QIMEN-PRISM

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 数字签名方案 QIMEN-PRISM。

1.1 设计理由

QIMEN-PRISM 的核心设计源于数字签名方案 salt-PRISM [BBC+26]（该论文是 PKC'25 的最佳论文）。它遵循相同的数学困难假设和协议体系。

数学问题： QIMEN-PRISM 的安全性依赖于椭圆曲线算术中的一个经典计算问题：给定一条超奇异椭圆曲线 E 和一个指定的大素数 q，在不知道其自同态环的情况下，计算一个从 E 出发的次数为 q 的同源。自 1971 年 Vélu 公式 [Vél71] 以来，同源的显式计算一直是被广泛研究的课题，后续大量工作涉及挠方法、模多项式、同源图和自同态环 [Cou96, Elk98, Koh96, LM00, BCL08, DG16]。那些次数分解为小素数的同源可以通过低次映射的链式复合来计算，但这种方法不适用于大素数次 q。经过五十多年的算法发展，最好的通用算法仍需 $\widetilde{O}(\sqrt{q})$ 量级的有限域操作 [BDLS20]，相对于 q 的比特长度呈指数级复杂度。

核心设计思路： QIMEN-PRISM 将上述计算差异转化为密码构造的核心思想是：对于大素数度同源，是否已知起始曲线的自同态环会造成显著的计算复杂度差异。给定超奇异椭圆曲线 E 的自同态环，可以高效构造从 E 出发的大素数度同源。因此，方案以曲线 E 作为公钥，并以其自同态环作为私钥。

底层身份鉴别协议的结构简单。验证者随机选取一个大素数 q 作为挑战，证明者利用秘密自同态环，从 E 出发构造一条次数为 q 的同源。由于验证者不知道该自同态环，因

此无法直接高效地验证这条同源。为解决这一问题，证明者利用 Kani 引理的算法化实现，将该素数度同源嵌入一条定义在椭圆曲线乘积之间的二维同源中。该二维同源的次数为光滑的 $(2^a, 2^a)$，其核构成证明者的应答。验证者仅利用公开曲线和挑战 q，便可通过 theta 坐标算术重构相应的光滑度同源链，并借助配对检验其同源的素数次数的真实性。

哈希-签名范式： 遵循哈希-签名范式，通过构造哈希到素数函数，身份鉴别协议转化为一个简单的签名方案。签名者将消息本地哈希为素数，即可非交互式地生成证明作为消息的签名。

参数选择与失败分析： QIMEN-PRISM 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 salt-PRISM 的改进

基于 salt-PRISM [BBC+26] 的核心设计，QIMEN-PRISM 引入了若干理论和工程创新，与先前的学术原型在多个面向上形成区分。其中的差异总结如下。

1. **更快的核心子过程：** QIMEN-PRISM 在核心子过程中引入的额外机制，结合精炼后的参数选择，使得更多的合适的方程解可以在连续阶段之间循环复用，包括 Cornacchia 计算和范数方程求解（详见例如 Remarks 3.2.1, 3.2.2 and 3.3.1）。这样避免了对独立的大素数搜索与重启，减少了 salt-PRISM 中的拒绝循环和冗余素性测试，从而在不损害安全性的前提下大幅提升整体性能。综上所述，这些改进使密钥生成和验证时间减少了约 6–9.5%，签名时间减少了约 27–40%，具体数值取决于安全级别。

2. **快速签名压缩：** QIMEN-PRISM 复用了同源求值过程中已经计算出的几何值和基变换矩阵。利用这些中间值，接收者可以重建部分签名数据而无须显式传输椭圆曲线点，从而将签名大小缩减约 33% 至 41%，仅引入轻微开销，具体数值取决于安全级别。压缩只增加约 2% 的签名成本和 15% 的验证成本。

3. **可证明安全的素性测试：** 哈希到素数过程对每个安全级别使用固定次数的 Miller–Rabin 迭代。迭代次数经过选择，使得接受合数整数的概率不超过 $2^{-\lambda_c}$，从而防止 [CGMT26] 中描述的哈希到伪素数伪造攻击。

4. **汇编级域算术内核：** 优化实现提供了使用 BMI2 和 ADX 指令的 x86-64 汇编过程。这些过程降低了模算术的成本。与可移植 C 实现相比，在 NGCC-1 和 NGCC-2 安全级别下，密钥生成运行时间分别约缩减 14% 和 30%，签名约缩减 15% 和 20%，验证约缩减 40%。

1.2 特性陈述

1.2.1 创新性

现有基于同源的数字签名方案大多遵循 sigma 协议与 Fiat–Shamir 范式，如 [DKL+20, DLLW23, DLRW24, BDD+24, NOC+24] 所示，它们均使用源于 [DKL+20] 的 SQIsign 框架。在这些构造中，[BDD+24] 是 NIST 标准化提案 SQIsign [AAA+25] 的主要基础。SQIsign 是目前唯一进入国际标准化评估流程的基于同源的数字签名方案，已推进至 NIST 附加数字签名流程的第三轮。

与之不同，salt-PRISM 以及由此而来的 QIMEN-PRISM 遵循独特的哈希-签名体系。它们的主要创新在于该范式下引入的特定签名关系和公开验证机制。下文统称 QIMEN-PRISM。注意以下协议层面的创新源自 salt-PRISM。

- **每轮新生成的素数的次与签名：** QIMEN-PRISM 是首个以随机大素数作为挑战的同源签名方案。具体而言，对于每条消息及其盐值（salt），方案以确定性方式派生一个新的大素数 q，并将其作为签名中底层一维同源的次数。因此，每个签名都对应一个独立生成的素数次同源实例。
- **随机化次数签名的 Kani 配对验证：** QIMEN-PRISM 是首个将 Kani 引理的算法化实现与配对检验相结合的同源签名方案，用于验证由大素数度同源生成的签名。该大素数在每次签名时均重新随机生成，并作为底层签名同源的次数。Kani 引理将这一素数度同源嵌入到椭圆曲线乘积之间一条双次数为 $(2^a, 2^a)$ 的光滑度同源中，而配对检验则用于确认重构得到的二维同源确实包含指定的素数次同源分量。

综合而言，这些特性使 QIMEN-PRISM 形成了一套独立的同源签名体系，而非对正在接受国际标准化评估的 SQIsign 框架的简单修改或参数调整。

1.2.2 简洁性

QIMEN-PRISM 的一个主要优势在于其设计更为简洁，这一点在与当前最先进的同源签名方案 SQIsign [AAA+25] 比较时尤为明显。SQIsign 建立在一个包含承诺、挑战和应答三个阶段的交互式身份鉴别协议之上，并通过 Fiat–Shamir 变换得到签名方案。这种结构增加了签名流程的复杂性，也使安全性分析和方案扩展更为繁琐。相比之下，QIMEN-PRISM 无需承诺阶段，而是直接采用陷门单向函数，并以哈希后签名的方式构造签名。

SQIsign 及其高维变体的安全证明依赖专门设计的强模拟预言机。为了证明诚实验证者零知识性质，这些预言机需要在不知道私钥的情况下模拟有效的协议交互记录。相比之下，QIMEN-PRISM 的安全性可以直接归约到大素数度同源的计算困难性，并在标准模型或标准量子随机预言机模型下给出较为直接的归约，无需引入复杂且特殊的模拟预言机。

1.2.3 灵活性与兼容性

QIMEN-PRISM 的极简和模块化设计为多样化工程环境提供了灵活性，并与现代密码增强变换兼容：

- **可扩展的哈希/XOF 后端：** 协议将高层哈希到素数过程与底层的可扩展输出函数（XOF）分离。对于标准的全球部署，原生支持 SHAKE256 实例化。对于符合国家商用密码标准的场合，它无缝集成了由 SM3 密码哈希算法驱动的 pseudoXOF 构造（遵循 GB/T 32905-2016）。这使得同一套协议机制可以通过简单的编译开关为不同的监管域编译。
- **BUFF 变换兼容性：** QIMEN-PRISM 完全兼容 [CDF+21, DFH+24, FHK25] 中提出的 BUFF 变换，以签名中 $2\lambda_c$ 比特的开销和一次哈希计算为代价达到安全概念。核心方案可以增强以实现高级安全属性，包括排他所有权、消息绑定签名和不可重签名性。
- **灵活的点压缩模式：** 签名同源的表示可以根据目标平台的带宽和计算特征进行定制。它支持以坐标为中心的模式（适合最小化验证开销，仅需一次域求逆），以及基系数模式，通过提供基生成提示将签名大小压缩至 $6\lambda_c + 4a$ 比特。

1.2.4 可扩展性

与传统的同源签名方案相比，QIMEN-PRISM 最突出的优势之一是其良好的可扩展性。长期以来，SQIsign 较为复杂的协议结构限制了其向高级多方协议和结构化密码原语的扩展。尽管经过多年研究，目前基于 SQIsign 构建的复杂密码原语仍然较为有限。相比之下，QIMEN-PRISM 采用直接的哈希后签名结构，可以较为自然地扩展到多种高级应用：

- **支持密钥随机化的签名：** 如 [BBS+26] 所示，QIMEN-PRISM 可以扩展为支持密钥随机化和消息适配。特别是，后者难以在 SQIsign 中实现。该扩展即使面对计算能力不受限制的敌手，也能够保证不可链接性，同时仍可在与 QIMEN-PRISM 相同的安全假设下证明不可伪造性。
- **良构性证明：** QIMEN-PRISM 支持对公钥和签名进行高效的良构性证明 [LM26]。借助此类证明，证明者可以在不泄露底层秘密信息的情况下，证明给定的公钥或签名满足协议规定的结构。这一性质有利于将 QIMEN-PRISM 作为基础构件集成到更高级的密码协议中，因为在这些协议中，恶意构造的公钥或签名可能破坏安全性分析。
- **门限签名：** 更为重要的是，QIMEN-PRISM 适合构建 T-out-of-N 门限签名协议 [Thr]，而 SQIsign 目前尚未实现这一功能。此外，[Thr] 给出了首个不依赖群作用的同源门限签名方案，因而不受针对群作用的次指数级量子攻击影响。与现有同类方案相比，该方案的联合公钥和签名尺寸以及通信开销均显著更小。

1.2.5 性能

- **紧凑性：** QIMEN-PRISM 具有较小的公钥和签名尺寸。在相同安全级别下，与 Falcon 和已标准化的 ML-DSA [PFH+22, LDK+22] 相比，QIMEN-PRISM 的公钥和签名尺寸分别仅为 Falcon 的 1/13.9 和 1/3.8，以及 ML-DSA 的 1/20.1 和 1/13.9（见下方 Table 1.1）。因此，QIMEN-PRISM 尤其适用于证书链传输等带宽受限场景。
- **合理的签名时间：** 虽然慢于 ML-DSA 和 Falcon，QIMEN-PRISM 效率水平合理，在大多数实际应用中切实可行。例如，在 NGCC-1 安全级别下，优化后的 64 位实

Table 1.1: NIST 安全级别 5（与使用 NGCC-2 的 QIMEN-PRISM 相同）下，NIST 已标准化或将标准化的后量子签名方案的公钥和签名大小对比，单位为字节。

方案	公钥	签名
QIMEN-PRISM (NGCC-2)	129 **B**	334 **B**
FN-DSA-1024 (Falcon)	1793 B (13.9×)	1280 B (3.8×)
ML-DSA-87 (Dilithium)	2592 B (20.1×)	4627 B (13.9×)
SLH-DSA-SHAKE-256s (SPHINCS$^+$)	64 B (0.5×)	29 792 B (89.2×)

现在 2.7 GHz 的 Intel Ultra 9 CPU 上，签名约需 24.5 ms，验证约需 3.6 ms（见 Table 7.4）。

- **快速验证：** QIMEN-PRISM 的验证速度显著快于签名速度，因此在需要高效验证的场景中具有明显优势。这使得 QIMEN-PRISM 特别适用于签名可离线生成、但需要进行大规模验证的应用，例如证书链验证、代码更新以及透明度机制中的签名验证。
- **与优化版 SQIsign 的比较：** 相比 SQIsign，QIMEN-PRISM 的设计更为简洁：签名过程中所需的二维同源计算更少，并且完全避免了一维同源映射的求值（evaluation）。进一步地，与采用 Qlapoti [BCRSE+26] 的官方 NIST 第二轮 SQIsign 提交版本的优化实现[1] 进行比较时，QIMEN-PRISM 的签名速度接近提升了两倍。

Table 1.2: NIST 安全类别 5 下参考实现基于同源的签名性能对比（在 Intel Ultra 9 CPU（2.7GHz）上，与 QIMEN-PRISM 相同的安全级别），单位为 10^6 CPU 时钟周期。

方案	KeyGen	Sign	Verify
QIMEN-PRISM (NGCC-2)	130.24	142.46	41.72
SQIsign+[BCRSE+26] (NIST-5)	129.94 (1.0x)	332.74 (2.3x)	35.87 (0.9x)

1.2.6 假设的多样性

QIMEN-PRISM 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-PRISM 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。

[1] https://github.com/KULeuven-COSIC/Qlapoti/tree/main/C-implementation

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-PRISM 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 1.3: QIMEN-PRISM 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 1.4: QIMEN-PRISM 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
	域参数
p	域特征，在参数集中取为 $p = f \cdot 2^e - 1$
e	满足 $2^e \mid p+1$ 的最大整数；决定了可用的 2 幂挠
f	$p = f \cdot 2^e - 1$ 中的奇余因子
a	满足 $0 < a < e$ 的整数参数；其具体值由参数集确定
	公钥和私钥对象
E_0	起始曲线，方程为 $y^2 = x^3 + x$
(P_0, Q_0)	$E_0[2^{a+2}]$ 的一组基
N_{sk}	秘密同源的素数次，取为大于 $2^{4\lambda_c}$ 的最小素数
I_{sk}	范数为 N_{sk} 的秘密理想
ϕ_{sk}	对应 I_{sk} 的秘密同源 $E_0 \to E_{\mathsf{pk}}$
E_{pk}	公钥曲线，即 ϕ_{sk} 的像曲线
P_{pk} 与 Q_{pk}	由 $\mathsf{hint}_{\mathsf{pk}}$ 恢复的 $E_{\mathsf{pk}}(\mathbb{F}_{p^2})[2^{a+2}]$ 的生成元
$\mathsf{hint}_{\mathsf{pk}}$	编码了 E_{pk} 上确定性挠基 $(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$ 的提示；存储在公钥中
M_{sk}	将 $(\phi_{\mathsf{sk}}(P_0), \phi_{\mathsf{sk}}(Q_0))$ 映到 $(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$ 的基变换矩阵
	哈希与挑战对象
$\mathsf{H}_{\mathsf{PRISM}}$	将 $(j(E), \mathsf{msg}, \mathsf{salt})$ 映射为长度为 $a-1$ 比特的奇整数的哈希函数
H_{a-2}	将比特串映射为 $[0, 2^{a-2})$ 中整数的辅助哈希函数
$\mathsf{PrimalityTest}$	应用于 $\mathsf{H}_{\mathsf{PRISM}}$ 输出的素性测试；以 $\mathrm{MR}_{\mathrm{iter}}$ 次 Miller–Rabin 迭代实例化
q	$\mathsf{H}_{\mathsf{PRISM}}$ 的素数输出，用于定义签名同源的次数 $q(2^a - q)$
I_{chall}	范数为 $q(2^a - q)$ 的挑战理想，在签名期间构造
salt	签名期间采样并包含在签名中的盐（salt）
n_{salt}	salt 的比特长度
	签名与验证对象
E_{sig}	签名曲线，签名同源的像曲线
σ	次数为 $q(2^a - q)$ 的签名同源 $E_{\mathsf{pk}} \to E_{\mathsf{sig}}$
P_{sig} 与 Q_{sig}	E_{sig} 上用于表示签名同源以供验证的点
M_{sig}	编码签名中使用的基变换矩阵
$\mathsf{hint}_{\mathrm{sig}}$	用于在 E_{sig} 上恢复挠基的提示
accept 与 reject	验证输出

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-PRISM 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 $k \in [0, 2^a]$

if $a = 1$ **then**
  return 穷举搜索满足 $\zeta_1 = \zeta_0^k$ 的 k
$a' \leftarrow \lfloor a/2 \rfloor$
$\zeta_0' \leftarrow \zeta_0^{2^{a-a'}}$
$\zeta_1' \leftarrow \zeta_1^{2^{a-a'}}$
$k' \leftarrow$ NormalizedDlog(a', ζ_0', ζ_1')
$\zeta_0'' \leftarrow \zeta_0^{2^{a'}}$
$\zeta_1'' \leftarrow \zeta_1 / \zeta_0^{k'}$
$k'' \leftarrow$ NormalizedDlog$(a - a', \zeta_0'', \zeta_1'')$
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-PRISM 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-PRISM 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。

