct} \leftarrow (E_B, E_{AB}, x(P_B^+)/z(P_B^+), x(Q_B^+)/z(Q_B^+), x(P_{AB}^+)/z(P_{AB}^+), x(Q_{AB}^+)/z(Q_{AB}^+), c)$

return ct

Algorithm 19 QIMEN-PIKE.KeyDecUncompressed(ct, sk)

Input: 规范化密文 $(E_B, E_{AB}, P_B^+, Q_B^+, P_{AB}^+, Q_{AB}^+, c)$ 和私钥 $\mathsf{sk} = (q, \alpha_2, \beta_2, \delta_D)$
Output: 消息 msg

$Q_B \leftarrow [D]Q_B^+$
$Y_B \leftarrow [2^a]Q_B^+$
$P_{AB} \leftarrow [D]P_{AB}^+$
$X_{AB} \leftarrow [2^a]P_{AB}^+$
$\widetilde{P_{AB}} \leftarrow [(-q\alpha_2)^{-1} \mod 2^a]P_{AB}, \widetilde{Q_{AB}} \leftarrow [(-q\beta_2)^{-1} \mod 2^a]Q_{AB}$
$E_M, \_, (Y_M, \_), (X_M, \_)$
$\quad \leftarrow$ Isogeny22Chain$((P_B, \widetilde{P_{AB}}), (Q_B, \widetilde{Q_{AB}}), [(Y_B, 0_{E_B}), (0_{E_{AB}}, X_{AB})])$
$t \leftarrow \left(q(2^{a-2} - q)C\delta_D\right)^{-1} \bmod D$
$w \leftarrow$ TateOdd$(E_M, D, X_M, Y_M, X_M - Y_M)$
$w' \leftarrow w^t$
$\widetilde{\mathsf{pad}} \leftarrow \mathsf{H}(\mathsf{KDF} \| \mathrm{tr}(w'))$
return $\mathsf{pad} \oplus c$

Algorithm 20 QIMEN-PIKE.KeyGenCompressed(λ)

Input: 内部安全参数 λ
Output: 私钥和公钥 $(\mathsf{sk}, \mathsf{pk})$

$(q, E_A, \widetilde{P_A^+}, \widetilde{Q_A^+}, T_A^+, R_A^-, S_A^-, T_A^-) \leftarrow$ QIMEN-PIKE.ModifiedCoreKeyGenIso()
$\alpha_2, \beta_2 \xleftarrow{\$} (\mathbb{Z}/2^{a-2}\mathbb{Z})^\times$, $\gamma_1 \xleftarrow{\$} (\mathbb{Z}/C_1\mathbb{Z})^\times$, $\gamma_2 \xleftarrow{\$} (\mathbb{Z}/C_2\mathbb{Z})^\times$, 以及 $\delta_D \xleftarrow{\$} (\mathbb{Z}/D\mathbb{Z})^\times$
$U_{A,C_2}, V_{A,C_2}, \mathsf{hint}_{C_2} \leftarrow$TorsionBasisGivenOrderToHint$(E_A, C_2, \mathsf{false})$
$s_{A,C_2}, t_{A,C_2}, u_{A,C_2}, v_{A,C_2} \leftarrow$ DlogOdd$(E_A, C_2, U_{A,C_2}, V_{A,C_2}, U_{A,C_2} - V_{A,C_2}, R_A^-, S_A^-)$
if s_{A,C_2} 在 $(\mathbb{Z}/C_2\mathbb{Z})^*$ 中可逆 **then**
    $\mathsf{label}_{C_2} \leftarrow 0$, $s_{C_2}^1 \leftarrow (s_{A,C_2})^{-1} t_{A,C_2}$, $s_{C_2}^2 \leftarrow (s_{A,C_2})^{-1} u_{A,C_2}$, $s_{C_2}^3 \leftarrow (s_{A,C_2})^{-1} v_{A,C_2}$
else
    $\mathsf{label}_{C_2} \leftarrow 1$, $s_{C_2}^1 \leftarrow (t_{A,C_2})^{-1} s_{A,C_2}$, $s_{C_2}^2 \leftarrow (t_{A,C_2})^{-1} u_{A,C_2}$, $s_{C_2}^3 \leftarrow (t_{A,C_2})^{-1} v_{A,C_2}$
计算 t 使得 $t \equiv \beta_2 \bmod 2^a$ 且 $t \equiv \gamma_1 \bmod C_1$
$U_{A,2^aC_1}, V_{A,2^aC_1}, \mathsf{hint}_{C_1} \leftarrow$TorsionBasisGivenOrderToHint$(E_A, 2^aC_1, \mathsf{true})$
$U_{A,2^a} \leftarrow [C_1]U_{A,2^aC_1}$,$U_{A,C_1} \leftarrow [2^a]U_{A,2^aC_1}$,$V_{A,2^a} \leftarrow [C_1]V_{A,2^aC_1}$,$V_{A,C_1} \leftarrow [2^a]V_{A,2^aC_1}$
$u_{A,2^a}, v_{A,2^a} \leftarrow$ Dlog2Single$(E_A, a, U_{A,2^a}, V_{A,2^a}, U_{A,2^a} - V_{A,2^a}, [C_1 t]\widetilde{Q_A^+})$
$u_{A,C_1}, v_{A,C_1} \leftarrow$ DlogOddSingle$(E_A, C_1, U_{A,C_1}, V_{A,C_1}, U_{A,C_1} - V_{A,C_1}, [2^a t]\widetilde{Q_A^+})$
$m_{2^a} \leftarrow (v_{A,2^a})^{-1} \bmod 2^a$, $s_{A,2^a} \leftarrow m_{2^a} \cdot u_{A,2^a} \bmod 2^a$
$m_{C_1}, s_{C_1}, \mathsf{label}_{C_1} \leftarrow$LabelComputation$(C_1, u_{A,C_1}, v_{A,C_1})$
计算 t_1 使得 $t_1 \equiv \alpha_2 m_{2^a} \bmod 2^a$, $t_1 \equiv \gamma_1 m_{C_1} \bmod C_1$ 且 $t_1 \equiv \delta_D \bmod D$
$P_A^+ \leftarrow [t_2]\widetilde{P_A^+}$
计算 t_2 使得 $t_2 \equiv \beta_2 m_{2^a} \bmod 2^a$ 且 $t_2 \equiv \beta_2 m_{C_1} \bmod C_1$
$T_1 \leftarrow$ LadderBiscalar$(P_A^+, Q_A^+, T_A^+, E_A, Dt_1, t_2)$
$P_A^+ \leftarrow [t_1]P_A^+$, $Q_A^+ \leftarrow [t_2]Q_A^+$
$T_2 \leftarrow$ ProjectiveDifference$([D]P_A^+, Q_A^+, E_A)$
$\mathsf{label}_+ \leftarrow (T_1 = T_2)$
$\mathsf{sk} \leftarrow (q, \alpha_2, \beta_2, \delta_D)$
$\mathsf{pk} \leftarrow (E_A, P_A^+, \mathsf{hint}_{C_1}, \mathsf{hint}_{C_2}, s_{A,2^a}, s_{C_1}, \mathsf{label}_{C_1}, \mathsf{label}_+, s_{C_2}^1, s_{C_2}^2, s_{C_2}^3, \mathsf{label}_{C_2})$
return $(\mathsf{sk}, \mathsf{pk})$

Algorithm 21 QIMEN-PIKE.KeyEncCompressed(pk, msg)

Input: 公钥 $\mathsf{pk} = (E_A, P_A^+, \mathsf{hint}_{C_1}, \mathsf{hint}_{C_2}, s_{A,2^a}, \mathsf{label}_{2^a}, s_{C_1}, \mathsf{label}_{C_1}, \mathsf{label}_+, s_{C_2}^1, s_{C_2}^2, s_{C_2}^3, \mathsf{label}_{C_2})$ 和消息 $\mathsf{msg} \in \{0,1\}^{2\lambda}$
Output: 密文 ct

$r_1 \xleftarrow{\$} \mathbb{Z}/C_1\mathbb{Z}$, $r_2 \xleftarrow{\$} \mathbb{Z}/C_2\mathbb{Z}$, $\omega_2 \xleftarrow{\$} (\mathbb{Z}/2^a\mathbb{Z})^\times$, $\eta_D, \zeta_D \xleftarrow{\$} (\mathbb{Z}/D\mathbb{Z})^\times$
$w' \leftarrow w_0^{\eta_D \zeta_D}$
$\mathsf{pad} \leftarrow \mathsf{H}(\mathsf{KDF} \| \mathrm{tr}(w'))$
$K_1 \leftarrow$ Ladder3pt$(R_0^1, S_0^1, R_0^1 - S_0^1, E_0, r_1)$
$K_2 \leftarrow$ Ladder3pt$(R_0^2, S_0^2, R_0^2 - S_0^2, E_0, r_2)$
$E'_B, (T_1, T_2, K'_2) \leftarrow$ OddIsogenyChain$(E_0, K_1, C_1, (P_0, Q_0^+, K_2))$
$E_B, (T_1, T_2) \leftarrow$ OddIsogenyChain$(E'_B, K'_2, C_2, (T_1, T_2))$
$U_{B,2^a}, V_{B,2^a}, \mathsf{hint}_B \leftarrow$ TorsionBasisToHint(E_B, a)
$u_{B,2^a}, v_{B,2^a} \leftarrow$ Dlog2Single$(E_B, a, U_{B,2^a}, V_{B,2^a}, [\omega_2]T_1)$
计算 s 使得 $s \bmod 2^a \equiv (\omega_2 v_{B,2^a})^{-1}$ 且 $s \bmod D \equiv \eta_D$
$Q_B^+ \leftarrow [s]T_2$
$U_{A,2^aC_1}, V_{A,2^aC_1} \leftarrow$ TorsionBasisGivenOrderFromHint$(E_A, 2^aC_1, \mathsf{hint}_{C_1})$
$U_{A,C_2}, V_{A,C_2} \leftarrow$ TorsionBasisGivenOrderFromHint$(E_A, C_2, \mathsf{hint}_{C_2})$
$S_A^- \leftarrow$ LadderBiscalar$(U_{A,C_2}, V_{A,C_2}, U_{A,C_2} - V_{A,C_2}, E_A, s_{C_2}^2, s_{C_3}^3)$
if $\mathsf{label}_{C_2} = 1$ **then**
    $R_A^- \leftarrow$ Ladder3pt$(V_{A,C_2}, U_{A,C_2}, U_{A,C_2} - V_{A,C_2}, E_A, s_{C_2}^1)$
    $T_A^- \leftarrow$ LadderBiscalar$(U_{A,C_2}, V_{A,C_2}, U_{A,C_2} - V_{A,C_2}, E_A, s_{C_2}^1 - s_{C_2}^2, 1 - s_{C_2}^3)$
else
    $R_A^- \leftarrow$ Ladder3pt$(U_{A,C_2}, V_{A,C_2}, U_{A,C_2} - V_{A,C_2}, E_A, s_{C_2}^1)$
    $T_A^- \leftarrow$ LadderBiscalar$(U_{A,C_2}, V_{A,C_2}, U_{A,C_2} - V_{A,C_2}, E_A, 1 - s_{C_2}^2, s_{C_2}^1 - s_{C_2}^3)$
$t_1, t_2 \leftarrow$ ScalarRecoveryFromLabel$(C_1, s_{C_1}, \mathsf{label}_{C_1})$
计算 s_1 使得 $s_1 \equiv s_{A,2^a} \bmod 2^a$ 且 $s_1 \equiv t_1 \bmod C_1$
计算 s_2 使得 $s_2 \equiv 1 \bmod 2^a$ 且 $s_2 \equiv t_2 \bmod C_1$
$Q_A^+ \leftarrow$ LadderBiscalar$(U_{A,2^aC_1}, V_{A,2^aC_1}, U_{A,2^aC_1} - V_{A,2^aC_1}, E_A, s_1, s_2)$
$T_3 \leftarrow$ ProjectiveDifference$([D]P_A^+, Q_A^+, E_A)$
if $\mathsf{label}_{\mathsf{pk}} = 0$ **then**
    $T_3 \leftarrow$ xADD$([D]P_A^+, Q_A^+, T_3)$
$K_1 \leftarrow$ Ladder3pt$([D]P_A^+, Q_A^+, [D]P_A^+ - Q_A^+, E_A, r_1)$
$K_1 \leftarrow [2^a]K_1$
$K_2 \leftarrow$ Ladder3pt$(R_A^-, S_A^-, T_A^-, E_A, r_2)$
$E'_{AB}, (T_1, T_2, K'_2) \leftarrow$ OddIsogenyChain$(E_A, K_1, C_1, ([C_1D]P_A^+, [C_1]Q_A^+, K_2))$
$E_{AB}, (T_1, T_2) \leftarrow$ OddIsogenyChain$(E'_{AB}, K'_2, C_2, (T_1, T_2))$
$U_{AB,2^a}, V_{AB,2^a}, \mathsf{hint}_{AB} \leftarrow$ TorsionBasisToHint(E_{AB}, a)
$u_{AB,2^a}, v_{AB,2^a} \leftarrow$ Dlog2Single$(E_{AB}, a, U_{AB,2^a}, V_{AB,2^a}, [\omega_2^{-1} \bmod 2^a]T_2)$
计算 s 使得 $s \bmod 2^a \equiv \omega_2 (u_{AB,2^a})^{-1}$ 且 $s \bmod D \equiv \zeta_D$
$P_{AB}^+ \leftarrow [s]T_1$
$s_{B,2^a} \leftarrow (v_{B,2^a})^{-1} u_{B,2^a} \bmod 2^a$, $s_{AB,2^a} \leftarrow (u_{AB,2^a})^{-1} v_{AB,2^a} \bmod 2^a$
$c \leftarrow \mathsf{pad} \oplus \mathsf{msg}$
$\mathsf{ct} = (E_B, E_{AB}, Q_B^+, P_{AB}^+, \mathsf{hint}_B, \mathsf{hint}_{AB}, s_{B,2^a}, s_{AB,2^a}, c)$
return ct

Algorithm 22 QIMEN-PIKE.KeyDecCompressed(ct, sk)

Input: 密文 $\mathsf{ct} = (E_B, E_{AB}, Q_B^+, P_{AB}^+, \mathsf{hint}_B, \mathsf{hint}_{AB}, s_{B,2^a}, s_{AB,2^a}, c)$ 和私钥 $\mathsf{sk} = (q, \alpha_2, \beta_2, \delta_D)$

Output: 消息 msg

$U_{B,2^a}, V_{B,2^a} \leftarrow$ TorsionBasisFromHint$(E_B, a, \mathsf{hint}_B)$
$U_{AB,2^a}, V_{AB,2^a} \leftarrow$ TorsionBasisFromHint$(E_{AB}, a, \mathsf{hint}_{AB})$
$Q_B \leftarrow [D]Q_B^+$
$Y_B \leftarrow [2^a]Q_B^+$
$P_{AB} \leftarrow [D]P_{AB}^+$
$X_{AB} \leftarrow [2^a]P_{AB}^+$
$P_B \leftarrow$ Ladder3pt$(V_{B,2^a}, U_{B,2^a}, V_{B,2^a} - U_{B,2^a}, E_B, s_{B,2^a})$
$Q_{AB} \leftarrow$ Ladder3pt$(U_{AB,2^a}, V_{AB,2^a}, U_{AB,2^a} - V_{AB,2^a}, E_{AB}, s_{AB,2^a})$
$\widetilde{P_{AB}} \leftarrow [(-q\alpha_2)^{-1} \mod 2^a]P_{AB}, \widetilde{Q_{AB}} \leftarrow [(-q\beta_2)^{-1} \mod 2^a]Q_{AB}$
$E_M, \_, (Y_M, \_), (X_M, \_)$
$\quad \leftarrow$ Isogeny22Chain$((P_B, \widetilde{P_{AB}}), (Q_B, \widetilde{Q_{AB}}), [(Y_B, 0_{E_{AB}}), (0_{E_B}, X_{AB})])$
$t \leftarrow \left(q(2^{a-2} - q)C\delta_D\right)^{-1} \mod D$
$w \leftarrow$ TateOdd$(E_M, D, X_M, Y_M, X_M - Y_M)$
$w' \leftarrow w^t$
$\widetilde{\mathsf{pad}} \leftarrow \mathsf{H}(\mathsf{KDF} \| \mathrm{tr}(w'))$
return $\mathsf{pad} \oplus c$

CHAPTER 5

参数集

本章规定 QIMEN-PIKE 所支持的各个具体参数集。一个参数集确定以下标量元组

$$\mathsf{par} = (\lambda, p, a, C_1, C_2, D), \qquad C := C_1 C_2.$$

内部参数 λ 取值为 $2\lambda_q$，其中 λ_q 为目标量子安全级别(security level)。这些值与Section 3.1中定义的基曲线及挠基一起，决定了方案所使用的公开参数。对这些参数的规范性要求见Section 3.1.1。本章将给出这些要求的具体数值实例，并列出相应的序列化长度

$$\ell_p := \left\lceil \frac{\log_2 p}{8} \right\rceil,$$

从而 $\mathbb{F}_p$ 的一个元素占用 ℓ_p 字节，$\mathbb{F}_{p^2}$ 的一个元素在最小字节编码下占用 $2\ell_p$ 字节。Table 7.1中报告的实现级大小统计基于 $\mathbb{F}_p$ 元素的编码槽长度 B_p^{enc}(即实现常量 `FP_ENCODED_BYTES`)。其中

$$B_p^{\mathrm{enc}} \in \{64, 96, 192\}$$

分别对应 NGCC-1、NGCC-2 和 NGCC-3。以 NGCC-1 为例：数学域元素在最小编码下需 60 字节，而序列化布局预留 64 字节的槽位。本文中的所有十六进制常量均为基 16 的非负整数。

5.1 具体参数集

具体常数如下。

NGCC-1.

$$\begin{aligned}
\lambda &= 160, \qquad \lceil \log_2 p \rceil = 479, \qquad a = 162, \\
C_1 &= 3^5 \cdot 7^{46}, \qquad C_2 = 11^{55}, \qquad C = C_1 C_2, \\
D &= \texttt{0xa9884081fc61c3910dd29f8b9a278f64abd4a91bcbcaf}, \\
p &= 2^{162} \cdot 3^5 \cdot 7^{46} \cdot D - 1,
\end{aligned}$$

NGCC-2.

$$\begin{aligned}
\lambda &= 256, \qquad \lceil \log_2 p \rceil = 765, \qquad a = 260, \\
C_1 &= 3 \cdot 5^{69}, \qquad C_2 = 7^{125}, \qquad C = C_1 C_2, \\
D &= \texttt{0x5239d20d58a01897cbfcc9d63b3cf9e34eeb43028119538d9503e} \\
&\quad \texttt{9c952d48d37a675ad7}, \\
p &= 2^{260} \cdot 3 \cdot 5^{69} \cdot D \cdot \texttt{0x190CA2A8D6DF6A79} - 1,
\end{aligned}$$

NGCC-3.

$$\begin{aligned}
\lambda &= 512, \qquad \lceil \log_2 p \rceil = 1534, \qquad a = 514, \\
C_1 &= 5 \cdot 7^{60}, \qquad C_2 = 11^{247}, \qquad C = C_1 C_2, \\
D &= \texttt{0x7827d2cf7534a7becfcfdf39f6058f7ed5c272e8e3a5fc87d9995} \\
&\quad \texttt{1e3af5445c77eea9c23e06ea7ff55ecb2124dcefad44cf3a4f361aa} \\
&\quad \texttt{d70f76964cb21884adc186a2c84a1a64414262c221f79b5c80defe6} \\
&\quad \texttt{9d80a585c0ba638e935d3bd2620299059658e862b43ef}, \\
p &= 2^{514} \cdot 5 \cdot 7^{60} \cdot D \cdot \texttt{0x2B271} - 1.
\end{aligned}$$

CHAPTER 6

测试向量

已知答案测试（KAT）向量覆盖所有 QIMEN-PIKE 参数集，存放在提交文件中的 `Test_Vector/`目录下。

对于 QIMEN-PIKE，文件为 `KAT_KEM_NGCC-1.txt`、`KAT_KEM_NGCC-2.txt` 和 `KAT_KEM_NGCC-3.txt`；每条记录包含一个确定性种子、一个公钥、一个私钥、一个密文以及生成的共享秘密。对应的 KAT 生成程序位于 `Implementations/`目录中。使用相同的构建选项，即可重新生成包级别的已知答案测试向量文件。

CHAPTER 7

性能分析

提交包中包含 C 语言的参考实现和优化实现，后者融合了汇编优化的有限域运算。两种实现均派生自 PIKE 实现。[1] 该代码库构建于 SQIsign 库 [AAA+25] 之上。

QIMEN-PIKE 实现的主要创新点如下：

- 定制参数选择：提出了一套专为 QIMEN-PIKE 优化的新参数集，在满足 NGCC 安全要求的同时，对有限域运算进行了精炼，以最大化计算效率。
- 广义同源框架：PIKE 实现仅限于素数幂次的一维同源，而本实现扩展了这一能力，支持任意次数。
- 密钥尺寸优化：通过精炼和集成 [CCLL26] 中的压缩技术，密文和公钥的存储开销显著减少。此外，QIMEN-PIKE 的增强压缩变体——利用确定性挠基进一步缩减数据量——目前仍在开发中。

SQIsign 库中的若干子模块提供了底层构建块，在 QIMEN-PIKE 的实现中保持不变。这些子模块为：

- `hd`：在 theta 模型中计算 $(2,2)$-同源的模块。
- `klpt`：KLPT[KLPT14] 模块。特别地，该模块用于在密钥生成中计算自同态。
- `quaternion`：四元数计算模块，同时支持基于 GMP 的任意精度运算和基于 DPE 的浮点运算。

参考实现提供以下 CMake 构建选项：

- `CMAKE_BUILD_TYPE=Release` 选择优化构建配置。
- `ENABLE_COMPRESSED=ON` 构建 NGCC KAT 生成程序所使用的压缩 PIKE 实现。
- `PIKE_NGCC_XOF_BACKEND` 选择 NGCC XOF 后端；支持的值为 `shake` 和 `sm3`。
- `ENABLE_NGCC_PIKE=ON` 构建 NGCC KAT 生成目标。KAT 输出目录由 `NGCC_PIKE_KAT_OUTPUT_DIR` 指定。

当 `ENABLE_TESTS=ON` 时，NGCC KAT 生成程序将注册为 CTest 条目。本实现提供 SHAKE 和 SM3 两种 NGCC XOF 后端：SHAKE 是标准且广泛使用的 XOF，而 NGCC 官方示例

[1] https://github.com/Kaizhan-Lin/PIKE-C-Implementation

代码提供了一个基于 SM3 的伪 XOF 构造。

7.1 密钥和密文尺寸

Table 7.1 列出了每个参数集的公钥、私钥和 密文 尺寸。PIKE 的尺寸使用编码槽长度 $B_p^{\text{enc}} = 64, 96, 192$，分别对应 NGCC-1、NGCC-2 和 NGCC-3。

Table 7.1: QIMEN-PIKE 密钥和密文尺寸（字节）。其中 pk、ct 和 sk 分别表示公钥、密文和私钥。

参数集	未压缩			压缩		
	pk	ct	sk	pk	ct	sk
NGCC-1	641 B	800 B	724 B	405 B	602 B	488 B
NGCC-2	961 B	1184 B	1103 B	595 B	882 B	737 B
NGCC-3	1921 B	2368 B	2220 B	1201 B	1746 B	1500 B

7.2 性能评估

本节报告 QIMEN-PIKE 参考实现和优化实现的性能数据，单位为 CPU 周期。

- **平台：** `x86_64` 机器上的 WSL：Intel(R) Core(TM) Ultra 9 275HX CPU @ 2.70 GHz，32 GB RAM。
- **编译器和构建系统：** GCC 13.3.0，使用 CMake `Release` 模式。
- **依赖项：** 实现链接 GMP 以支持多精度整数运算。SHAKE/FIPS202、AES/CTR-DRBG、SM3 以及 `randombytes` 程序均打包在代码库中，而非由外部密码库提供。C 语言中的有限域运算由 [Sco24] 中的自动化脚本生成。该代码库构建于 SQIsign 实现仓库 [AAA+25] 之上。
- **实现：**参考实现由 `Implementations/` 构建，汇编优化实现由 `Optimized_Implementation/` 构建。
- **哈希/XOF 后端：**遵循 NGCC 提交要求，两种构建均通过配置 `PIKE_NGCC_XOF_BACKEND=sm3` 使用 SM3 后端，该配置将 `PIKE_XOF_BACKEND` 设为 2。
- **编译设置：**有效的 C 编译标志包括 `-O3`、`-DNDEBUG`、`-std=c99`、`-fvisibility=hidden` 和 `-funroll-loops`；两种构建均定义了 `RADIX_64`、`TARGET_AMD64` 和 `TARGET_OS_UNIX`。
- **参数集：** 报告的基准测试使用 NGCC-1、NGCC-2 和 NGCC-3。
- **测量：** 每个报告的 KEM 操作均为 300 次运行的平均值。

7.2.1 参考实现

Table 7.2 给出了参考实现的性能数据。

Table 7.2: 参考实现 QIMEN-PIKE KEM 在使用 SM3 作为 XOF 时，未压缩与压缩变体的性能，单位为 10^6 CPU 周期，在 Intel(R) Core(TM) Ultra 9 275HX CPU 上测量，300 次运行取平均。

参数集	未压缩			压缩		
	KeyGen	Encaps	Decaps	KeyGen	Encaps	Decaps
NGCC-1	94.15	34.73	61.43	159.61	61.27	98.02
NGCC-2	350.45	121.33	222.20	608.92	223.68	360.88
NGCC-3	3005.44	1029.46	1794.11	4606.19	1803.16	2845.02

Table 7.3: 参考实现 QIMEN-PIKE KEM 在使用 SM3 作为 XOF 时，未压缩与压缩变体的吞吐量，单位为每秒操作数（ops/s），在 Intel(R) Core(TM) Ultra 9 275HX CPU 上测量，300 次运行取平均。

参数集	未压缩			压缩		
	KeyGen	**Encaps**	**Decaps**	**KeyGen**	**Encaps**	**Decaps**
NGCC-1	30.4	84.7	47.5	17.4	47.5	29.5
NGCC-2	7.9	22.0	12.8	4.5	13.0	8.1
NGCC-3	0.9	2.62	1.48	0.62	1.47	0.97

7.2.2 优化实现

优化构建使用了

`PIKE_BUILD_TYPE=ref-mbmi2-madx`,

启用了 x86-64 BMI2/ADX 汇编有限域运算。Table 7.4 给出了该平台上的汇编优化实现性能。

Table 7.4: 汇编优化 QIMEN-PIKE KEM 在使用 SM3 作为 XOF 时，未压缩与压缩变体的性能，单位为 10^6 CPU 周期，在 Intel(R) Core(TM) Ultra 9 275HX CPU 上测量，300 次运行取平均。

参数集	未压缩			压缩		
	KeyGen	Encaps	Decaps	KeyGen	Encaps	Decaps
NGCC-1	76.16	26.94	47.93	129.31	48.00	77.45
NGCC-2	292.40	93.86	171.85	481.88	172.94	280.08
NGCC-3	2480.87	803.12	1405.36	3639.78	1421.09	2243.85

Table 7.5: 优化实现 QIMEN-PIKE KEM 在使用 SM3 作为 XOF 时，未压缩与压缩变体的吞吐量，单位为每秒操作数（ops/s），在 Intel(R) Core(TM) Ultra 9 275HX CPU 上测量，300 次运行取平均。

参数集	未压缩			压缩		
	KeyGen	**Encaps**	**Decaps**	**KeyGen**	**Encaps**	**Decaps**
NGCC-1	34.9	104.2	54.7	20.7	55.0	36.8
NGCC-2	10.5	32.8	17.8	5.6	17.0	10.8
NGCC-3	1.08	3.78	2.15	0.83	2.07	1.31

CHAPTER 8

安全性

本章讨论 QIMEN-PIKE 的主要安全特性，遵循 [CCLL26] 和 [BM25] 中的分析。首先，Section 8.1讨论自同态环问题（endomorphism ring problem）的复杂度——这是支撑超奇异同源密码安全性的核心困难假设。接着，Section 8.2在Theorem 8.2.1的困难性下，给出 QIMEN-PIKE 在量子随机预言机模型（quantum random-oracle model）中满足 IND-CCA2 安全性的结论，并简要讨论Theorem 8.2.1与 POKÉ[BM25] 安全性之间的关系。然后，Section 8.3考察针对 QIMEN-PIKE 的不同具体攻击的复杂度。最后，Section 8.4总结各攻击，并证明我们所选的参数满足 NGCC 所要求的经典和量子安全级别。

8.1 自同态环问题

自同态环问题是所有（超奇异）同源密码学的共同基础问题，包括 QIMEN-PIKE，因此其困难性是攻破本方案所需代价的一个上界。如 Section 2.6.2 所述，超奇异椭圆曲线 E 在 $\mathbb{F}_{p^2}$ 上的自同态环同构于一个极大序 $\mathcal{O} \subset \mathcal{B}_{p,\infty}$。自同态环问题要求显式地计算出该序。

Problem 8.1.1 (自同态环问题)**.** 给定一条超奇异椭圆曲线 $E/\mathbb{F}_{p^2}$，返回 $b_1, b_2, b_3 \in \mathcal{B}_{p,\infty}$ 以及自同态 $\beta_1, \beta_2, \beta_3$ 的高效表示，使得由

$$1 \mapsto \mathrm{Id}, \qquad b_1 \mapsto \beta_1, \qquad b_2 \mapsto \beta_2, \qquad b_3 \mapsto \beta_3$$

定义的从 $\mathcal{O} := \langle 1, b_1, b_2, b_3 \rangle_{\mathbb{Z}}$ 到 $\mathrm{End}(E)$ 的 $\mathbb{Z}$ 线性映射是一个环同构。

这里，两条超奇异椭圆曲线 E, E' 在 $\mathbb{F}_{p^2}$ 上的同源 $\varphi : E \to E'$ 的高效表示，是指一个求值算法，其运行时间是 $\log p$、$\deg \varphi$ 以及输入点 $P \in E(\mathbb{F}_{p^{2k}})$ 的定义域扩张次数 k 的多项式函数。自同态环问题的若干变种已有研究，例如仅计算 b_1, b_2, b_3，使得由 $1, b_1, b_2, b_3$ 生成的序存在到 $\mathrm{End}(E)$ 的同构；这有时称为极大序问题。所有这些变种均已被证明是多项式时间等价的 [Wes22]。针对 Problem 8.1.1的最快经典算法，运行时间为 $\tilde{O}(p^{1/2})$ [DG16]。这些算法可借助 Grover 量子加速，得到运行时间为 $\tilde{O}(p^{1/4})$ 的量子方法 [Gro96, BJS14]。在这两种情形下，这些方法的内存需求均很低。因此，为达到 λ_q 比特的量子安全级别，素数 p 至少需要约 $4\lambda_q$ 比特。

8.2 理论安全性

后量子安全的 KEM QIMEN-PIKE 通过先构造 QIMEN-PIKE.PKE（一个 IND-CPA PKE，遵循 ElGamal 范式，见Section 3.2），然后应用 Fujisaki-Okamoto 转换 [FO99] 得到 QIMEN-PIKE.KEM（一个 IND-CCA2 KEM，见Section 3.3）。

QIMEN-PIKE 安全性的证明路径为：首先识别支撑 QIMEN-PIKE.PKE IND-CPA 安全性的困难问题，然后在该问题的困难性下，借助 Fujisaki-Okamoto 转换的经典结果，导出 QIMEN-PIKE.KEM 的 IND-CCA2 安全性。该核心问题称为计算 PIKE 问题（Computational PIKE Problem），定义如下。

Problem 8.2.1 （计算 PIKE 问题（Computational PIKE Problem）[CCLL26, 问题 9]）**.** 该问题由 pp 参数化。$\mathsf{pp} = (p, a, C, D, E_0, (P_0, Q_0), (R_0, S_0), (X_0, Y_0))$，其中 E_0 是定义在 $\mathbb{F}_{p^2}$ 上的超奇异椭圆曲线，其自同态环已知，$\{P_0, Q_0\}$、$\{R_0, S_0\}$ 和 $\{X_0, Y_0\}$ 分别是 $E_0[2^a]$, $E_0[C]$ 和 $E_0[D]$ 的基。

挑战者首先随机生成一个同源 $\phi : E_0 \to E_A$，次数为 $q(2^{a-2} - q)$，其中 $q < 2^{a-2}$ 为某个未知值且与 2、C 和 D 互质，然后计算 $\mathsf{x}_1 = (E_A, P_A, Q_A, R_A, S_A, X_A)$，其中

$$P_A, Q_A, R_A, S_A, X_A = \phi([\alpha_2]P_0,\ [\beta_2]Q_0,\ [\gamma_C]R_0,\ [\gamma_C]S_0,\ [\delta_D]X_0)$$

且 $\alpha_2, \beta_2, \gamma_C, \delta_D$ 分别从 $(\mathbb{Z}/n\mathbb{Z})^\times$ 中均匀随机采样，其中 $n = 2^a, 2^a, C, D$。

然后，挑战者随机生成一个次数为 C 的同源 $\psi : E_0 \to E_B$，并记 $\psi' : E_A \to E_{AB}$ 为其前推 $[\phi]_*\psi$。然后，挑战者计算 $\mathsf{x}_2 = (P_B, Q_B, Y_B, P_{AB}, Q_{AB}, X_{AB})$，定义如下：

$$\begin{aligned} &P_B, Q_B, Y_B, P_{AB}, Q_{AB}, X_{AB} \\ &= \psi([\omega_2]P_0,\ [\omega_2^{-1}]Q_0,\ [\eta_D]Y_0),\ \psi'([\omega_2]P_A,\ [\omega_2^{-1}]Q_A,\ [\zeta_D]X_A), \end{aligned}$$

其中 $\omega_2, \eta_D, \zeta_D$ 分别从 $(\mathbb{Z}/n\mathbb{Z})^\times$ 中均匀随机抽取，$n = 2^a, D, D$。

给定 $\mathsf{pp}, \mathsf{x}_1, \mathsf{x}_2$，计算 $e(X_0, Y_0)^{\eta_D\zeta_D}$。

我们将敌手 $\mathcal{A}$ 的优势记为 $\mathsf{Adv}^{\mathsf{CPIKE}}_{\mathsf{pp}}(\mathcal{A}) = \Pr[\mathcal{A}\text{ win.}]$，其中概率取自游戏中所用的随机性和 $\mathcal{A}$ 的随机性。

8.2.1 QIMEN-PIKE.PKE 的 IND-CPA 安全性

QIMEN-PIKE.PKE 的正确性首先由 [CCLL26, 命题 8] 建立，此后该方案的 IND-CPA 安全性在Theorem 8.2.1下得到证明。

Theorem 8.2.2 ([CCLL26, 定理 10])**.** 若*Theorem 8.2.1*是困难的，则通过将 H 建模为随机预言机，*Chapter 3*中描述的 *PKE* 方案 Π_{PKE} 是 *IND-CPA* 安全的。确切地，给定一个针对 Π_{PKE} *IND-CPA* 的敌手 $\mathcal{A}$，其优势为 ϵ，对随机预言机进行 Q 次查询，运行时间为 T，我们可以构造一个运行时间为 $O(T)$ 的敌手 $\mathcal{B}$，针对*Theorem 8.2.1*，使得

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi_{\mathsf{PKE}},\mathsf{pp}}(\mathcal{A}) \le 2Q \cdot \mathsf{Adv}^{\mathsf{CPIKE}}_{\mathsf{pp}}(\mathcal{B}).$$

8.2.2 PIKE.KEM 的 IND-CCA2 安全性

[HHK17, JZM19, DFMS22] 的结果足以在经典随机预言机模型或量子随机预言机模型下，从 QIMEN-PIKE.PKE 的 IND-CPA 安全性推出 QIMEN – PIKE.CCAKEM 的 IND-CCA2 安全性。形式化陈述如下。

Theorem 8.2.3 (PIKE.KEM 的 IND-CCA2 安全性)**.** 设 $\mathcal{M}$ 表示 *QIMEN-PIKE.PKE* 的消息空间。若 *QIMEN-PIKE.PKE* 是 *IND-CPA* 安全的，则 *QIMEN-PIKE.KEM* 在经典 *ROM* 和 *QROM* 中是 *IND-CCA2* 安全的。具体地，对于任意一个针对 *QIMEN-PIKE.KEM* 的 *IND-CCA* 敌手 $\mathcal{A}$，其对随机预言机 G 的查询至多 q_G 次，对随机预言机 H_{ct} 和 KDF 的查询合计至多 q_H 次，我们有：

(1) 在经典随机预言机模型 *[HHK17]* 中，存在一个针对 *QIMEN-PIKE.PKE* 的 *IND-CPA* 敌手 $\mathcal{B}_{\mathrm{cROM}}$，使得

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathit{QIMEN\text{-}PIKE.KEM},\mathsf{pp}}(\mathcal{A}) \le 3\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathit{QIMEN\text{-}PIKE.PKE},\mathsf{pp}}(\mathcal{B}_{\mathrm{cROM}}) + \Delta_{\mathrm{cROM}}(\mathcal{A}),$$

其中

$$\Delta_{\mathrm{cROM}}(\mathcal{A}) := \frac{2q_G + q_H + 1}{|\mathcal{M}|}.$$

(2) 在量子随机预言机模型 *[JZM19]* 中，存在一个针对 *QIMEN-PIKE.PKE* 的 *IND-CPA* 敌手 $\mathcal{B}_{\mathrm{qROM}}$，使得

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathit{QIMEN\text{-}PIKE.KEM},\mathsf{pp}}(\mathcal{A}) \le 2\sqrt{q_G + q_H + 1}\,\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathit{QIMEN\text{-}PIKE.PKE},\mathsf{pp}}(\mathcal{B}_{\mathrm{qROM}}) + \Delta_{\mathrm{qROM}}(\mathcal{A}),$$

其中

$$\Delta_{\mathrm{qROM}}(\mathcal{A}) := \frac{2q_H}{\sqrt{|\mathcal{M}|}} + \frac{2(q_G + q_H + 1)^2}{|\mathcal{M}|}.$$

8.2.3 与 POKÉ 的（不）等价性

一个自然的问题是：Theorem 8.2.1与支撑 POKÉ 安全性的 C-POKÉ 问题 [BM25, 问题 7] 之间是否存在单向归约。对此，[CCLL26] 指出，严格意义上的答案是否定的——两者设置中存在细微但实质性的差异。在 PIKE 的公钥中，暴露的挠点比 POKÉ 少一个，但其阶更大；此外，PIKE 虽然提供了一个更高阶的挠点，但并未像 POKÉ 那样暴露基赋值。因此，就密钥恢复型攻击而言，很难断言哪种公钥本质上更健壮。

对于 E_B 和 E_{AB} 上的密文挠数据，POKÉ 在 E_B 上发布两个挠点求值，而 PIKE 分别在 E_B 和 E_{AB} 上各发布一个。这些结构差异表明，要在两个假设之间建立干净的等价关系，可能性不大。

8.3 实际安全性

本节使用Fig. 3.1中的记号。

8.3.1 同源恢复问题

遵循 [CCLL26]，为分析 QIMEN-PIKE 的实际安全性，从Theorem 8.2.1派生出两个子问题，每个子问题都是恢复一个秘密同源。第一个问题涉及接收方的秘密同源，对应于恢复私钥；第二个问题涉及发送方的秘密同源，对应于恢复临时同源（nonce isogeny）。解决其中任一问题即足以攻破 QIMEN-PIKE 的安全性。需注意，在两条给定的超奇异椭圆曲线之间寻找同源的一般问题，等价于Problem 8.1.1[PW24, HLMW26]。

Problem 8.3.1 (接收方同源恢复问题（Receiver Isogeny Recovery Problem）[CCLL26, 问题 13])**.** 该问题由与Theorem 8.2.1相同的公开参数 $\mathsf{pp} = (p, a, C, D, E_0, (P_0, Q_0), (R_0, S_0), (X_0, Y_0))$ 参数化。挑战者采样一个同源 $\phi : E_0 \to E_A$，次数为 $q(2^{a-2} - q)$，其中 $q < 2^{a-2}$ 不为敌手所知且与 2、C 和 D 互质。然后采样掩码标量 $\alpha_2, \beta_2 \xleftarrow{\$} (\mathbb{Z}/2^a\mathbb{Z})^\times$，$\gamma_C \xleftarrow{\$} (\mathbb{Z}/C\mathbb{Z})^\times$，和 $\delta_D \xleftarrow{\$} (\mathbb{Z}/D\mathbb{Z})^\times$，并设 $\mathsf{x}_1 := (E_A, P_A, Q_A, R_A, S_A, X_A)$，其中

$$\begin{pmatrix} P_A \\ Q_A \end{pmatrix} = \begin{pmatrix} \alpha_2 & 0 \\ 0 & \beta_2 \end{pmatrix} \phi \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix}, \qquad \begin{pmatrix} R_A \\ S_A \end{pmatrix} = \begin{pmatrix} \gamma_C & 0 \\ 0 & \gamma_C \end{pmatrix} \phi \begin{pmatrix} R_0 \\ S_0 \end{pmatrix}, \qquad X_A = [\delta_D]\, \phi(X_0). \tag{8.1}$$

给定 $(\mathsf{pp}, \mathsf{x}_1)$，恢复 ϕ（的显式表示）。

Problem 8.3.2 (发送方同源恢复问题（Sender Isogeny Recovery Problem）[CCLL26, 问题 14])**.** 该问题由与上述相同的公开参数 pp 参数化。挑战者首先如Theorem 8.3.1生成 $\mathsf{x}_1 := (E_A, P_A, Q_A, R_A, S_A, X_A)$。然后均匀随机采样一个次数为 C 的同源 $\psi : E_0 \to E_B$，并令 $\psi' : E_A \to E_{AB}$ 表示前推 $[\phi]_*\psi$。最后采样 $\omega_2 \xleftarrow{\$} (\mathbb{Z}/2^a\mathbb{Z})^\times$ 和 $\eta_D, \zeta_D \xleftarrow{\$} (\mathbb{Z}/D\mathbb{Z})^\times$，并通过以下定义 $\mathsf{x}_2 := (E_B, P_B, Q_B, Y_B, E_{AB}, P_{AB}, Q_{AB}, X_{AB})$：

$$\begin{pmatrix} P_B \\ Q_B \end{pmatrix} = \begin{pmatrix} \omega_2 & 0 \\ 0 & \omega_2^{-1} \end{pmatrix} \cdot \psi \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix}, \qquad \begin{pmatrix} P_{AB} \\ Q_{AB} \end{pmatrix} = \begin{pmatrix} \omega_2 & 0 \\ 0 & \omega_2^{-1} \end{pmatrix} \cdot \psi' \begin{pmatrix} P_A \\ Q_A \end{pmatrix}, \tag{8.2}$$
$$Y_B = [\eta_D]\, \psi(Y_0), \qquad X_{AB} = [\zeta_D]\, \psi'(X_A).$$

给定 $(\mathsf{pp}, \mathsf{x}_1, \mathsf{x}_2)$，恢复 ψ 或 ψ'。

在分别研究它们的具体复杂度之前，首先评估Theorems 8.3.1 and 8.3.2在面对已知同源恢复方法时的安全性。遵循 [CCLL26, 第 5.2 节] 的分析，以下重点考察三类攻击 [CV23, DFP24, BM25]。第一类攻击 [CV23, DFP24] 推广了当敌手获得已知次数同源的掩码基赋值时的密钥恢复型攻击；其核心观察是某些对角掩码矩阵与自同态交换，可以在特定设置下归约到 SIDH 风格的恢复上。这并不适用于Theorem 8.3.1：在Eq. (8.1)中，关于 ϕ 的关键次数信息被行列式均匀分布的独立掩码标量所隐藏，无法通过配对恢复。对于Theorem 8.3.2，其掩码结构不同于这些攻击所利用的交换对角形式。

第二类攻击将类似思路扩展到恶意选取挠基下的 FESTA 型设置 [CV23]。这在确定性（规范）基生成下不适用于Theorem 8.3.2中的 E_B 和 E_{AB} 分量，而且通信记录并未暴露足够多的挠点以使归约策略得以施行。

最后，[BM25] 在特定掩码条件下给出了针对 $(q, 2^{a-2} - q)$-同源的高效恢复攻击；Eq. (8.1)中的第一个方程不符合所需形式，因此该技术不适用于Theorem 8.3.1。

更一般地，如 [DFP24] 的 Fig. 1 所示，针对此处所用掩码同源实例的最知名通用方法仍然需要指数级时间。总之，目前不存在从Theorem 8.2.1的通信记录中单个分量高效恢复同源的攻击。

以下对适用于这些问题的攻击进行精确的复杂度分析。

8.3.2 密钥恢复（针对Theorem 8.3.1的攻击）

私钥由 α_2、β_2、δ_D 和 $\deg\phi$ 组成。利用 Tate 配对结合 SIDH 密钥恢复攻击，可将恢复这些量归约为求解Theorem 8.3.1——即计算秘密同源 ϕ。这类攻击的关键在于：利用秘密同源 ϕ 下的挠点像来高效重构该同源。

然而，恢复私钥并不需要恢复其每一个分量。此外，虽然 γ_C 不是私钥的一部分，但知晓它同样足以恢复 ϕ。以下逐一讨论这些要点。

假设 $C > \sqrt{\deg\phi}$。在此假设下，知晓秘密值 q（等价于知晓 $\deg\phi$）即可使敌手恢复 ϕ。实际上，由于 C 是光滑的，γ_C 可以通过 Tate 配对计算高效恢复，从而得到 $\phi(R_0)$ 和 $\phi(S_0)$。然后可以将这些数据输入 SIDH 密钥恢复过程以获得秘密同源 ϕ。

若敌手获得了掩码标量 γ_C，则可以通过 Tate 配对恢复 $\deg\phi \mod C$。由于 q 的大小与 $\sqrt{\deg\phi}$ 近似，敌手可以通过求解二次方程

$$q(2^{a-2} - q) \equiv \deg\phi \mod C$$

来计算 q。这意味着敌手可以再次使用 SIDH 密钥恢复过程恢复私钥 ϕ。

此外，[BM25, 第 6.1 节] 指出，可以通过二维同源 Φ 来表示 ϕ 从而恢复它，该同源的核为

$$\ker\Phi = \langle([-\alpha_2 q]P_0, [\alpha_2]\phi(P_0)), ([-\alpha_2 q]Q_0, [\alpha_2]\phi(Q_0))\rangle.$$

当 $\alpha_2 = \beta_2$ 时，点 $[\alpha_2]\phi(P_0) = P_1$ 和 $[\alpha_2]\phi(Q_0) = Q_1$ 都是公开的，可以从 Tate 配对计算中提取出 $-\alpha_2 q \bmod 2^a$。因此，可以高效计算 Φ 的核。

通过设定 $\alpha_2 \neq \beta_2$ 可以防御此攻击。然而，若给定比值 α_2/β_2，敌手仍然可以进行类似的恢复。实际上，通过计算 $[\alpha_2/\beta_2]Q_1 = [\alpha_2]\phi(Q_0)$，同样可以使用 Tate 配对得到 $-\alpha_2 q \bmod 2^a$。然后 Φ 的核可以高效计算，因为 $-\alpha_2 q \bmod 2^a$、$P_1 = [\alpha_2]\phi(P_0)$ 和 $[\alpha_2/\beta_2]Q_1 = [\alpha_2]\phi(Q_0)$ 都是已知的。

总之，恢复秘密同源 ϕ 可归约为获取以下任一信息：秘密值 q、掩码标量 γ_C 或比值 α_2/β_2。以下逐一分析每个秘密值的计算难度，以推导安全参数。

- 关于秘密值 q，大约有 2^{a-2} 个候选，为 $O(2^a)$。因此，敌手可以通过暴力搜索在经典时间 $O(2^a)$ 内恢复 q。若拥有量子资源，利用 Grover 算法 [Gro96]，该复杂度可降至 $O(2^{a/2})$。
- 掩码标量 γ_C 在 $(\mathbb{Z}/C\mathbb{Z})^\times$ 上均匀分布，其基数约为 C。但需要注意，实际上需要 $\gamma_C \bmod C'$（其中 C' 满足 $C' \mid C$ 且 $C' \approx \sqrt{\deg\phi} \approx 2^{a-2}$）来恢复 ϕ，因为只需要 C'-挠点来计算 q。因此，通过测试 $O(2^a)$ 个 $\gamma_C \bmod C'$ 的候选，敌手可以计算 ϕ。因此，该方法在经典计算机上的复杂度为 $O(2^a)$，在量子计算机上为 $O(2^{a/2})$。

- 掩码标量 α_2 和 β_2 假设在 $(\mathbb{Z}/2^a\mathbb{Z})^\times$ 上均匀分布。因此，它们的比值 α_2/β_2 同样在 $(\mathbb{Z}/2^a\mathbb{Z})^\times$ 上均匀分布，该集合的基数约为 2^a；故敌手可以通过穷举搜索在经典时间 $O(2^a)$ 和量子时间 $O(2^{a/2})$ 内恢复比值 α_2/β_2。

综上，恢复 PIKE 的私钥在经典计算机上至少需要 $O(2^a)$ 时间，在量子计算机上至少需要 $O(2^{a/2})$ 时间。

8.3.3 临时同源恢复（针对Theorem 8.3.2的攻击）

若能恢复临时同源 ψ（或 ψ'），则可相应推导出共享秘密。恢复 ψ 主要有两种方法：

- 通过 E_0 和 E_B 之间的中间相遇攻击恢复 ψ；
- 先恢复掩码临时值 ω_2，然后应用 SIDH 密钥恢复攻击获得 ψ。

在第一种方法中，敌手计算两个同源，它们的次数乘积为 C，分别从 E_0 和 E_B 出发，每个同源的次数约为 $\sqrt{C}$。若它们的像曲线相同，则复合它们得到 E_0 和 E_B 之间次数为 C 的同源。在理想情况下，当 ψ 是唯一这样的同源时，该重构立即成功。一般地，E_0 和 E_B 之间可能存在多个次数为 C 的同源，但找到其中任何一个在经典计算机上已需要 $O(\sqrt{C})$ 次操作。借助量子资源，使用基于量子行走 [Tan09] 的碰撞查找算法，复杂度可降至 $O(C^{1/3})$。这给出了用该方法恢复 ψ 的经典和量子代价的下界。

在第二种方法中，敌手通过穷举搜索确定掩码标量 ω_2。注意到 ω_2 从群 $(\mathbb{Z}/2^a\mathbb{Z})^\times$ 中均匀随机采样，该群的大小约为 2^a。

因此，第二种方法在经典计算机上需要 $O(2^a)$ 时间。通过量子计算可降低此复杂度，得到运行时间为 $O(2^{a/2})$。

8.3.4 消息恢复

PIKE 的共享密钥为 $e_D(X_0, Y_0)^{\eta_D\zeta_D}$。敌手可能尝试直接猜测 $e_D(X_0, Y_0)^{\eta_D\zeta_D}$。

由于 η_D 和 ζ_D 均匀随机选取，共享值在 μ_D（大小为 D 的群）中也是均匀分布的。因此，敌手在经典计算机上成功揭示共享值需要 $O(D)$ 时间，在量子计算机上通过 Grover 算法则需要 $O(\sqrt{D})$ 时间。

8.4 参数安全评估

表 8.1和8.2分别报告了以Chapter 5中的参数实例化 QIMEN-PIKE 时，最知名经典攻击和量子攻击的代价，确认达到了目标安全级别。本节表格使用以下记号。