2.2.2 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 $R=(x_R,y_R)$：

$$x_R=B\lambda^2-(x_P+x_Q)-A, \tag{2.5}$$

以及

$$y_R=\lambda(x_P-x_R)-y_P, \tag{2.6}$$

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 $\{r_+, r_-\} = \{x_{P-Q}, x_{P+Q}\}$：

$$r_\pm = \frac{B_{XZ} \pm \sqrt{B_{XZ}^2 - B_{ZZ}B_{XX}}}{B_{ZZ}}, \tag{2.7}$$

其中

$$\begin{aligned} B_{XX} &= (x_P x_Q - 1)^2, \\ B_{XZ} &= (x_P x_Q + 1)(x_P + x_Q) + 2A x_P x_Q, \\ B_{ZZ} &= (x_P - x_Q)^2. \end{aligned} \tag{2.8}$$

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 (2.8)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

对于 $m \in \mathbb{Z}$ 和一条超奇异曲线 $E/\mathbb{F}_{p^2}$，定义 $E[m]$ 为 E 的 m-挠子群，它包含所有满足 $[m]P = 0_E$ 的点 $P \in E(\overline{\mathbb{F}_p})$。若 $m \nmid p$，则 $E[m] \cong \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$。对于极大超奇异曲线，当 $m \mid p+1$ 时，有 $E[m] \subseteq E(\mathbb{F}_{p^2})$。因此，存在点 $R, S \in E[m]$ 生成 $E[m]$，但这组生成元并不唯一。这样的一对点称为 $E[m]$ 的基，记作 $E[m] = \langle R, S \rangle$。若仅使用 x 坐标，这样的基 (R, S) 可以用 x_R、x_S 以及 $R+S$ 或 $R-S$ 的 x 坐标来表示，记作 X_{RS}。因此，参考实现用三元组 (x_R, x_S, x_{RS}) 表示基 (R, S)。严格来说，这种表示法仅能将 (R, S) 确定到相差一个全局符号，即 (R, S) 和 $(-R, -S)$ 的表示相同。

本文档中，e 表示满足 $2^e \mid p+1$ 的最大整数，其中 p 为有限域的基素数。在 QIMEN-PRISM 中，需要为 $E[2^a]$ $(a \le e)$ 生成基 (R, S)，对此已有充分研究 [ZSP+18, CEMR24]。本文在 TorsionBasisToHint 中描述了 $E[2^a]$ 的基 (R, Q) 的确定性生成方法。该方法修改自 [ZSP+18, Alg. 3.1]，适用于 $A \neq 0$ 的 Montgomery 曲线 $E_A/\mathbb{F}_{p^2}$。具体而言，算法判断 A 是否为平方元，以选出一个合适的 $x_R \leftarrow -A/(1 + i \cdot b)$，其中 b 为平方元。选出的 x_R 是 $\mathbb{F}_{p^2}$ 中的非平方元。然后，若 $x_R \in E(\mathbb{F}_{p^2})$，则令 $x_S = -x_R - A$，x_S 也必为非平方元。在此之后，将 x_R 和 x_S 乘以辅因子 $(p+1)/2^a$ 以清除协因子分量，所得 (R, S) 共同构成 $E[2^a]$ 的基。算法的调整如下：因 $b \in \mathbb{F}_p$ 在 $\mathbb{F}_{p^2}$ 中总是平方元，故当 A 为非平方

元时，对任意 $\mathbb{F}_p$ 中的 b，$x_R = b \cdot A$ 均满足所需形式。可以证明，对于用此算法采样的基 (R, S)，点 $R+S$ 满足 $2^{a-1}(R+S) = (0,0)$，这是后续算法中假设的性质。实际需要的是 S 而非 $R+S$ 满足此性质，因此将 (x_R, x_S, x_{RS}) 置换为 (x_R, x_{RS}, x_S)，从而以 $(R, R+S)$ 作为基。给定 x_R 和 x_S 来计算 x_{RS} 必须是确定性的。然而，仅从 x_R 和 x_S 无法区分 x_{RS} 是 $R+S$ 还是 $R-S$ 的 x 坐标。只要实现采用与参考实现相同的确定性选择，这并不会引发问题。参考实现使用 ProjectiveDifference，参见 Section C.2。对于本算法不适用的特殊椭圆曲线 E_0，可在参数生成阶段预先计算 $E_0[2^e]$ 的基，因此无需精细优化。

提示值（hint） 为 $E_A[2^e]$ 采样这样的挠基可以加快验证者的速度：该算法依赖 A 的平方性以及查找 x_P 所需的索引。这两者都可以作为提示值（hint）包含在签名中——签名者使用 TorsionBasisToHint 生成基和相应的 hint，并将 hint 写入签名；验证者使用 TorsionBasisFromHint，以 hint 为输入，得到基 (x_R, x_S, x_{RS})。注意，TorsionBasisFromHint 返回的基能保证 x_R 和 x_S 两个值位于同一个扭（twist）上，即使这些值未经验证是 E 上的点。这不会引起问题，因为在使用这些点时会验证它们的阶，从而隐含地确保它们位于 E 上。参考实现如 Chapters C and D 中所述进行此操作。

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-PRISM 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

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

Algorithm 3 TorsionBasisToHint(A, a)

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

if A 是平方元 **then**
  $h_A \leftarrow 1$
else
  $h_A \leftarrow 0$
$h \leftarrow 0$
if A 是平方元 **then**
  repeat
    $h \leftarrow h + 1$
    $x_R \leftarrow -1/(1 + i \cdot h) \cdot A$
  until $(1 + h^2)$ 不是平方元且 $x_R \in E_A(\mathbb{F}_{p^2})$
else
  repeat
    $h \leftarrow h + 1$
    $x_R \leftarrow h \cdot A$
  until $x_R \in E_A(\mathbb{F}_{p^2})$
$x_{RS} \leftarrow -x_R - A$
$x_R \leftarrow$ 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$(x_R, x_{RS}, (A : 1))$
if $h \ge 128$ **then**
  $h \leftarrow 0$
return (x_R, x_S, x_{RS}) 和 (h_A, h)

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-PRISM 中配对的使用

Tate 配对。 Tate 配对 t_n 可用于快速求解离散对数问题。这基于 Tate–Lichtenbaum 配对的一个性质：对任意 $P \in E[n]$，有 $t_n(P, P) = 1$。例如，设 $E[n]$ 有基 $P_1, P_2 \in E(\mathbb{F}_q)$，并

Algorithm 4 TorsionBasisFromHint(A, a, h_A, h)

Input: 非零仿射 Montgomery 系数 A、整数 $a < e$，以及两个 hint h_A, h
Output: $E[2^a]$ 的 x-仅基 (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})

设 $\zeta = t_n(P_1, P_2)$ 为 n 次本原单位根。若需要将 $Q \in E[n]$ 表示为 $Q = [a_1]P_1 + [a_2]P_2$，则

$$t_n(P_1, Q) = t_n(P_1, [a_1]P_1 + [a_2]P_2) = t_n(P_1, P_1)^{a_1} \cdot t_n(P_1, P_2)^{a_2} = t_n(P_1, P_2)^{a_2} = \zeta^{a_2}.$$

类似地，$t_n(P_2, Q) = t_n(P_2, P_1)^{a_1} = \zeta^{-a_1}$。因此，给定 ζ，只需计算两个 n 级 Tate 配对，然后在 μ_n 中通过 NormalizedDlog 求解两个离散对数，即可得到 a_1, a_2。在 QIMEN-PRISM 中，对 $n = 2^a$ $(a \le e)$ 的情形，以上述方式使用 Tate 配对来计算基变换所对应的矩阵。具体而言，设 (P_1, P_2) 是 $E[2^e]$ 的基（这确保 $\zeta = t_{2^e}(P_1, P_2)$ 为本原 2^e 次单位根），(Q_1, Q_2) 是 $E[2^a]$ 的基，则可由 Tate 配对计算出 x_1, x_2, x_3, x_4 使得

$$\begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \cdot \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}.$$

本文在 ChangeOfBasis 中更精确地描述了这一过程。此外，若要计算将 $E[2^a]$ 的基 (Q_1, Q_2) 变为基 $([2^{e-a}]P_1, [2^{e-a}]P_2)$ 的 x_i，可以应用上述算法并求所得矩阵的逆。

计算 t_n 有多种高效方法；关于 QIMEN-PRISM 中的选择及这些算法的细节，参见 Chapter E。

Weil 配对。 在 QIMEN-PRISM 中，Weil 配对在签名和验证中均用于验证同源的次数。这利用了 Weil 配对的性质：当 $\deg\phi$ 与 n 互素时，$e_n(\phi P, \phi Q) = e_n(P, Q)^{\deg\phi}$。在 QIMEN-PRISM 中，$n = 2^a$ 的情形应用了此性质：计算 $\zeta = e_{2^a}(P, Q)$ 和 $\zeta' = e_{2^a}(\phi P, \phi Q)$，其中 (P, Q) 为 2^a-挠的基，这意味着 ζ 是 2^a 次单位根。验证 $\zeta' = \zeta^m$ 即表明 $\deg\phi = m \bmod 2^a$。

Algorithm 5 ChangeOfBasis$_{2^a}(E,(P_1,P_2),(Q_1,Q_2))$

Input: $E[2^e]$ 的基 (P_1,P_2) 和 $E[2^a]$ 的基 (Q_1,Q_2)
Output: 基变换矩阵 (x_i) $(1\le i\le 4)$，使得 $Q_1=[x_1]P_1+[x_2]P_2$ 且 $Q_2=[x_3]P_1+[x_4]P_2$

$\zeta \leftarrow t_{2^a}(P_1,P_2)$
$\zeta_1 \leftarrow t_{2^a}(Q_1,P_2)$, $\zeta_2 \leftarrow 1/t_{2^a}(Q_1,P_1)$, $\zeta_3 \leftarrow t_{2^a}(Q_2,P_2)$, $\zeta_4 \leftarrow 1/t_{2^a}(Q_2,P_1)$
for i 从 1 到 4 **do**
    $x_i \leftarrow 2^{e-a}\cdot$ NormalizedDlog(ζ,ζ_i)
return (x_1,x_2,x_3,x_4)

在签名中，此性质用于 IdealToIso 中以识别正确的像曲线（codomain）；而在验证中，用于 CheckIsogeny 中以验证同源 ϕ 的次数。

Weil 配对的高效实现方式是计算两个 Tate 配对的比值；关于 QIMEN-PRISM 中的选择及这些算法的细节，参见 Chapter E。

2.4 一维同源

Remark 2.4.1. *QIMEN-PRISM* 并不显式使用或计算一维同源，故我们在此省略了相关细节。不过，关于一维同源的一般性描述，对于理解*Section 2.5*中引入的二维同源依然很有帮助。

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，φ 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 $\deg(\varphi) = \max\{\deg(f_1), \deg(f_2)\}$。

2.5 二维同源

QIMEN-PRISM 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

$$(P_1, P_2) + (Q_1, Q_2) := (P_1 + Q_1,\ P_2 + Q_2), \tag{2.9}$$

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

$$C : y^2 = f(x), \tag{2.10}$$

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-PRISM 的使用： 对于 QIMEN-PRISM，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-PRISM 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。

2.5.1 $(2, 2)$-同源的实现细节

Chapter D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 D，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。
- 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。

[2] 与椭圆曲线类似，可以在所有 PPAS 上定义 Weil 配对。

- 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-PRISM 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 D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。

[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。

CHAPTER 3

四元数

本节介绍四元数，它们是构造 QIMEN-PRISM 的关键代数对象。Section 3.1 讨论整数、矩阵和格的基本算术。Section 3.2 定义四元数代数、四元数序、四元数理想，并引入在四元数序中求解范数方程的算法。最后，Section 3.3 解释理想与同源之间的联系，并给出将四元数理想高效转换为同源的算法。

3.1 整数、矩阵和格 *

3.1.1 整数算术

QIMEN-PRISM 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-PRISM 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 $a \neq 0$ 且 $b \neq 0$，则要求 $1 \leq au \leq |ab|/g$ 且 $-|ab|/g < bv \leq 0$。注意，GMP 的 XGCD 算法并不强制输出 $u \neq 0$，而 QIMEN-PRISM 需要这一保证；
- 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-PRISM 参考实现所采用的大整数库。

Algorithm 6 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

3.1.2 矩阵算术与 Hermite 标准形

基本整数矩阵算术 QIMEN-PRISM 需要操作若干小维度的整数矩阵（如 2×2, 4×4、$n \le 16$ 的 $4 \times n$）。基本运算（如矩阵-向量和矩阵-矩阵乘法）可以使用教科书方法实现。对于 2×2 矩阵的行列式和求逆，可以使用标准公式：

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc, \qquad \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

对于 4×4 矩阵的行列式和求逆，类似的 Laplace 公式 [Ebe07] 使用 78 次环运算计算行列式和伴随矩阵。

Hermite 标准形（HNF） Hermite 标准形（HNF）是整系数矩阵约化阶梯形的推广。称矩阵 H 为（列式）HNF，若其满足以下条件：

- 上三角（即 $h_{ij} = 0$ 对 $j < i$），且任意零列位于左侧；
- 非零行的首系数（即主元）总是严格高于其上方行的首系数，且为正；

- 主元左侧元素为零，主元右侧元素为非负且严格小于主元。

称矩阵 A 以 H 为 HNF，若 H 为 HNF 且存在幺模矩阵 U 使得 $AU = H$。这里幺模指 U 具有整系数且行列式为 ± 1。此时 A 和 H 具有相同的列空间，且 H 唯一，从而为 A 的列空间提供了规范表示。任意矩阵的 HNF 计算算法参见 [Coh93, 2.4.2]。HNF 给出了计算矩阵 HNF 的算法示例。

在 QIMEN-PRISM 中，理想由其基表示，即一个 4×4 矩阵，其列为基向量。该矩阵的 HNF 用作理想的规范表示，同时也用于检查理想的相等性、计算它们的和与交，以及计算一个理想在另一个理想中的指数。更多细节参见 Section 3.2 和 Section 3.2.2。

3.1.3 格与格约化

设 V 为 d 维 $\mathbb{Q}$-向量空间。V 中的一个满秩格（full rank lattice）$\mathcal{L} \subseteq V$ 是 V 的某个 $\mathbb{Q}$-基的 $\mathbb{Z}$-张成，即 $\mathcal{L} = \mathbb{Z}b_0 + \cdots + \mathbb{Z}b_{d-1}$，其中 $\{b_0, \ldots, b_{d-1}\}$ 是 V 的基。本文档仅涉及满秩格，以下提及" 格" 时均指满秩格。

QIMEN-PRISM 中考虑的格位于一个二次空间中。二次空间是一个有限维向量空间，其上定义了如下的对称双线性型。

Definition 3.1.1 (对称双线性型)**.** $\mathbb{Q}$-向量空间 V 上的对称双线性型是一个映射，对任意 $a, b \in V$ 关联一个值 $\langle a, b\rangle \in \mathbb{Q}$，具有以下性质：

- $\langle a, b\rangle = \langle b, a\rangle$,
- $\langle a + b, c\rangle = \langle a, c\rangle + \langle b, c\rangle$,
- $\langle \lambda a, b\rangle = \lambda\langle a, b\rangle$,

对任意 $a, b \in V$ 和任意 $\lambda \in \mathbb{Q}$。两个向量 $a, b \in V$ 称为正交，若 $\langle a, b\rangle = 0$。

任意对称双线性型都关联一个二次型，定义为 $Q(a) = \langle a, a\rangle$。反过来，双线性型可由二次型重建，公式为 $\langle a, b\rangle = (Q(a + b) - Q(a) - Q(b))/2$。

向量 a 称为迷向（isotropic），若 $Q(a) = 0$。若对所有 $a \neq 0$ 有 $Q(a) > 0$，则称二次型为正定，此时双线性型和二次空间也称为正定。在这种情况下，称 $Q(a)$ 为向量 a 的长度。正定对称双线性型的一个例子是 $\mathbb{Q}^n$ 的内积 $a \cdot b$，其关联的二次型为欧氏范数的平方 $\|a\|^2$。