$\star$: 此攻击的复杂度也对应于针对临时同源恢复问题的第一类攻击。

§: 指数向上取整，例如将 $2^{79.7\cdots}$ 记为 2^{80}。

Table 8.1: QIMEN-PIKE 的参数选择与经典攻击复杂度

		NGCC-1	**NGCC-2**	**NGCC-3**
			参数	
	a	162	260	514
	C_1	$3^5 \cdot 7^{46}$	$3 \cdot 5^{69}$	$5 \cdot 7^{60}$
	C_2	11^{55}	7^{125}	11^{247}
	C	C_1C_2	C_1C_2	C_1C_2
	D	0xa9884081fc61c3910dd29f8b9a278f64abd4a91bcbcaf	0x5239d20d58a01897cbfcc9d63b3cf9e34eeb43028119538d9503e9c952d48d37a675ad7	0x7827d2cf7534a7becfcfdf39f6058f7ed5c272e8e3a5fc87d99951e3af5445c77eea9c23e06ea7ff55ecb2124dcefad44cf3a4f361aad70f76964cb21884adc186a2c84a1a64414262c221f79b5c80defe69d80a585c0ba638e935d3bd2620299059658e862b43ef
	D'		0x190CA2A8D6DF6A79	0x2B271
	p	$2^{162} \cdot 3^5 \cdot 7^{46} \cdot D - 1$	$2^{260} \cdot 3 \cdot 5^{69} \cdot D \cdot D' - 1$	$2^{514} \cdot 5 \cdot 7^{60} \cdot D \cdot D' - 1$
所需经典安全级别		128	256	512
经典攻击	**复杂度**		**复杂度估计** [§]	
针对 Problem 8.1.1	$\tilde{O}(p^{1/2})$	2^{239}	2^{382}	2^{767}
密钥恢复*	$\tilde{O}(2^a)$	2^{162}	2^{260}	2^{514}
临时同源恢复	$\tilde{O}(C^{1/2})$	2^{164}	2^{256}	2^{513}
消息恢复	$\tilde{O}(D)$	2^{179}	2^{282}	2^{831}

Table 8.2: QIMEN-PIKE 的参数选择与量子攻击复杂度

		NGCC-1	**NGCC-2**	**NGCC-3**
			参数	
	a	162	260	514
	C_1	$3^5 \cdot 7^{46}$	$3 \cdot 5^{69}$	$5 \cdot 7^{60}$
	C_2	11^{55}	7^{125}	11^{247}
	C	C_1C_2	C_1C_2	C_1C_2
	D	0xa98840 81fc61c3 910dd29f 8b9a278f 64abd4a9 1bcbcaf	0x5239d20d58 a01897cbfcc9 d63b3cf9e34e eb4302811953 8d9503e9c952 d48d37a675ad 7	0x7827d2cf753 4a7becfcfdf39 f6058f7ed5c27 2e8e3a5fc87d9 9951e3af5445c 77eea9c23e06e a7ff55ecb2124 dcefad44cf3a4 f361aad70f769 64cb21884adc1 86a2c84a1a644 14262c221f79b 5c80defe69d80 a585c0ba638e9 35d3bd2620299 059658e862b43 ef
	D'		0x190CA2A8D6 DF6A79	0x2B271
	p	$2^{162} \cdot 3^5 \cdot 7^{46} \cdot D - 1$	$2^{260} \cdot 3 \cdot 5^{69} \cdot D \cdot D' - 1$	$2^{514} \cdot 5 \cdot 7^{60} \cdot D \cdot D' - 1$
所需量子安全级别		80	128	256
量子攻击	**复杂度**		**复杂度估计** [§]	
针对 Problem 8.1.1	$\tilde{O}(p^{1/4})$	2^{120}	2^{191}	2^{383}
密钥恢复*	$\tilde{O}(2^{a/2})$	2^{81}	2^{130}	2^{257}
临时同源恢复	$\tilde{O}(C^{1/3})$	2^{109}	2^{171}	2^{342}
消息恢复	$\tilde{O}(D^{1/2})$	2^{90}	2^{141}	2^{415}

CHAPTER 9

失败概率分析

本章列举并分析 QIMEN-PIKE 中各类算法可能出现的失败情形。具体而言，本章首先讨论失败概率分析所依据的启发式假设，然后计算三个安全级别下各易失败算法的失败概率，并在分析中解释相关参数的选择依据。

在 QIMEN-PIKE 中，可能抛出异常的算法包括GeneralizedRepresentInteger以及二维同源链相关程序。密钥生成期间，以下异常可能在QIMEN-PIKE.CoreKeyGenIso内部发生：调用GeneralizedRepresentInteger时，有界搜索可能未找到目标范数的四元数元素；调用Isogeny22Chain时，内部某个二维同源子程序可能失败。这些异常会以密钥生成失败的形式向上传播给调用者。解密期间，QIMEN-PIKE.PKE.Decrypt同样可能向上传播Isogeny22Chain的异常；QIMEN-PIKE.Decaps捕获该异常后，返回基于秘密值 z 和密文哈希导出的隐式拒绝回退密钥。

以下各节分别分析GeneralizedRepresentInteger和Isogeny22Chain的失败概率。GeneralizedRepresentInteger的失败概率分析与 [AAA+25] 中的分析高度一致，仅将数值调整为本方案设定下的对应值。Isogeny22Chain的分析与 [AAA+25] 几乎完全相同，故以星号上标标注该节。

9.1 GeneralizedRepresentInteger的失败概率分析

GeneralizedRepresentInteger成功当且仅当算法执行了 Line 10，即CornacchiaGeneral返回了 M' 表示为两平方和的一个表示。在下述启发式假设（Heuristic 9.1.1）下，M' 为模 4 余 1 的素数的概率约为 $1/(2\log M')$；这是CornacchiaGeneral成功的充分条件，由此可得成功概率的一个保守下界。在 QIMEN-PIKE 中，$M = q(2^{a-2} - q)C$（如Chapter 5中指定），其中 $q < 2^{a-2}$ 在QIMEN-PIKE.CoreKeyGenIso中成立。对于所有三个 PIKE 参数集，每当 $M' := 4M - p(z^2 + t^2) > 0$ 时，有 $M' < 4M < p^{1.5}$。因此 $\log M' \leq 1.5 \log p$，成功概率的保守下界为 $1/(3\log(p))$。

Heuristic 9.1.1. 由二次型表示的整数在素性和同余条件方面表现得如同相同大小的随机整数。

令 B 表示GeneralizedRepresentInteger中循环的迭代次数，则失败概率的上界

为

$$\left(1-\frac{1}{3\log(p)}\right)^{B}.$$

一旦 B 大于 $3\lambda_c \log p$——在 QIMEN-PIKE 中始终成立——GeneralizedRepresentInteger的失败率便以可忽略的速率 $e^{-\lambda_c}$ 为上界。

9.2 Isogeny22Chain的失败概率分析 *

在 QIMEN-PIKE 的密钥生成和解密过程中，需计算若干如下形式的同源链：

$$E_1 \times E_2 \xrightarrow{\Phi_1} A_1 \xrightarrow{\Phi_2} A_2 \cdots A_{e-2} \xrightarrow{\Phi_{e-1}} A_{e-1} \xrightarrow{\Phi_e} E_3 \times E_4.$$

Section 2.5中给出的算法并非通用的，且可能以两种方式失败。可以论证，在诚实执行过程中这些失败以可忽略的概率发生，因此在密钥生成中可以忽略。如果在解密过程中发生，则以压倒性概率表明密文是畸形的，从而令QIMEN-PIKE.Decaps返回隐式拒绝回退密钥。

第一种失败情形发生在链的最终步骤之前遇到分裂时，即当 $1 \le k < e$ 时，A_k 具有乘积 theta 结构。为估计密钥生成和解密中失败概率的上界，引入以下启发式假设。

Heuristic 9.2.1. 在QIMEN-PIKE.PKE.KeyGen和QIMEN-PIKE.PKE.Decrypt中计算的 $(2,2)$ 同源链上出现的曲面 A_k，其行为如同均匀随机超特异 PPAS。

超奇异椭圆曲线乘积 $E_1 \times E_2$ 的个数为 $O(p^2)$，而超特异 PPAS 的个数为 $O(p^3)$，因此在该启发式假设下，$(2,2)$ 同源链的计算以 $\widetilde{O}(p^{-1})$ 的概率失败。

第二种失败情形发生在第一次粘合同源 $E_1 \times E_2 \to A$ 中，当ThetaChangeOfBasis中计算的基变换矩阵为 0 时。这仅在坐标乘积 $X_1 \cdot X_2$ 在核下的迹为零时发生，其中 $(X_1 : Z_1)$ 和 $(X_2 : Z_2)$ 分别是 E_1 和 E_2 上的 Montgomery 坐标。

Heuristic 9.2.2. 令 $E_1 \times E_2$ 为在QIMEN-PIKE.PKE.KeyGen和QIMEN-PIKE.PKE.Decrypt的诚实执行过程中，其上进行二维同源计算的椭圆曲线乘积。则乘积坐标 $X_1 \cdot X_2$ 在粘合核下的迹的行为，如同 $\mathbb{F}_{p^2}$ 上一维向量空间中的独立随机元素。

显然遇到零迹的概率为 $O(p^{-2})$，且该计算仅涉及 $O(1)$ 个二维同源；因此第二种类型失败的总失败概率为 $O(p^{-2})$。

Bibliography

[AAA+25] Marius A. Aardal, Gora Adj, Diego F. Aranha, Andrea Basso, Isaac Andrés Canales Martínez, Jorge Chávez-Saab, Maria Corte-Real Santos, Pierrick Dartois, Luca De Feo, Max Duparc, Jonathan Komada Eriksen, Tako Boris Fouotsa, Décio Luiz Gazzoni Filho, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Luciano Maino, Michael Meyer, Kohei Nakagawa, Hiroshi Onuki, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Giacomo Pope, Krijn Reijnders, Damien Robert, Francisco Rodríguez-Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign. Technical report, National Institute of Standards and Technology, 2025.

[AAB+22] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[BJS14] Jean-François Biasse, David Jao, and Anirudh Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves. In Willi Meier and Debdeep Mukhopadhyay, editors, *INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 428–442, New Delhi, India, December 14–17, 2014. Springer, Cham, Switzerland.

[BM25] Andrea Basso and Luciano Maino. POKÉ: A compact and efficient PKE from higher-dimensional isogenies. In *EUROCRYPT 2025, Part II*, LNCS, pages 94–123. Springer, Cham, Switzerland, June 2025.

[BMP23] Andrea Basso, Luciano Maino, and Giacomo Pope. FESTA: Fast encryption from supersingular torsion attacks. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 98–126, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

[Can89] David G Cantor. On arithmetical algorithms over finite fields. *Journal of Combinatorial Theory, Series A*, 50(2):285–300, 1989.

[CCLL26] Shiping Cai, Mingjie Chen, Yi-Fu Lai, and Kaizhan Lin. Pike: Faster isogeny-based public key encryption withăpairing-assisted decryption. In Shi Bai and Edoardo Persichetti, editors, *Public-Key Cryptography – PKC 2026*, pages 58–91, Cham, 2026. Springer Nature Switzerland.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 423–447, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[CH17] Craig Costello and Hüseyin Hisil. A simple and compact algorithm for SIDH with arbitrary degree isogenies. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 303–329, Hong Kong, China, December 3–7, 2017. Springer, Cham, Switzerland.

[Cor08] Giuseppe Cornacchia. Su di un metodo per la risoluzione in numeri interi dell'equazione $\sum_{h=0}^{n} c_h x^{n-h} y^h = p$. *Giornale di Matematiche di Battaglini*, 46:33–90, 1908.

[CS18] Craig Costello and Benjamin Smith. Montgomery curves and their arithmetic - the case of large characteristic fields. *Journal of Cryptographic Engineering*, 8(3):227–240, September 2018.

[CV23] Wouter Castryck and Frederik Vercauteren. A polynomial time attack on instances of M-SIDH and FESTA. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 127–156, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

[DFMS22] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 677–706, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[DFP24] Luca De Feo, Tako Boris Fouotsa, and Lorenz Panny. Isogeny problems with level structure. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 181–204, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[DG16] Christina Delfs and Steven D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *DCC*, 78(2):425–440, 2016.

[DMPR24] Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert. An algorithmic approach to (2, 2)-isogenies in the theta model and applications to isogeny-based cryptography. In *ASIACRYPT 2024, Part III*, LNCS, pages 304–338. Springer, Singapore, Singapore, December 7–11, 2024.

[FMP23] Tako Boris Fouotsa, Tomoki Moriya, and Christophe Petit. M-SIDH and MD-SIDH: Countering SIDH attacks by masking information. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 282–309, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, page 537–554, Berlin, Heidelberg, 1999. Springer-Verlag.

[FR94] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of computation*, 62(206):865–874, 1994.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

[HLMW26] Arthur Herlédan Le Merdy and Benjamin Wesolowski. Unconditional foundations forăsupersingular isogeny-based cryptography. In Benny Applebaum and Huijia (Rachel) Lin, editors, *Theory of Cryptography*, pages 266–297, Cham, 2026. Springer Nature Switzerland.

[JAC+22] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[JD11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum*

Cryptography - 4th International Workshop, PQCrypto 2011, pages 19–34, Tapei, Taiwan, November 29 – December 2 2011. Springer, Berlin, Heidelberg, Germany.

[JZM19] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Tighter security proofs for generic key encapsulation mechanism in the quantum random oracle model. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 227–248, Chongqing, China, May 8–10, 2019. Springer, Cham, Switzerland.

[KLPT14] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion-isogeny path problem. *LMS Journal of Computation and Mathematics*, 17(A):418–432, 2014.

[Lic69] Stephen Lichtenbaum. Duality theorems for curves over p-adic fields. *Inventiones mathematicae*, 7(2):120–136, 1969.

[LLC+24] Kaizhan Lin, Jianming Lin, Shiping Cai, Weize Wang, and Chang-An Zhao. Compressed M-SIDH: an instance of compressed SIDH-like schemes with isogenies of highly composite degrees. *DCC*, 92(6):1823–1843, 2024.

[LM26] Yi-Fu Lai and Luciano Maino. Toward zkSNARK-assisted isogeny-based cryptography. Cryptology ePrint Archive, Paper 2026/1096, 2026.

[LR23] David Lubicz and Damien Robert. Fast change of level and applications to isogenies. *Research in Number Theory*, 9(1):7, 2023.

[Mil86] James S Milne. Jacobian varieties. In *Arithmetic geometry*, pages 167–212. Springer, 1986.

[MMP+23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 448–471, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[MN90] François Morain and Jean-Louis Nicolas. On Cornacchia's algorithm for solving the diophantine equation $u^2 + dv^2 = m$, 1990.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.

[Mon87] Peter L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987.

[MS25] Tomoki Moriya and Miha Stopar. LIT-SiGamal: An efficient isogeny-based PKE based on a LIT diagram. In *ASIACRYPT 2025, Part IV*, LNCS, pages 275–306. Springer, Singapore, Singapore, December 7–11, 2025.

[NO24] Kohei Nakagawa and Hiroshi Onuki. QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras. LNCS, pages 75–106, Santa Barbara, CA, USA, August 2024. Springer, Cham, Switzerland.

[Per12] Edoardo Persichetti. *Improving the Efficiency of Code-Based Cryptography*. PhD thesis, University of Auckland, November 2012.

[PH78] Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1):106–110, January 1978.

[PRR+25] Giacomo Pope, Krijn Reijnders, Damien Robert, Alessandro Sferlazza, and Benjamin Smith. Simpler and faster pairings from the montgomery ladder. *CiC*, 2(2):29, 2025.

[PW24] Aurel Page and Benjamin Wesolowski. The supersingular endomorphism ring and one endomorphism problems are equivalent. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 388–417, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[Rob23] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 472–503, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[RS17] Joost Renes and Benjamin Smith. qDSA: Small and secure digital signatures with curve-based Diffie-Hellman key pairs. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 273–302, Hong Kong, China, December 3–7, 2017. Springer, Cham, Switzerland.

[SAB+20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[Sco24] Michael Scott. Modular arithmetic for cryptographers. Technology Innovation Institute, White Paper, 2024.

[Tan09] Seiichiro Tani. Claw finding algorithms using quantum walk. *Theoretical Computer Science*, 410(50):5285–5297, 2009.

[Tat62] John Tate. Duality theorems in galois cohomology over number fields. In *Proc. Internat. Congr. Mathematicians (Stockholm, 1962)*, pages 288–295, 1962.

[Vél71] Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences*, 273:238–241, 1971.

[Wei40] André Weil. Sur les fonctions algebriques á corps de constantes finis, volume i. *Oeuvres Scientifiques, Paris*, pages 257–259, 1940.

[Wei57] André Weil. *Zum Beweis des Torellischen Satzes: CL Siegel zum 60. Geburtstage*. Vandenhoeck & Ruprecht, 1957.

[Wes22] Benjamin Wesolowski. The supersingular isogeny path and endomorphism ring problems are equivalent. In *62nd FOCS*, pages 1100–1111, Denver, CO, USA, February 7–10, 2022. IEEE Computer Society Press.

CHAPTER A

实现概览

本章简要介绍 QIMEN-PIKE 实现的各个组成模块，并说明模块间的交互关系。附录的其他章节详述这些模块内部的算法。

intbig: 为四元数算术（quat）提供多精度算术支持。

gf: 为椭圆曲线算术（ec）提供有限域运算。本模块的完整描述见Chapter B。

ec: 实现椭圆曲线算术的核心算法，尤其是高维同源（hd）和理想翻译（qlapoti）所需的关键部分。本模块的完整描述见Chapter C和Chapter D。

hd: 实现 $(2,2)$-同源链的计算算法，供理想翻译（qlapoti）和核心方案功能（QIMEN-PIKE.KeyGen, QIMEN-PIKE.Encaps, QIMEN-PIKE.Decaps ）调用。本模块的完整描述见Chapter E。

biext: 为 Weil 和 Tate 配对计算提供立方算术。在实现中，它是 ec 的一个子模块。但由于该子模块较为复杂，Chapter F专门描述其实现细节。

quat: 提供核心四元数算术，特别是理想翻译（qlapoti）的核心算法。QIMEN-PIKE 使用的四元数描述见Sections 2.6.2 and 2.6.3。

id2iso: 提供将四元数翻译为同源核的算法，主要用于QIMEN-PIKE.KeyGen。

此外还有两个辅助模块：precomp 负责预计算供多个模块使用的常量和方案参数，而 common 提供基础密码学功能，如哈希函数和伪随机数生成器。

Figure A.1展示了各模块之间的依赖关系以及关键协议算法 QIMEN-PIKE.KeyGen, QIMEN-PIKE.Encaps, QIMEN-PIKE.Decaps 。

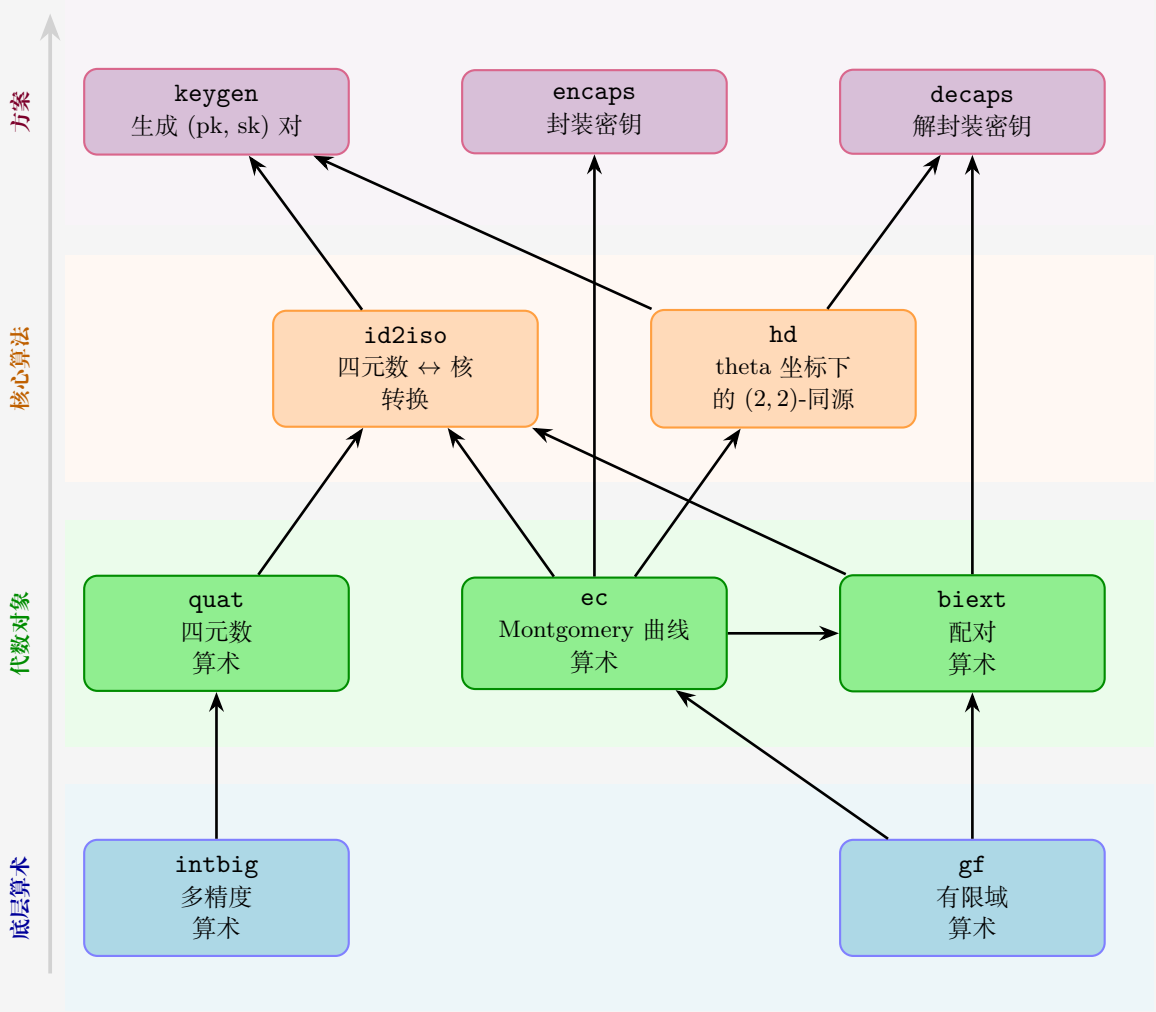

Figure A.1: QIMEN-PIKE 参考实现的模块结构，垂直方向表示抽象层级的递升。

CHAPTER B

有限域算术（实现细节）*

QIMEN-PIKE 中的所有高层运算最终都归结为 $\mathbb{F}_p$ 及其二次扩域 $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$ 上的算术（Section 2.1）。实现分为三层：

1. $\mathbb{F}_p$ Montgomery 算术内核，由 Scott 的代码生成器 [Sco24] 按非饱和表示自动生成；
2. 编码/解码包装层，提供常数时间工具函数；
3. 基于 $\mathbb{F}_p$ 构建的 $\mathbb{F}_{p^2}$ 算术。

所有代码均为可移植 C99，使用 `__uint128_t` 处理双宽度乘积。代码面向 64 位平台，对所有依赖秘密的输入均以常数时间运行。

B.1 元素表示

$\mathbb{F}_p$ 的每个元素以 n 个无符号 64 位 limb 构成的数组存储，采用非饱和表示：每个 limb 最多携带 $r < 64$ 个有效比特。元素 $a \in \mathbb{F}_p$ 被编码为 $(a_0, a_1, \ldots, a_{n-1})$，满足

$$a \equiv \sum_{j=0}^{n-1} a_j \cdot 2^{r\,j} \pmod{p}, \qquad 0 \le a_j < 2^r.$$

每个 limb 的 $64 - r$ 个比特为中间结果预留进/借位空间，从而加法和减法无需立即进位和借位。

参数 n 和 r 满足 $n \cdot r \ge \lceil \log_2 p \rceil$；Table B.1列出了Section 5.1中每个参数的具体值。

Table B.1: 参考实现的域元素表示参数。其中 n 为 64 位 limb 的个数，r 为基（每个 limb 的有效比特数）。

参数组	$\lceil \log_2 p \rceil$	Limb 数 n	基 r	有效比特 $n{\cdot}r$
NGCC-1	479	8	61	488
NGCC-2	765	13	59	767
NGCC-3	1534	26	60	1560

在内部，所有 $\mathbb{F}_p$ 元素均以 Montgomery 形式 [Mon85] 表示：a 的存储值为 $\tilde{a} = a \cdot R \bmod p$，其中 $R = 2^{n \cdot r}$。转换（`nres`/`redc`）仅在导入/导出边界发生。$\mathbb{F}_{p^2} = \mathbb{F}_p(i)$ 的一个元素是一对 $\mathbb{F}_p$ 元素 $(\mathsf{re}, \mathsf{im})$，表示 $\mathsf{re} + \mathsf{im} \cdot i$。

对于序列化，$\mathbb{F}_p$ 元素经 `redc` 约化后以小端序写入 $\ell_p = \lceil \log_2 p / 8 \rceil$ 字节。一个 $\mathbb{F}_{p^2}$ 元素是其实部和虚部的连接，占用 $2\ell_p$ 字节。

B.2 $\mathbb{F}_p$ 上的算术

$\mathbb{F}_p$ 算术内核由 Scott 的 `monty.py` 工具 [Sco24] 生成。该工具输出可移植 C 代码，采用Section B.1所述非饱和基 2^r 下的 Montgomery 表示。在此设定下，非饱和布局简化了高层实现中的进位处理：进位以移位

和掩码方式传播，约化逻辑直接表达在 limb 层面。生成的代码还经过脚本对随机测试输入进行验证，并附带额外的溢出检查。该内核还提供基于按位掩码的常数时间条件选择和交换，其分支和内存访问模式不依赖秘密值。

- **模加法和模减法**在 limb 层面实现。加法先逐 limb 求和，再经一次进位传播遍历（`prop`）：在第 (i) 个 limb 上，提取超过该 limb 允许位宽的高位作为进位，保留低位，并将该进位加入更高一位 limb，总开销为 $O(n)$ 字操作。当中间结果为负时，由无分支修正步骤（`flatten`）加回 p。减法类似处理。在整个算术过程中，中间值通常维持在 $2p$ 以下，而非每次运算后都约化到规范区间。
- **乘法**将 Schoolbook 乘积与 Montgomery 约化合并为单次遍历。对每一列，算法累加相关部分和、确定对应的约化数字，并立即将该数字的贡献并入当前状态，从而省去完整的双宽度中间量。每个 limb 乘积由 `__uint128_t` 内建类型处理。
- **平方**利用对称性 $a_j a_k = a_k a_j$ 减少非对角线乘积的数量。
- **求逆、平方根**和 **Legendre 符号**共用同一条加法链求幂，该链在 [Sco24] 中称为 progenitor，计算 $\pi = a^{(p-3)/4}$。由于 $p \equiv 3 \pmod 4$，平方根恢复为 $\sqrt{a} = \pi \cdot a = a^{(p+1)/4}$；参见Equation (2.1)。Legendre 符号由 $\pi^2 \cdot a = a^{(p-1)/2}$ 得到，逆元由 $\pi^4 \cdot a = a^{p-2}$ 得到。

B.3 $\mathbb{F}_{p^2}$ 上的算术

$\mathbb{F}_{p^2}$ 中的大多数算术归约为 $\mathbb{F}_p$ 算术。

- $\mathbb{F}_{p^2}$ 中的加法、减法、取反和折半按分量逐项进行，每个开销为两次 $\mathbb{F}_p$ 运算，
- $\mathbb{F}_{p^2}$ 中的乘法使用Algorithm 23中的类 Karatsuba 公式，将开销从四次 $\mathbb{F}_p$ 乘法（Section 2.1.2）降至三次，代价为增加两次额外加法。
- $\mathbb{F}_{p^2}$ 中的平方使用Algorithm 24，利用恒等式 $(a_0 + a_1 i)^2 = (a_0 + a_1)(a_0 - a_1) + 2a_0 a_1\, i$，因此仅需两次 $\mathbb{F}_p$ 乘法，
- 求逆使用Section 2.1.2的基于范数的方法：计算 $N = a_0^2 + a_1^2 \in \mathbb{F}_p$，求 N 的逆，再将共轭 $(a_0, -a_1)$ 乘以 N^{-1}，
- k 个元素的批量求逆使用 Montgomery 的批量求逆技巧 [Mon87]，将 k 次求逆降至一次求逆外加 $3(k-1)$ 次 $\mathbb{F}_{p^2}$ 乘法：前缀积向前累积，对总和求逆，各个逆元沿反向恢复。
- $a \in \mathbb{F}_{p^2}$ 的二次互反性检查归约为：利用Section 2.1.2检查 $a_0^2 + a_1^2$ 是否为 $\mathbb{F}_p$ 中的平方元，
- 平方根利用Eq. (2.2)与 $\mathbb{F}_p$ progenitor 计算；两种情形（$\chi = 1$ vs. $\chi = -1$）通过常数时间条件选择处理。

我们在Table B.2中总结了这些开销，供后续各节使用。

Table B.2: 实现章节中使用的开销记法。

符号	运算	$\mathbb{F}_p$ 开销
M	一次 $\mathbb{F}_{p^2}$ 乘法	3 次 $\mathbb{F}_p$ 乘法和 6 次 $\mathbb{F}_p$ 加法
S	一次 $\mathbb{F}_{p^2}$ 平方	2 次 $\mathbb{F}_p$ 乘法和 3 次 $\mathbb{F}_p$ 加法
a	一次 $\mathbb{F}_{p^2}$ 加法或减法	2 次 $\mathbb{F}_p$ 加/减法

Algorithm 23 Fp2Mul(a, b)

Input: $a = a_0 + a_1 i,\ b = b_0 + b_1 i \in \mathbb{F}_{p^2}$
Output: $c = a \cdot b \in \mathbb{F}_{p^2}$

$t_0 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1)$
$t_1 \leftarrow a_1 \cdot b_1$
$c_0 \leftarrow a_0 \cdot b_0$
$c_1 \leftarrow t_0 - t_1 - c_0$ ▷ $= a_0 b_1 + a_1 b_0$
$c_0 \leftarrow c_0 - t_1$ ▷ $= a_0 b_0 - a_1 b_1$
return $c = c_0 + c_1\, i$

Algorithm 24 Fp2Sqr(a)

Input: $a = a_0 + a_1 i \in \mathbb{F}_{p^2}$
Output: $c = a^2 \in \mathbb{F}_{p^2}$

$c_1 \leftarrow 2\, a_0 \cdot a_1$
$c_0 \leftarrow (a_0 + a_1) \cdot (a_0 - a_1)$ ▷ $= a_0^2 - a_1^2$
return $c = c_0 + c_1\, i$

B.4 离散对数

本节描述有限域上的离散对数算法。这些计算涉及 $\mathbb{F}_{p^2}^{\times}$ 的乘法子群。我们区分 2-adic 情形与奇数阶情形。

B.4.0.1 2 幂阶子群

首先考虑阶为 2^e 的子群。令 $\zeta \in \mu_{2^e} \subset \mathbb{F}_{p^2}^{\times}$ 为阶 2^e 的元素，且 $\eta = \zeta^k$。标准递归方法如下：首先由 $\eta^{2^{e-1}} \in \{\pm 1\}$ 确定 k 的最低有效比特，去除最低有效位对应的因子，将问题从阶 2^e 归约到阶 2^{e-1}。重复此过程即可恢复 k 的完整二进制展开。

Algorithm 25 DLogPowerOfTwo(ζ, η, e)

Input: $\zeta \in \mathbb{F}_{p^2}^{\times}$ 的阶为 2^e，$\eta = \zeta^k \in \langle \zeta \rangle$
Output: $k \in \mathbb{Z}/2^e\mathbb{Z}$ 满足 $\eta = \zeta^k$

if $e = 1$ **then**
    if $\eta = 1$ **then**
        return 0
    else
        return 1
if $\eta^{2^{e-1}} = 1$ **then**
    $b \leftarrow 0$
else
    $b \leftarrow 1$
$\eta' \leftarrow \eta \cdot \zeta^{-b}$
$k' \leftarrow$ DLogPowerOfTwo$(\zeta^2, \eta', e-1)$
return $b + 2k' \bmod 2^e$

B.4.0.2 奇数阶子群

现在考虑奇数阶情形。设 ℓ 为奇素数，$e \geq 1$，$\zeta \in \mu_{\ell^e} \subset \mathbb{F}_{p^2}^{\times}$ 为阶 ℓ^e 的元素，$\eta = \zeta^k$。目标是从 $\eta = \zeta^k$ 恢复 $k \bmod \ell^e$。在此情形中，我们仅需标准的通用方法。

首先考虑素数阶情形 $e = 1$。此时需根据 $\eta = \zeta^k$，从阶为 ℓ 的循环群 $\langle\zeta\rangle$ 中恢复 $k \in \mathbb{Z}/\ell\mathbb{Z}$。由于此处涉及的群规模较小，通用算法已足够。我们使用大步小步法（baby-step giant-step）：预计算小步 ζ^j $(0 \le j < m$，其中 $m = \lceil\sqrt{\ell}\rceil)$，然后搜索形如 $\eta\zeta^{-im}$ 的大步，检查是否与已有值匹配。一旦找到匹配，即得 $k \equiv im + j \pmod{\ell}$。

Algorithm 26 DLogPrimeOrder(ζ, η, ℓ)

Input: $\zeta \in \mathbb{F}_{p^2}^{\times}$ 的阶为 ℓ，$\eta = \zeta^k \in \langle\zeta\rangle$
Output: $k \in \mathbb{Z}/\ell\mathbb{Z}$ 满足 $\eta = \zeta^k$

$m \leftarrow \lceil\sqrt{\ell}\rceil$
$T \leftarrow \emptyset$
$u \leftarrow 1$
for $j = 0$ to $m - 1$ **do**
    在 T 中存储 (u, j)
    $u \leftarrow u \cdot \zeta$
$c \leftarrow \zeta^{-m}$
$v \leftarrow \eta$
for $i = 0$ to $m - 1$ **do**
    if v 出现在 T 中，对应项为 (v, j) **then**
        return $im + j \bmod \ell$
    $v \leftarrow v \cdot c$
return failure

对于一般素数幂情形，记

$$k = k_0 + k_1\ell + \cdots + k_{e-1}\ell^{e-1}, \qquad 0 \le k_j < \ell.$$

然后使用标准的 Pohlig–Hellman 逐位提升法。在第 j 阶段，首先消除已恢复各数字的贡献。然后将修正后的目标升至 ℓ^{e-1-j} 次幂，使其归入由 $\zeta^{\ell^{e-1}}$ 生成的素数阶子群。由此，每个基 ℓ 数字 k_j 通过调用一次Algorithm 26即可恢复。

Algorithm 27 DLogPrimePower(ζ, η, ℓ, e)

Input: $\zeta \in \mathbb{F}_{p^2}^{\times}$ 的阶为 ℓ^e，$\eta = \zeta^k \in \langle\zeta\rangle$
Output: $k \in \mathbb{Z}/\ell^e\mathbb{Z}$ 满足 $\eta = \zeta^k$

$g \leftarrow \zeta^{\ell^{e-1}}$
$k \leftarrow 0$
for $j = 0$ to $e - 1$ **do**
    $u \leftarrow \eta \cdot \zeta^{-k}$
    $v \leftarrow u^{\ell^{e-1-j}}$
    $k_j \leftarrow$ DLogPrimeOrder(g, v, ℓ)
    $k \leftarrow k + k_j\ell^j$
return $k \bmod \ell^e$

最后，对于一般光滑奇数阶

$$n = \prod_{i=1}^{r} \ell_i^{e_i},$$

对每个素数幂分量分别应用Algorithm 27，然后通过中国剩余定理组合模 $\ell_i^{e_i}$ 下的余数。

CHAPTER C

椭圆曲线算术（实现细节）*

如Section 2.2.1所述，我们始终使用 $\mathbb{F}_{p^2}$ 上的 Montgomery 曲线。曲线系数以射影形式 $(A : C)$ 存储，其中 A/C 为 Montgomery 系数。同时预计算量 $(A_{24} : C_{24}) = (A + 2C : 4C)$ 以加速倍点运算。

C.1 x-only 算术

点以 x-only 坐标 $(X : Z)$ 表示，其中 $x = X/Z$。y 坐标仅在显式需要时才计算，因为同源过程只使用 x 坐标数据。在此表示下，点只确定到符号层面，这对下文所述过程已经足够。两个基本算法是xDBL和xADD。这些算法为协议中的标量乘法提供了常数时间原语。

- xDBL（Algorithm 28）使用射影曲线常数 $(A_{24} : C_{24})$ 计算点的倍点，开销为 $2\mathrm{S} + 4\mathrm{M} + 4\mathrm{a}$，
- xADD（Algorithm 29）从 P、Q 及已知差 $P - Q$ 计算 $P + Q$，开销为 $2\mathrm{S} + 4\mathrm{M} + 6\mathrm{a}$，
- xDBLADD（Algorithm 30）在单个子程序中组合xDBL和xADD，并复用中间值，总开销为 $4\mathrm{S} + 8\mathrm{M} + 8\mathrm{a}$，
- Ladder给定 m 和 P，计算标量倍 $[m]P$。当 m 为 k 比特标量时，每个比特调用一次xDBLADD，总计 $4k\mathrm{S} + 8k\mathrm{M} + 8k\mathrm{a}$。
- Ladder3pt给定 m、P、Q 及其差，计算 $P + [m]Q$，开销与Ladder相似。
- LadderBiscalar给定 m、n、P、Q 及其差，计算 $[m]P + [n]Q$。当 m 和 n 均为 k 比特时，算法先调用一次xADD，之后每次循环迭代执行一次xDBL和两次xADD，总开销为 $(6k + 2)\mathrm{S} + (12k + 4)\mathrm{M} + (16k + 6)\mathrm{a}$。

Algorithm 28 xDBL$(P, (A_{24} : C_{24}))$

Require: $P = (X_P : Z_P)$，曲线 E 的 Montgomery 常数 $(A_{24} : C_{24})$
Ensure: $[2]P = (X_{2P} : Z_{2P})$

1. $t_0 \leftarrow X_P + Z_P$
2. $t_1 \leftarrow X_P - Z_P$
3. $t_0 \leftarrow t_0^2$
4. $t_1 \leftarrow t_1^2$
5. $t_2 \leftarrow t_0 - t_1$
6. $Z_{2P} \leftarrow t_1 \cdot C_{24}$
7. $X_{2P} \leftarrow t_0 \cdot Z_{2P}$
8. $t_0 \leftarrow t_2 \cdot A_{24}$
9. $t_0 \leftarrow t_0 + Z_{2P}$
10. $Z_{2P} \leftarrow t_0 \cdot t_2$
11. **return** $(X_{2P} : Z_{2P})$

. ▷ 开销: $2\mathrm{S} + 4\mathrm{M} + 4\mathrm{a}$

Algorithm 29 xADD$(P, Q, P - Q)$

Require: $P = (X_P : Z_P)$，$Q = (X_Q : Z_Q)$，$P - Q = (X_{P-Q} : Z_{P-Q})$
Ensure: $P + Q = (X_{P+Q} : Z_{P+Q})$

1. $t_0 \leftarrow X_P + Z_P$
2. $t_1 \leftarrow X_P - Z_P$
3. $t_2 \leftarrow X_Q + Z_Q$
4. $t_3 \leftarrow X_Q - Z_Q$
5. $t_0 \leftarrow t_0 \cdot t_3$
6. $t_1 \leftarrow t_1 \cdot t_2$
7. $t_2 \leftarrow t_0 + t_1$
8. $t_3 \leftarrow t_0 - t_1$
9. $t_2 \leftarrow t_2^2$
10. $t_3 \leftarrow t_3^2$
11. $X_{P+Q} \leftarrow Z_{P-Q} \cdot t_2$
12. $Z_{P+Q} \leftarrow X_{P-Q} \cdot t_3$
13. **return** $(X_{P+Q} : Z_{P+Q})$

▷ 开销: $2\mathrm{S} + 4\mathrm{M} + 6\mathrm{a}$

Algorithm 30 xDBLADD$(P, Q, P-Q, (A_{24} : C_{24}))$

Require: $P = (X_P : Z_P)$, $Q = (X_Q : Z_Q)$, $P - Q = (X_{P-Q} : Z_{P-Q})$, E 的 Montgomery 常数 $(A_{24} : C_{24})$
Ensure: $[2]P = (X_{2P} : Z_{2P})$ 和 $P + Q = (X_{P+Q} : Z_{P+Q})$

1. $t_0 \leftarrow X_P + Z_P$
2. $t_1 \leftarrow X_P - Z_P$
3. $X_{2P} \leftarrow t_0^2$
4. $t_2 \leftarrow X_Q - Z_Q$
5. $X_{P+Q} \leftarrow X_Q + Z_Q$
6. $t_0 \leftarrow t_0 \cdot t_2$
7. $Z_{2P} \leftarrow t_1^2$
8. $t_1 \leftarrow t_1 \cdot X_{P+Q}$
9. $t_2 \leftarrow X_{2P} - Z_{2P}$
10. $Z_{2P} \leftarrow Z_{2P} \cdot C_{24}$
11. $X_{2P} \leftarrow X_{2P} \cdot Z_{2P}$
12. $X_{P+Q} \leftarrow A_{24} \cdot t_2$
13. $Z_{P+Q} \leftarrow t_0 - t_1$
14. $Z_{2P} \leftarrow Z_{2P} + X_{P+Q}$
15. $X_{P+Q} \leftarrow t_0 + t_1$
16. $Z_{2P} \leftarrow Z_{2P} \cdot t_2$
17. $Z_{P+Q} \leftarrow Z_{P+Q}^2$
18. $X_{P+Q} \leftarrow X_{P+Q}^2$
19. $Z_{P+Q} \leftarrow Z_{P+Q} \cdot X_{P-Q}$
20. $X_{P+Q} \leftarrow X_{P+Q} \cdot Z_{P-Q}$
21. **return** $\big((X_{2P} : Z_{2P}), (X_{P+Q} : Z_{P+Q})\big)$ ▷ 开销: $4\mathrm{S} + 8\mathrm{M} + 8\mathrm{a}$

Algorithm 31 Ladder(P, E, m)

Input: 射影点 $P = (X_P : Z_P)$, 曲线 E 的 Montgomery 常数 $(A_{24} : C_{24})$, 正整数标量 $m = (m_{k-1}, \ldots, m_0)_2$
Output: 射影点 $[m]P = (X_{[m]P} : Z_{[m]P})$

$((X_0 : Z_0), (X_1 : Z_1)) \leftarrow ((1 : 0), (X_P : Z_P))$
for $i \leftarrow k-1$ **down to** 0 **do**
  if $m_i = 1$ **then**
    $((X_1 : Z_1), (X_0 : Z_0)) \leftarrow$ xDBLADD$((X_1 : Z_1), (X_0 : Z_0), (X_P : Z_P), (A_{24} : C_{24}))$
  else
    $((X_0 : Z_0), (X_1 : Z_1)) \leftarrow$ xDBLADD$((X_0 : Z_0), (X_1 : Z_1), (X_P : Z_P), (A_{24} : C_{24}))$
$(X_{[m]P} : Z_{[m]P}) \leftarrow (X_0 : Z_0)$
return $[m]P = (X_{[m]P} : Z_{[m]P})$ ▷ 开销: $4k\mathrm{S} + 8k\mathrm{M} + 8k\mathrm{a}$

Algorithm 32 Ladder3pt$(P, Q, P-Q, (A_{24} : C_{24}), m)$

Input: 射影点 $P = (X_P : Z_P)$, $Q = (X_Q : Z_Q)$, $P - Q = (X_{P-Q} : Z_{P-Q})$, 曲线 E 的 Montgomery 常数 $(A_{24} : C_{24})$, 正整数标量 $m = (m_{k-1}, \ldots, m_0)_2$
Output: 射影点 $P + [m]Q = (X_{P+[m]Q} : Z_{P+[m]Q})$

$((X_0 : Z_0), (X_1 : Z_1), (X_2 : Z_2)) \leftarrow ((X_P : Z_P), (X_Q : Z_Q), (X_{P-Q} : Z_{P-Q}))$
for $i \leftarrow 0$ **to** $k-1$ **do**
  if $m_i = 1$ **then**
    $((X_0 : Z_0), (X_1 : Z_1)) \leftarrow$ xDBLADD$((X_0 : Z_0), (X_1 : Z_1), (X_2 : Z_2), (A_{24} : C_{24}))$
  else
    $((X_0 : Z_0), (X_2 : Z_2)) \leftarrow$ xDBLADD$((X_0 : Z_0), (X_2 : Z_2), (X_1 : Z_1), (A_{24} : C_{24}))$
$(X_{P+[m]Q} : Z_{P+[m]Q}) \leftarrow (X_1 : Z_1)$
return $P + [m]Q = (X_{P+[m]Q} : Z_{P+[m]Q})$ ▷ 开销: $4k\mathrm{S} + 8k\mathrm{M} + 8k\mathrm{a}$

Algorithm 33 LadderBiscalar$(P, Q, P-Q, (A_{24}:C_{24}), m, n)$

Input: $P=(X_P:Z_P)$, $Q=(X_Q:Z_Q)$, $P-Q=(X_{P-Q}:Z_{P-Q})$, 曲线 E 的 Montgomery 常数 $(A_{24}:C_{24})$, 正整数标量 $m=(m_{k-1}\cdots m_0)_2$, $n=(n_{k-1}\cdots n_0)_2$
Output: $[m]P+[n]Q=(X_{[m]P+[n]Q}:Z_{[m]P+[n]Q})$

若 m 为偶数且 n 为奇数，则 $(\sigma_0,\sigma_1)\leftarrow(1,0)$；否则 $(\sigma_0,\sigma_1)\leftarrow(0,1)$
$m'\leftarrow m$, $n'\leftarrow n$
if m 为偶数 **then**
    $m'\leftarrow m'-1$
if n 为偶数 **then**
    $n'\leftarrow n'-1$
$b_0\leftarrow(0m'_{k-1}\cdots m'_0)_2$, $b_1\leftarrow(0n'_{k-1}\cdots n'_0)_2$, $b\leftarrow(b_0,b_1)$
for $i\leftarrow 0$ **to** $k-1$ **do**
    $r_{2i}\leftarrow b_{\sigma_0,i}\oplus b_{\sigma_0,i+1}$, $r_{2i+1}\leftarrow b_{\sigma_1,i}\oplus b_{\sigma_1,i+1}$
    if $r_{2i+1}=1$ **then**
        $(\sigma_0,\sigma_1)\leftarrow(\sigma_1,\sigma_0)$
$R_0\leftarrow(1:0)$, $T=(T_0,T_1)\leftarrow(P,Q)$, $R_1\leftarrow T_{\sigma_0}$, $R_2\leftarrow T_{(\sigma_0+1) \bmod 2}$
$D_1\leftarrow R_1$, $D_2\leftarrow R_2$, $R_2\leftarrow$ xADD$(R_1,R_2,P-Q)$
$F_1\leftarrow R_2$, $F_2\leftarrow P-Q$
for $i\leftarrow k-1$ **down to** 0 **do**
    $h\leftarrow r_{2i}+r_{2i+1}$, $T_0\leftarrow R_{h \bmod 2}$, $T\leftarrow(T_0,R_2)$
    $T_0\leftarrow$ xDBL$\big(T_{\lfloor h/2\rfloor},(A_{24}:C_{24})\big)$, $T_1\leftarrow R_{r_{2i+1}}$, $T_2\leftarrow R_{r_{2i+1}+1}$
    if $r_{2i+1}=1$ **then**
        $(D_1,D_2)\leftarrow(D_2,D_1)$
    $T_1\leftarrow$ xADD(T_1,T_2,D_1), $T_2\leftarrow$ xADD(R_0,R_2,F_1)
    if $h \bmod 2=1$ **then**
        $(F_1,F_2)\leftarrow(F_2,F_1)$
    $(R_0,R_1,R_2)\leftarrow(T_0,T_1,T_2)$
$(X_{[m]P+[n]Q}:Z_{[m]P+[n]Q})\leftarrow R_{((m \bmod 2)\oplus 1)+((n \bmod 2)\oplus 1)}$
return $[m]P+[n]Q=(X_{[m]P+[n]Q}:Z_{[m]P+[n]Q})$ ▷ $(6k+2)$S + $(12k+4)$M + $(16k+6)$a

C.2 射影 x-only 子程序

我们使用两个 x-only 子程序。

- IsomorphismMontgomeryCurves将点通过 Montgomery 曲线之间的同构进行映射，见Section 2.2.1.1。
- ProjectiveDifference从 $x(P)$ 和 $x(Q)$ 计算 $x(P\pm Q)$ 的确定性选取，见Section 2.2.2.3。
- RecoverCodomain给定两个点 P、Q 及其差 $P-Q$ 的 x-only 射影形式，计算曲线系数 $(A:C)$。

Algorithm 34 IsomorphismMontgomeryCurves(E, P, Q, E')

Input: 曲线 E 和 E' 的 Montgomery 系数 $(A:C)$ 和 $(A':C')$，以及 E 上的点 $P=(X_P:Z_P)$, $Q=(X_Q:Z_Q)$
Output: P 和 Q 在 E 与 E' 之间同构下的像 $P'=(X_{P'}:Z_{P'})$, $Q'=(X_{Q'}:Z_{Q'})$

$\lambda_x\leftarrow(2A'^3-9A'C'^2)(3C^3-A^2C)$
$\lambda_z\leftarrow(2A^3-9AC^2)(3C'^3-A'^2C')$
if $\lambda_x=0$ 或 $\lambda_z=0$ **then**
    raise ("IsomorphismMontgomeryCurves: 输入曲线无效。")
$X_{P'}\leftarrow\lambda_x(3X_PCC'+AC'Z_P)-\lambda_zA'CZ_P$
$Z_{P'}\leftarrow 3\lambda_zCC'Z_P$
$X_{Q'}\leftarrow\lambda_x(3X_QCC'+AC'Z_Q)-\lambda_zA'CZ_Q$
$Z_{Q'}\leftarrow 3\lambda_zCC'Z_Q$
return $(X_{P'}:Z_{P'}),(X_{Q'}:Z_{Q'})$

Algorithm 35 ProjectiveDifference$(P, Q, (A : C))$

Input: 射影点 $P = (X_P : Z_P)$ 和 $Q = (X_Q : Z_Q)$，Montgomery 系数 $(A : C)$
Output: 确定性 x 坐标 x_{PQ}，为 x_{P-Q} 或 x_{P+Q}

$B_{XX} \leftarrow C \cdot (X_P X_Q - Z_P Z_Q)^2$
$B_{XZ} \leftarrow C \cdot (X_P X_Q + Z_P Z_Q)(X_P Z_Q + Z_P X_Q) + 2A X_P X_Q Z_P Z_Q$
$B_{ZZ} \leftarrow C \cdot (X_P Z_Q - Z_P X_Q)^2$
$\gamma \leftarrow C \cdot (C \cdot Z_P \cdot Z_Q)^2$
$B_{XX} \leftarrow \gamma \cdot B_{XX}$, $B_{XZ} \leftarrow \gamma \cdot B_{XZ}$, $B_{ZZ} \leftarrow \gamma \cdot B_{ZZ}$
$\delta \leftarrow \sqrt{B_{XZ}^2 - B_{XX} B_{ZZ}}$
$X_{PQ} \leftarrow \delta + B_{XZ}, \quad Z_{PQ} \leftarrow B_{ZZ}$
return $x_{PQ} = (X_{PQ}, Z_{PQ})$

Algorithm 36 RecoverCodomain$(P, Q, P - Q)$

Require: $P_1 = P = (X_1 : Z_1)$，$P_2 = Q = (X_2 : Z_2)$，$P_3 = P - Q = (X_3 : Z_3)$
Ensure: Montgomery 曲线系数 $(A : C)$

1. $x_{123} \leftarrow X_1 X_2 X_3$
2. $z_{123} \leftarrow Z_1 Z_2 Z_3$
3. $xz_1 \leftarrow X_1 Z_2 Z_3$
4. $xz_2 \leftarrow X_2 Z_1 Z_3$
5. $xz_3 \leftarrow X_3 Z_1 Z_2$
6. $t_1 \leftarrow xz_1 + xz_2 + xz_3$
7. $t_2 \leftarrow (X_1 - Z_1)(X_2 - Z_2)(X_3 - Z_3)$
8. $A \leftarrow t_2 - x_{123} - t_1 + 2z_{123}$
9. $x_{123} \leftarrow 4x_{123}$
10. $C \leftarrow x_{123} z_{123}$
11. $A \leftarrow A^2 - x_{123} t_1$
12. **return** $(A : C)$