给定二次空间 V 的基 $b_0, \ldots, b_{d-1}$，其 *Gram* 矩阵是一个对称矩阵，其 (i, j)-元为 $\langle b_i, b_j\rangle$。该 Gram 矩阵唯一确定了格上的双线性型（该格由 $b_0, \ldots, b_{d-1}$ 张成）。二次空间中的格不一定存在正交基。格约化的目标是计算格的一组基，使其在精确意义下" 尽可能正交"。

Definition 3.1.2 (约化基)**.** 设 $b_0, \ldots, b_{d-1}$ 为正定二次空间的一组基。定义 Gram–Schmidt 向量 $\bar{b}_i$ 为

$$\bar{b}_i = b_i - \sum_{j=0}^{i-1} \mu_{i,j}\,\bar{b}_j, \qquad \text{其中} \quad \mu_{i,j} = \frac{r_{i,j}}{r_{j,j}} \quad \text{且} \quad r_{i,j} = \langle b_i, \bar{b}_j\rangle. \tag{8}$$

称该基为(η, δ)-约化的，对于参数 $\frac{1}{2} < \eta < 1$ 和 $\frac{1}{4} < \delta < 1$，若：

- $|\mu_{i,j}| < \eta$ 对 $0 \le j < i < d$，且
- $\langle \bar{b}_i, \bar{b}_i \rangle \ge (\delta - \mu_{i,i-1}^2) \langle \bar{b}_{i-1}, \bar{b}_{i-1} \rangle$ 对 $1 \le i < d$。

若一个对称矩阵是某组 (η, δ)-约化基的 Gram 矩阵，则称该矩阵为(η, δ)-约化的。

QIMEN-PRISM 中考虑的对称双线性型将在 Section 3.2.1 中给出。在 QIMEN-PRISM 中，IdealToIso 算法用于密钥生成和签名，其中一个关键步骤是计算输入格的约化基。在欧氏空间中（双线性型由 $\mathbb{Q}^n$ 的内积给出），此步骤使用 LLL 算法 [LLL82]。而 QIMEN-PRISM 中的双线性型不再是标准形式，因此采用 [NS09] 中的格约化算法 $\mathrm{L2}_{\eta,\delta}$。该算法的细节见 Section F.1。

浮点数 QIMEN-PRISM 在实现格约化算法 $\mathrm{L2}_{\eta,\delta}$ 时使用有限精度的浮点数（参见 Section F.1）。大多数平台的原生浮点类型不足以满足 QIMEN-PRISM 的需求，但任何具有至少 24 位尾数和至少 20 位指数的浮点类型，对所有安全级别来说都足够宽松。

构造此类类型的一个标准方法是将一个原生浮点类型用于存放尾数，搭配一个原生整数类型用于存放指数。参考实现使用"double plus exponent" 头文件库 [PZ24] 来实现这一点。

使用浮点数实现格约化算法时，输出的基受许多因素影响：具体的操作顺序（浮点数不满足结合律）、所使用的舍入模式，甚至编译器标志的选择，都会导致输出差异。因此，QIMEN-PRISM 的替代实现可能难以精确复现 Chapter 6 中描述的已知答案测试（KAT）。不过，这并不妨碍生成有效签名——这些签名仍可通过 Algorithm 21 的正确实现进行验证。

3.2 四元数与理想 *

3.2.1 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

$$\mathbf{i}^2 = -1, \qquad \mathbf{j}^2 = -p, \qquad \mathbf{ij} = -\mathbf{ji} = \mathbf{k}. \tag{3.1}$$

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. (3.1)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 $\mathrm{nrd}(\alpha) := \alpha\bar{\alpha} = a^2 + b^2 + p(c^2 + d^2)$。

映射

$$\langle \alpha, \beta \rangle = \mathrm{tr}(\alpha\bar{\beta}) \tag{3.2}$$

是一个对称双线性型，它使得 $B_{p,\infty}$ 成为一个二次空间。此外，

$$\langle \alpha, \alpha \rangle = \mathrm{tr}(\alpha\bar{\alpha}) = 2\ \mathrm{nrd}(\alpha),$$

这意味着该二次空间是正定的。

3.2.2 四元数格

一个满秩格 $\Lambda \subseteq \mathcal{B}_{p,\infty}$ 由一组 $\mathbb{Q}$-线性无关四元数的基 $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ 定义。按照四元数元素的表示惯例，基表示为矩阵 L 的列，即

$$(\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4) = (1\ \mathbf{i}\ \mathbf{j}\ \mathbf{k}) \cdot L.$$

参考实现中提取公分母，将 L 表示为整数矩阵 M 和公分母 r，使得 $L = M/r$。这属于实现细节。

格 Λ 的对偶定义为

$$\Lambda^\star = \{f \in \mathcal{B}_{p,\infty}^\star \mid \forall x \in \Lambda,\ f(x) \in \mathbb{Z}\},$$

其中 $\mathcal{B}_{p,\infty}^\star$ 表示线性函数 $B_{p,\infty} \to \mathbb{Q}$ 的空间。设 $(1^\star, \mathbf{i}^\star, \mathbf{j}^\star, \mathbf{k}^\star)$ 为 $(1, \mathbf{i}, \mathbf{j}, \mathbf{k})$ 的对偶基，即对于 $\alpha = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$，

$$1^\star(\alpha) = a, \qquad \mathbf{i}^\star(\alpha) = b, \qquad \mathbf{j}^\star(\alpha) = c, \qquad \mathbf{k}^\star(\alpha) = d.$$

这是对偶格 $(\mathbb{Z} + \mathbf{i}\mathbb{Z} + \mathbf{j}\mathbb{Z} + \mathbf{k}\mathbb{Z})^\star$ 的基。若 Λ 由基 $(1\ \mathbf{i}\ \mathbf{j}\ \mathbf{k}) \cdot L$ 生成，则 $\Lambda^\star$ 由 $L^{-1} \cdot (1^\star\ \mathbf{i}^\star\ \mathbf{j}^\star\ \mathbf{k}^\star)^t$ 生成。

下文将轻微滥用记号，将定义格基的矩阵的列（相应地，行）与格（相应地，对偶格）本身等同起来。基本运算的计算方式如下。

相等性 (Algorithm 67)：检查 L_1 和 L_2 具有相同的 HNF [Coh93, 2.4.3]。

和 (Algorithm 68)：若 L_1 和 L_2 为格，拼接其矩阵 $L_1 \mid L_2$ 并计算 HNF 得到 $L_1 + L_2$。

交 (Algorithm 70)：若 L_1 和 L_2 为格，计算其对偶格 $L_1^\star$ 和 $L_2^\star$；则 $L_1 \cap L_2$ 是 $L_1^\star + L_2^\star$ 的对偶。

乘法 (Algorithm 71)：若 L_1 和 L_2 为格，其乘积 L_1L_2 的一种计算方式是：构造 L_2 的基 $\alpha_1,\dots,\alpha_4$ 所对应的右乘矩阵 $A_1,\dots,A_4$，然后计算和

$$A_1L_1 + A_2L_1 + A_3L_1 + A_4L_1.$$

包含（元素） (Algorithm 72)：给定元素 $\alpha \in \mathcal{B}_{p,\infty}$ 和格 L，通过求解线性方程组 $LX = \alpha$ 并验证 X 具有整数项来检查 $\alpha \in L$。

包含（格） (Algorithm 73)：检查格 L_1 是否包含于格 L_2，可以通过检查 L_1 的所有 4 个基向量在 L_2 中的包含性，或测试 $L_1 + L_2$ 与 L_2 的相等性。

指数 (Algorithm 74)：当 $L_1 \subset L_2$ 时，L_1 在 L_2 中的指数，记作 $[L_2 : L_1]$，是有限商群 L_2/L_1 的阶。该值等于 $|\det(L_1)/\det(L_2)|$，可使用四维行列式算法计算。

基约化 (Algorithm 78)：称一个格是约化的，当其基按照 Section 3.1.3 的定义是约化的，且其中的双线性型具体取为 Eq. (3.2)。约化基通过 $\mathrm{L2}_{\eta,\delta}$ 算法计算。基 $(1,\mathbf{i},\mathbf{j},\mathbf{k})$ 的 Gram 矩阵是对角矩阵 G，其对角元为 $(2,2,2p,2p)$。若另一组基 $(\alpha_1,\alpha_2,\alpha_3,\alpha_4)$ 写作 $(\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4) = (1\ \mathbf{i}\ \mathbf{j}\ \mathbf{k})M$（$M$ 为某矩阵），则其 Gram 矩阵为 $G' = M^tGM$。

3.2.3 四元数序与理想

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-PRISM 使用的四元数代数含有一个极大序，基为 $(1,\mathbf{i},\frac{1+\mathbf{j}}{2},\frac{1+\mathbf{k}}{2})$；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 $(1,\mathbf{i},\mathbf{j},\mathbf{k})$。

设 $\mathcal{O}$ 为一个序。$\mathcal{O}$ 的左（右）整理想（此后简称 $\mathcal{O}$ 的左（右）理想）是 $\mathcal{O}$ 的子格，且在 $\mathcal{O}$ 的左（右）乘下封闭。理想的左序定义为

$$\mathcal{O}_L(I) = \{\alpha \in \mathcal{B}_{p,\infty} \mid \alpha I \subset I\}$$

右序 $\mathcal{O}_R(I)$ 类似定义。此时，I 显然是 $\mathcal{O}_L(I)$ 的左理想。左序为 $\mathcal{O}_L$ 且右序为 $\mathcal{O}_R$ 的理想称为 $\mathcal{O}_L$ 和 $\mathcal{O}_R$ 的连接理想（connecting ideal），记作 $(\mathcal{O}_L,\mathcal{O}_R)$-理想。理想 I 的约化范数记作 $\mathrm{nrd}(I)$，定义为 I 中所有元素的约化范数的最大公约数。它是一个整数，且等于

$$\sqrt{[\mathcal{O}_L(I) : I]} = \sqrt{[\mathcal{O}_R(I) : I]}.$$

任意理想可写为

$$I = \mathcal{O}_L(I)\alpha + \mathcal{O}_L(I)\ \mathrm{nrd}(I)$$

其中 $\alpha \in \mathcal{O}_L(I)$（对 $\mathcal{O}_R(I)$ 有类似的表达式）。为简化记号，对任意序 $\mathcal{O}$ 记 $\mathcal{O}\alpha + \mathcal{O}N = \mathcal{O}(\alpha, N)$。

满足 $\mathcal{O}_R(I) = \mathcal{O}_L(J)$ 的理想 I 和 J，其乘积 IJ 按格的乘法定义。由此 IJ 也是（整）理想，且 $\mathcal{O}_L(IJ) = \mathcal{O}_L(I)$、$\mathcal{O}_R(IJ) = \mathcal{O}_R(J)$。理想范数关于理想乘积满足乘性。

在序上通过共轭定义等价关系，在左 $\mathcal{O}$-理想上通过右标量乘法定义等价关系。两个序 $\mathcal{O}_1$ 和 $\mathcal{O}_2$ 等价，若存在 $\beta \in \mathcal{B}_{p,\infty}^{\times}$ 使得 $\beta\mathcal{O}_1 = \mathcal{O}_2\beta$。两个左 $\mathcal{O}$-理想 I 和 J 等价，若存在 $\beta \in \mathcal{B}_{p,\infty}^{\times}$ 使得 $I = J\beta$。若后者成立，则 $\mathcal{O}_R(I)$ 和 $\mathcal{O}_R(J)$ 等价，因为 $\beta\mathcal{O}_R(I) = \mathcal{O}_R(J)\beta$。

理想用格表示，且其相等性、属于、和、交和乘法的计算与格相同。上述基本运算可用于计算形如 $\mathcal{O}(\alpha, N)$ 的理想：先用格乘法计算 $\mathcal{O}\alpha$ 和 $\mathcal{O}N$，再用格和得到 $\mathcal{O}(\alpha, N)$。反过来，给定约化范数为 N 的理想 I，可通过枚举 I 中的元素，直到找到满足 $\gcd(\mathrm{nrd}(\alpha), N^2) = N$ 的 α，即得 $I = \mathcal{O}(\alpha, N)$。参考实现中使用此方法，详见 IdealGenerator。

下面列出在 QIMEN-PRISM 中使用的一些理想相关算法。这些算法的细节见 Section F.2。

理想逆 (Algorithm 83)：由于 QIMEN-PRISM 中所有理想都是连接极大序的，它们都有逆。这种理想 I 的逆为

$$I^{-1} = \frac{1}{\mathrm{nrd}(I)}\,\overline{I}$$

其中 $\overline{I}$ 是 I 中元素的共轭构成的格。I^{-1} 不是理想，而是一个秩-4 格，且对于格乘法，$II^{-1} = \mathcal{O}_L(I)$ 且 $I^{-1}I = \mathcal{O}_R(I)$。

理想的左序和右序 (Algorithm 84)：由于 QIMEN-PRISM 中所有理想都是连接极大序的理想，理想 I 的左序和右序由其逆给出：$\mathcal{O}_L(I) = II^{-1}$、$\mathcal{O}_R(I) = I^{-1}I$。

计算连接理想 (Algorithm 87)：给定两个序 $\mathcal{O}_L$ 和 $\mathcal{O}_R$，计算连接理想为

$$I = N\mathcal{O}_L\mathcal{O}_R,$$

其中 N 是 $\mathcal{O}_L \cap \mathcal{O}_R$ 在 $\mathcal{O}_L$ 中的指数的平方根。

理想的拉回与前推： 回顾 [DKL+20, Lemma 3] 中的两个定义。给定两个理想 I, J，满足 $\mathcal{O}_R(J) = \mathcal{O}_L(I)$ 且范数互素，定义拉回（pullback）理想为 $\mathcal{O}_L(J)$-理想

$$[J]^*I = JI + \mathrm{nrd}(I)\mathcal{O}_L(J).$$

类似地，当 I, J 是两个范数互素的左 $\mathcal{O}$-理想时，定义前推（pushforward）理想为左 $\mathcal{O}_R(J)$-理想

$$[J]_*I = J^{-1}(J \cap I).$$

可以验证 $[J]^*([J]_*I) = I$。

3.2.4 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。

3.2.4.1 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 7 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 $\perp$
$r \leftarrow \text{ModularSQRT}(-1, m)$
$s \leftarrow m$
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-PRISM 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-PRISM 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 8 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 $(\mathrm{Re}(z), \mathrm{Im}(z))$

3.2.4.2 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

$$x^2 + y^2 + p(z^2 + t^2) = 4M \tag{3.3}$$

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. (3.3)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 $M' := 4M - p(z^2 + t^2)$。当 M' 为模 4 余 1 的素数时，调用Cornacchia求解二元二次方程 $x^2 + y^2 = M'$。

Algorithm 9 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 PrimalityTest(M') **then**
    if $M' \equiv 1 \pmod 4$ **then**
      found $\leftarrow$ true
      $(x, y) \leftarrow$ Cornacchia(M')
      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")

Remark 3.2.1. 由于 $x^2 + y^2 = M' \equiv 1 \pmod 4$，整数 x 和 y 具有相反的奇偶性。此外，由 $M' = 4M - p(z^2 + t^2)$，其中 p 和 M' 均为奇数，整数 z 和 t 也具有相反的奇偶性。因此，要么 $x \equiv t \pmod 2$ 且 $y \equiv z \pmod 2$，要么这两个匹配关系互换。在后一种情况下，将 (x, y) 替换为 (y, x) 即可得到所需的同余关系。这种交换保持 *Cornacchia* 方程不变，因为 $x^2 + y^2 = y^2 + x^2$。因此，在可能的交换之后，$\gamma = (x + y\mathbf{i} + z\mathbf{j} + t\mathbf{k})/2$ 属于 $\mathcal{O}_0$。

[AAA+25] 也使用了类似的思想，但仅限于架构特定的条件，且未用于 *[BBC+26]*。引入此额外的交换机制后，*QIMEN-PRISM* 在所有安全级别下的签名时间缩短了 *15%–20%*。

3.2.4.3 计算理想

QIMEN-PRISM 需要计算两种不同类型的随机理想，由以下两个算法分别处理：

RandFixNormIdeal 给定一个足够大且不被 p 整除的整数 N，该算法均匀随机采样一个约化范数为 N 的左 $\mathcal{O}_0$-理想。当 N 不是素数时，应以参数 prime 设为 false 调用。此时RandFixNormIdeal先用GeneralizedRepresentInteger寻找 $\mathcal{O}_0$ 中范数为 N 的元素 γ。如果GeneralizedRepresentInteger失败，算法约定将该失败向上传播。然后将此四元数与 $\mathcal{O}_0/N\mathcal{O}_0$ 中的一个均匀随机四元数 β 相乘，其中 β 的范数与 N 互素。所得 $\gamma\beta$ 的范数仍能被 N 整除。算法随后返回由 N 和 $\gamma\beta$ 生成的左 $\mathcal{O}_0$-理想。为了使