▷ 开销: $1\mathrm{S} + 14\mathrm{M} + 12\mathrm{a}$

C.3 Jacobian 坐标

QIMEN-PIKE 协议中有若干步骤需要点的 y 坐标，而 x-only 算术忽略了该坐标。例如，完成 $(2,2)$ 同源链求值后，需要将得到的像点通过小阶子群上的离散对数表示为挠基的线性组合；而要确定每个点的正确符号，又必须知道 y 坐标。密钥生成过程中同样需要 y：自同态 θ 在非光滑挠点上求值，这需要执行对符号敏感的双标量乘法 $\theta(P) = [a]P + [b]\iota(P)$。解密过程中也需要 y：E_B 上的挠基必须提升到完整坐标，以正确定向 $(2,2)$ 同源的核。在所有这些情形中，我们切换到满足 $BY^2 = X^3 + AX^2Z^2 + XZ^4$ 的 Jacobian 坐标 $(X : Y : Z)$。Montgomery 形式与 Jacobian 形式之间的转换公式如下：

$$(X_M : Y_M : Z_M) = (X_J : Y_J/Z_J : Z_J^2),$$
$$(X_J : Y_J : Z_J) = (X_M - AZ_M^2/3,\ Y_M,\ Z_M).$$

下文给出 Jacobian 坐标下算术的伪代码。基本运算如下：

- DBL（Algorithm 38）以点 P 的 Jacobian 坐标和规范化 Montgomery 系数 $a = A/C$ 为输入，输出倍点 $[2]P$ 的 Jacobian 坐标。
- ADD（Algorithm 37）以两点 P 和 Q 的 Jacobian 坐标以及规范化 Montgomery 系数 $a = A/C$ 为输入，输出和 $P + Q$ 的 Jacobian 坐标。
- ADDComponents（Algorithm 39）以两个不同点 P, Q 的 Jacobian 坐标和规范化 Montgomery 系数 $a = A/C$ 为输入，输出 (u, v, w) 使得 $P + Q$ 和 $P - Q$ 的射影 x-only 坐标由下式给出：

$$x(P + Q) = (u - v : w), \qquad x(P - Q) = (u + v : w).$$

Algorithm 37 ADD$(P, Q, (A : C))$

Require: Montgomery 曲线 E 上的 Jacobian 点 $P = (X_P : Y_P : Z_P)$ 和 $Q = (X_Q : Y_Q : Z_Q)$，满足 $P \neq Q$，$Q \neq -P$，且 $P, Q \neq \mathcal{O}$
Ensure: Jacobian 点 $P + Q = (X_{P+Q} : Y_{P+Q} : Z_{P+Q})$

1. $t_0 \leftarrow Z_P^2$
2. $t_1 \leftarrow t_0 \cdot Z_P$
3. $t_2 \leftarrow Z_Q^2$
4. $t_3 \leftarrow t_2 \cdot Z_Q$
5. $t_1 \leftarrow t_1 \cdot Y_Q$
6. $t_3 \leftarrow t_3 \cdot Y_P$
7. $t_1 \leftarrow t_1 - t_3$
8. $t_0 \leftarrow t_0 \cdot X_Q$
9. $t_2 \leftarrow t_2 \cdot X_P$
10. $t_4 \leftarrow t_0 - t_2$
11. $t_0 \leftarrow t_0 + t_2$
12. $t_5 \leftarrow Z_P \cdot Z_Q$
13. $Z_{P+Q} \leftarrow t_4 \cdot t_5$
14. $t_5 \leftarrow t_5^2$
15. $t_5 \leftarrow t_5 \cdot A$
16. $t_0 \leftarrow t_0 + t_5$
17. $t_6 \leftarrow t_4^2$
18. $t_5 \leftarrow t_0 \cdot t_6$
19. $X_{P+Q} \leftarrow t_1^2$
20. $X_{P+Q} \leftarrow X_{P+Q} - t_5$
21. $t_3 \leftarrow t_3 \cdot t_4$
22. $t_3 \leftarrow t_3 \cdot t_6$
23. $t_2 \leftarrow t_2 \cdot t_6$
24. $Y_{P+Q} \leftarrow t_2 - X_{P+Q}$
25. $Y_{P+Q} \leftarrow Y_{P+Q} \cdot t_1$
26. $Y_{P+Q} \leftarrow Y_{P+Q} - t_3$
27. **return** $(X_{P+Q} : Y_{P+Q} : Z_{P+Q})$

▷ 开销: $5\mathrm{S} + 15\mathrm{M} + 7\mathrm{a}$

Algorithm 38 DBL$(P, (A : C))$

Require: $P = (X_P : Y_P : Z_P)$ 和曲线 E 的 Montgomery 系数 $(A : C)$
Ensure: $[2]P = (X_{[2]P} : Y_{[2]P} : Z_{[2]P})$

1. $t_0 \leftarrow X_P^2$
2. $t_1 \leftarrow t_0 + t_0$
3. $t_0 \leftarrow t_0 + t_1$
4. $t_1 \leftarrow Z_P^2$
5. $t_2 \leftarrow X_P \cdot A$
6. $t_2 \leftarrow t_2 + t_2$
7. $t_2 \leftarrow t_1 + t_2$
8. $t_2 \leftarrow t_1 \cdot t_2$
9. $t_2 \leftarrow t_0 + t_2$
10. $Z_{[2]P} \leftarrow Y_P \cdot Z_P$
11. $Z_{[2]P} \leftarrow Z_{[2]P} + Z_{[2]P}$
12. $t_0 \leftarrow Z_{[2]P}^2$
13. $t_0 \leftarrow t_0 \cdot A$
14. $t_1 \leftarrow Y_P^2$
15. $t_1 \leftarrow t_1 + t_1$
16. $t_3 \leftarrow X_P + X_P$
17. $t_3 \leftarrow t_1 \cdot t_3$
18. $X_{[2]P} \leftarrow t_2^2$
19. $X_{[2]P} \leftarrow X_{[2]P} - t_0$
20. $X_{[2]P} \leftarrow X_{[2]P} - t_3$
21. $X_{[2]P} \leftarrow X_{[2]P} - t_3$
22. $Y_{[2]P} \leftarrow t_3 - X_{[2]P}$
23. $Y_{[2]P} \leftarrow Y_{[2]P} \cdot t_2$
24. $t_1 \leftarrow t_1^2$
25. $Y_{[2]P} \leftarrow Y_{[2]P} - t_1$
26. $Y_{[2]P} \leftarrow Y_{[2]P} - t_1$
27. **return** $(X_{[2]P} : Y_{[2]P} : Z_{[2]P})$

▷ 开销: $6\mathrm{S} + 6\mathrm{M} + 14\mathrm{a}$

Algorithm 39 ADDComponents$(P, Q, (A : C))$

Require: $P = (X_P : Y_P : Z_P)$，$Q = (X_Q : Y_Q : Z_Q)$，满足 $P \neq Q$，和曲线 E 的 Montgomery 系数 $(A : C)$
Ensure: (u, v, w) 使得 $P + Q$ 和 $P - Q$ 的 x-only 坐标为 $P + Q = (u - v : w)$ 和 $P - Q = (u + v : w)$

1. $t_0 \leftarrow Z_P^2$
2. $t_1 \leftarrow Z_Q^2$
3. $t_2 \leftarrow X_P \cdot t_1$
4. $t_3 \leftarrow t_0 \cdot X_Q$
5. $t_4 \leftarrow Y_P \cdot Z_Q$
6. $t_4 \leftarrow t_4 \cdot t_1$
7. $t_5 \leftarrow Z_P \cdot Y_Q$
8. $t_5 \leftarrow t_5 \cdot t_0$
9. $t_0 \leftarrow t_0 \cdot t_1$
10. $t_6 \leftarrow t_4 \cdot t_5$
11. $t_4 \leftarrow t_4^2$
12. $t_5 \leftarrow t_5^2$
13. $t_4 \leftarrow t_4 + t_5$
14. $t_5 \leftarrow t_2 + t_3$
15. $t_7 \leftarrow t_3 + t_3$
16. $t_7 \leftarrow t_5 - t_7$
17. $t_7 \leftarrow t_7^2$
18. $t_1 \leftarrow A \cdot t_0$
19. $t_1 \leftarrow t_5 + t_1$
20. $t_1 \leftarrow t_1 \cdot t_7$
21. $u \leftarrow t_4 - t_1$
22. $v \leftarrow t_6 + t_6$
23. $w \leftarrow t_6 \cdot t_0$
24. **return** (u, v, w)

▷ 开销: $5\mathrm{S} + 11\mathrm{M} + 7\mathrm{a}$

C.4 挠基

对于固定的起始曲线 $E_0 : y^2 = x^3 + x$，挠子群 $E_0[2^a]$、$E_0[C]$ 和 $E_0[D]$ 的基已预计算。对于动态生成的曲线 E_A、E_B 和 E_{AB}，挠基通常由 E_0 的预计算基经相应同源链求值得到，而非从零计算。

对于 2 幂挠，可以完全保持在 x-only 坐标中。从仿射 Montgomery 系数 A 出发，搜索一个形状合适的点，通过关系 $x_{R+S} = -x_R - A$ 构造第二个点。然后将两个候选点乘以余因子 $(p+1)/2^e$ 投影到 $E[2^e]$ 上，再通过ProjectiveDifference恢复剩余的 x 坐标。由此得到 (x_R, x_S, x_{R+S}) 形式的确定性基，该形式非常适合整个实现中使用的基于阶梯的算术。

对于奇素数幂挠 $E[\ell^e]$，通常没有类似的 x-only 捷径，因此标准方法使用完整坐标。具体做法是：采

样随机点，用余因子 $(p+1)/\ell^e$ 将其映射到 ℓ^e 挠上 [1]，并拒绝任何阶严格小于 ℓ^e 的点。第二个点仅在与第一个点线性无关时才予以接受。为处理一般挠群，需将问题分解为上述素数幂情形，对每种情形应用相似策略。具体而言，对于一般整数 $N = \prod_i \ell_i^{e_i}$，我们使用同一套子程序在 $E[N]$ 中采样点。点 $P \in E[N]$ 具有精确阶 N 当且仅当对每个素因子 $\ell_i \mid D$ 满足 $[N/\ell_i]P \neq 0_E$。因此，为获得 $E[N]$ 的基，采样两点 $P, Q \in E[N]$，对两者检查精确阶条件，最后验证线性无关性。

挠基提示(Hints)。 提示可以加速挠基生成。在 2 幂挠情形中，生成基需要知道曲线系数 A 的二次剩余性以及用于确定 x_P 的具体索引。协议将这些值作为提示提供。具体做法是：一方使用TorsionBasisToHint同时计算基及其关联提示，将提示直接纳入公钥或密文；另一方则执行TorsionBasisFromHint，利用这些提示高效恢复基 (x_R, x_S, x_{RS})。需注意，TorsionBasisFromHint返回的基中两个值 x_R 和 x_S 位于同一扭上，即使这些值并未显式验证为 E 上的点。只要后续使用时验证其阶，就不会造成任何问题——验证阶本身就蕴含了这些点在 E 上这一事实。参考实现按Chapters C and E所述完成此操作。其他情形中，TorsionBasisGivenOrderFromHint使用的提示避免了昂贵的平方根运算，且无需执行无关性测试。

Algorithm 40 TorsionBasisToHint(A, a)

Input: 非零仿射 Montgomery 系数 A 和整数 $a \leq e$
Output: $E[2^a]$ 的 x-only 基 (x_R, x_S, x_{RS}) 以及两个提示 h_A, h

1: **if** A 为平方元 **then**
2: $h_A \leftarrow 1$
3: **else**
4: $h_A \leftarrow 0$
5: $h \leftarrow 0$
6: **if** A 为平方元 **then**
7: **repeat**
8: $h \leftarrow h + 1$
9: $x_R \leftarrow -1/(1 + i \cdot h) \cdot A$
10: **until** $(1 + h^2)$ 非平方元且 $x_R \in E_A(\mathbb{F}_{p^2})$
11: **else**
12: **repeat**
13: $h \leftarrow h + 1$
14: $x_R \leftarrow h \cdot A$
15: **until** $x_R \in E_A(\mathbb{F}_{p^2})$
16: $x_{RS} \leftarrow -x_R - A$
17: $x_R \leftarrow \text{Ladder}((x_R : 1), (A + 2 : 4), \frac{p+1}{2^a})$
18: $x_{RS} \leftarrow \text{Ladder}((x_{RS} : 1), (A + 2 : 4), \frac{p+1}{2^a})$
19: $x_S \leftarrow \text{ProjectiveDifference}(x_R, x_{RS}, (A : 1))$
20: **if** $h \geq 128$ **then**
21: $h \leftarrow 0$
22: **return** (x_R, x_S, x_{RS}) 和 (h_A, h)

在 QIMEN-PIKE 的压缩实现QIMEN-PIKE.KeyGenCompressed中，我们首先恢复标量 u_{A,C_1} 和 v_{A,C_1}，满足 $[2^a]Q_A^+ = [u_{A,C_1}]U_{A,C_1} + [v_{A,C_1}]V_{A,C_1}$。随后需要压缩这些标量的规模，以保证公钥紧凑。当 C_1 为素数幂时做法很简单：对 u_{A,C_1} 或 v_{A,C_1} 求逆，再将另一标量乘以此逆元即可。但在我们的设定中，挠阶是复合的，形如 $C_1 = \prod_{i=1}^n \ell_i^{e_i}$ 且 $n > 1$。因此，u_{A,C_1} 和 v_{A,C_1} 可能都不可逆模 C_1。

解决方法是采用 [LLC+24, Section 4.3] 的方案。关键观察是，对每个素数幂因子 $\ell_i^{e_i}$ $(i = 1, 2, \ldots, n)$，u_{A,C_1} 或 v_{A,C_1} 中至少有一个可逆。为显式追踪每个因子处求逆了哪个标量，我们引入一个 n 比特标签 $\mathsf{label} = (l_n l_{n-1} \cdots l_1)_2$。如LabelComputation中详述，若 u_{A,C_1} 可逆模 $\ell_i^{e_i}$ 则置 $l_i = 0$，否则置 $l_i = 1$。

[1] 对于二次扭 E_t，将余因子替换为 $(p-1)/\ell^e$。

Algorithm 41 TorsionBasisFromHint(A, a, h_A, h)

Input: 非零仿射 Montgomery 系数 A，整数 $a < e$，以及两个提示 h_A, h
Output: $E[2^a]$ 的 x-only 基 (x_R, x_S, x_{RS})

if $h = 0$ **then**
  $(x_R, x_S, x_{RS}), (h_A, h) \leftarrow$ TorsionBasisToHint(A, a)
  return (x_R, x_S, x_{RS})
else
  if $h_A = 1$ **then**
    $x_R \leftarrow -1/(1 + i \cdot h) \cdot A$
  else
    $x_R \leftarrow h \cdot A$
  $x_{RS} \leftarrow -x_R - A$
  $x_R \leftarrow$ Ladder$((x_R : 1), (A + 2 : 4), \frac{p+1}{2^a})$
  $x_{RS} \leftarrow$ Ladder$((x_{RS} : 1), (A + 2 : 4), \frac{p+1}{2^a})$
  $x_S \leftarrow$ ProjectiveDifference$(x_R, x_{RS}, (A : 1))$
  return (x_R, x_S, x_{RS})

该算法通过中国剩余定理将每个局部的求逆结果聚合为标签 label 以及全局标量 $m, s \in \mathbb{Z}/C_1\mathbb{Z}$，满足

$$m = \begin{cases} u_{A,C_1}^{-1} \bmod \ell_i^{e_i}, \text{若 } l_i = 0, \\ v_{A,C_1}^{-1} \bmod \ell_i^{e_i}, \text{否则}, \end{cases}, s = \begin{cases} mv_{A,C_1} \bmod \ell_i^{e_i}, \text{若 } l_i = 0, \\ mu_{A,C_1} \bmod \ell_i^{e_i}, \text{否则}, \end{cases}, \tag{C.1}$$

以及对应标签 label。

仅使用压缩标量 s 和标签 label，ScalarRecoveryFromLabel重构两个完整标量 $s_1, s_2 \in \mathbb{Z}/C_1\mathbb{Z}$ 使得

$$s_1 = \begin{cases} mv_{A,C_1} \bmod \ell_i^{e_i}, \text{若 } l_i = 0, \\ 1 \bmod \ell_i^{e_i}, \text{否则}, \end{cases}, s_2 = \begin{cases} 1 \bmod \ell_i^{e_i}, \text{若 } l_i = 0, \\ mu_{A,C_1} \bmod \ell_i^{e_i}, \text{否则}. \end{cases} \tag{C.2}$$

依照 [LLC+24, Proposition 4] 的证明，容易验证：对每个 $i = 1, 2, \cdots, n$，点 $[C_1/\ell_i^{e_i}]([s_1]U_{A,C_1} + [s_2]V_{A,C_1})$ 生成子群 $\langle [C_1/\ell_i^{e_i}]([2^a]Q_A^+) \rangle$。此性质在无需知道 Q_A^+ 的显式坐标的情况下，保证了压缩加密的正确性。

Algorithm 42 TorsionBasisGivenOrderToHint(E, N, flag)

Input: 超奇异 Montgomery 曲线 $E/\mathbb{F}_{p^2}$，整数 $N = \prod_{i=1}^{n} \ell_i^{e_i}$，以及指示基是定义在曲线 $E(\mathbb{F}_{p^2})$ 上还是其二次扭取线上的标志 flag

Output: $E[N]$ 的 x-only 基 $(P, Q, P-Q)$ 以及提示 $h = (h_0, h_1)$

$h_0 \leftarrow 0$, found $\leftarrow$ false
令余因子 c_N 为用于将点投影到 N 挠上的整数
while found $=$ false **do**
    $h_0 \leftarrow h_0 + 1$
    $P \leftarrow (1 + h_0 i : 1)$
    $F \leftarrow X(P)(C^2 X(P)^2 + ACX(P) + C^2)$
    if lssquare$(F) \neq$ flag **then**
        continue
    $P \leftarrow [c_N]P$
    $P' \leftarrow [\prod_{i=1}^{n} \ell_i^{e_i - 1}]P$
    if P' 在每个 ℓ_i 分量上具有非零投影 **then**
        found $\leftarrow$ true
$h_1 \leftarrow 0$, found $\leftarrow$ false
while found $=$ false **do**
    $h_1 \leftarrow h_1 + 1$
    $Q \leftarrow (1 + (h_0 + h_1)i : 1)$
    $F \leftarrow X(Q)(C^2 X(Q)^2 + ACX(Q) + C^2)$
    if lssquare$(F) \neq$ flag **then**
        continue
    $Q \leftarrow [c_N]Q$
    $Q' \leftarrow [\prod_{i=1}^{n} \ell_i^{e_i - 1}]Q$
    if Q' 与 P' 无关 **then**
        found $\leftarrow$ true
$T \leftarrow$ ProjectiveDifference(P, Q, E)
return $(P, Q, T), (h_0, h_1)$

Algorithm 43 TorsionBasisGivenOrderFromHint$(E, N, hint)$

Input: 超奇异 Montgomery 曲线 $E/\mathbb{F}_{p^2}$，整数 N 和提示 $h = (h_0, h_1)$

Output: $E[N]$ 的 x-only 基 $\mathcal{B} = (P, Q, P-Q)$

令余因子 c_N 为用于将点投影到 N 挠上的整数
$P \leftarrow (1 + h_0 i : 1)$
$P \leftarrow [c_N]P$
$Q \leftarrow (1 + (h_0 + h_1)i : 1)$
$Q \leftarrow [c_N]Q$
$T \leftarrow$ ProjectiveDifference(P, Q, E)
return (P, Q, T)

Algorithm 44 LabelComputation(N, u, v)

Input: 整数 $N = \prod_{i=1}^{n} \ell_i^{e_i}$ 和两个标量 $u, v \in \mathbb{Z}/N\mathbb{Z}$
Output: 标量 $m, s \in \mathbb{Z}/N\mathbb{Z}$ 和 n 比特标签 $\mathsf{label} = (l_n l_{n-1} \cdots l_1)_2$ 使得

$$m = \begin{cases} u^{-1} \bmod \ell_i^{e_i}, \text{若 } l_i = 0, \\ v^{-1} \bmod \ell_i^{e_i}, \text{否则}, \end{cases}, s = \begin{cases} mv \bmod \ell_i^{e_i}, \text{若 } l_i = 0, \\ mu \bmod \ell_i^{e_i}, \text{否则}, \end{cases}$$

$label \leftarrow 0$
for $i = 1, 2, \cdots, n$ **do**
  if $u \bmod \ell_i = 0$ **then**
    $label = label + 2^i$, $m' \leftarrow v^{-1} \bmod \ell^{e_i}$, $s' \leftarrow m' \cdot u$
  else
    $m' \leftarrow u^{-1} \bmod \ell^{e_i}$, $s' \leftarrow m' \cdot v$
  if $i = 1$ **then**
    $m \leftarrow m'$, $s \leftarrow s'$, $m'' \leftarrow m'$, $s'' \leftarrow s'$, $M \leftarrow \ell_1^{e_1}$
  else
    计算 m 使得 $m \equiv m' \bmod \ell_i^{e_i}$ 且 $m \equiv m'' \bmod M$
    计算 s 使得 $s \equiv s' \bmod \ell_i^{e_i}$ 且 $s \equiv s'' \bmod M$
    $m'' \leftarrow m, s'' \leftarrow s, M \leftarrow M \cdot \ell_i^{e_i}$
return m, s, label

Algorithm 45 ScalarRecoveryFromLabel$(N, s, label)$

Input: 整数 $N = \prod_{i=1}^{n} \ell_i^{e_i}$, 标量 $s \in \mathbb{Z}/N\mathbb{Z}$ 和 n 比特标签 $\mathsf{label} = (l_n l_{n-1} \cdots l_1)_2$
Output: 标量 $s_1, s_2 \in \mathbb{Z}/N\mathbb{Z}$ 使得

$$s_1 = \begin{cases} s \bmod \ell_i^{e_i}, \text{若 } l_i = 0, \\ 1 \bmod \ell_i^{e_i}, \text{否则}, \end{cases}, s_2 = \begin{cases} 1 \bmod \ell_i^{e_i}, \text{若 } l_i = 0, \\ s \bmod \ell_i^{e_i}, \text{否则}, \end{cases}$$

for $i = 1, 2, \cdots, n$ **do**
  if $l_i = 1$ **then**
    $s_1' \leftarrow 1$, $s_2' \leftarrow s \bmod \ell^{e_i}$
  else
    $s_1' \leftarrow s \bmod \ell^{e_i}$, $s_2' \leftarrow 1$
  if $i = 1$ **then**
    $s_1 \leftarrow s_1'$, $s_2 \leftarrow s_2'$, $s_1'' \leftarrow s_1'$, $s_2'' \leftarrow s_2'$, $M \leftarrow \ell_1^{e_1}$
  else
    计算 s_1 使得 $s_1 \equiv s_1' \bmod \ell_i^{e_i}$ 且 $s_1 \equiv s_1'' \bmod M$
    计算 s_2 使得 $s_2 \equiv s_2' \bmod \ell_i^{e_i}$ 且 $s_2 \equiv s_2'' \bmod M$
    $s_1'' \leftarrow s_1, s_2'' \leftarrow s_2, M \leftarrow M \cdot \ell_i^{e_i}$
return s_1, s_2

CHAPTER D

一维同源（实现细节）*

QIMEN-PIKE 使用一维光滑次数同源，将其分解为由小素数同源计算组成的链。QIMEN-PIKE 还需要光滑奇数次同源来计算自同态。可用的奇数次因子是整除 $p+1$ 或 $p-1$ 的小素数幂。对于每个素数 $\ell = 2d+1$，同源计算采用 Costello 和 Hisil [CH17] 提出的混合 Montgomery–Twisted Edwards 公式。Montgomery 射影坐标 $(X : Z)$ 通过 $y = (X - Z, X + Z)$ 映射到 Twisted Edwards 坐标。Edwards 模型中的差分倍点和加法（Algorithms 46 and 47）则用于构造核点集。

Algorithm 46 $\textsc{yDBL}(P, (A : C))$

Require: Twisted Edwards 点 $P = (Y_P : Z_P)$，曲线 E 的 Montgomery 常数 $(A : C)$ **Ensure:** Twisted Edwards 形式下的 $[2]P = (Y_{2P} : Z_{2P})$

1. $t_0 \leftarrow Y_P^2$
2. $t_1 \leftarrow Z_P^2$
3. $Z \leftarrow C \cdot t_0$
4. $X \leftarrow Z \cdot t_1$
5. $t_1 \leftarrow t_1 - t_0$
6. $t_0 \leftarrow A \cdot t_1$
7. $Z \leftarrow Z + t_0$
8. $Z \leftarrow Z \cdot t_1$
9. $Y_{2P} \leftarrow X - Z$
10. $Z_{2P} \leftarrow X + Z$
11. **return** $(Y_{2P} : Z_{2P})$

▷ 代价：$2\mathrm{S} + 3\mathrm{M} + 3\mathrm{a}$

Algorithm 47 $\textsc{yADD}(P, Q, P-Q)$

Require: Twisted Edwards 点 $P = (Y_P : Z_P)$, $Q = (Y_Q : Z_Q)$, $P - Q = (Y_{PQ} : Z_{PQ})$
Ensure: Twisted Edwards 形式下的 $P + Q = (Y_R : Z_R)$

1. $a \leftarrow Z_P \cdot Y_Q$
2. $b \leftarrow Y_P \cdot Z_Q$
3. $c \leftarrow (a + b)^2$
4. $d \leftarrow (a - b)^2$
5. $X \leftarrow (Z_{PQ} - Y_{PQ}) \cdot c$
6. $Z \leftarrow (Z_{PQ} + Y_{PQ}) \cdot d$
7. $Y_R \leftarrow X - Z$
8. $Z_R \leftarrow X + Z$
9. **return** $(Y_R : Z_R)$

▷ 代价：$2\mathrm{S} + 2\mathrm{M} + 6\mathrm{a}$

Algorithm 48给出了核点集的构造方法。

Algorithm 48 $\textsc{KernelPointSet}(\ell, P, (A : C))$

Input: 奇素数 $\ell = 2d + 1$；Montgomery 形式下的核生成元 $P = (X_P : Z_P)$；曲线常数 $(A : C)$
Output: Twisted Edwards 形式下的核点集 $\{K_j\}_{j=0}^{d-1}$

$K_0 \leftarrow (X_P - Z_P,\ X_P + Z_P)$ ▷ Montgomery → Edwards
$K_1 \leftarrow \textsc{yDBL}(K_0, (A : C))$
for $j = 2, \ldots, d - 1$ **do**
    $K_j \leftarrow \textsc{yADD}(K_{j-1}, K_0, K_{j-2})$ ▷ $K_j = [j+1]P$
return $\{K_0, K_1, \ldots, K_{d-1}\}$

OddIsogenyCodomain（Algorithm 49）分三步计算像曲线：首先计算 Edwards 模型中所有核点坐标之积；然后通过从左至右的二进制求幂将 Edwards 曲线参数 a 和 d 提升至 ℓ 次幂；最后映射回 Montgomery 形式。

Algorithm 49 OddIsogenyCodomain$(\ell, (A : C))$

Input: 奇素数 $\ell = 2d+1$；Montgomery 常数 $(A : C)$；核集 $\{K_j\}_{j=0}^{d-1}$（预计算）
Output: 像 Montgomery 系数 $(A' : C')$

$B_y \leftarrow K_0.y$; $B_z \leftarrow K_0.z$ ▷ 累积乘积
for $j = 1, \ldots, d-1$ **do**
  $B_y \leftarrow B_y \cdot K_j.y$; $B_z \leftarrow B_z \cdot K_j.z$
$d_E \leftarrow A - C$ ▷ Edwards 参数 d
$a_E \leftarrow A$; $d'_E \leftarrow d_E$
for $j = 1, \ldots, \lceil \log_2 \ell \rceil - 1$ **do** ▷ 二进制求幂计算 a_E^ℓ, d_E^ℓ
  $a_E \leftarrow a_E^2$; $d'_E \leftarrow d'^2_E$
  if ℓ 的第 j 比特为 1 **then**
    $a_E \leftarrow a_E \cdot A$; $d'_E \leftarrow d'_E \cdot d_E$
$B_y \leftarrow B_y^8$; $B_z \leftarrow B_z^8$ ▷ 提升至 8 次幂（三次平方）
$A' \leftarrow a_E \cdot B_z$
$C' \leftarrow A' - d'_E \cdot B_y$
return $(A' : C')$

OddIsogenyEval（Algorithm 51）使用 Costello 和 Hisil 的 CrissCross 算法计算点在同源下的像。

Algorithm 50 CrissCross$(\alpha, \beta, \gamma, \delta)$

Input: $\alpha, \beta, \gamma, \delta \in \mathbb{F}_{p^2}$
Output: (r_0, r_1) 其中 $r_0 = \alpha\delta + \beta\gamma$, $r_1 = \alpha\delta - \beta\gamma$

$t_1 \leftarrow \alpha \cdot \delta$; $t_2 \leftarrow \beta \cdot \gamma$
$r_0 \leftarrow t_1 + t_2$; $r_1 \leftarrow t_1 - t_2$
return (r_0, r_1) ▷ 代价：$2\mathrm{M} + 2\mathrm{a}$

Algorithm 51 OddIsogenyEval(ℓ, P)

Input: 奇素数 $\ell = 2d+1$；射影点 $P = (X_P : Z_P)$；核集 $\{K_j\}_{j=0}^{d-1}$（预计算）
Output: P 在 ℓ-同源下的像 $Q = (X_Q : Z_Q) \in E'$

$S_0 \leftarrow X_P + Z_P$; $S_1 \leftarrow X_P - Z_P$
$(R_0, R_1) \leftarrow$ CrissCross$(K_0.z,\ K_0.y,\ S_0,\ S_1)$
for $j = 1, \ldots, d-1$ **do**
  $(T_0, T_1) \leftarrow$ CrissCross$(K_j.z,\ K_j.y,\ S_0,\ S_1)$
  $R_0 \leftarrow R_0 \cdot T_0$; $R_1 \leftarrow R_1 \cdot T_1$
$R_0 \leftarrow R_0^2$; $R_1 \leftarrow R_1^2$
$X_Q \leftarrow X_P \cdot R_0$; $Z_Q \leftarrow Z_P \cdot R_1$
return $(X_Q : Z_Q)$

在 QIMEN-PIKE 中，协议中的奇数次同源链被分解为次数 ℓ^e 的子链，其中 ℓ 为奇素数。相应的计算由两个算法配合完成：一个顶层算法处理完整的奇数链，一个递归程序处理每个奇素数幂子链。每个素数幂子链均采用平衡策略计算。

Algorithm 52 OddIsogenyChain(E, K, N, pts)

Input: Montgomery 曲线 E，E 上阶为 $N = \prod_{i=1}^{n} \ell_i^{e_i}$ 的奇核分量的生成元 K，以及待求值的点列表 $\text{pts} = (Q_1, \ldots, Q_m)$。
Output: 像曲线 E' 以及 pts 中所有点在奇数次同源下的像。

eval_pts $\leftarrow (Q_1, \ldots, Q_m, K)$
for $i = 1$ to n **do**
    $P \leftarrow \text{eval\_pts}[m+1]$
    for $j = i+1$ to n **do**
        $P \leftarrow \text{Ladder}(P, E, \ell_j^{e_j})$
    $\mathcal{S} \leftarrow ()$
    PrimePowerRec$(E, P, i, e_i, \mathcal{S}, \text{eval_pts})$
pts $\leftarrow (\text{eval\_pts}[1], \ldots, \text{eval\_pts}[m])$
return (E, pts)

Algorithm 53 PrimePowerRec$(E, P, \ell_i, h, \mathcal{S}, \text{pts})$

Input: Montgomery 曲线 $E = (A : C)$，阶为 ℓ_i^h 的点 P，核点栈 $\mathcal{S}$，以及需要求值的点列表 pts。
Output: 将 E 替换为相应 ℓ_i^h-子链后的像曲线，并将 $\mathcal{S} \cup \text{pts}$ 中的每个点替换为其像。

if $h = 0$ **then**
    return
if $h = 1$ **then**
    $\mathcal{K} \leftarrow \text{KernelPointSet}(\ell_i, P, (A : C))$
    $E_{\text{new}} \leftarrow \text{OddIsogenyCodomain}(\ell_i, (A : C))$
    for 每个 $S \in \mathcal{S}$ **do**
        $S \leftarrow \text{OddIsogenyEval}(\ell_i, S)$
    for 每个 $Q \in \text{pts}$ **do**
        $Q \leftarrow \text{OddIsogenyEval}(\ell_i, Q)$
    $E \leftarrow E_{\text{new}}$
    return
$r \leftarrow \lfloor h/1.5 \rfloor, \quad s \leftarrow h - r$
将 P 压入 $\mathcal{S}$
$P_{\text{right}} \leftarrow \text{Ladder}(P, E, \ell_i^s)$
PrimePowerRec$(E, P_{\text{right}}, i, r, \mathcal{S}, \text{pts})$
$P_{\text{left}} \leftarrow \mathcal{S}$ 的栈顶，并将其从 $\mathcal{S}$ 弹出
PrimePowerRec$(E, P_{\text{left}}, i, s, \mathcal{S}, \text{pts})$

CHAPTER E

二维同源（实现细节）*

本节描述次数为 2^n 的二维同源（即 $(2^n, 2^n)$-同源）在 level-2 theta 坐标下的实现。$(2,2)$-同源链的算法源自 Dartois、Maino、Pope 和 Robert 的工作 [DMPR24]。本节运算开销以基域 $\mathbb{F}_{p^2}$ 上的基本算术操作计量，分别记为 S（平方）、M（乘法）、I（求逆）和 a（加法）。我们首先介绍 theta 坐标上的运算，这是计算此类同源链的基础。

- Section E.1 引入 level-2 theta 坐标，
- Section E.2 描述了用 theta 坐标对点倍点的公式，

椭圆曲线乘积 $E_1 \times E_2 \to E_3 \times E_4$ 之间的 $(2,2)$-同源链分三个阶段执行。

1. 粘合同源 (gluing isogeny)：从乘积曲面 $E_1 \times E_2$ 映射到带有固定 theta 结构的主极化阿贝尔曲面，参见 Section E.4；
2. 在 theta 坐标中迭代一般 $(2,2)$-同源，参见 Section E.3；
3. 当最后陪域阿贝尔曲面同构于椭圆曲线乘积 $E_3 \times E_4$ 时，将当前的 theta 结构转化为曲线乘积上的 theta 结构，并返回点的坐标。参见 Section E.5。

E.1 level-2 theta 坐标

主极化阿贝尔曲面上的运算使用 level-2 theta 坐标。level-2 theta 坐标提供了一种非常适合 $(2,2)$-同源公式的射影模型, 它们在 theta 坐标模型中同时支持高效倍点、陪域曲面恢复和同源求值。

E.1.1 Montgomery 曲线上

设 E 为 $\mathbb{F}_{p^2}$ 上的 Montgomery 曲线，由

$$By^2 = x^3 + Ax^2 + x, \qquad A, B \in \mathbb{F}_{p^2}.$$

定义。我们仍用 x-only 坐标 $(X : Z)$ 表示 E 上的点。level-2 theta 坐标给出了相同点的另一种坐标表示，同样只定义到符号。取 $E[4]$ 的一组基 (T_1', T_2')，使得

$$T_1' = (-1 : 1), \qquad T_2' = (r : s),$$

则对应的 theta 零点为

$$(a : b) = (r + s : r - s).$$

一旦选定了 $(a : b)$，Montgomery 坐标到 theta 坐标的变换为

$$(X : Z) \longmapsto (\theta_0 : \theta_1) = \big(a(X - Z) : b(X + Z)\big),$$

逆变换为

$$(\theta_0 : \theta_1) \longmapsto (X : Z) = \big(a\theta_1 + b\theta_0 : a\theta_1 - b\theta_0\big).$$

无穷远点取 $(1:0)$。在 (2,2)-同源链计算的边界处，椭圆曲线乘积上的点需要在 Montgomery 坐标与乘积曲面上的 level-2 theta 坐标之间转换。进入 (2,2)-同源链时，我们将 (E_1) 和 (E_2) 上的点转换为 $(E_1 \times E_2)$ 的 product theta coordinates；链结束后，再通过 splitting 阶段的坐标变换恢复为两个椭圆曲线上的坐标。前一方向的转换过程见 Algorithm 54。

Algorithm 54 MontgomeryToTheta$(P, 0)$

Require: Montgomery 坐标中的点 $P = (X : Z)$ 以及 theta 零点 $0 = (a : b)$
Ensure: theta 坐标中的点 $P = (\theta_0 : \theta_1)$

1. $\theta_0 \leftarrow X - Z$ 2. $\theta_0 \leftarrow a \cdot \theta_0$ 3. $\theta_1 \leftarrow X + Z$
4. $\theta_1 \leftarrow b \cdot \theta_1$ 5. **return** $(\theta_0 : \theta_1)$. ▷ 代价: 2M + 2a

E.1.2 主极化阿贝尔曲面上

设 A 是定义在 $\mathbb{F}_{p^2}$ 上的主极化阿贝尔曲面。我们用射影坐标 $(x : y : z : w)$ 表示 A 上的 level-2 theta 结构。若 A 是某条曲线的 Jacobian，则这组坐标表示 Jacobian 上的一个点，但只能确定到符号；若 $A \simeq E_1 \times E_2$，则它表示一对椭圆曲线点，同样带有自然的符号歧义。和椭圆曲线情形类似，整个 theta 坐标系 theta 零点 $0_A = (a : b : c : d) := (x(0) : y(0) : z(0) : w(0))$ 确定。同一个曲面上可以选取不同的 theta 结构，对应的射影坐标系也会不同；这些坐标系之间可以通过显式的线性变换相互转换。在实现中，我们会在每个中间像曲面上固定一个 theta 结构，并在之后的倍点和求值公式中始终使用这一套坐标。

E.1.3 乘积 theta 坐标

设 $A = E_1 \times E_2$ 为 Montgomery 椭圆曲线的乘积，令

$$(\theta_0 : \theta_1) \in E_1, \qquad (\theta_0' : \theta_1') \in E_2$$

为两个分量的 level-2 theta 坐标。A 上对应的乘积 theta 坐标为 $(x : y : z : w) = (\theta_0\theta_0' : \theta_1\theta_0' : \theta_0\theta_1' : \theta_1\theta_1')$。这正是乘积 theta 结构天然带有的坐标。然而，链的中间步骤所涉及的阿贝尔曲面并不用此坐标表示。因此，粘合步需要从乘积 theta 坐标到像曲面上所用固定 theta 结构的基变换。反之，在最终的分裂之后，我们从该固定 theta 结构返回到乘积 theta 结构，以恢复两个椭圆曲线分量的 Montgomery 表示。

E.2 使用 theta 坐标的倍点公式

设 A 为主极化阿贝尔曲面，其 theta 零点为 $0_A = (a : b : c : d)$。记

$$\mathcal{H}(x, y, z, w) := (x + y + z + w,\ x - y + z - w,\ x + y - z - w,\ x - y - z + w),$$
$$\mathcal{S}(x, y, z, w) := (x^2, y^2, z^2, w^2)$$

分别为 Hadamard 变换和逐分量平方算子。Hadamard 变换代价为 8a，$\mathcal{S}$ 代价为 4S。

首先从 theta 零点导出对偶 theta 零点 $(a' : b' : c' : d') := \mathcal{H}(a, b, c, d)$。对于 theta 结构附带的典范 2-同源，对偶同源的 theta 零点为 $(\alpha^2 : \beta^2 : \gamma^2 : \delta^2) = \mathcal{H}(a^2 : b^2 : c^2 : d^2)$。 这些常数正是 theta 坐标中倍点所需的量。

公式中反复出现的常数可以预计算，以降低重复倍点的代价。ThetaPrecomp 用于计算这些常数。得到这些常量后，ThetaDBL 依次应用逐分量平方、Hadamard 变换和逐分量乘法，即可得到 $[2]P$ 的 theta 坐标。在进入下一个同源之前，通常要对同一个陪域曲面 theta 零点做多次连续倍点，该预计算结果在链的每一层都会被复用。

Algorithm 55 ThetaPrecomp(0_A)

Require: theta 零点 $0_A = (a : b : c : d)$
Ensure: 辅助常数 consts

1. $(A, B, C, D) \leftarrow \mathcal{H} \circ \mathcal{S}(a, b, c, d)$
2. $t_1 \leftarrow A \cdot B$
3. $t_2 \leftarrow C \cdot D$
4. $ABC \leftarrow t_1 \cdot C$
5. $ABD \leftarrow t_1 \cdot D$
6. $ACD \leftarrow t_2 \cdot A$
7. $BCD \leftarrow t_2 \cdot B$
8. $t_1 \leftarrow a \cdot b$
9. $t_2 \leftarrow c \cdot d$
10. $abc \leftarrow t_1 \cdot c$
11. $abd \leftarrow t_1 \cdot d$
12. $acd \leftarrow t_2 \cdot a$
13. $bcd \leftarrow t_2 \cdot b$
14. consts $\leftarrow \{abc, abd, acd, bcd, ABC, ABD, ACD, BCD\}$
15. **return** consts

▷ 代价: 4S + 12M (+8a)

Algorithm 56 ThetaDBL(P, consts)

Require: A 上点 P 的 theta 坐标，A 的 theta 零点为 $0_A = (a : b : c : d)$，且 consts = ThetaPrecomp(0_A)
Ensure: $[2]P$ 的 theta 坐标

1. $(x_P, y_P, z_P, w_P) \leftarrow P$
2. $(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8) \leftarrow$ consts
3. $(X_{2P}, Y_{2P}, Z_{2P}, W_{2P}) \leftarrow \mathcal{S} \circ \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, w_P)$
4. $X_{2P} \leftarrow X_{2P} \cdot c_8$
5. $Y_{2P} \leftarrow Y_{2P} \cdot c_7$
6. $Z_{2P} \leftarrow Z_{2P} \cdot c_6$
7. $W_{2P} \leftarrow W_{2P} \cdot c_5$
8. $(X_{2P}, Y_{2P}, Z_{2P}, W_{2P}) \leftarrow \mathcal{H}(X_{2P}, Y_{2P}, Z_{2P}, W_{2P})$
9. $X_{2P} \leftarrow X_{2P} \cdot c_4$
10. $Y_{2P} \leftarrow Y_{2P} \cdot c_3$
11. $Z_{2P} \leftarrow Z_{2P} \cdot c_2$
12. $W_{2P} \leftarrow W_{2P} \cdot c_1$
13. **return** $(X_{2P} : Y_{2P} : Z_{2P} : W_{2P})$

▷ 代价: 8S + 8M + 16a

E.3 一般 $(2, 2)$-同源计算

对于链的一般同源计算。设 $\Phi : A \to B$ 为主极化阿贝尔曲面之间的 $(2, 2)$-同源，原曲面和像曲面都以某个给定的 theta 结构表示。在给定 $\ker(\Phi)$ 上方的挠点，恢复 B 的 theta 零点、对偶 theta 零点和它的逆。实现中提供了三个陪域曲面恢复程序，根据核上方可用的挠点阶选择不同的程序。

- 当核上方的相容 8-挠点已知时，可直接使用 8-挠点版本的通用 codomain 公式，即 Section E.3.1中的GenericCodomainWith8Torsion。
- 在链末端附近，可能只剩下相容的 4-挠点。此时陪域曲面恢复需要额外开平方根，参见 Section E.3.2，GenericCodomainWith4Torsion。
- 在链计算的最后一步，当仅有核生成元本身时，直接从原曲面的 theta 零点恢复陪域曲面，参见 Section E.3.3，GenericCodomain。
- 无论使用哪种方法计算陪域曲面，都可以使用 GenericEval 对像点进行求值，参见 Section E.3.4。

本文始终在与所选 theta 结构相容的核上工作。这个相容性是后续公式成立的前提：只有在该条件下，核才能确定陪域曲面的 theta 结构。对一条 ((2,2))-同源链而言，初始粘合同源首先选定相容的 theta 结构；之后，每一步都通过传递的挠点继续确定下一层的 theta 结构，并在需要时检查这些点是否具有预期形状，从而保证相容性沿链传递。

E.3.1 使用核上方的 8-挠点恢复陪域

设 $T_1'', T_2'' \in A[8]$ 为与当前 theta 结构相容的 (8)-挠点，并满足 $\ker(\Phi) = \langle [4]T_1'' \rangle \oplus \langle [4]T_2'' \rangle$。可以直接调用 GenericCodomainWith8Torsion 得到三类陪域曲面量：陪域曲面的对偶 theta 零点 $(\alpha : \beta : \gamma : \delta)$、其射影逆 $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$ 以及陪域曲面上的 theta 零点 $0_B = (a_2 : b_2 : c_2 : d_2)$。

此时可以采用成本最低的陪域恢复路径。对 (T_1'') 和 (T_2'') 应用 $H \circ S$ 后得到的两个四元组，已经给出了由 duplication formula 恢复对偶 theta 零点所需的乘法关系。因此，$(\alpha : \beta : \gamma : \delta)$ 及其射影逆可以直接确定，唯一的不确定性来自整体射影因子。整个过程无需开平方根。换言之，只要链上保存有相容的 (8)-挠点，陪域 theta 零点的恢复以及后续计算都可以在这一套无开方的流程中完成。

在链中重复调用这一过程时，需要排除两类情形。首先，在每一步 generic codomain 得到之后，需要检查恢复出的对偶 theta 零点 $(\alpha : \beta : \gamma : \delta)$ 是否四个分量均非零。该条件作用于，若 $\alpha\beta\gamma\delta = 0$，则当前

陪域 B 已是椭圆曲线乘积情形，即链发生提前 split。第二，即使没有提前 spilt，被传递挠点的像在陪域曲面上也必须是相容的 4-挠点。下面的迷向性检查正是为了验证这一点。

迷向性与相容性检查。 令 $P = \mathcal{H} \circ \mathcal{S}(T_1'')$，$Q = \mathcal{H} \circ \mathcal{S}(T_2'')$，并令 $I = (\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$ 为陪域曲面上的对偶 theta 零点的逆。CheckIsotropic 中的检查确保所选 8-挠点的像恰好具有所需的形式，从而能够在下一核的上方充当相容的 4-挠点。

具体而言，T_1'' 的像点的对偶 theta 坐标必须具有形状 $(x : x : y : y)$，T_2'' 的像点的对偶 theta 坐标必须形如 $(z : w : z : w)$。等价地，检查以下四个关系：

$$x_P\alpha^{-1} = y_P\beta^{-1}, \qquad z_P\gamma^{-1} = w_P\delta^{-1},$$

以及

$$x_Q\alpha^{-1} = z_Q\gamma^{-1}, \qquad y_Q\beta^{-1} = w_Q\delta^{-1}.$$

当这些关系同时成立时，$\Phi(T_1'')$ 与 $\Phi(T_2'')$ 确为陪域上的相容 4-挠提升：它们位于下一步 (2)-挠核生成元的上方，并与 B 上选定的 theta 结构相容。

Algorithm 57 GenericCodomainWith8Torsion(T_1'', T_2'')

Input: T_1'' 和 T_2'' 的 theta 坐标，满足 $\ker(\Phi) = \langle [4]T_1'' \rangle \oplus \langle [4]T_2'' \rangle$
Output: 对偶同源 theta 零点 $(\alpha : \beta : \gamma : \delta)$、其逆 $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$、以及 B 上的 theta 零点 0_B

$x_{T_1''}, y_{T_1''}, z_{T_1''}, w_{T_1''} \leftarrow T_1''$
$x_{T_2''}, y_{T_2''}, z_{T_2''}, w_{T_2''} \leftarrow T_2''$
$(x\alpha, x\beta, y\gamma, y\delta) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_1''}, y_{T_1''}, z_{T_1''}, w_{T_1''})$
$(z\alpha, w\beta, z\gamma, w\delta) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_2''}, y_{T_2''}, z_{T_2''}, w_{T_2''})$
if $0 \in \{x\alpha, x\beta, z\alpha, w\beta, z\gamma, w\delta\}$ **then**
    raise 异常: ("`generic-codomain-8torsion` 失败: 提前分裂")
$x\alpha w\beta \leftarrow x\alpha \cdot w\beta$
$z\alpha x\beta \leftarrow z\alpha \cdot x\beta$
$\alpha \leftarrow z\alpha \cdot x\alpha w\beta$
$\beta \leftarrow w\beta \cdot z\alpha x\beta$
$\gamma \leftarrow z\gamma \cdot x\alpha w\beta$
$\delta \leftarrow w\delta \cdot z\alpha x\beta$
$z\gamma w\delta \leftarrow z\gamma \cdot w\delta$
$\alpha^{-1} \leftarrow x\beta \cdot z\gamma w\delta$
$\beta^{-1} \leftarrow x\alpha \cdot z\gamma w\delta$
$\gamma^{-1} \leftarrow \delta$
$\delta^{-1} \leftarrow \gamma$
$P \leftarrow (x\alpha : x\beta : y\gamma : y\delta)$
$Q \leftarrow (z\alpha : w\beta : z\gamma : w\delta)$
$I \leftarrow (\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$
if not CheckIsotropic(P, Q, I) **then**
    raise 异常: ("`generic-codomain-8torsion` 失败: P,Q 非迷向")
$(a_2, b_2, c_2, d_2) \leftarrow \mathcal{H}(\alpha, \beta, \gamma, \delta)$
$0_B \leftarrow (a_2 : b_2 : c_2 : d_2)$
return $(\alpha : \beta : \gamma : \delta)$, $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$, 0_B ▷ 总代价（不含检查）：$8\mathrm{S} + 9\mathrm{M} + 24\mathrm{a}$

Algorithm 58 CheckIsotropic(P, Q, I)

Require: theta 坐标 $P = (x_P, y_P, z_P, w_P)$ 和 $Q = (x_Q, y_Q, z_Q, w_Q)$，以及对偶 theta 逆点 $I = (\alpha^{-1}, \beta^{-1}, \gamma^{-1}, \delta^{-1})$
Ensure: 布尔值，指示相应 4-挠点是否为迷向

1. $t_1 \leftarrow x_P \cdot \alpha^{-1}$
2. $t_2 \leftarrow y_P \cdot \beta^{-1}$
3. **if** $t_1 \neq t_2$, **return false**
4. $t_1 \leftarrow z_P \cdot \gamma^{-1}$
5. $t_2 \leftarrow w_P \cdot \delta^{-1}$
6. **if** $t_1 \neq t_2$, **return false**
7. $t_1 \leftarrow x_Q \cdot \alpha^{-1}$
8. $t_2 \leftarrow z_Q \cdot \gamma^{-1}$
9. **if** $t_1 \neq t_2$, **return false**
10. $t_1 \leftarrow y_Q \cdot \beta^{-1}$
11. $t_2 \leftarrow w_Q \cdot \delta^{-1}$
12. **if** $t_1 \neq t_2$, **return false**
13. **return true** ▷ 代价: 最多 8M

E.3.2 使用核上方 4-挠点

现在假设核上方不再有相容的 8-挠点可用，但已知点 $T_1' \in A[4]$ 满足 $[2]T_1' \in \ker(\Phi)$。此时，陪域曲面仍可通过 GenericCodomainWith4Torsion 恢复。

与前一种情形不同的是，可用的挠点不再能唯一地通过乘法确定对偶 theta 坐标。因此，该算法将 $\mathcal{H} \circ \mathcal{S}(T_1')$ 提与原曲线的 theta 零点

$$(\alpha^2, \beta^2, \gamma^2, \delta^2) = \mathcal{H} \circ \mathcal{S}(a, b, c, d), \qquad 0_A = (a : b : c : d),$$

来恢复陪域曲面的 theta 零点。

此程序仅在可用挠点从 8 阶下降到 4 阶时使用，它只出现在链的最后若干个非终止步中。与 8-挠点情形相比，公式代价更高且形式更不均匀。

Algorithm 59 GenericCodomainWith4Torsion$(T_1', 0_A)$

Require: 4 阶点 T_1' 的 theta 坐标，满足 $[2]T_1' \in \ker(\Phi)$，以及 theta 零点 $0_A = (a : b : c : d)$
Ensure: 对偶同源 theta 零点 $(\alpha : \beta : \gamma : \delta)$、其逆 $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$、以及 B 上的 theta 零点 0_B