用GeneralizedRepresentInteger，N 应相对于 p 足够大。在 QIMEN-PRISM 的使用场景中，这一条件成立。若 N 已知为素数，RandFixNormIdeal应使用参数 `prime` 设为 `true` 调用。此时不使用GeneralizedRepresentInteger，而是采样一个迹为零的随机四元数，其范数 n 满足 $-n$ 为模 N 的平方剩余。

Remark 3.2.2. 在 *QIMEN-PRISM* 中，*RandFixNormIdeal*仅在*QIMEN-PRISM.GenIso* 中以 *`prime`* 为 *`false`* 调用，且 $N = q(a^a - q)$。论文中未明确说明，但我们观察到在 salt-*PRISM* *[BBC+26]* 中，此时*GeneralizedRepresentInteger*在 *Line 10*的输入是 $q^2(2^a - q)$ 而非 $q(2^a - q)$。在 *QIMEN-PRISM* 中，我们将其改为 $q(2^a - q)$。这不影响正确性，也不削弱安全性：后续步骤会采样 β，从而随机化返回的理想。我们在实验中观察到，此修改使*QIMEN-PRISM.Sign* 在所有安全级别下对 *NGCC* 的速度提升至原来的约 *1.12* 到 *1.17* 倍。原因是：给*GeneralizedRepresentInteger*更小的输入，提高了找到素数的概率，同时素数测试步骤也变得更快。

Algorithm 10 RandFixNormIdeal(N, `prime`)

Input: 正整数 N（不被 p 整除）为某个左 $\mathcal{O}_0$ 理想的范数，以及一个布尔值 `prime` 指示 N 是否为素数。

Output: 一个范数为 N 的随机左理想 J'，或引发异常。

```
found ← false
if prime then
    while not found do
        g1, g2, g3 ← [0, N − 1] 中独立的均匀随机整数
        γ ← g1·i + g2·j + g3·k
        found ← (1 = Legendre(− nrd(γ), N))
        if found then
            γ ← γ + ModularSQRT(− nrd(γ), N)
else
    γ ← GeneralizedRepresentInteger(N)
while not found do
    x, y, z, w ← [1, N] 中均匀随机选取的整数
    β ← x + y·i + z·j + w·k
    found ← (gcd(nrd(β), N) = 1)
J' ← 由 γβ 和 N 生成的左 O0-理想
return J'
```

RandomEquivalentPrimeIdeal 给定一个左 $\mathcal{O}_0$-理想，此算法找到一个与之等价的左 $\mathcal{O}_0$-理想 J，使得 $\mathrm{nrd}(J)$ 为一个小素数。注意，在该算法中，输出 J 在所有等价理想中的分布不影响方案的安全性。给定一个整理想 I，RandomEquivalentPrimeIdeal找到一个等价理想 J（即 $I = J\alpha$，其中 $\alpha \in \mathcal{B}_{p,\infty}^{\times}$），且具有不同（即素数且有界）的范数。为此，

它使用满射

$$\chi_I(\alpha) = I\frac{\bar{\alpha}}{\mathrm{nrd}(I)}$$

将 $I \setminus \{0\}$ 映射到与 I 等价的理想集合。

该算法在界 $[-\texttt{QUAT\_repres\_bound\_input}, \texttt{QUAT\_repres\_bound\_input}]$ 内采样常数 c_i，构造 $\beta = \sum_{i=1}^{4} c_i\alpha_i$，其中 $(\alpha_1, \ldots, \alpha_4)$ 为 I 的一个约化基；若 J 的范数为素数，则输出 $J = \chi_I(\beta)$。从启发式角度看，输出理想的范数可望约为 $\approx \sqrt{p}$，这应远小于 QIMEN-PRISM 中输入理想的范数。若在规定试验次数内（由Section 4.1.1中指定的算法参数 `QUAT_repres_bound_input` 确定）未找到范数为素数的等价理想，则算法报告失败。

Algorithm 11 RandomEquivalentPrimeIdeal(I)

Input: I，一个左 $\mathcal{O}_0$-理想。
Output: $J \sim I$ 具有小素数范数，若不成功则引发异常。

初始化 `counter` $\leftarrow 0$
$G_I \leftarrow \mathrm{Gram}(I)$。
$((\alpha_1, \alpha_2, \alpha_3, \alpha_4), \_) \leftarrow \mathrm{L2}_{\eta,\delta}(I, G_I)$
while `counter` $< (2 \cdot \texttt{QUAT\_equiv\_bound\_coeff} + 1)^4$ **do**
    `counter` $\leftarrow$ `counter` $+ 1$
    从 $[-b, \ldots, b]$ 中均匀随机采样 c_1, c_2, c_3, c_4，其中界 $b = \texttt{QUAT\_equiv\_bound\_coeff}$
    $\beta \leftarrow \sum_{i=1}^{4} c_i\alpha_i$
    $J \leftarrow \chi_I(\beta)$
    if $\mathrm{nrd}(J)$ 为素数 **then return** J
引发异常: ("RandomEquivalentPrimeIdeal failed")

3.3 理想到同源的转换

本节解释如何将四元数理想转换为同源。Section 3.3.1 回顾理想与同源之间的对应关系。Section 3.3.2 描述二维同源的表示，Section 3.3.3 描述 Qlapoti 算法——该转换的一个关键子程序。最后，Section 3.3.4 给出 IdealToIso，一种将理想高效转换为对应同源的算法。

3.3.1 理想与同源的对应

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。

设 E 为一条椭圆曲线，$\mathcal{O}$ 为一个序，且带有同构 $\mathcal{O} \simeq \mathrm{End}(E)$。该数据建立了以下两

个集合之间的双射（bijection）：

$$\left\{ \begin{array}{c} \mathcal{O} \text{ 的左理想} \\ (\text{范数与 } p \text{ 互素}) \end{array} \right\} \quad \text{与} \quad \left\{ E \text{ 的有限子群} \right\}.$$

结合有限子群与可分同源之间的对应，得到双射

$$\left\{ \begin{array}{c} \mathcal{O} \text{ 的左理想} \\ (\text{范数与 } p \text{ 互素}) \end{array} \right\} \quad \longleftrightarrow \quad \left\{ \begin{array}{c} \text{从 } E \text{ 出发的} \\ \text{可分同源} \end{array} \right\} / \sim,$$

其中 $\sim$ 表示模掉同构后复合的等价关系，即 $\varphi \sim \varphi'$ 若存在同构 ι 使 $\varphi' = \iota \circ \varphi$。具体而言，理想 I 映到有限子群

$$E[I] := \{P \in E \mid \alpha(P) = 0_E,\ \forall \alpha \in I\},$$

当 I 写成 $I = \mathcal{O}\langle \alpha, N \rangle$ 时，这简化为

$$E[I] := \ker \alpha \cap E[N].$$

给定这样的理想 I，将其对应的可分同源记为 φ_I；该同源以 $E[I]$ 为核。反过来，给定从 E 出发的可分同源 φ，将其对应的理想记为 I_φ，其显式表达式为：

$$I_\varphi = \{\alpha \in \mathcal{O} \mid \alpha(P) = 0_E,\ \forall P \in \ker \varphi\}.$$

QIMEN-PRISM 中使用素数 $p \equiv 3 \pmod 4$。对于这样的素数，曲线

$$E_0 : y^2 = x^3 + x$$

是超奇异的，且其自同态环 $\mathrm{End}(E_0)$ 同构于

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

该同构的显式构造为：将 $\mathbf{j}$ 映到 Frobenius 自同态 $(x, y) \mapsto (x^p, y^p)$，将 $\mathbf{i}$ 映到 E_0 上的自同构 $(x, y) \mapsto (-x, \sqrt{-1}y)$，其中 $\sqrt{-1}$ 是 -1 在 $\mathbb{F}_{p^2}$ 中的平方根。本文档剩余部分固定上述对 E_0、$\mathcal{O}_0$ 以及同构 $\mathrm{End}(E_0) \simeq \mathcal{O}_0$ 的选择。符号 $\mathcal{O}$ 表示任意极大序，可能等于 $\mathcal{O}_0$。假设给定了同源 $\varphi : E_0 \to E$ 以及对应的 $(\mathcal{O}_0, \mathcal{O})$-理想 I。则固定的同构 $\mathcal{O}_0 \simeq \mathrm{End}(E_0)$ 诱导同构 $\mathcal{O} \simeq \mathrm{End}(E)$。令 $N = \mathrm{nrd}(I)$。由于 I 是 $(\mathcal{O}_0, \mathcal{O})$-理想，有 $N\mathcal{O} \subseteq \mathcal{O}_0$。该诱导同构通过以下方式获得：先将 $N\alpha \in N\mathcal{O}$ 视为 $\mathcal{O}_0$ 的元素，再将其映为

$$\varphi \circ \alpha \circ \widehat{\varphi} \in \mathrm{End}(E).$$

然后通过标量扩张将同构扩展到整个 $\mathcal{O}$。在以下算法中，所有计算或者在 E_0 上进行，或者在一条已知数对 (φ, I) 的曲线 E 上进行。因此不必显式指定对应中使用的同构——它总是隐式定义的 $\mathcal{O} \simeq \mathrm{End}(E)$。

3.3.2 二维同源的表示

设 d_1, d_2, a 为正整数，满足 $\gcd(d_1, d_2) = 1$ 且 $d_1 + d_2 = 2^a$。考虑以下椭圆曲线之间的同源交换图：

$$\begin{array}{ccc} E_1 & \xrightarrow{\varphi_1} & F_1 \\ {\scriptstyle\varphi_2}\downarrow & & \downarrow{\scriptstyle\varphi_2'} \\ F_2 & \xrightarrow[\varphi_1']{} & E_2 \end{array} \tag{3.4}$$

其中 $\deg(\varphi_1) = \deg(\varphi_1') = d_1$ 且 $\deg(\varphi_2) = \deg(\varphi_2') = d_2$。则由

$$\Phi : E_1 \times E_2 \to F_1 \times F_2$$

定义的同源 $\begin{pmatrix} \varphi_1 & \widehat{\varphi_2'} \\ -\varphi_2 & \widehat{\varphi_1'} \end{pmatrix}$ 是一个 $(2^a, 2^a)$-同源，其核为

$$\{([d_1]P, \varphi_2' \circ \varphi_1(P)) \mid P \in E_1[2^a]\}. \tag{3.5}$$

给定 E_1, E_2, d_1, d_2, e 以及 ψ 在 $E_1[2^{a+2}]$ 上的限制，利用椭圆曲线乘积之间的某个同构 ι，可通过 Isogeny22Chain 计算同源 $\iota \circ \Phi$。

在 QIMEN-PRISM 的验证中，验证者给定 $(E_1, P_1, Q_1, E_2, P_2, Q_2, a, q)$，需检查是否存在同源 $\varphi : E_1 \to E_2$，其次数为 $q(2^a - q)$ 且满足 $\varphi(P_1) = [q]P_2$、$\varphi(Q_1) = [q]Q_2$。若存在，则该同源 φ 可自然地从形如 Eq. (3.4) 的交换图中导出：取 $d_1 = q$（或 $2^a - q$）、$d_2 = 2^a - q$（或 q）。因此，验证 φ 的存在性等价于计算 $\iota \circ \Phi$——具体是检查 $\iota \circ \Phi(P, 0_{E_2})$ 的第一个分量是否为 $\varphi(P)$，其中 φ 为次数 q 或 $2^a - q$ 的同源、$P \in E_1[2^a]$。最终判定由下文所述的 Weil 配对计算完成。

设 (P, Q) 为 $E_1[2^a]$ 的基，$(P_1, P_2) = \iota \circ \Phi(P, 0_{E_2})$，$(Q_1, Q_2) = \iota \circ \Phi(Q, 0_{E_2})$。则有

$$e_{2^a}(P_1, Q_1) = e_{2^a}(P, Q)^{d_1}.$$

由于 $2^a > q, 2^a - q$ 且 $e_{2^a}(P, Q)$ 是本原 2^a 次单位根，检查 $e_{2^a}(P_1, Q_1)$ 等于 $e_{2^a}(P, Q)^q$ 还是 $e_{2^a}(P, Q)^{2^a - q}$，即可确定 d_1 的值。若等于其中之一，则 φ 存在；否则不满足。CheckIsogeny 中的验证即按此方式执行。若此项检查失败，验证算法拒绝该签名，判定其为无效。

3.3.3 Qlapoti 算法

Qlapoti 算法最初在 [BCRSE+26] 中引入，是 IdealToIso 的关键子程序。给定左 $\mathcal{O}_0$-理想 J 和正整数 k，该算法计算两个左 $\mathcal{O}_0$-理想 I_1 和 I_2，两者均等价于 J 且满足

$$\mathrm{nrd}(I_1) + \mathrm{nrd}(I_2) = 2^k.$$

在 QIMEN-PRISM 中，$k = e - 2$，其中 e 是 Section 4.1.1 中定义的参数。等价地，该算法寻找 $\beta_1, \beta_2 \in J$ 满足

$$\mathrm{nrd}(\beta_1) + \mathrm{nrd}(\beta_2) = 2^k \,\mathrm{nrd}(J). \tag{3.6}$$

Algorithm 12 CheckIsogeny$(E_1, P_1, Q_1, E_2, P_2, Q_2, a, q)$

Input: $(E_1, P_1, Q_1, E_2, P_2, Q_2, a, q)$，其中 (P_1, Q_1) 是 $E_1[2^{a+2}]$ 的基，(P_2, Q_2) 是 $E_2[2^{a+2}]$ 的基。

Output: 若存在同源 $\varphi : E_1 \to E_2$ 其次数为 $q(2^a - q)$ 且使得 $\varphi(P_1) = [q]P_2$、$\varphi(Q_1) = [q]Q_2$，则返回 true；否则返回 false。

$K_1 \leftarrow ([q]P_1, P_2)$
$K_2 \leftarrow ([q]Q_1, Q_2)$
try
    $\_, [(P', \_), (Q', \_)] \leftarrow$ Isogeny22Chain$(K_1, K_2, [(P_1, 0_{E_2}), (Q_1, 0_{E_2})])$
catch
    return false
if $e_{2^a}(P', Q') = e_{2^a}(P_1, Q_1)^m$ 对 $m = q$ 或 $2^a - q$ **then**
    return true
return false

下面概述寻找 β_1, β_2 的 Qlapoti 算法流程。

将 J 写作 $J = \mathcal{O}_0\langle N, \alpha\rangle$，其中 $N = \mathrm{nrd}(J)$。对 $i = 1, 2$，令 $\beta_i = \gamma_i N + \gamma_i'\alpha$，其中 $\gamma_i, \gamma_i' \in \mathcal{O}_0$。限制到 $\gamma_i = a_i + b_i\mathbf{i}$、$\gamma_i' = 1$，并记 $\alpha = a_\alpha + b_\alpha\mathbf{i} + c_\alpha\mathbf{j} + d_\alpha\mathbf{k}$。在此限制下，Eq. (3.6) 等价于：

$$N \cdot (a_1^2 + b_1^2 + a_2^2 + b_2^2) + 2a_\alpha(a_1 + a_2) + 2b_\alpha(b_1 + b_2) = 2^k - 2r, \tag{3.7}$$

其中 $r = \mathrm{nrd}(\alpha)/N$。

求解 Eq. (3.7) 的第一步是寻找满足以下同余式的小解 (s, t)：

$$2a_\alpha x + 2b_\alpha y = 2^k - 2r \pmod N.$$

这一步称为短同余步骤，详见 QlapotiShortCongruence。QlapotiShortCongruence 算法还使用了 [Coh93, Algorithm 3.1.14] 的二维 Gauss 约化过程 Dim2ReducedBasis，其中 $\mathbb{Z}^2$ 中秩二格的有序基 $(\boldsymbol{b}_0, \boldsymbol{b}_1)$ 称为约化的，若

$$\|\boldsymbol{b}_0\|^2 \le \|\boldsymbol{b}_1\|^2 \qquad \text{且} \qquad |\boldsymbol{b}_0 \cdot \boldsymbol{b}_1| \le \frac{1}{2}\|\boldsymbol{b}_0\|^2.$$

等价地说，$\boldsymbol{b}_1$ 已通过最近整数舍入关于 $\boldsymbol{b}_0$ 进行了大小约化，且 $\boldsymbol{b}_0$ 不长于 $\boldsymbol{b}_1$。

Algorithm 13 Dim2ReducedBasis(L)

Input: 秩二格 $L = (\boldsymbol{a}, \boldsymbol{b}) \subseteq \mathbb{Z}^2$。
Output: L 的约化基。

if $\|\boldsymbol{a}\|^2 < \|\boldsymbol{b}\|^2$ **then**
  交换 $\boldsymbol{a}$ 与 $\boldsymbol{b}$
while true **do**
  $r \leftarrow \mathrm{round}((\boldsymbol{a} \cdot \boldsymbol{b})/\|\boldsymbol{b}\|^2)$ ▷ 最近整数舍入
  $\boldsymbol{t} \leftarrow \boldsymbol{a} - r\boldsymbol{b}$
  if $\|\boldsymbol{t}\|^2 < \|\boldsymbol{b}\|^2$ **then**
    $\boldsymbol{a} \leftarrow \boldsymbol{b}$ 且 $\boldsymbol{b} \leftarrow \boldsymbol{t}$
  else
    break
if $\|\boldsymbol{t}\|^2 < \|\boldsymbol{a}\|^2$ **then**
  $\boldsymbol{a} \leftarrow \boldsymbol{t}$
return $(\boldsymbol{b}, \boldsymbol{a})$

Algorithm 14 QlapotiShortCongruence(A, B, M, N)

Input: 整数 A, B, M, N，使得 A 与 B 中至少有一个模 N 可逆。
Output: 满足 $As + Bt \equiv M \pmod N$ 的短数对 $(s, t) \in \mathbb{Z}^2$。

swapped $\leftarrow$ false
if $\gcd(A, N) \neq 1$ **then**
  交换 A 与 B；swapped $\leftarrow$ true
$u \leftarrow A^{-1} \bmod N$
$x \leftarrow Bu \bmod N$ 且 $T \leftarrow Mu \bmod N$
$L \leftarrow \langle (N - x, 1), (N, 0) \rangle \subseteq \mathbb{Z}^2$ ▷ 秩二整数格
$R \leftarrow$ Dim2ReducedBasis(L)
$D \leftarrow \det(R)$ 且 $R_{\mathrm{inv}} \leftarrow D \cdot R^{-1}$ ▷ R^{-1} 的整数分子
$\boldsymbol{v} \leftarrow (-T, 0)$
$(c_1, c_2) \leftarrow R_{\mathrm{inv}}\boldsymbol{v}$
$c_i \leftarrow \mathrm{round}(c_i/D)$ 对 $i = 1, 2$ ▷ 最近整数舍入
$\boldsymbol{c} \leftarrow (c_1, c_2)$
$\boldsymbol{w} \leftarrow R\boldsymbol{c}$
$(s, t) \leftarrow \boldsymbol{w} - \boldsymbol{v}$
if swapped **then**
  交换 s 与 t
return (s, t)

找到 s 和 t 之后，代入 $a_2 = s - a_1$ 和 $b_2 = t - b_1$，将方程除以 N 并乘以 2，则 Eq. (3.7)

化为标准的平方和问题：

$$(2a_1 - s)^2 + (2b_1 - t)^2 = \frac{2(2^k - 2r - 2a_\alpha s - 2b_\alpha t)}{N} - s^2 - t^2,$$

该问题可用 Cornacchia 算法求解。解出 a_1, b_1，再计算 a_2, b_2 即得 β_1, β_2。上述流程在 Qlapoti 中详细描述。该算法并非总是成功，详细的失败分析见 Chapter 9。

Algorithm 15 QLAPOTI(J, k)

Input: 左 $\mathcal{O}_0$-理想 J，整数 k。
Output: 满足以下条件的元素 $\beta_1, \beta_2 \in J$：
$\operatorname{nrd}(\beta_1) + \operatorname{nrd}(\beta_2) = 2^k \operatorname{nrd}(J)$ 且 $\gcd(\operatorname{nrd}(\beta_1)/\operatorname{nrd}(J), \operatorname{nrd}(\beta_2)/\operatorname{nrd}(J)) = 1$。
或引发异常。

$I \leftarrow$ RANDOMEQUIVALENTPRIMEIDEAL(J)
$N_I \leftarrow \operatorname{nrd}(I)$
$\texttt{counter} \leftarrow 0, \quad \texttt{bound} \leftarrow \left\lceil 0.607927 N_I \cdot \frac{2^k}{4p} \right\rceil$
while $\texttt{counter} < \texttt{bound}$ **do**
  $\texttt{counter} \leftarrow \texttt{counter} + 1$
  $\alpha \leftarrow$ IDEALGENERATOR(I)
  $a_\alpha, b_\alpha, c_\alpha, d_\alpha \leftarrow \alpha$ 在 $1, \mathbf{i}, \mathbf{j}, \mathbf{k}$ 中的坐标。
  $r \leftarrow \operatorname{nrd}(\alpha)/N_I$
  if $\gcd(2a_\alpha, N_I) \neq 1$ 且 $\gcd(2b_\alpha, N_I) \neq 1$ **then**
    continue
  $(s, t) \leftarrow$ QLAPOTISHORTCONGRUENCE$(2a_\alpha, 2b_\alpha, 2^k - 2r, N_I)$
  $z \leftarrow 2(2^k - 2r - 2a_\alpha s - 2b_\alpha t)/N_I - s^2 - t^2$
  if $z < 0$ **then**
    continue
  if $z \equiv 0 \pmod 4$ 且不是 $s = t = 0 \pmod 2$ **then**
    continue
  if $z \equiv 1 \pmod 4$ 且不是 $s \neq t \pmod 2$ **then**
    continue
  if $z \equiv 2 \pmod 4$ 且不是 $s = t = 1 \pmod 2$ **then**
    continue
  if $z \equiv 3 \pmod 4$ **then**
    continue
  $\texttt{sol} \leftarrow$ CORNACCHIAGENERAL(z)
  if $\texttt{sol} = \perp$ **then**
    continue
  $z_0, z_1 \leftarrow \texttt{sol}$
  if $z_0 \not\equiv s \pmod 2$ **then**
    交换 z_0, z_1
  $a_1 \leftarrow (z_0 + s)/2, \quad b_1 \leftarrow (z_1 + t)/2$
  $a_2 \leftarrow s - a_1, \quad b_2 \leftarrow t - b_1$
  $\beta_1 \leftarrow N_I(a_1 + b_1\mathbf{i}) + \alpha, \quad \beta_2 \leftarrow N_I(a_2 + b_2\mathbf{i}) + \alpha$
  return β_1, β_2
raise Exception("Qlapoti 失败")

Remark 3.3.1 (替代表示与实现失败)**.** *Cornacchia* 方程的解 $z = z_0^2 + z_1^2$ 中，独立改变

z_0 和 z_1 的符号可得到更多解。在上述回代下，这些符号变化将 a_1 与 a_2 交换，和/或将 b_1 与 b_2 交换。因此，同一个 *Cornacchia* 解给出四个有序表示：

$$(a_1, b_1; a_2, b_2), \qquad (a_1, b_2; a_2, b_1), \qquad (a_2, b_1; a_1, b_2), \qquad (a_2, b_2; a_1, b_1).$$

对应的四元数元素对为

$$\begin{aligned}
(\beta_1^{(0)}, \beta_2^{(0)}) &= (N_I(a_1 + b_1\mathbf{i}) + \alpha,\ N_I(a_2 + b_2\mathbf{i}) + \alpha),\\
(\beta_1^{(1)}, \beta_2^{(1)}) &= (N_I(a_1 + b_2\mathbf{i}) + \alpha,\ N_I(a_2 + b_1\mathbf{i}) + \alpha),\\
(\beta_1^{(2)}, \beta_2^{(2)}) &= (N_I(a_2 + b_1\mathbf{i}) + \alpha,\ N_I(a_1 + b_2\mathbf{i}) + \alpha),\\
(\beta_1^{(3)}, \beta_2^{(3)}) &= (N_I(a_2 + b_2\mathbf{i}) + \alpha,\ N_I(a_1 + b_1\mathbf{i}) + \alpha).
\end{aligned}$$

所有四对都满足输出条件中的范数方程。后续实现对 $(\beta_1^{(v)}, \beta_2^{(v)}), v \in \{0, 1, 2, 3\}$ 施加了额外的条件。因此，一个在抽象 *Qlapoti* 算法中有效的输出数对，仍可能被后续子程序拒绝。*QIMEN-PRISM* 利用 *Cornacchia* 子程序中的多个解来减少循环次数——在密钥生成和签名中，根据安全级别可提升约 *5%* 至 *9%* 的速度。

3.3.4 理想到同源算法

IdealToIso 是本节的主要算法。给定左 $\mathcal{O}_0$-理想 J，它计算对应同源 φ_J 的像曲线 E_J，以及 φ_J 在下述固定挠基上的赋值。

预计算数据。 该算法假设 $p = f \cdot 2^e - 1$（f 为小整数），并取满足 $a \le e$ 的正整数 a。回顾 Section 3.3.1，$\left(1, \mathbf{i}, \frac{1+\mathbf{ij}}{2}, \frac{1+\mathbf{k}}{2}\right)$ 是 $\mathcal{O}_0$ 的固定整基。在 QIMEN-PRISM 中，IdealToIso 使用以下预计算数据：

- $E_0[2^{a+2}]$ 的基 (P_0, Q_0)，以及矩阵 $M_1, \ldots, M_4 \in \mathsf{M}_2(\mathbb{Z}/2^{a+2}\mathbb{Z})$，表示这四个基元素关于 (P_0, Q_0) 的作用；
- $E_0[2^e]$ 的基 (P_0^e, Q_0^e)，以及矩阵 $M_1^e, \ldots, M_4^e \in \mathsf{M}_2(\mathbb{Z}/2^e\mathbb{Z})$，表示关于 (P_0^e, Q_0^e) 的相同作用。

算法概述。 算法首先使用 Qlapoti 找到理想 $I_1, I_2 \sim J$，其约化范数 d_1, d_2 互素且满足 $d_1 + d_2 = 2^{e-2}$。将 I_i 写为 $I_i = J\frac{\overline{\beta_i}}{\mathrm{nrd}(J)}$（$i = 1, 2$，$\beta_i \in J$），则 $\widehat{\varphi_{I_i}} \circ \varphi_J = \beta_i$（$i = 1, 2$）。因此，自同态 $\widehat{\varphi_{I_2}} \circ \varphi_{I_1} = \frac{\beta_2\overline{\beta_1}}{\mathrm{nrd}(J)} \in \mathcal{O}_0$。上述同源构成如下椭圆曲线之间的交换图：

$$\begin{array}{ccc}
E_0 & \xrightarrow{\varphi_{I_1}} & E_J \\
{\scriptstyle [\varphi_{I_1}]^*\widehat{\varphi_{I_2}}}\downarrow & & \downarrow{\scriptstyle \widehat{\varphi_{I_2}}} \\
E' & \xrightarrow[[\varphi_{I_1}]^*\widehat{\varphi_{I_2}}]{} & E_0.
\end{array}$$

Algorithm 16 IdealToIso(J)

Input: 左 $\mathcal{O}_0$-理想 J。
Output: 同源 φ_J 的像曲线 E_J，以及 $\varphi_J(P_0), \varphi_J(Q_0)$。

$\beta_1, \beta_2 \leftarrow$ Qlapoti$(J, e-2)$
$d_1 \leftarrow \mathrm{nrd}(\beta_1)/\mathrm{nrd}(J)$
$c_1, c_2, c_3, c_4 \leftarrow \frac{\beta_2\overline{\beta_1}}{\mathrm{nrd}(J)}$ 在 $1, \mathbf{i}, \frac{\mathbf{i}+\mathbf{j}}{2}, \frac{1+\mathbf{k}}{2}$ 中的坐标
$M_\theta \leftarrow c_1M_1^e + c_2M_2^e + c_3M_3^e + c_4M_4^e$ ▷ $\theta = \frac{\beta_2\overline{\beta_1}}{\mathrm{nrd}(J)}$
$(\widetilde{P_0^e}, \widetilde{Q_0^e})^T \leftarrow M_\theta(P_0^e, Q_0^e)^T$
$K_P \leftarrow ([d_1]P_0^e, \widetilde{P_0^e})$
$K_Q \leftarrow ([d_1]Q_0^e, \widetilde{Q_0^e})$
$E_1 \times E_2,\ [(P, P'), (Q, Q')] \leftarrow$ Isogeny22Chain$(K_P, K_Q, [(P_0, 0_{E_0}), (Q_0, 0_{E_0})])$
if $e_{2^{a+2}}(P, Q) = e_{2^{a+2}}(P_0, Q_0)^{d_1}$ **then**
    $E_J \leftarrow E_1, P_J \leftarrow P, Q_J \leftarrow Q$
else
    $E_J \leftarrow E_2, P_J \leftarrow P', Q_J \leftarrow Q'$
$c_1, c_2, c_3, c_4 \leftarrow \beta_1$ 在 $1, \mathbf{i}, \frac{\mathbf{i}+\mathbf{j}}{2}, \frac{1+\mathbf{k}}{2}$ 中的坐标
$M_{\beta_1} \leftarrow c_1M_1 + c_2M_2 + c_3M_3 + c_4M_4$
$(P_J, Q_J)^T \leftarrow [d_1^{-1} \bmod 2^{a+2}]M_{\beta_1}(P_J, Q_J)^T$
return E_J, P_J, Q_J

由

$$\Phi : E_0 \times E_0 \to E_J \times E'$$

定义的同源 $\begin{pmatrix} \varphi_{I_1} & \widehat{\varphi_{I_2}} \\ -[[\varphi_{I_1}]^*\widehat{\varphi_{I_2}}]_*\varphi_{I_1} & [\widehat{\varphi_{I_2}}]_*\varphi_{I_1} \end{pmatrix}$ 是一个 $(2^{e-2}, 2^{e-2})$-同源，其核为

$$\{([d_1]P, \widehat{\varphi_{I_2}} \circ \varphi_{I_1}(P)) \mid P \in E_0[2^{e-2}]\}.$$

为得到同源 φ_J 在 P_0, Q_0 上的赋值，算法首先计算 Φ，从中提取 φ_{I_1} 在 P_0, Q_0 上的赋值。最后，由于 $\widehat{\varphi_{I_i}} \circ \varphi_J = \beta_1$，有 $\varphi_J(P_0, Q_0) = [d_1^{-1} \bmod 2^{a+2}]\varphi_{I_1} \circ \beta_i(P_0, Q_0)$。

CHAPTER 4

协议

4.1 协议参数

本节基于 [BBC+26, §6] 描述 QIMEN-PRISM。

4.1.1 参数需求

下文用到的主要参数列举如下。方案运行在有限域 $\mathbb{F}_{p^2}$ 上，其中 $p = f \cdot 2^e - 1$。固定以下参数。

- a 是整数，$0 < a < e$。其确切值在 Chapter 5 中定义。
- E_0 是方程为 $y^2 = x^3 + x$ 的曲线。
- (P_0, Q_0) 是 $E_0[2^{a+2}]$ 的基。
- N_{sk} 是大于 $2^{4\lambda_c}$ 的最小素数。
- n_{salt} 表示 Section 4.1.2 中使用的 salt（盐值）的位长度。
- H_{a-2} 是一个哈希函数，将输入映射到正整数 $< 2^{a-2}$。方案中使用的具体哈希函数在 Chapter 5 中描述。
- `FP_ENCODED_BYTES` 是能够表示 $\mathbb{F}_p$ 中所有元素的最小字节数；定义为 $8\lfloor(\log_2(p) + 63)/64\rfloor$。

辅助算法参数固定如下。

- `QUAT_equiv_bound_coeff` 是 RandomEquivalentPrimeIdeal 中使用的系数界；对三组参数集均为 64。
- $\mathrm{MR}_{\mathrm{iter}}$ 是伪素性测试中使用的 Miller–Rabin 迭代轮数；对于 NGCC-1、NGCC-2 和 NGCC-3，分别为 28、63 和 144。
- `cornacchia_prime_list` 是 CornacchiaGeneral 使用的固定列表；由 2 和前 100 个满足 $\ell \equiv 1 \pmod 4$ 的奇素数组成，对三组参数集均相同。

为便于阅读，所涉算法省略这些辅助量作为显式输入，其用法详见 Chapters 2 and 3。

Algorithm 17 QIMEN-PRISM.KeyGen(λ_c, λ_q)