1. $(x\alpha\beta, \_, x\gamma\delta, \_) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_1'}, y_{T_1'}, z_{T_1'}, w_{T_1'})$
2. $(\alpha^2, \beta^2, \gamma^2, \delta^2) \leftarrow \mathcal{H} \circ \mathcal{S}(a, b, c, d)$
3. $\alpha\beta \leftarrow \sqrt{\alpha^2\beta^2}$
4. $\alpha\gamma \leftarrow \sqrt{\alpha^2\gamma^2}$
5. $\beta \leftarrow \alpha\beta \cdot \alpha\gamma$
6. $\delta^{-1} \leftarrow \beta \cdot x\gamma\delta$
7. $\beta \leftarrow \beta \cdot x\alpha\beta$
8. $\delta \leftarrow x\gamma\delta \cdot \alpha\beta \cdot \alpha^2$
9. $\alpha \leftarrow x\alpha\beta \cdot \alpha^2$
10. $\gamma \leftarrow \alpha \cdot \gamma^2$
11. $\alpha \leftarrow \alpha \cdot \alpha\gamma$
12. $\alpha^{-1} \leftarrow x\alpha\beta \cdot \delta^2$
13. $\gamma^{-1} \leftarrow \alpha^{-1} \cdot \beta^2$
14. $\alpha^{-1} \leftarrow \alpha^{-1} \cdot \gamma^2$
15. $\beta^{-1} \leftarrow \alpha^{-1} \cdot \alpha\beta$
16. $\alpha^{-1} \leftarrow \alpha^{-1} \cdot \beta^2$
17. $\gamma^{-1} \leftarrow \gamma^{-1} \cdot \alpha\gamma$
18. $\delta^{-1} \leftarrow \delta^{-1} \cdot \beta^2$
19. $(a_2, b_2, c_2, d_2) \leftarrow \mathcal{H}(\alpha, \beta, \gamma, \delta)$
20. $0_B \leftarrow (a_2 : b_2 : c_2 : d_2)$
21. **return** $(\alpha : \beta : \gamma : \delta)$, $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$, 0_B

E.3.3 仅使用核生成元

在链的最后阶段，核上方可能没有更高阶挠点可用，仅剩下 $\ker(\Phi)$ 的生成元 $T_1, T_2 \in A[2]$。此时，可仅从原曲面 theta 零点出发，通过 GenericCodomain 重建像曲面。

从操作层面看，这是因为一旦知道核与 theta 结构相容，剩余的陪域曲面信息已经编码在

$$(\alpha^2, \beta^2, \gamma^2, \delta^2) = \mathcal{H} \circ \mathcal{S}(a, b, c, d), \qquad 0_A = (a : b : c : d)$$

之中。从这些信息恢复陪域曲面需要计算三次平方根，因此这是三种一般程序中代价最高的。正因如此，它仅用于链的最后一步。

Algorithm 60 GenericCodomain(0_A)

Require: theta 常数 $0_A = (a : b : c : d)$
Ensure: 对偶同源 theta 零点 $(\alpha : \beta : \gamma : \delta)$、其逆 $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$、以及 B 上的 theta 零点 0_B

1. $(\alpha^2, \beta^2, \gamma^2, \delta^2) \leftarrow \mathcal{H} \circ \mathcal{S}(a, b, c, d)$
2. $\alpha \leftarrow \alpha^2$
3. $\beta \leftarrow \alpha^2 \cdot \beta^2$
4. $\gamma \leftarrow \alpha^2 \cdot \gamma^2$
5. $\delta \leftarrow \alpha^2 \cdot \delta^2$
6. $\beta \leftarrow \sqrt{\beta}$
7. $\gamma \leftarrow \sqrt{\gamma}$
8. $\delta \leftarrow \sqrt{\delta}$
9. $\alpha^{-1} \leftarrow \gamma^2 \cdot \delta^2$
10. $\beta^{-1} \leftarrow \alpha^{-1} \cdot \beta$
11. $\alpha^{-1} \leftarrow \alpha^{-1} \cdot \beta^2$
12. $\gamma^{-1} \leftarrow \delta^2 \cdot \beta^2 \cdot \gamma$
13. $\delta^{-1} \leftarrow \gamma^2 \cdot \beta^2 \cdot \delta$
14. $(a_2, b_2, c_2, d_2) \leftarrow \mathcal{H}(\alpha, \beta, \gamma, \delta)$
15. $0_B \leftarrow (a_2 : b_2 : c_2 : d_2)$
16. **return** $(\alpha : \beta : \gamma : \delta)$, $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$, 0_B

E.3.4 一般同源求值

一旦一般 $(2, 2)$-同源的陪域曲面已经恢复——无论通过 GenericCodomainWith8Torsion、GenericCodomainWith4Torsion 还是 GenericCodomain——同源本身的求值均采用统一的方式。确切地说，假设陪域曲面 theta 零点 0_B 和对偶 theta 逆点

$$I = (\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$$

已知，则任意点 $P \in A$ 可以仅使用其 theta 坐标和 I 在 Φ 下求值，如 GenericEval 所示。该公式与倍点程序类似：先对 P 应用 $\mathcal{H} \circ \mathcal{S}$，再用 I 的对应分量对四个坐标进行逐坐标乘法，最后应用 Hadamard 变换，得到 $\Phi(P)$ 的 theta 坐标。特别地，求值仅依赖于对偶 theta 零点的逆，而陪域曲面 theta 零点的主要用于后续的倍点。

Algorithm 61 GenericEval(P, I)

Input: P 的 theta 坐标和对偶 theta 零点逆 $I = (\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$
Output: $\Phi(P)$ 的 theta 坐标

1: $x_P, y_P, z_P, w_P \leftarrow P$
2: $\alpha^{-1}, \beta^{-1}, \gamma^{-1}, \delta^{-1} \leftarrow I$
3: $(X_P, Y_P, Z_P, W_P) \leftarrow \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, w_P)$
4: $X' \leftarrow \alpha^{-1} X_P$, $Y' \leftarrow \beta^{-1} Y_P$, $Z' \leftarrow \gamma^{-1} Z_P$, $W' \leftarrow \delta^{-1} W_P$
5: $(x_{\Phi(P)}, y_{\Phi(P)}, z_{\Phi(P)}, w_{\Phi(P)}) \leftarrow \mathcal{H}(X', Y', Z', W')$
6: **return** $(x_{\Phi(P)} : y_{\Phi(P)} : z_{\Phi(P)} : w_{\Phi(P)})$ ▷ 总代价: $4\mathrm{S} + 4\mathrm{M} + 16\mathrm{a}$

平移作用和基变换。 在讨论粘合步之前，还需交代一个要素。在乘积曲面

$$E_1 \times E_2$$

上，天然的乘积 theta 结构与粘合像曲面上使用的固定 theta 结构并不一致，因此需要一个从乘积 theta 坐标到所选 theta 结构的线性坐标变换。

该基变换建立在椭圆曲线上合适的 4-挠点平移作用之上。ActionByTranslation 计算每个因子上对应的 2×2 平移矩阵，ThetaChangeOfBasis 将它们组装成一个 4×4 矩阵 N，随后在粘合同源中复用。该矩阵纯将点从乘积 theta 坐标转换到固定 theta 结构中——陪域曲面及后续所有中间曲面均在该结构下表示。

Algorithm 62 ActionByTranslation(P, Q)

Input: $E_1 \times E_2$ 上的四挠点 $P = (P_1, P_2)$ 和 $Q = (Q_1, Q_2)$
Output: 数组 mats，包含给出 P_1, Q_1 在 E_1 上以及 P_2, Q_2 在 E_2 上平移作用的 2×2 矩阵

$P' = (P'_1, P'_2) \leftarrow [2]P$
$Q' = (Q'_1, Q'_2) \leftarrow [2]Q$
for i 从 1 到 2 **do**
    记 $P_i = (X_i^{(P)} : Z_i^{(P)})$，$Q_i = (X_i^{(Q)} : Z_i^{(Q)})$
    记 $P'_i = (U_i^{(P)} : W_i^{(P)})$，$Q'_i = (U_i^{(Q)} : W_i^{(Q)})$
    $\delta_i^{(P)} \leftarrow W_i^{(P)} X_i^{(P)} - U_i^{(P)} Z_i^{(P)}$
    $\delta_i^{(Q)} \leftarrow W_i^{(Q)} X_i^{(Q)} - U_i^{(Q)} Z_i^{(Q)}$
通过批量求逆计算下式的逆：

$$\delta_1^{(P)},\ \delta_2^{(P)},\ \delta_1^{(Q)},\ \delta_2^{(Q)},\ Z_1^{(P)},\ Z_2^{(P)},\ Z_1^{(Q)},\ Z_2^{(Q)}$$

初始化 mats $\leftarrow [\,]$
pts $\leftarrow [P, Q]$
for i 从 1 到 2 **do**
    for j 从 1 到 2 **do**
        $R \leftarrow \text{pts}[j]$
        $M_{0,0} \leftarrow -U_i^{(R)} Z_i^{(R)} \cdot (\delta_i^{(R)})^{-1}$
        $M_{0,1} \leftarrow -W_i^{(R)} Z_i^{(R)} \cdot (\delta_i^{(R)})^{-1}$
        $M_{1,0} \leftarrow U_i^{(R)} X_i^{(R)} \cdot (\delta_i^{(R)})^{-1} - X_i^{(R)} \cdot (Z_i^{(R)})^{-1}$
        $M_{1,1} \leftarrow -M_{0,0}$
        $M \leftarrow (M_{u,v})_{0 \le u,v \le 1}$
        将 M 追加到 mats
return mats

E.4 粘合 $(2, 2)$-同源

现在我们描述链的第一步，即粘合同源 $\Phi : E_1 \times E_2 \to A$。输入包含 $E_1 \times E_2$ 上的 8-挠点 T''_1, T''_2，满足 $\ker(\Phi) = \langle [4]T''_1 \rangle \oplus \langle [4]T''_2 \rangle$。

计算流程为：先从乘积 theta 坐标变换到粘合步使用的选定 theta 结构，再计算陪域曲面 theta 零点，最后通过 Φ 对点求值。

E.4.1 粘合 $(2, 2)$-同源陪域曲面计算

第一个任务是将点从 $E_1 \times E_2$ 的乘积 theta 结构转换到 A 上使用的 theta 结构。先对输入做倍点：$T'_1 = [2]T''_1, T'_2 = [2]T''_2$，得到相容的 4-挠点，然后对 T'_1, T'_2 调用 ThetaChangeOfBasis。该程序利用 ActionByTranslation 算出的平移矩阵，返回一个 4×4 的基变换矩阵 N。

矩阵 N 将 $E_1 \times E_2$ 上的乘积 theta 坐标变换到选定的 theta 结构中。因此对于点 $P = (P_1, P_2)$，程序 ProductToTheta 先由 P_1 和 P_2 的一维 theta 坐标构造乘积 theta 坐标，再应用 N。若输入点以 Montgomery 坐标给出，则先通过 MontgomeryToTheta 获取各分坐标。

然后算法将 N 应用于两个 8-挠点，把它们从乘积 theta 结构变换到选定的 theta 结构。接着对变换后的点应用 $\mathcal{H} \circ \mathcal{S}$，所得的中间量用于恢复对偶 theta 零点 $(\alpha : \beta : \gamma : 0)$、其射影逆 $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : 0)$ 以及陪域曲面 theta 零点 $0_A = \mathcal{H}(\alpha, \beta, \gamma, 0)$。

粘合陪域曲面与一般情形的区别在于：在选定的 theta 结构下，其对偶同源 theta 零点呈特殊形状 $(\alpha : \beta : \gamma : 0)$。因此，粘合并未引入一套完全独立的计算体系；相反，它只是陪域曲面恢复的退化情形——最后一个对偶坐标恒为零。

除上述量之外，粘合步还返回一个辅助点 $J = (x : x : y : y) = \widehat{\Phi(T_1'')}$。$J$ 和 N 都在 GluingEval 中复用：N 负责在选定 theta 结构中映射输入点。

有三项检查至关重要。第一，应用 $\mathcal{H} \circ \mathcal{S}$ 之后，最后一个坐标必须为零，否则计算不在选定 theta 结构所规定的粘合配置中。第二，用于构造 α, β, γ 及其逆的其余射影因子必须非零。第三，倍点 $[2]T_1''$ 和 $[2]T_2''$ 必须迷向且与选定 theta 结构相容。这些条件确保链从正确的坐标系启动，且后续一般程序无需经过任何模型变换即可直接使用。

Algorithm 63 ThetaChangeOfBasis(P, Q)

Input: $E_1 \times E_2$ 中 4 阶点 $P = (P_1, P_2)$ 和 $Q = (Q_1, Q_2)$，满足粘合核为 $\langle [2]P, [2]Q \rangle$
Output: 4×4 基变换矩阵 N

if P_1, P_2, Q_1, Q_2 的阶非 4，或 $[2]P_1 = [2]P_2$，或 $[2]Q_1 = [2]Q_2$ **then**
    raise 异常: (“`theta-change-of-basis` 失败: 粘合核是对角的或非迷向”)
$(G, G', H, H') \leftarrow$ ActionByTranslation(P, Q)
$t_1 \leftarrow G_{0,0} \cdot H_{0,0} + G_{0,1} \cdot H_{1,0}$
$t_2 \leftarrow G_{1,0} \cdot H_{0,0} + G_{1,1} \cdot H_{1,0}$
$t_3 \leftarrow G'_{0,0} \cdot H'_{0,0} + G'_{0,1} \cdot H'_{1,0}$
$t_4 \leftarrow G'_{1,0} \cdot H'_{0,0} + G'_{1,1} \cdot H'_{1,0}$
$N_{0,0} \leftarrow G_{0,0} \cdot G'_{0,0} + H_{0,0} \cdot H'_{0,0} + t_1 \cdot t_3 + 1$
$N_{0,1} \leftarrow G_{0,0} \cdot G'_{1,0} + H_{0,0} \cdot H'_{1,0} + t_1 \cdot t_4$
$N_{0,2} \leftarrow G_{1,0} \cdot G'_{0,0} + H_{1,0} \cdot H'_{0,0} + t_2 \cdot t_3$
$N_{0,3} \leftarrow G_{1,0} \cdot G'_{1,0} + H_{1,0} \cdot H'_{1,0} + t_2 \cdot t_4$
$N_{1,0} \leftarrow H'_{0,0} \cdot N_{0,0} + H'_{0,1} \cdot N_{0,1}$
$N_{1,1} \leftarrow H'_{1,0} \cdot N_{0,0} + H'_{1,1} \cdot N_{0,1}$
$N_{1,2} \leftarrow H'_{0,0} \cdot N_{0,2} + H'_{0,1} \cdot N_{0,3}$
$N_{1,3} \leftarrow H'_{1,0} \cdot N_{0,2} + H'_{1,1} \cdot N_{0,3}$
$N_{2,0} \leftarrow G_{0,0} \cdot N_{0,0} + G_{0,1} \cdot N_{0,2}$
$N_{2,1} \leftarrow G_{0,0} \cdot N_{0,1} + G_{0,1} \cdot N_{0,3}$
$N_{2,2} \leftarrow G_{1,0} \cdot N_{0,0} + G_{1,1} \cdot N_{0,2}$
$N_{2,3} \leftarrow G_{1,0} \cdot N_{0,1} + G_{1,1} \cdot N_{0,3}$
$N_{3,0} \leftarrow G_{0,0} \cdot N_{1,0} + G_{0,1} \cdot N_{1,2}$
$N_{3,1} \leftarrow G_{0,0} \cdot N_{1,1} + G_{0,1} \cdot N_{1,3}$
$N_{3,2} \leftarrow G_{1,0} \cdot N_{1,0} + G_{1,1} \cdot N_{1,2}$
$N_{3,3} \leftarrow G_{1,0} \cdot N_{1,1} + G_{1,1} \cdot N_{1,3}$
$N \leftarrow (N_{i,j})_{0 \le i,j \le 3}$
return N

Algorithm 64 ProductToTheta(pts, N)

Input: 点列表 pts，其中对 $P \in \text{pts}$ 有 $P = (P_1, P_2) \in E_1 \times E_2$，以及基变换矩阵 N
Output: $A \cong E_1 \times E_2$ 上相应 theta 坐标中的点

eval_pts $\leftarrow [\,]$
$L \leftarrow \#\text{pts}$
for j 从 1 到 L **do**
  $P = (P_1, P_2) \leftarrow \text{pts}[j]$
  记 $P_1 = (\theta_0 : \theta_1)$，$P_2 = (\theta'_0 : \theta'_1)$
  $x \leftarrow \theta_0 \cdot \theta'_0$
  $y \leftarrow \theta_0 \cdot \theta'_1$
  $z \leftarrow \theta_1 \cdot \theta'_0$
  $w \leftarrow \theta_1 \cdot \theta'_1$
  $P' \leftarrow N \cdot (x : y : z : w)$
  将 P' 追加到 eval_pts
return eval_pts

Algorithm 65 GluingCodomain(T''_1, T''_2)

Input: $E_1 \times E_2$ 中的 8-挠点 T''_1, T''_2，满足 $\ker(\Phi) = \langle [4]T''_1 \rangle \oplus \langle [4]T''_2 \rangle$
Output: 对偶同源 theta 零点 $(\alpha : \beta : \gamma : 0)$、其逆 $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : 0)$、$A$ 上的 theta 零点 0_A、对偶 theta 点 $J = \widehat{\Phi(T''_1)}$、以及基变换矩阵 N

$T'_1 \leftarrow [2](T''_1)$
$T'_2 \leftarrow [2](T''_2)$
$N \leftarrow$ ThetaChangeOfBasis(T'_1, T'_2)
$[P_1, P_2] \leftarrow$ ProductToTheta$([T''_1, T''_2], N)$
$x_1, y_1, z_1, w_1 \leftarrow P_1$
$x_2, y_2, z_2, w_2 \leftarrow P_2$
$(X_1, Y_1, Z_1, W_1) \leftarrow \mathcal{H} \circ \mathcal{S}(x_1, y_1, z_1, w_1)$
$(X_2, Y_2, Z_2, W_2) \leftarrow \mathcal{H} \circ \mathcal{S}(x_2, y_2, z_2, w_2)$
if $W_1 \neq 0$ 或 $W_2 \neq 0$ **then**
  raise 异常: (“`gluing-codomain` 失败: 最后一个坐标非零”)
if $X_1 = 0$ 或 $X_2 = 0$ 或 $Y_1 = 0$ 或 $Z_2 = 0$ **then**
  raise 异常: (“`gluing-codomain` 失败: 射影因子为零”)
$\alpha \leftarrow X_1 \cdot X_2$
$\beta \leftarrow Y_1 \cdot X_2$
$\gamma \leftarrow X_1 \cdot Z_2$
$\alpha^{-1} \leftarrow Y_1 \cdot Z_2$
$\beta^{-1} \leftarrow \gamma$
$\gamma^{-1} \leftarrow \beta$
$x \leftarrow X_1 \cdot \alpha^{-1}$
$y \leftarrow Z_1 \cdot \gamma^{-1}$
$J \leftarrow (x : x : y : y)$
if $(Y_1 \cdot \beta^{-1} \neq x)$ **or** $(X_2 \cdot \alpha^{-1} \neq Y_2 \cdot \beta^{-1})$ **then**
  raise 异常: (“`gluing-codomain` 失败: $[2]T''_1$ 和 $[2]T''_2$ 非迷向”)
$(a_2, b_2, c_2, d_2) \leftarrow \mathcal{H}(\alpha, \beta, \gamma, 0)$
$0_A \leftarrow (a_2 : b_2 : c_2 : d_2)$
return $(\alpha : \beta : \gamma : 0)$, $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : 0)$, 0_A, J, N

E.4.2 粘合 $(2,2)$-同源求值

一般点 $P=(P_1,P_2)\in E_1\times E_2$ 随后通过 GluingEval 进行求值。该程序的坐标变换与陪域曲面计算所用的变换相同，但不能直接归约到 GenericEval。原因在于，粘合陪域曲面有一个对偶 theta 零点坐标为零，若套用一般赋值公式，$\Phi(P)$ 的一个对偶坐标将无法确定。GluingEval 正是为了恢复这一缺失信息，从而完整地计算出 $\Phi(P)$ 的 theta 坐标。

为此，算法使用了已在 GluingCodomain 中用过的 8-挠点 T_1''。在每个椭圆因子上，AddComponents 计算与 P_i 和 T_i 关联的局部数据。这些局部数据编码了 P 和平移点 $P+T_1''$ 两者的贡献。然后算法将它们组合成两个四元组，应用基变换矩阵 N，并用辅助点 J 来固定剩余的射影缩放。

此外，当输入点具有 $(P_1,0)$ 或 $(0,P_2)$ 时，还存在更简单的特殊求值情形。此时，GluingEvalSpecial 仅使用对偶 theta 逆点和基变换矩阵即可对点求值。这一捷径对 QIMEN-PIKE 很有用。

Algorithm 66 GluingEval(P,T_1'',J,N)

Input: $E_1\times E_2$ 中的点 P、满足 $[4]T_1''\in\ker(\Phi)$ 的 8-挠点 T_1''、点 $J=(x:x:y:y)$、以及 GluingCodomain 返回的基变换矩阵 N

Output: $\Phi(P)$ 的 theta 坐标

$P_1,P_2\leftarrow P$
$T_1,T_2\leftarrow T_1''$
$x,y\leftarrow J$
$u_1,v_1,z_1\leftarrow$ AddComponents(P_1,T_1,E_1)
$u_2,v_2,z_2\leftarrow$ AddComponents(P_2,T_2,E_2)
$U\leftarrow(u_1u_2+v_1v_2,\ u_1z_2,\ z_1u_2,\ z_1z_2)$
$V\leftarrow(v_1u_2+u_1v_2,\ v_1z_2,\ z_1v_2,\ 0)$
$U\leftarrow N\cdot U$
$V\leftarrow N\cdot V$
$U\leftarrow\mathcal{S}(U)$
$V\leftarrow\mathcal{S}(V)$
$X_\pm,Y_\pm,Z_\pm,W_\pm\leftarrow\mathcal{H}(U-V)$
$X_{\Phi(P)},Y_{\Phi(P)},Z_{\Phi(P)},W_{\Phi(P)}\leftarrow X_\pm\cdot y,\ Y_\pm\cdot y,\ Z_\pm\cdot x,\ W_\pm\cdot x$
$(x_{\Phi(P)},y_{\Phi(P)},z_{\Phi(P)},w_{\Phi(P)})\leftarrow\mathcal{H}(X_{\Phi(P)},Y_{\Phi(P)},Z_{\Phi(P)},W_{\Phi(P)})$
return $(x_{\Phi(P)}:y_{\Phi(P)}:z_{\Phi(P)}:w_{\Phi(P)})$

Algorithm 67 GluingEvalSpecial(P,I,N)

Input: $E_1\times E_2$ 中形如 $(P_1,0)$ 或 $(0,P_2)$ 的点 P、对偶 theta 逆点 $I=(\alpha^{-1}:\beta^{-1}:\gamma^{-1}:0)$、以及基变换矩阵 N

Output: $\Phi(P)$ 的 theta 坐标

$[\widetilde{P}]\leftarrow$ ProductToTheta$([P],N)$
$x_P,y_P,z_P,w_P\leftarrow\widetilde{P}$
$\alpha^{-1},\beta^{-1},\gamma^{-1},\_\leftarrow I$
$(X_P,Y_P,Z_P,0)\leftarrow\mathcal{H}\circ\mathcal{S}(x_P,y_P,z_P,w_P)$
$(X_{\Phi(P)},Y_{\Phi(P)},Z_{\Phi(P)})\leftarrow X_P\cdot\alpha^{-1},\ Y_P\cdot\beta^{-1},\ Z_P\cdot\gamma^{-1}$
$(x_{\Phi(P)},y_{\Phi(P)},z_{\Phi(P)},w_{\Phi(P)})\leftarrow\mathcal{H}(X_{\Phi(P)},Y_{\Phi(P)},Z_{\Phi(P)},0)$
return $(x_{\Phi(P)}:y_{\Phi(P)}:z_{\Phi(P)}:w_{\Phi(P)})$

E.5 分裂坐标变换

现在我们转到链的最后转换。此时已计算了最后一个一般 $(2,2)$-同源 $\Phi:A\to B$。像曲面 B 同构于椭圆曲线的乘积，但仍使用固定 theta 结构表示，而非乘积 theta 坐标。因此，最后阶段应分两部分来看。

1. 首先，完全如前一小节计算最后一个一般像曲面，根据仍可用的挠点信息使用三种一般像曲面程序之一。

2. 其次，计算一个显式同构，将该像曲面的 theta 零点传送到 $E_1 \times E_2$ 的乘积 theta 结构中。这由 SplittingIsomorphism 处理，同样分为两部分。

   - SplittingIsomorphism 首先确定唯一指标对 (i,j) 使得 $U_{i,j}(0_A) = 0$，其中量 $U_{i,j}$ 是使用子算法 GetIndexSplitting 从 theta 零点构造的 $(2,2)$ 级 theta 表达式。显式地，

   $$U_{i,j}(0_A) = \sum_{t=0}^{3} \chi(i,t)\, 0_A[t]\, 0_A[j \oplus t],$$

   其中 $\oplus$ 表示按位 XOR，χ 是相应的符号特征。
   - 一旦找到了正确的对 (i,j)，SplittingIsomorphism 选择相应的预计算矩阵 $M \in \mathrm{Mat}_{4\times 4}$，其作用将当前 theta 结构传回乘积 theta 结构。

经此坐标变换之后，像曲面即以真正的乘积形式表示。此后，使用 ThetaToProduct 从变换后的 theta 零点恢复两个椭圆因子的 Montgomery 系数，并使用 ThetaProductPointToMontgomery 将求值点的 theta 乘积坐标转换回各因子上的 Montgomery 坐标。

Algorithm 68 GetIndexSplitting(0_A)

Input: theta 结构 $A \cong E_1 \times E_2$ 的 theta 零点 0_A
Output: 满足 $U_{i,j}(0_A) = 0$ 的指标对 (i,j)

inds $\leftarrow \{(0,0),(0,1),(0,2),(0,3),(1,0),(1,2),(2,0),(2,1),(3,0),(3,3)\}$
count $\leftarrow 0$
got_index $\leftarrow$ **false**
for k 从 1 到 #inds **do**
    $(i,j) \leftarrow$ inds$[k]$
    $U \leftarrow \sum_{t=0}^{3} \chi(i,t) \cdot 0_A[t] \cdot 0_A[j \oplus t]$
    if $U = 0$ **then**
        count $\leftarrow$ count $+ 1$
        ind $\leftarrow (i,j)$
        got_index $\leftarrow$ **true**
if count $\neq 1$ 或 **not** got_index **then**
    raise 异常: ("`get-index-splitting` 失败: 找到零个或多个零指标")
return ind