Input: 目标经典安全参数 λ_c 和量子安全参数 λ_q。
Output: 私钥 sk 和公钥 pk。

while true **do**
  $I_{\mathsf{sk}} \leftarrow \text{RandFixNormIdeal}(N_{\mathsf{sk}}, \text{True})$
  try
    $I_{\mathsf{sk}} \leftarrow \text{RandomEquivalentPrimeIdeal}(I_{\mathsf{sk}})$
    $E_{\mathsf{pk}}, \varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0) \leftarrow \text{IdealToIso}(I_{\mathsf{sk}})$
  catch
    continue
  $P_{\mathsf{pk}}, Q_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}} \leftarrow \text{TorsionBasisToHint}(E_{\mathsf{pk}}, a+2)$
  $M_{\mathsf{sk}} \leftarrow \text{ChangeOfBasis}_{2^{a+2}}(E_{\mathsf{pk}}, (\varphi_{\mathsf{sk}}(P_0), \varphi_{\mathsf{sk}}(Q_0)), (P_{\mathsf{pk}}, Q_{\mathsf{pk}}))$
  $\mathsf{pk} \leftarrow (E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}})$
  $\mathsf{sk} \leftarrow (E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}}, I_{\mathsf{sk}}, M_{\mathsf{sk}})$
  return sk, pk

4.1.2 哈希函数

参照 [BBC+25, BBC+26] 的构造，需要一个哈希函数 $\mathsf{H}_{\mathsf{PRISM}}$，它将输入映射到长度恰好为 $a-1$ 位的奇整数集合。哈希函数 $\mathsf{H}_{\mathsf{PRISM}} : \mathbb{F}_{p^2} \times \{0,1\}^* \times \{0,1\}^{n_{\mathsf{salt}}} \to \{2^{a-1} + 2x + 1 | 0 \le x < 2^{a-2}\}$ 接受三个输入：一条超奇异曲线的 j-不变量、一条消息 msg 和一个 $\mathsf{salt} \in \{0,1\}^{n_{\mathsf{salt}}}$；返回一个长度为 $a-1$ 位的奇整数。具体而言，$\mathsf{H}_{\mathsf{PRISM}}(j(E), \mathsf{msg}, \mathsf{salt})$ 先计算 $h \leftarrow \mathsf{H}_{a-2}(j(E) \| \mathsf{msg} \| \mathsf{salt})$，再返回 $q = 2^{a-1} + 2h + 1$。

4.2 密钥生成

密钥生成过程在 QIMEN-PRISM.KeyGen中 描述。首先，使用 RandFixNormIdeal 采样一个素数范数 N_{sk} 的随机理想 I_{sk}。其次，使用 IdealToIso 计算同源 $\phi_{\mathsf{sk}} : E_0 \to E_{\mathsf{pk}}$ 的像曲线——该同源由 RandFixNormIdeal 对应——以及 $\phi_{\mathsf{sk}}(P_0)$ 和 $\phi_{\mathsf{sk}}(Q_0)$。然后，使用 TorsionBasisToHint 确定性地计算 $E_{\mathsf{pk}}(\mathbb{F}_{p^2})[2^{a+2}]$ 的生成元 $P_{\mathsf{pk}}, Q_{\mathsf{pk}}$ 以及一个 hint $\mathsf{hint}_{\mathsf{pk}}$。最后，计算将 $(\phi_{\mathsf{sk}}(P_0), \phi_{\mathsf{sk}}(Q_0))$ 映为 $(P_{\mathsf{pk}}, Q_{\mathsf{pk}})$ 且满足 mod 2^{a+2} 同余的矩阵 M_{sk}：

$$M_{\mathsf{sk}} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$$

使得 $P_{\mathsf{pk}} = [m_{11}]\phi_{\mathsf{sk}}(P_0) + [m_{12}]\phi_{\mathsf{sk}}(Q_0)$ 且 $Q_{\mathsf{pk}} = [m_{21}]\phi_{\mathsf{sk}}(P_0) + [m_{22}]\phi_{\mathsf{sk}}(Q_0)$。该矩阵使用 $\text{ChangeOfBasis}_{2^{a+2}}$ 计算。公钥设为 $(E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}})$，私钥为 $(E_{\mathsf{pk}}, I_{\mathsf{sk}}, M_{\mathsf{sk}})$。

Algorithm 18 QIMEN-PRISM.GenIso(sk, q)

Input: 私钥 $\mathsf{sk} = (E_{\mathsf{pk}}, I_{\mathsf{sk}}, M_{\mathsf{sk}})$ 和素数 $q \in (2^{a-1}, 2^a)$。
Output: 签名曲线 E_{sig} 以及点 $(P_{\mathrm{sig}}, Q_{\mathrm{sig}})$。

$N \leftarrow q(2^a - q)$
$I_{\mathrm{chall}} \leftarrow \textsc{RandFixNormIdeal}\,(N, \mathsf{False})$
$I \leftarrow \textsc{IdealIntersection}(I_{\mathsf{sk}}, I_{\mathrm{chall}})$
$E_{\mathsf{sig}}, \varphi(P_0), \varphi(Q_0) \leftarrow \textsc{IdealToIso}(I)$
$t \leftarrow q^{-1} \mod 2^{a+2}$
$P_{\mathrm{sig}} \leftarrow [t \cdot m_{11}]\varphi(P_0) + [t \cdot m_{12}]\varphi(Q_0)$
$Q_{\mathrm{sig}} \leftarrow [t \cdot m_{21}]\varphi(P_0) + [t \cdot m_{22}]\varphi(Q_0)$
return $(E_{\mathsf{sig}}, P_{\mathrm{sig}}, Q_{\mathrm{sig}})$

Algorithm 19 PrimeGenerator(E, msg)

Input: 曲线 E 和消息 msg。
Output: 整数 q 和 salt。

while True **do**
  $\mathsf{salt} \xleftarrow{\$} \{0,1\}^{n_{\mathsf{salt}}}$
  $q \leftarrow \mathsf{H}_{\mathsf{PRISM}}(E, \mathsf{msg}, \mathsf{salt})$
  if PrimalityTest(q) **then**
    Break
return (q, salt)

4.3 签名

签名过程在 QIMEN-PRISM.Sign中 描述。签名步骤使用 Section 4.1.2 的哈希函数，其同源生成过程由 QIMEN-PRISM.GenIso给 出。将 pk 解析为 E_{pk}，sk 解析为 $(E_{\mathsf{pk}}, I_{\mathsf{sk}}, M_{\mathsf{sk}})$。该算法的目标是：对次数为 $q(2^a - q)$ 的同源 $\sigma : E_{\mathsf{pk}} \to E_{\mathsf{sig}}$，输出其像曲线 E_{sig}，以及 $([q^{-1} \mod 2^{a+2}]\sigma(P_{\mathsf{pk}}), [q^{-1} \mod 2^{a+2}]\sigma(Q_{\mathsf{pk}}))$。

第一步是生成一个范数为 $q(2^a - q)$ 的 $\mathcal{O}_0$-理想 I_{chall}：先由 PrimeGenerator得 到素数 q，再通过 RandFixNormIdeal 算法完成。算法 PrimeGenerator接 受曲线 E 和消息 msg 作为输入。它从 $\{0,1\}^{n_{\mathsf{salt}}}$ 中均匀随机采样一个 salt，对 $(E, \mathsf{msg}, \mathsf{salt})$ 应用 $\mathsf{H}_{\mathsf{PRISM}}$ 得到 q，再对 q 运行 PrimalityTest。最后输出 (q, salt)。同源 $\varphi = \sigma \circ \phi_{\mathsf{sk}} : E_0 \to E_{\mathsf{sig}}$ 对应的理想由 $I = I_{\mathsf{sk}} \cap I_{\mathrm{chall}}$ 给出，使用 IdealToIso 可获得 φ 的表示并计算 $\varphi(P_0), \varphi(Q_0)$。将 E_{sig} 设为 φ 的像曲线，并令 $t = q^{-1} \mod 2^{a+2}$，最终得到

$$P_{\mathrm{sig}} = [t \cdot m_{11}]\varphi(P_0) + [t \cdot m_{12}]\varphi(Q_0), \quad Q_{\mathrm{sig}} = [t \cdot m_{21}]\varphi(P_0) + [t \cdot m_{22}]\varphi(Q_0).$$

Algorithm 20 QIMEN-PRISM.Sign($\mathsf{sk}, \mathsf{msg}$)

Input: 私钥 $\mathsf{sk} = (E_{\mathsf{pk}}, I_{\mathsf{sk}}, M_{\mathsf{sk}})$ 和消息 msg。
Output: 签名曲线 E_{sig} 和点 $(P_{\mathrm{sig}}, Q_{\mathrm{sig}})$，以及 salt salt。

$(q, \mathsf{salt}) \leftarrow \textsc{PrimeGenerator}(E_{\mathsf{pk}}, \mathsf{msg})$
$E_{\mathsf{sig}}, P_{\mathrm{sig}}, Q_{\mathrm{sig}} \leftarrow \text{QIMEN-PRISM.}\textsc{GenIso}(\mathsf{sk}, \mathsf{pk}, q)$
$P'_{\mathrm{sig}}, Q'_{\mathrm{sig}}, \mathsf{hint}_{\mathrm{sig}} \leftarrow \textsc{TorsionBasisToHint}(E_{\mathsf{sig}}, a+2)$
$M_{\mathrm{sig}} \leftarrow \textsc{ChangeOfBasis}_{2^{a+2}}(E_{\mathsf{sig}}, (P'_{\mathrm{sig}}, Q'_{\mathrm{sig}}), (P_{\mathrm{sig}}, Q_{\mathrm{sig}}))$
$\sigma \leftarrow (E_{\mathsf{sig}}, M_{\mathrm{sig}}, \mathsf{hint}_{\mathrm{sig}})$
return (σ, salt)

Algorithm 21 QIMEN-PRISM.Verif($\mathsf{pk}, \mathsf{msg}, \sigma, \mathsf{salt}$)

Input: 公钥 $\mathsf{pk} = (E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}})$、消息 msg 以及签名 (σ, salt)。
Output: accept 或 reject。

$q \leftarrow \mathsf{H}_{\mathsf{PRISM}}(E_{\mathsf{pk}}, \mathsf{msg}, \mathsf{salt})$
if $\mathsf{PrimalityTest}(q) = \mathsf{False}$ **then**
    return reject
$(P_{\mathsf{pk}}, Q_{\mathsf{pk}}) \leftarrow \textsc{TorsionBasisFromHint}(E_{\mathsf{pk}}, \mathsf{hint}_{\mathsf{pk}}, a+2)$
将 σ 解析为 $E_{\mathsf{sig}}, M_{\mathrm{sig}}, \mathsf{hint}_{\mathrm{sig}}$
$P_{\mathrm{sig}}, Q_{\mathrm{sig}} \leftarrow \textsc{TorsionBasisFromHint}(E_{\mathsf{sig}}, \mathsf{hint}_{\mathrm{sig}}, a+2)$
$P_{\mathrm{sig}}, Q_{\mathrm{sig}} \leftarrow M_{\mathrm{sig}} \cdot (P_{\mathrm{sig}}, Q_{\mathrm{sig}})$
if $\textsc{CheckIsogeny}(E_{\mathsf{pk}}, P_{\mathsf{pk}}, Q_{\mathsf{pk}}, E_{\mathsf{sig}}, P_{\mathrm{sig}}, Q_{\mathrm{sig}}, a, q)$ **then**
    return accept
else
    return reject

4.4 验证

现定义 QIMEN-PRISM.Verif中描述的验证算法。将输入的插值同源 iso 解析为 $(E_{\mathsf{sig}}, P_{\mathrm{sig}}, Q_{\mathrm{sig}})$，其中 $P_{\mathrm{sig}} = [q^{-1} \bmod 2^{a+2}]\,\sigma(P)$ 且 $Q_{\mathrm{sig}} = [q^{-1} \bmod 2^{a+2}]\,\sigma(Q)$。回顾 $E_{\mathsf{pk}}[2^{a+2}] = \langle P_{\mathsf{pk}}, Q_{\mathsf{pk}} \rangle$。首先，以 $(E_{\mathsf{pk}}, \mathsf{msg}, \mathsf{salt})$ 调用哈希函数 $\mathsf{H}_{\mathsf{PRISM}}$，并对输出 q 运行 $\mathsf{PrimalityTest}$。然后，需要检查这些点是否能插值出一个同源 $\sigma : E_{\mathsf{pk}} \to E_{\mathsf{sig}}$，其次数为 $q(2^a - q)$。为此使用函数 $\textsc{CheckIsogeny}(E_{\mathsf{pk}}, P_{\mathsf{pk}}, Q_{\mathsf{pk}}, E_{\mathsf{sig}}, P_{\mathrm{sig}}, Q_{\mathrm{sig}}, a, q)$；当且仅当存在同源 $\varphi : E_{\mathsf{pk}} \to E_{\mathsf{sig}}$，其次数为 $q(2^a - q)$，且满足 $[q]P_{\mathrm{sig}} = \varphi(P_{\mathsf{pk}})$ 和 $[q]Q_{\mathrm{sig}} = \varphi(Q_{\mathsf{pk}})$ 时，该函数返回 true。

4.5 二进制格式

签名方案涉及若干数学对象，本节规定其字节编码方式以供网络上传输。含以下类型的组件对象。

- $\mathbb{F}_p$ 中的元素编码为 0 到 $p-1$ 之间的无符号整数，采用小端序，使用 `FP_ENCODED_BYTES` 字节。
- $\mathbb{F}_{p^2}$ 中的元素编码为实部编码与虚部编码的拼接，两部分均按 $\mathbb{F}_p$ 元素编码。
- $\mathbb{Z}$ 中的整数采用小端序二进制补码表示编码；字节数由各实例的技术规范确定。
- 椭圆曲线以其 Montgomery 系数 $A \in \mathbb{F}_{p^2}$ 编码。
- 理想 I 编码为以下两者的拼接：$\mathrm{nrd}(I)$ 的编码——一个 `FP_ENCODED_BYTES`-字节整数——以及 IdealGenerator(I) 输出的四元数编码。其中四元数编码是四个 `FP_ENCODED_BYTES`-字节整数的拼接，分别表示 $1, i, j, k$-基中的整数系数。
- 2×2 整数矩阵 $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ 编码为 a、b、c 和 d 的编码的拼接。
- hint 由两个值 (h_A, h) 组成；h_A 为 0 或 1，h 为 7-bit 整数。hint 编码为一个字节，其中最低有效位表示标志 h_A。
- salt salt 长度为 n_{salt} 位，编码为 $n_{\mathsf{salt}}/8$ 字节。

公钥 公钥编码为 QIMEN-PRISM.KeyGen第 5 步中 E_{pk} 的编码与 QIMEN-PRISM.KeyGen第 8 步中 hint $\mathsf{hint}_{\mathsf{pk}}$ 的编码的拼接。

私钥 私钥编码为以下对象按顺序的编码拼接：

- 公钥 pk 的编码；
- QIMEN-PRISM.KeyGen第 4 步中的理想 I_{sk}；
- QIMEN-PRISM.KeyGen第 9 步中的 2×2 矩阵 M_{sk}。

签名 签名编码为以下对象按顺序的编码拼接：

- QIMEN-PRISM.Sign第 2 步中 E_{sig} 的 Montgomery 系数；
- QIMEN-PRISM.Sign第 4 步中的 2×2 矩阵 M_{sig}；
- QIMEN-PRISM.Sign第 3 步中的 hint $\mathsf{hint}_{\mathrm{sig}}$；
- QIMEN-PRISM.Sign第 1 步中的 salt salt。

CHAPTER 5

参数集

NGCC-1.	$p = 69 \cdot 2^{313} - 1,$	$a = 224,$	$n_{\mathsf{salt}} = 232,$	$\mathrm{MR}_{\mathrm{iter}} = 28,$
NGCC-2.	$p = 27 \cdot 2^{500} - 1,$	$a = 320,$	$n_{\mathsf{salt}} = 336,$	$\mathrm{MR}_{\mathrm{iter}} = 63,$
NGCC-3.	$p = 15 \cdot 2^{1004} - 1,$	$a = 576,$	$n_{\mathsf{salt}} = 592,$	$\mathrm{MR}_{\mathrm{iter}} = 144.$