Algorithm 69 SplittingIsomorphism(0_A)

Require: $A \cong E_1 \times E_2$ 的 theta 零点 0_A
Ensure: 同构矩阵 M，其对 0_A 的作用返回与乘积 theta 结构关联的 theta 零点

1. $(i,j) \leftarrow$ GetIndexSplitting(0_A)

$(i,j) = (0,0)$:

2. $M \leftarrow \begin{pmatrix} 1 & \sqrt{-1} & 1 & \sqrt{-1} \\ 1 & -\sqrt{-1} & -1 & \sqrt{-1} \\ 1 & -\sqrt{-1} & -1 & -\sqrt{-1} \\ -1 & \sqrt{-1} & -1 & \sqrt{-1} \end{pmatrix}$

$(i,j) = (1,0)$:

3. $M \leftarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix}$

$(i,j) = (2,0)$:

4. $M \leftarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix}$

$(i,j) = (3,0)$:

5. $M \leftarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & 1 & 1 & -1 \end{pmatrix}$

$(i,j) = (0,1)$:

6. $M \leftarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}$

$(i,j) = (2,1)$:

7. $M \leftarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \end{pmatrix}$

$(i,j) = (0,2)$:

8. $M \leftarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}$

$(i,j) = (1,2)$:

9. $M \leftarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

$(i,j) = (0,3)$:

10. $M \leftarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$

$(i,j) = (3,3)$:

11. $M \leftarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

12. **return** M

Algorithm 70 ThetaToProduct(0_A)

Require: 带 theta 乘积结构的 PPAS A 的 theta 零点 $0_A = (a : b : c : d)$
Ensure: E_1 和 E_2 的 Montgomery 系数 $(A_1 : C_1)$ 和 $(A_2 : C_2)$，使得 $A \cong E_1 \times E_2$

1. $(a,b,c,d) \leftarrow 0_A$
2. $t_1 \leftarrow ad$
3. $t_2 \leftarrow bc$
4. **if** $t_1 \neq t_2$, **raise** ("`theta-to-product` 失败: 0_A 不来自乘积 theta 结构")
5. $x \leftarrow a^4$
6. $y \leftarrow b^4$
7. $A_2 \leftarrow x + y$
8. $C_2 \leftarrow x - y$
9. $A_2 \leftarrow -2A_2$
10. $z \leftarrow c^4$
11. $A_1 \leftarrow x + z$
12. $C_1 \leftarrow x - z$
13. $A_1 \leftarrow -2A_1$
14. **if** $C_1 = 0$ 或 $C_2 = 0$, **raise** ("`theta-to-product` 失败")
15. **return** $(A_1 : C_1), (A_2 : C_2)$

Algorithm 71 ThetaProductPointToMontgomery$(P, 0_A)$

Input: 带乘积结构的 PPAS A 的 theta 点 $P = (x : y : z : w)$ 和 theta 零点 $0_A = (a : b : c : d)$
Output: $P = (P_1, P_2) \in E_1 \times E_2$ 的 Montgomery 坐标 $(X(P_1) : Z(P_1))$ 和 $(X(P_2) : Z(P_2))$

1: $(a,b,c,d) \leftarrow 0_A$
2: $(x,y,z,w) \leftarrow P$
3: $X_1 \leftarrow a \cdot z + c \cdot x$
4: $Z_1 \leftarrow a \cdot z - c \cdot x$
5: $X_2 \leftarrow a \cdot y + b \cdot x$
6: $Z_2 \leftarrow a \cdot y - b \cdot x$
7: **return** $(X_1 : Z_1), (X_2 : Z_2)$

E.6 计算椭圆曲线乘积之间的 $(2,2)$-同源链

现在可以描述完整的 $(2,2)$-同源链计算

$$\Phi = \Phi_e \circ \cdots \circ \Phi_1 : E_1 \times E_2 \longrightarrow E_3 \times E_4.$$

计算 Isogeny22Chain 具有清晰的三阶段结构：:

1. 从乘积曲面到主极化阿贝尔曲面 theta 坐标模型的粘合步，
2. 完全在该模型内的一系列一般 $(2,2)$-同源，
3. 返回到椭圆曲线乘积的最终分裂步。

我们按照此结构来组织算法陈述。除主程序 Isogeny22Chain 外，首先给出一个辅助程序 ChainStrategyComputation，然后给出三个阶段子程序：GluingStep、GenericStep 和 SplittingStep。

在本小节的其余部分，我们沿用实现中采用的富挠点输入约定：该程序接受 $E_1 \times E_2$ 中阶为 2^{e+2} 的两个点 P, Q 作为输入。这意味着输入比实际核多出两层 2-幂挠点，从而为链计算提供相容的 8-挠点。

以下子程序的使用方式如下。

1. 辅助程序 ChainStrategyComputation 管理策略栈。它重复应用当前倍点程序，直到栈顶提供下一次陪域曲面计算所需的 8-挠点输入为止。
2. GluingStep 执行初始粘合步。它使用 ChainStrategyComputation 先下降到初始核上方的相容 8-挠点，通过 GluingCodomain 计算粘合陪域曲面，再通过 GluingEval 或 GluingEvalSpecial 将所有待处理点变换到 theta 坐标。
3. 得到第一个陪域曲面之后，剩余链完全在 theta 坐标中计算。GenericStep 在 theta 坐标中计算所有中间的一般 $(2,2)$-同源。在每个中间层，它使用 ChainStrategyComputation 中的 ThetaDBL，通过 GenericCodomainWith8Torsion 恢复下一个陪域曲面，并通过 GenericEval 对所有待处理点求值。
4. 最后，SplittingStep 应用 SplittingIsomorphism，将陪域曲面变换回乘积 theta 坐标。恢复两个 Montgomery 因子，并将所有求值点转换回 Montgomery 坐标。

Algorithm 72 ChainStrategyComputation(`strat_pts`, `orders`)

Input: 相同长度的列表 `strat_pts` 和 `orders`，其最后条目表示当前栈顶
Output: 更新后的列表 `strat_pts` 和 `orders`，其最后一对点为 8-挠点

while `orders`$[k] \neq 1$ **do**
  $k \leftarrow k + 1$
  if `orders`$[k-1] \geq 16$ **then**
    $n \leftarrow \lfloor$`orders`$[k-1]/2 \rfloor$
  else
    $n \leftarrow$ `orders`$[k-1] - 1$
  $(R, S) \leftarrow$ `strat_pts`$[k-1]$
  for j 从 1 到 n **do**
    $(R, S) \leftarrow \mathrm{DBL}(R, S)$
  `strat_pts`$[k] \leftarrow (R, S)$
  `orders`$[k] \leftarrow$ `orders`$[k-1] - n$
return `strat_pts`, `orders`

Algorithm 73 GluingStep(P, Q, e, pts)

Input: $E_1 \times E_2$ 中阶为 2^{e+2} 的点 P, Q，以及形如 $(R_1, 0)$ 或 $(0, R_2)$ 的点数组 pts
Output: 第一个陪域曲面 theta 零点 0_A、倍点常数、传递的求值点、以及剩余的策略状态

strat_pts $\leftarrow [(P, Q)]$
orders $\leftarrow [e]$
$k \leftarrow 0$
(strat_pts, orders) $\leftarrow$ ChainStrategyComputation(strat_pts, orders, k)
$(T_1'', T_2'') \leftarrow$ strat_pts$[k]$
$(\_, I, 0_A, J, N) \leftarrow$ GluingCodomain(T_1'', T_2'')
pts $\leftarrow [\,$GluingEvalSpecial$(R, I, N) \mid R \in$ pts$\,]$
for i 从 0 到 $k-1$ **do**
    strat_pts$[i] \leftarrow$ GluingEval(strat_pts$[i], T_1'', J, N$)
    orders$[i] \leftarrow$ orders$[i] - 1$
$k \leftarrow k - 1$
consts $\leftarrow$ ThetaPrecomp(0_A)
return 0_A, consts, pts, strat_pts, orders

Algorithm 74 GenericStep$(0_A, \text{consts}, \text{pts}, \text{strat_pts}, \text{orders})$

Input: 当前 theta 零点 0_A、倍点常数 consts、需要求值的点 pts、以及核栈 strat_pts, orders
Output: 最终 theta 零点 0_A 和求值点 pts

while $k \geq 0$ 且 orders$[k] \neq 0$ **do**
    (strat_pts, orders) $\leftarrow$ ChainStrategyComputation(strat_pts, orders) ▷ 使用 DBL = (ThetaDBL(_, consts)
    $(T_1'', T_2'') \leftarrow$ strat_pts$[k]$
    $(0_B', I, 0_B) \leftarrow$ GenericCodomainWith8Torsion(T_1'', T_2'')
    pts $\leftarrow [\,$GenericEval$(R, I) \mid R \in$ pts$\,]$
    consts $\leftarrow$ ThetaPrecomp(0_B)
    for i 从 0 到 $k-1$ **do**
        strat_pts$[i] \leftarrow$ GenericEval(strat_pts$[i], I$)
        orders$[i] \leftarrow$ orders$[i] - 1$
    $k \leftarrow k - 1$
return 0_B, pts

Algorithm 75 SplittingStep$(0_A, \text{pts})$

Input: 同构于乘积的曲面的 theta 零点 0_A，以及求值点 pts
Output: 像乘积 $E_3 \times E_4$ 以及两个因子上 Montgomery 坐标中的求值点

$M \leftarrow$ SplittingIsomorphism(0_A)
$0_A \leftarrow M \cdot 0_A$
pts $\leftarrow [\, M \cdot R \mid R \in$ pts$\,]$
$(A_3 : C_3), (A_4 : C_4) \leftarrow$ ThetaToProduct(0_A)
pts $\leftarrow [\,$ThetaProductPointToMontgomery$(R, 0_A) \mid R \in$ pts$\,]$
令 E_3, E_4 为由 Montgomery 系数 $(A_3 : C_3)$ 和 $(A_4 : C_4)$ 定义的椭圆曲线
return $E_3 \times E_4$, pts

介绍了辅助程序和三个阶段子程序之后，现在将它们组合成主程序 Isogeny22Chain。该主算法遵循上述三阶段结构：依次调用 GluingStep、GenericStep 和 SplittingStep。

Algorithm 76 IsogenyChain(P, Q, pts)

Input: $E_1 \times E_2$ 中阶为 2^{e+2} 的点 P, Q，以及形如 $(R_1, 0)$ 或 $(0, R_2)$ 的点数组 pts
Output: 链的陪域曲面 $E_3 \times E_4$，其中 $\ker(\Phi) = \langle [4]P, [4]Q \rangle$，以及求值点 $[\Phi(R) \mid R \in \text{pts}]$

1: $(0_A, \text{consts}, \text{pts}, \text{strat\_pts}, \text{orders}) \leftarrow$ GluingStep(P, Q, e, pts)
2: $(0_A, \text{pts}) \leftarrow$ GenericStep$(0_A, \text{consts}, \text{pts}, \text{strat_pts}, \text{orders})$
3: $(E_3 \times E_4, \text{pts}) \leftarrow$ SplittingStep$(0_A, \text{pts})$
4: **return** $E_3 \times E_4, \text{pts}$

CHAPTER F

配对计算（实现细节）*

本节描述计算配对所用的立方算术。本文内容基于 Pope、Reijnders、Robert、Sferlazza 和 Smith 的立方算术 [PRR+25]。立方算术与差分算术略有不同。以 $\widetilde{P}$（带波浪线）表示 $P \in E$ 在此算术下的表示，称为立方点。从立方点

$$\widetilde{P} = (x(P) : z(P)), \quad \widetilde{Q} = (x(Q) : z(Q)), \quad \widetilde{P+Q} = (x(P+Q) : z(P+Q)), \quad \widetilde{0} = (1:0)$$

出发，一个立方阶梯产生 $[n]P$ 和 $[n]P+Q$ 的立方点，称为这些点的仿射提升。当 $[n]P$ 为无穷远点或 2 挠点时，仿射提升 $\widetilde{[n]P}$ 和 $\widetilde{[n]P+Q}$ 与起始点 $\widetilde{0}$ 和 $\widetilde{Q}$ 相差标量因子 λ, λ'。立方配对的核心在于：比值 λ/λ' 已足以计算 Tate 或 Weil 配对。因此不必通过 Miller 函数求值——从阶梯输出携带的射影缩放信息即可计算这些配对。[1]

F.1 立方算术

立方配对计算使用与 Montgomery 阶梯相同的 x-only 原语。但立方加法与差分加法略有不同，以正确跟踪仿射缩放。我们使用四个基本程序。

- CubicalDbl计算立方点 $\widetilde{P}$ 的立方倍点 $\widetilde{2P}$，
- CubicalDiffAdd给定 $\widetilde{P}$、$\widetilde{Q}$ 和 $\widetilde{P-Q}$，计算 $\widetilde{P+Q}$，
- CubicalTranslate用于立方阶梯中最终倍点的平移，
- CubicalRatio恢复同一点 $P \in E$ 的两个仿射提升之间的比值 λ。

Algorithm 77 CubicalDbl$(E, \widetilde{P})$

Input: Montgomery 曲线 $E : y^2 = x^3 + Ax^2 + x$，立方点 $\widetilde{P} = (X(P) : Z(P))$
Output: 立方倍点 $2\widetilde{P} = (X_2 : Z_2)$

1: $a \leftarrow (X(P) + Z(P))^2$
2: $b \leftarrow (X(P) - Z(P))^2$
3: $c \leftarrow a - b$
4: $X_2 \leftarrow a \cdot b$
5: $Z_2 \leftarrow c \cdot \left(b + \frac{A+2}{4} \cdot c\right)$
6: **return** (X_2, Z_2)

[1] 此处省略了立方差分加法输出中除以 4 的显式操作。对于 $\mathbb{F}_{p^2}$ 上的约化 Tate 配对，最终求幂会移除这些因子。对于 Weil 配对，它们可计算为两个平方 Tate 配对的比值。

Algorithm 78 CubicalDiffAdd$(E, \widetilde{P}, \widetilde{Q}, x(P-Q))$

Input: Montgomery 曲线 $E : y^2 = x^3 + Ax^2 + x$；立方点 $\widetilde{P} = (X(P) : Z(P))$，$\widetilde{Q} = (X(Q) : Z(Q))$ 及其差的 x 坐标 $x(P-Q)$

Output: 立方差分加法 $\widetilde{P+Q} = (X_2, Z_2)$

1: $a \leftarrow X(P) + Z(P)$
2: $b \leftarrow X(P) - Z(P)$
3: $c \leftarrow X(Q) + Z(Q)$
4: $d \leftarrow X(Q) - Z(Q)$
5: $X_2 \leftarrow (a \cdot d + b \cdot c)^2$
6: $Z_2 \leftarrow (a \cdot d - b \cdot c)^2$
7: $X_2 \leftarrow X_2 / x(P-Q)$
8: **return** $(X_2 : Z_2)$

Algorithm 79 CubicalTranslate$(\widetilde{P}, \widetilde{T})$

Input: 立方 Montgomery 坐标 $\widetilde{P} = (X(P) : Z(P))$ 和 $\widetilde{T} = (X(T) : Z(T))$，其中 $T = (X(T) : Z(T))$ 为 2 挠点

Output: 立方平移 $\widetilde{P+T} = (X(P+T), Z(P+T))$

1: $X \leftarrow X(T)X(P) - Z(T)Z(P)$
2: $Z \leftarrow Z(T)X(P) - X(T)Z(P)$
3: **if** $Z(T) = 0$ **then**
4: $\quad Z \leftarrow -Z$
5: **if** $X(T) = 0$ **then**
6: $\quad X \leftarrow -X$
7: **return** (X, Z)

Algorithm 80 CubicalRatio$(\widetilde{P_1}, \widetilde{P_2})$

Input: 立方 Montgomery 坐标 $\widetilde{P_1} = (X(P_1) : Z(P_1))$, $\widetilde{P_2} = (X(P_2) : Z(P_2))$ 满足 $P_1 = (X(P_1) : Z(P_1)) = P_2 = (X(P_2) : Z(P_2))$

Output: 比值 λ 使得 $X(P_2) = \lambda X(P_1)$ 且 $Z(P_2) = \lambda Z(P_1)$

1: **if** $X(P_1) = 0$ **then**
2: $\quad$ **return** $Z(P_2)/Z(P_1)$
3: **else**
4: $\quad$ **return** $X(P_2)/X(P_1)$

F.2 偶数次配对

当配对阶 $N = 2n$ 为偶数时，配对的平方会丢失一比特信息。此时将阶 N 的阶梯替换为阶 n 的阶梯，再施加由 2 挠点 $[n]P$ 给出的仿射平移。本技术规范仅考虑 $N = 2^e$ 的情形。

Algorithm 81 CubicalLadderPowertwo$(E, e, \widetilde{P+Q}, \widetilde{P}, x(Q))$

Input: Montgomery 曲线 $E : y^2 = x^3 + Ax^2 + x$；整数 e；立方点 $\widetilde{P+Q} = (X(P+Q) : Z(P+Q))$，$\widetilde{P} = (X(P) : Z(P))$；$Q$ 的 x 坐标
Output: 立方点 $[2^e]\widetilde{P}$ 和 $[2^e]\widetilde{P} + \widetilde{Q}$

$n_{PQ} \leftarrow \widetilde{P+Q}$
$n_P \leftarrow \widetilde{P}$
for $k \leftarrow 1$ **to** e **do**
    $n_{PQ} \leftarrow$ CubicalDiffAdd$(E, n_{PQ}, n_P, x(Q))$
    $n_P \leftarrow$ CubicalDbl(E, n_P)
return (n_P, n_{PQ})

Algorithm 82 Tate$(E, e, x(P), x(Q), x(P+Q))$

Input: 超奇异 Montgomery 曲线 $E : y^2 = x^3 + Ax^2 + x$；整数 e；点 $P, Q, P+Q \in E(\mathbb{F}_{p^2})$ 的 x 坐标，且 $2^e P = 0_E$
Output: 约化 Tate 配对 $t_{2^e}(P, Q) \in \mu_{2^e}$

$(n_P, n_{PQ}) \leftarrow$ CubicalLadderPowertwo$(E, e-1, (x(P+Q), 1), (x(P), 1), x(Q))$
$\widetilde{O} \leftarrow$ CubicalTranslate(n_P, n_P)
$\widetilde{Q'} \leftarrow$ CubicalTranslate(n_{PQ}, n_P)
$\lambda \leftarrow$ CubicalRatio$((x(Q) : 1), \widetilde{Q'})$ / CubicalRatio$((1 : 0), \widetilde{O})$
return $\lambda^{(p^2-1)/2^e}$

Algorithm 83 Dlog2Single$(E, e, x(P), x(Q), x(R), x(P-Q), x(P-R), x(R-Q))$

Input: 超奇异 Montgomery 曲线 $E/\mathbb{F}_{p^2}$；整数 e；$P, Q, R \in E[2^e]$ 的 x 坐标；$x(P-Q)$, $x(P-R)$, $x(R-Q)$
Output: 标量 $a, b \in \mathbb{Z}/2^e\mathbb{Z}$ 满足 $R = [a]P + [b]Q$

$S_P \leftarrow (x(P) : 1)$, $S_Q \leftarrow (x(Q) : 1)$, $S_R \leftarrow (x(R) : 1)$
$(U_1, V_1, D_1) \leftarrow (P, Q, x(P-Q))$
$(U_2, V_2, D_2) \leftarrow (P, R, x(P-R))$
$(U_3, V_3, D_3) \leftarrow (R, Q, x(R-Q))$
for $j \leftarrow 1$ **to** 3 **do**
    $L_j \leftarrow (D_j, 1)$
    $M_j \leftarrow (D_j, 1)$
for $i \leftarrow 1$ **to** $e-1$ **do**
    for $j \leftarrow 1$ **to** 3 **do**
        $L_j \leftarrow \text{CubicalDiffAdd}(E, L_j, S_{U_j}, x(V_j))$
        $M_j \leftarrow \text{CubicalDiffAdd}(E, M_j, S_{V_j}, x(U_j))$
    $S_P \leftarrow \text{CubicalDbl}(E, S_P)$
    $S_Q \leftarrow \text{CubicalDbl}(E, S_Q)$
    $S_R \leftarrow \text{CubicalDbl}(E, S_R)$
$S'_P \leftarrow \text{CubicalTranslate}(S_P, S_P)$
$S'_Q \leftarrow \text{CubicalTranslate}(S_Q, S_Q)$
$S'_R \leftarrow \text{CubicalTranslate}(S_R, S_R)$
for $j \leftarrow 1$ **to** 3 **do**
    $L_j \leftarrow \text{CubicalTranslate}(L_j, S_{U_j})$
    $M_j \leftarrow \text{CubicalTranslate}(M_j, S_{V_j})$
    $\theta_j \leftarrow \text{CubicalRatio}(M_j, S'_{V_j}) / \text{CubicalRatio}(L_j, S'_{U_j})$
$\eta_b \leftarrow \theta_2$
$\eta_a \leftarrow \theta_3$
return (ω, η_b, η_a)

F.3 奇数次配对

设 $\ell \geq 3$ 为奇数，且 $\ell \nmid p$，$P \in E[\ell]$，$Q \in E$。此时立方算术直接给出非约化 Tate 配对的平方。具体而言：长度为 ℓ 的立方阶梯产生 $[\ell]P$ 和 $[\ell]P + Q$ 的仿射提升，其单值比（monodromy ratio）在相差 ℓ 次幂的意义下给出 $e_{T,\ell}(P, Q)^2$。

由于 2 模 ℓ 可逆，这对约化 Tate 配对不造成任何困难。最终求幂后，额外取 $2^{-1} \bmod \ell$ 次幂即可恢复配对本身。类似地，Weil 配对的平方由两个 Tate 配对的商得到。

Algorithm 84 CubicalLadder$(E, n, x(P), x(Q), x(P-Q))$

Input: Montgomery 曲线 $E : y^2 = x^3 + Ax^2 + x$；整数 $n = \sum_{i=0}^{b-1} n_i 2^i$ 且 $n_b = 0$；规范化输入 $(x(P) : 1)$，$(x(Q) : 1)$，$(x(P-Q) : 1)$

Output: 立方点 $\widetilde{[n]P}$ 和 $\widetilde{[n]P + Q}$

$S_0 \leftarrow (1, 0)$
$S_1 \leftarrow (x(P) : 1)$
$T \leftarrow (x(Q) : 1)$
$d_0 \leftarrow x(Q)$
$d_1 \leftarrow x(P-Q)$
for $i \leftarrow b-1$ **downto** 0 **do**
    $R \leftarrow$ CubicalDiffAdd$(E, S_0, S_1, x(P))$
    if $n_i \oplus n_{i+1} = 1$ **then**
        交换 S_0, S_1
        交换 d_0, d_1
    $T \leftarrow$ CubicalDiffAdd(E, T, S_0, d_0)
    $S_0 \leftarrow$ CubicalDbl(E, S_0)
    $S_1 \leftarrow R$
return (S_0, T)

Algorithm 85 TateOdd$(E, N, x(P), x(Q), x(P-Q))$

Input: 超奇异 Montgomery 曲线 $E/\mathbb{F}_{p^2}$；奇数 N 满足 $N \nmid p$；$P, Q, P-Q$ 的 x 坐标，其中 $P \in E[N]$

Output: 约化 Tate 配对 $t_N(P, Q) \in \mu_N$

$(\widetilde{O}, \widetilde{Q'}) \leftarrow$ CubicalLadder$(E, N, x(P), x(Q), x(P-Q))$
$s \leftarrow$ CubicalRatio$((x(Q) : 1), \widetilde{Q'})$ / CubicalRatio$((1 : 0), \widetilde{O})$
$\lambda \leftarrow s^{(p^2-1)/\ell}$
return λ

Algorithm 86 DlogOddSingle$(E, C, x(P), x(Q), x(P-Q), R)$

Input: 超奇异 Montgomery 曲线 $E/\mathbb{F}_{p^2}$；$P, Q, P-Q, R \in E[N]$ 的 x 坐标

Output: 标量 $(a, b) \in (\mathbb{Z}/N\mathbb{Z})^2$ 满足 $R = [a]P + [b]Q$

$\omega \leftarrow$ TateOdd$(E, N, x(P), x(Q), x(P-Q))$
$\eta_b \leftarrow$ TateOdd$(E, N, x(P), x(R), x(P-R))$
$\eta_a \leftarrow$ TateOdd$(E, N, x(R), x(Q), x(R-Q))$
$a \leftarrow \log_\omega \eta_a$
$b \leftarrow \log_\omega \eta_b$
if $[a]P + [b]Q \neq R$ **then**
    $a \leftarrow N - a$
return (a, b)

Algorithm 87 DlogOdd$(E, N, x(P), x(Q), x(P-Q), x(P_1), x(P_2))$

Input: 超奇异 Montgomery 曲线 $E/\mathbb{F}_{p^2}$；$P, Q, P-Q, P_1, P_2 \in E[N]$ 的 x 坐标
Output: 标量 $(a_1, b_1, a_2, b_2) \in (\mathbb{Z}/N\mathbb{Z})^4$ 满足 $P_1 = [a_1]P + [b_1]Q$ 和 $P_2 = [a_2]P + [b_2]Q$

$\omega \leftarrow$ TateOdd$(E, N, x(P), x(Q), x(P-Q))$
$\eta_{b_1} \leftarrow$ TateOdd$(E, N, x(P), x(P_1), x(P-P_1))$
$\eta_{a_1} \leftarrow$ TateOdd$(E, N, x(P_1), x(Q), x(P_1-Q))$
$\eta_{b_2} \leftarrow$ TateOdd$(E, N, x(P), x(P_2), x(P-P_2))$
$\eta_{a_2} \leftarrow$ TateOdd$(E, N, x(P_2), x(Q), x(P_2-Q))$
$a_1 \leftarrow \log_\omega \eta_{a_1}$
$a_2 \leftarrow \log_\omega \eta_{a_2}$
$b_1 \leftarrow \log_\omega \eta_{b_1}$
$b_2 \leftarrow \log_\omega \eta_{b_2}$
if $[a_1]P + [b_1]Q \neq P_1$ **then**
    $a_1 \leftarrow N - a_1$
if $[a_2]P + [b_2]Q \neq P_2$ **then**
    $a_2 \leftarrow N - a_2$
return (a_1, b_1, a_2, b_2)