本技术规范为辅助哈希函数定义两种获批的可扩展输出函数实例化

$$\mathsf{H}_{a-2} : \{0,1\}^* \to [0, 2^{a-2})$$

：SHAKE256 实例化和 pseudoXOF 实例化，其中 pseudoXOF 是基于 SM3 的函数，NGCC 提供了具体定义（GB/T 32918.4-2016，第 5.4.3 节）。根据提交规定，pseudoXOF 仅用于审查，不构成安全的 XOF。

两种实例化均应使用域分离前缀 HPRISM。

SHAKE256 实例化。

$$\mathsf{H}_{a-2}(x) = \mathrm{trunc}_{a-2}\big(\mathsf{SHAKE256}(\mathsf{HPRISM} \| x)\big),$$

其中 trunc_{a-2} 表示截断到前 $a-2$ 个输出位，解释为 $[0, 2^{a-2})$ 中的非负整数。Section 4.1.2 的哈希函数由此实例化为

$$\mathsf{H}_{\mathsf{PRISM}}(j(E), \mathsf{msg}, \mathsf{salt}) = 2^{a-1} + 2\mathsf{H}_{a-2}\big(j(E) \| \mathsf{msg} \| \mathsf{salt}\big) + 1.$$

pseudoXOF 实例化。

$$\mathsf{H}_{a-2}(x) = \mathrm{trunc}_{a-2}\big(\mathsf{pseudoXOF}(\mathsf{HPRISM} \| x)\big),$$

其中 trunc_{a-2} 表示截断到前 $a-2$ 个输出位，解释为 $[0, 2^{a-2})$ 中的非负整数。Section 4.1.2 的哈希函数由此实例化为

$$\mathsf{H}_{\mathsf{PRISM}}(j(E), \mathsf{msg}, \mathsf{salt}) = 2^{a-1} + 2\mathsf{H}_{a-2}\big(j(E) \| \mathsf{msg} \| \mathsf{salt}\big) + 1.$$

实现者应选用其中一种并贯彻始终。从同一 XOF 派生其他函数时，须使用不同的域分离前缀。

CHAPTER 6

测试向量

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

对于 QIMEN-PRISM，文件为 `KAT_SIG_NGCC-1.txt`、`KAT_SIG_NGCC-2.txt` 和 `KAT_SIG_NGCC-3.txt`；每条记录包含一个确定性种子、一个公钥、一个私钥、一条消息以及生成的签名。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 PRISM-salt 实现。[1] 该代码库构建于 SQIsign 库 [AAA+25] 之上。

SQIsign 库中的若干子模块提供了底层构建块，在 QIMEN-PRISM 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。
- `id2iso`：对由理想产生的二维同源进行求值的模块。
- `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_SIGN=ON` 构建签名实现。
- `PRISM_NGCC_XOF_BACKEND` 选择 NGCC XOF 后端；支持的值为 `shake` 和 `sm3`。
- `ENABLE_NGCC_PRISM=ON` 构建 NGCC KAT 生成目标。KAT 输出目录由 `NGCC_PRISM_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 官方示例代码提供了一个基于 SM3 的伪 XOF 构造。

7.1 密钥和签名尺寸

Table 7.1 列出了每个参数集的公钥、私钥和 签名 尺寸。

[1] https://github.com/mariascrs/PRISM_v2

Table 7.1: QIMEN-PRISM 各安全级别的密钥和签名尺寸（字节）。

参数集	公钥	私钥	签名
NGCC-1	81	441	225
NGCC-2	129	701	334
NGCC-3	257	1401	622

7.2 性能评估

本节报告 QIMEN-PRISM 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 `PRISM_NGCC_XOF_BACKEND=sm3` 使用 SM3 后端，该配置将 `PRISM_XOF_BACKEND` 设为 2。
- **编译设置：**有效的 C 编译标志包括 `-O3`、`-DNDEBUG`、`-std=c11`、`-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：** 每个报告的签名 操作均为 300 次运行的平均值。

7.2.1 参考实现

Table 7.2 给出了当前使用 SM3 的参考实现性能数据。

Table 7.2: 参考实现 QIMEN-PRISM 签名性能，使用 SM3 作为 XOF，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	KeyGen	Sign	Verify
NGCC-1	66.95	78.14	14.23
NGCC-2	129.66	166.12	41.33
NGCC-3	684.67	969.23	231.23

Table 7.3: 参考实现 QIMEN-PRISM 签名在使用 SM3 作为 XOF 时的吞吐量，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	KeyGen	Sign	Verify
NGCC-1	40.33	34.55	189.74
NGCC-2	20.82	16.25	65.33
NGCC-3	3.94	2.79	11.68

7.2.2 优化实现

优化构建使用了额外标志

```
PRISM_BUILD_TYPE=broadwell
```

为 NGCC-1 和 NGCC-2 启用 x86-64 BMI2/ADX 汇编有限域运算，而 NGCC-3 使用相同的可移植 C 核心。Table 7.4 给出了汇编优化实现的性能数据。目前 NGCC-3 尚无汇编优化。

Table 7.4: 汇编优化 QIMEN-PRISM 签名性能，使用 SM3 作为 XOF，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	KeyGen	Sign	Verify
NGCC-1	57.32	66.27	9.71
NGCC-2	90.24	122.52	24.57

Table 7.5: 优化实现 QIMEN-PRISM 签名在使用 SM3 作为 XOF 时的吞吐量，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	KeyGen	Sign	Verify
NGCC-1	47.10	40.74	278.06
NGCC-2	29.92	22.04	109.89

CHAPTER 8

安全性

本章讨论 QIMEN-PRISM 数字签名的主要安全方面，遵循对 salt-PRISM [BBC+26] 所做的分析。首先，在 Section 8.1 中讨论 QIMEN-PRISM 基础中的数学困难性假设。Section 8.2 给出一个理论结果：在量子随机预言机模型（QROM）下，该协议的安全性可基于这些困难性假设得到证明。然后，在 Section 8.3 中，进一步讨论针对方案更专门的攻击及其各自复杂度。包括密钥恢复、针对哈希函数的直接攻击，以及对不可重签名性（non-resignability）的简要讨论。

8.1 困难问题

8.1.1 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-PRISM，因此其困难性是攻破本方案所需代价的一个上界。如 Section 3.3 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。

8.1.2 大素数次数同源问题

QIMEN-PRISM 的安全性依赖以下问题的困难性：给定满足 $\#E(\mathbb{F}_{p^2}) = (p+1)^2$ 的超奇异椭圆曲线 $E/\mathbb{F}_{p^2}$，计算一个指定的大次数 $q \neq p$ 的同源（的高效表示）。该问题总能在 $\tilde{O}(\max\{k^2, k\sqrt{q}\})$ 时间内解决，其中 $k \mid q-1$ 是满足 $(-p)^k \equiv 1 \bmod q$ 的最小整数 [NO25, Lemma 3]。求解主要依赖 [BDLS20] 中的平方根-Vélu 算法。量 k 等于所需域扩张的次数——在该扩张中可找到 q 阶点。该问题的以下变体更准确地刻画了 EUF-CMA 攻击模型，其中攻击者在发起攻击前可以访问任意多个大素数次数同源：

Problem 8.1.2 (大素数次数同源问题，LPDI)**.** 设 $\{q_i\}_{i\in[N]}$ 和 $\bar{q}$ 为 $N+1$ 个素数，独立且均匀随机地从 a-bit 素数的集合 Primes_a 中采样。给定 Ell_p（$\mathbb{F}_{p^2}$ 上超奇异椭圆曲线的集合）中一个均匀随机元素 E、一组 N 个次数为 $q_i(2^a - q_i)$ 的同源 $\{\phi_i : E \to E_i\}_{i\in[N]}$（其中 ϕ_i 在次数为 $q_i(2^a-q_i)$ 的同源中均匀随机采样），以及 $\bar{q}$，计算一个次数为 $\bar{q}(2^a-\bar{q})$ 的同源（的高效表示）。

若 E 的自同态环已知，则该问题可在多项式时间内解决——这本质上就是 QIMEN-PRISM 中陷门的原理。学术界普遍相信（参见 [BBC+26, DLRW24]），即使获知从 E 出发的随机大素数次数同源，也不会降低计算其自同态环的难度。

一般情况下预期 $k \approx 2^a$，此时计算 q-同源的代价为 $\tilde{O}(2^{3a/2})$。但 k 可能异常地偏小，从而使攻击速度加快。例如，若 $k=1$，则复杂度降至 $\tilde{O}(2^{a/2})$。[NO25] 详细研究了 $k \ll 2^a$ 的概率，并将其用于一种伪造攻击——该攻击原本是针对 PRISM-id 设计的，但同样适用于 QIMEN-PRISM——其运行时间为 $\tilde{O}(2^{6a/7})$。如 [BBC+26, Remark 8] 所建议，要避免此类攻击，可以从安全素数（safe primes）的子集中采样 $q_i, \bar{q}$；安全素数是指满足 $(q-1)/2$ 也是素数的素数。安全素数很常见：其出现频率约为 $1/a^2$。因此，取 $a = n = \lambda_c + 64$（该选择是为缓解针对哈希函数的攻击而做出的）足以保证存在足够多的满足位长 a 的有效素数。不过，在现有参数范围内无需采用这一对策，因为它会延长签名时间。事实上，对于最大的经典安全级别 $\lambda_c = 512$，比值 $6(512+64)/7 \approx 493.71$ 表面上似乎不足，但这一缺口完全由 $\tilde{O}(2^{6a/7})$ 中隐藏的常数因子和对数因子所弥补——这些因子源于对约 $2^{a/2}$ 个多项式在 $\mathbb{F}_{p^2}$ 上约 $2^{2a/7}$ 次的扩张上进行结式计算。

最后，以当前技术水平来看，该问题似乎不存在显著的量子加速。

8.2 理论安全性

Definition 8.2.1. 对任意概率多项式时间敌手 $\mathcal{A}$，定义 $\mathsf{Adv}^{p,N}_{\mathsf{LDPI}}(\mathcal{A})$ 为 $\mathcal{A}$ 在解决以素数 p 和给定的 N 个同源为输入的 LDPI 实例中的优势，即

$$\mathsf{Adv}^{\mathsf{LDPI}}_{p,N}(\mathcal{A}) := \Pr\left[\begin{array}{l|l} \bar{\phi} : E \to \bar{E} & \{q_i\}_{i\in[N]}, \bar{q} \xleftarrow{\$} \mathsf{Primes}_a^{N+1}; \\ \text{是次数为} \bar{q}(2^a-\bar{q}) \text{ 的} & E \xleftarrow{\$} \mathsf{Ell}_p; \\ \text{同源} & \{\phi_i : E \to E_i\}_{i\in[N]} \text{ 均匀随机采样}, \\ & \text{使得 } \deg(\phi_i) = q_i(2^a-q_i); \\ & \bar{\phi} \leftarrow \mathcal{A}\Big(E, \{\phi_i : E \to E_i\}_{i\in[N]}, \bar{q}\Big) \end{array}\right].$$

基于 LDPI 问题的困难性假设，QIMEN-PRISM 协议满足签名的标准安全概念（EUF-CMA）。

Theorem 8.2.2 ([BBC+25, Theorem 2])**.** 设 $\mathcal{A}$ 是针对 *QIMEN-PRISM*（以素数 p）的 EUF-CMA 安全的 *PPT* 敌手，最多进行 N_{sign} 次签名查询和 N_{H} 次对随机预言机 $\mathsf{H} : \mathsf{Ell}_p \times \{0,1\}^* \times \{0,1\}^{n_{\mathsf{salt}}} \to \{0,1\}^a$ 的（量子）查询。设 $\gamma_a = \#\mathsf{Primes}_a / 2^a$。在量子随机预言机模型（*QROM*）中，存在一个针对 $N = N_{\mathsf{sign}}$ 的 *LDPI* 问题的敌手 $\mathcal{B}$ 使得

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{salt\text{-}PRISM}}(\mathcal{A}) \le (2N_{\mathsf{H}}+1)^2 \mathsf{Adv}^{\mathit{LDPI}}_{p,N_{\mathsf{sign}}}(\mathcal{B}) + \frac{N_{\mathsf{sign}}(N_{\mathsf{sign}}+N_{\mathsf{H}}+2)^2}{2^{a-3}} + 3N_{\mathsf{sign}}\sqrt{\frac{N_{\mathsf{H}}+N_{\mathsf{sign}}+1}{\gamma_a 2^{n_{\mathsf{salt}}}}}\,.$$

8.3 实际安全性

8.3.1 密钥恢复

QIMEN-PRISM 中的私钥主要由理想 I_{sk} 给出，该理想对应于次数为 N_{sk} 的同源 $\phi_{\mathsf{sk}} : E_0 \to E_{\mathsf{pk}}$。秘密矩阵 M_{sk} 也是私钥的一部分，但它直接从秘密理想 I_{sk} 计算得出，其唯一用途是高效表示 ϕ_{sk}。

该理想 I_{sk} 在范数为 N_{sk} 的 $N_{\mathsf{sk}}+1$ 个左 $\mathcal{O}_0$-理想中均匀随机采样。由于 N_{sk} 是大于 $2^{4\lambda_c}$ 的最小素数，对所有范数为 N_{sk} 的 $\mathcal{O}_0$-理想进行穷举搜索是不可行的。目前，尚不清楚是否存在更好的搜索方法。

遵循 [DLRW24, §4] 中的讨论，素数 N_{sk} 足够大以保证公钥椭圆曲线 E_{pk} 与随机曲线不可区分。可以证明，计算随机超奇异椭圆曲线的自同态环这一问题，可归约到恢复对应私钥的问题，例如使用 [DLRW24, Algorithm 8]。如前所述，学术界普遍相信获知从 E 出发的随机大素数次数同源并不会降低自同态环问题的难度，在此前提下密钥恢复等价于问题 8.1.1。

8.3.2 签名伪造

碰撞攻击：

[BBC+26, §4.4] 指出，若不使用 salt，参数 a 需满足 $2^{\lambda_c} \le 2^{(a-2)/2}\sqrt{a}$，否则攻击者可针对哈希函数 $\mathsf{H}_{\mathsf{PRISM}}$ 发起碰撞攻击，仅需 $o(2^{\lambda_c})$ 次操作即可完成签名伪造。由此可估计 a 的位长度至少约为 $2\lambda_c - \log \lambda_c$。具体而言，碰撞攻击首先离线查找碰撞 $\mathsf{H}_{\mathsf{PRISM}}(j(E_{\mathsf{pk}}), \mathsf{msg}_1, \mathsf{counter}_1) = \mathsf{H}_{\mathsf{PRISM}}(j(E_{\mathsf{pk}}), \mathsf{msg}_2, \mathsf{counter}_2)$（$\mathsf{msg}_1 \neq \mathsf{msg}_2$，其中 $\mathsf{counter}_i$ 是反复调用 $\mathsf{H}_{\mathsf{PRISM}}$ 后使输出落入 Primes_a 的最小计数器值），然后调用签名预言机一次（输入 msg_1），便为 msg_2 获得有效签名。

依照 [BBC+26, §4.4] 的分析，用随机 salt 替代计数器可大幅削弱离线碰撞攻击，从而可以使用更小的参数 a，在保持相同安全级别的同时获得更实用的方案。尽管 Theorem 8.2.2 的 QROM 安全证明已涵盖这一点，仍有必要讨论加 salt 设置下依然存在的碰撞攻击。这也说明了 QIMEN-PRISM 在 Chapter 5 中选择参数 a 和 n_{salt} 的理由。

- 离线后在线： 对上述攻击的直接改编是：不断选取随机的消息和 salt 对，直到获得一个素数碰撞 $\mathsf{H}_{\mathsf{PRISM}}(j(E_{\mathsf{pk}}), \mathsf{msg}_1, \mathsf{salt}_1) = \mathsf{H}_{\mathsf{PRISM}}(j(E_{\mathsf{pk}}), \mathsf{msg}_2, \mathsf{salt}_2)$ $(\mathsf{msg}_1 \neq \mathsf{msg}_2)$。然而，此时无法再通过单次调用签名预言机将其转化为签名伪造：预言机返回的是对 msg_1 的签名 $(\sigma, \mathsf{salt}_1')$，其中 salt_1' 为某个随机 salt，且以压倒性概率满足 $\mathsf{H}_{\mathsf{PRISM}}(j(E_{\mathsf{pk}}), \mathsf{msg}_1, \mathsf{salt}_1') \neq \mathsf{H}_{\mathsf{PRISM}}(j(E_{\mathsf{pk}}), \mathsf{msg}_2, \mathsf{salt}_2)$，因此 $(\sigma, \mathsf{salt}_2)$ 并不是对 msg_2 的有效签名。
  不过，通过多次调用签名预言机，攻击者可以期望最终获得一个满足 $\mathsf{salt}_1' = \mathsf{salt}_1$ 的签名。记调用次数为 N_{sign}，则预期成功概率为 N_{sign}/h，其中 h 是满足 $\mathsf{H}_{\mathsf{PRISM}}(j(E_{\mathsf{pk}}), \mathsf{msg}_1, \mathsf{salt}) \in \mathsf{Primes}_a$ 的 $\mathsf{salt} \in \{0,1\}^{n_{\mathsf{salt}}}$ 的数量。h 的估计值自然取为 $2^{n_{\mathsf{salt}}}\gamma_a$。因此，为保证此概率可忽略，即 $\frac{N_{\mathsf{sign}}}{2^{n_{\mathsf{salt}}}\gamma_a} \leq 2^{-\lambda_c}$，需要 $n_{\mathsf{salt}} \geq \lambda_c + \log_2(N_{\mathsf{sign}}) - \log_2(\gamma_a)$。
- 在线后离线： a 的规模由以下攻击场景确定：攻击者首先调用签名预言机，获得对 N_{sign} 个消息 msg_i 的 N_{sign} 个签名 $(\sigma_i, \mathsf{salt}_i)$；然后固定任意消息 msg' 并不断采样随机 salt salt'，直到出现碰撞 $\mathsf{H}_{\mathsf{PRISM}}(j(E_{\mathsf{pk}}), \mathsf{msg}_i, \mathsf{salt}_i) = \mathsf{H}_{\mathsf{PRISM}}(j(E_{\mathsf{pk}}), \mathsf{msg}', \mathsf{salt}')$，此时 $(\sigma_i, \mathsf{salt}_i)$ 即成为对 msg' 的有效签名。salt' 产生碰撞的概率约为 $N_{\mathsf{sign}}/2^a$。为保证此概率可忽略，需要 $a \geq \lambda_c + \log_2(N_{\mathsf{sign}})$。

以下优势界也证实了上述界限：

$$\frac{\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{salt\text{-}PRISM}}(\mathcal{A})}{N_{\mathsf{H}} + N_{\mathsf{sign}}} \leq \mathsf{Adv}^{\mathsf{LDPI}}_{p,N_{\mathsf{sign}}}(\mathcal{B}) + \frac{N_{\mathsf{sign}}}{2^a} + \frac{N_{\mathsf{sign}}}{\gamma_a 2^{n_{\mathsf{salt}}-1}},$$

该式在 ROM 下由 [BBC+26, Theorem 2] 建立。这里攻击代价的下界为 $N_{\mathsf{H}} + N_{\mathsf{sign}}$。此处选择在 ROM 而非 QROM 下分析，理由如下：QROM 中出现的二次损失是证明技术引入的人工产物，这在先哈希后签名（hash-and-sign）的范式下是常见的。因此，对于 λ_c 位经典安全且 $N_{\mathsf{sign}} = 2^{64}$，设置 $a = \lambda_c + 64$ 和 $n_{\mathsf{salt}} = \lambda_c + 64 + \log_2(a)$ 是充分的。

最后需指出，关于 a 的条件是容易满足的。事实上，根据先前对问题 8.1.1 困难度的讨论，素数 p 本身已要求约 $4\lambda_q$ 位，这超过了 $2\lambda_c$ 位。因此，由构造可知，可访问的 2-挠中天然存在足够的位数来满足所需的 a 位。

哈希到伪素数攻击：

在 QIMEN-PRISM 中，算法 PrimeGenerator使 用 Miller–Rabin 方法测试 $\mathsf{H}_{\mathsf{PRISM}}$ 输出的素性。Miller–Rabin 算法存在非零的错误概率，即可能将一个合数错误地判定为素数。该概率随 Miller–Rabin 的迭代轮数（记作 $\mathrm{MR}_{\mathrm{iter}}$）的增加而降低。如 [CGMT26] 所示，敌手可以利用此类错误伪造签名。

为了抵御此类攻击，参数 $\mathrm{MR}_{\mathrm{iter}}$ 的取值需保证 Miller–Rabin 测试的错误概率小于 $2^{-\lambda_c}$。这一选择基于 [DLP93] 中针对平均情况的分析。具体而言，[DLP93, Theorem 6] 对随机奇数 k-bit 输入经过 t 轮迭代后 Miller–Rabin 测试的（当 $k \geq 21$ 且 $t \geq k/9$ 时）错误概率 $p_{k,t}$ 提供以下上界：

$$p_{k,t} < \frac{7}{20}k2^{-5t} + \frac{1}{7}k^{15/4}2^{-k/2-2t} + 12k2^{-k/4-3t}.$$

对于三个安全级别 128, 256 和 512，Miller–Rabin 方法测试的整数的对应位长分别为 $a = 224$、320 和 576。因此，分别设置 $\mathrm{MR}_{\mathrm{iter}} = 28$、63 和 144，使得错误概率满足 $p_{224,28} \leq 2^{-128}$、$p_{320,63} \leq 2^{-256}$ 和 $p_{576,144} \leq 2^{-544}$。

需指出，以上分析聚焦于经典安全性，因为目前尚无已知方法能利用量子计算机加速对伪素数哈希的搜索。

8.3.3 不可重签名性（Non Re-signability）

如 [BBC+26] 所述，将原曲线同源（domain isogeny）E_{pk} 纳入哈希参数（此举几乎无代价），即可实现 [CDF+21] 中定义的不可重签名性（non re-signability）性质。这使得每个签名与其签名者的公钥绑定。由此可以防止敌手将某个在某公钥下验证有效的签名，重用于另一不同公钥下来伪造有效签名。

8.4 参数安全性评估

Table 8.1 汇总了当前已知的针对 QIMEN-PRISM（采用 Chapter 5 的参数）的最佳经典攻击和量子攻击的代价，并确认目标安全级别已达成。表中使用以下记号：

⋆: 所报告的复杂度仅涉及攻击的第一步，即找到一个哈希输出落在伪素数上。完成该攻击还需要额外的代价高昂的操作，这些操作显著增加了总体计算代价。更多细节参见 [CGMT26]。

†: 复杂度估计与目标安全级别之间的差距由 $\mathbb{F}_p$ 上的算术运算代价弥补，这些代价未计入渐近估计中。

‡: 复杂度估计与目标安全级别之间的 18-bit 缺口由 $\tilde{O}$ 中固有的隐藏常数和对数因子弥补，如 Section 8.1.2 中所讨论。

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

Table 8.1: QIMEN-PRISM 的参数选择与攻击复杂度

		NGCC-1	**NGCC-2**	**NGCC-3**
		参数		
	p	$69 \cdot 2^{313} - 1$	$27 \cdot 2^{500} - 1$	$15 \cdot 2^{1004} - 1$
	a	224	320	576
	n_{salt}	232	336	592
	$\mathrm{MR}_{\mathrm{iter}}$	28	63	144
	N_{sign}	2^{64}	2^{64}	2^{64}
所需经典安全强度等级		128	256	512
经典攻击	**复杂度类**	**复杂度估计** §		
针对问题 8.1.1	$\tilde{O}(p^{1/2})$	2^{160}	$2^{252\dagger}$	$2^{504\dagger}$
针对 Problem 8.1.2	$\tilde{O}(2^{6a/7})$	2^{192}	2^{274}	$2^{494\ddagger}$
离线--在线	$\tilde{O}(\frac{2^{n_{\mathsf{salt}}}\gamma_a}{N_{\mathsf{sign}}})$	2^{160}	2^{263}	2^{518}
在线--离线	$\tilde{O}(\frac{2^a}{N_{\mathsf{sign}}})$	2^{160}	2^{256}	2^{512}
哈希到伪素数		$2^{128\star}$	$2^{256\star}$	$2^{544\star}$
所需量子安全强度等级		80	128	256
量子攻击	**复杂度类**	**复杂度估计** §		
针对问题 8.1.1	$\tilde{O}(p^{1/4})$	2^{80}	$2^{126\dagger}$	$2^{252\dagger}$

CHAPTER 9

失败概率分析

本章列举并分析 QIMEN-PRISM 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-PRISM 中，失败的处理方式取决于协议阶段。密钥生成期间，RandFixNormIdeal的初始调用使用 `prime=true` 分支。若RandomEquivalentPrimeIdeal或IdealToIso随后抛出异常，QIMEN-PRISM.KeyGen会 捕获该异常并重新开始密钥生成。签名期间，QIMEN-PRISM.GenIso抛 出的异常将作为签名失败向上传播。验证期间，CheckIsogeny内部捕获异常后将其转换为 false，从而令QIMEN-PRISM.Verif返 回 reject。

可能的失败来源如下：

- RandomEquivalentPrimeIdeal：在规定搜索空间内未能找到具有素约化范数的等价理想时抛出异常。
- GeneralizedRepresentInteger：有界搜索未能找到目标范数的四元数元素时抛出异常。当RandFixNormIdeal以 `prime=false` 调用时，此异常向上传播。
- Qlapoti：有界生成元搜索未能找到合适的元素 β_1, β_2 时抛出异常。
- IdealToIso：因Qlapoti或Isogeny22Chain的异常向上传播而失败。
- CheckIsogeny：捕获其对Isogeny22Chain的调用所抛出的任何异常，并返回 false。

以下各节分别分析RandomEquivalentPrimeIdeal、GeneralizedRepresentInteger、Qlapoti以及Isogeny22Chain的失败概率。RandomEquivalentPrimeIdeal和GeneralizedRepI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。Qlapoti的分析与 [BCRSE+26] 一致。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 RandomEquivalentPrimeIdeal的失败概率分析

该算法搜索一个与输入理想 I 等价且具有素范数的理想。实际中，这等价于寻找 $\beta \in I$ 使得 $\mathrm{nrd}(\beta)/\mathrm{nrd}(I)$ 为素数。给定 I 的一个 L2 约化基 $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ 后，这又等价于寻找由

如下二次型表示的素数：

$$q_I : (c_1, c_2, c_3, c_4) \longmapsto \frac{\mathrm{nrd}(c_1\alpha_1 + c_2\alpha_2 + c_3\alpha_3 + c_4\alpha_4)}{\mathrm{nrd}(I)}.$$

令 $B :=$ `QUAT_equiv_bound_coeff`，其取值在Section 4.1.1中指定。算法从盒子 $[-B, B]$ 中采样 c_1, c_2, c_3, c_4，产生 $(2B+1)^4$ 个候选。由 [DLRW24, Lemma 48]，由于 $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ 构成一个 L2 约化基，我们有

$$q_I(c_1, c_2, c_3, c_4) \lesssim \frac{8p}{\pi^2} \cdot B^2.$$

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。

根据Heuristic 9.1.1，将所表示的整数视为该大小的随机整数，则单个候选为素数的概率至少为 $1/(2\log(p))$。因此，总失败概率的上界为

$$\left(1 - \frac{1}{2\log(p)}\right)^{(2B+1)^4}.$$

参数选择下失败率的上界估计见Table 9.1。

NGCC Level	p	`QUAT_equiv_bound_coeff`	failure rate
1	$69 \cdot 2^{313} - 1$	64	$< 2^{-904129}$
2	$27 \cdot 2^{500} - 1$	64	$< 2^{-571357}$
3	$15 \cdot 2^{1004} - 1$	64	$< 2^{-286030}$

Table 9.1: 将RandomEquivalentPrimeIdeal应用于 QIMEN-PRISM 参数时的失败率上界。

9.2 GeneralizedRepresentInteger的失败概率分析

GeneralizedRepresentInteger成功当且仅当算法执行了 Line 9。在Heuristic 9.1.1下，M' 为模 4 余 1 的素数的概率约为 $1/(2\log M')$。QIMEN-PRISM 中取 $M = q(2^a - q)$（如Section 4.1中指定），且 $M' := 4M - p(z^2 + t^2) > 0$，则 M' 的上界为 $p^{1.5}$，从而成功概率至少为 $1/(3\log(p))$。

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

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

一旦 B 大于 $3\lambda_c \log p$——在 QIMEN-PRISM 中始终成立——GeneralizedRepresentInteger的失败率便以可忽略的速率 $2^{-\lambda_c}$ 为上界。

9.3 Qlapoti的失败概率分析

本节遵循 [BCRSE+26, Section 4] 中给出的 Qlapoti 失败概率分析。与原文一致，引入以下随机变量来建模QLAPOTI的失败概率：

- 令 $\delta := \frac{n}{\sqrt{p}}$ 为定义在所有理想类 $[I]$（$I \subset \mathcal{O}_0$）域上的随机变量，其中 n 表示类 $[I]$ 中最小的等价理想的范数；
- 对固定的 n，令 $\epsilon_n := \frac{\sqrt{s^2+t^2}}{\sqrt{n}}$ 为定义在域 $a, b, c \in \mathbb{Z}/n\mathbb{Z}$ 上的随机变量，满足 $\gcd(a, n) = 1$ 且 $n \nmid b$，其中 (s, t) 是线性方程 $ax + by = c \bmod n$ 的最小范数解。

对离散随机变量 X，以 f_X 记概率质量函数，以 F_X 记累积分布函数。

遵循 [BCRSE+26, Section 4] 中的分析，对密码学大小的 p，用 δ_{wc} 表示在所有理想类 $[I]$（$I \subset \mathcal{O}_0$）上 δ 的最坏情形值，并取 $\delta_{wc} = 2\sqrt{2}/\pi$。类似地，用 δ_{ac} 表示在所有理想类 $[I]$（$I \subset \mathcal{O}_0$）上 δ 的平均情形值，并取 $\delta_{ac} = 0.37$。

本节计算在最坏情形和平均情形下，为找到一个好的 (α, s, t) 元组（即 Line 12中定义的 z 为正数）所需的 α 个数，分别记为 $\#\alpha$ (wc) 和 $\#\alpha$ (ac)。所需的 α 个数通过首先计算 u_{wc} 和 u_{ac} 来估算：令 $u := 1/\delta\sqrt{2f}$，其中 f 为 p 中的辅因子，u_{wc} 和 u_{ac} 分别取 δ 为 δ_{wc} 和 δ_{ac} 计算得到。依据 [BCRSE+26, Section 4]，有 $\epsilon \le 1/\delta\sqrt{2f}$。这意味着需要尝试的 α 个数由 $1/F_\epsilon(u)$ 显式给出。由于 [BCRSE+26, Section 4] 仅包含 $F_\epsilon(u)$ 的图形估计，本文重复了其实验，所得 $\#\alpha$ 取值见Table 9.2。

NGCC Level	p	e	u_{wc}	$\#\alpha$ (wc)	u_{ac}	$\#\alpha$ (ac)
1	$69 \cdot 2^{313} - 1$	311	0.095	35.6	0.23	6.3
2	$27 \cdot 2^{500} - 1$	498	0.151	14.2	0.368	2.7
3	$15 \cdot 2^{1004} - 1$	1002	0.203	8.0	0.493	1.7

Table 9.2: 生成一个好的 (α, s, t) 元组所需的最坏情形和平均情形 α 个数。

此外，s, t, z 还需满足某些同余条件，且 Line 23需要返回一个解（本节采用最坏情形分析，要求 z 为模 4 余 1 的素数）。综合这些条件，最坏情形界为

$$8 \log(\sqrt{p}/(2f\delta))$$

，如 [BCRSE+26, Equation (13)] 所示。由此得到QLAPOTI成功所需的 α 个数，如下表所示。

NGCC Level	p	e	$\#\alpha$ (wc)	$\#\alpha$ (ac)
1	$69 \cdot 2^{313} - 1$	311	30153	4510
2	$27 \cdot 2^{500} - 1$	498	19522	3715
3	$15 \cdot 2^{1004} - 1$	1002	22256	4734

Table 9.3: QLAPOTI完成所需的最坏情形和平均情形 α 个数。

根据 [BCRSE+26, Section 4] 的分析，若 n 是与输入理想等价的最小理想的范数，则生成元 α 的期望个数为

$$\approx 0.607927n \cdot \frac{2^{e-2}}{4p}.$$

本节采用同样的假设：每个 α 的成功概率相同。因此 Qlapoti 终止的概率服从几何分布。令 c 记该分布的均值，即终止所需的平均 α 个数，则单个 α 的成功概率为 $q = 1/c$。Qlapoti 在 k 步（即 k 个 α）内未能终止的概率为 $(1-q)^k$，因此要求

$$(1-q)^k < 2^{-(m+1)},$$

其中 m 为某个正整数。取对数并利用对小量 x 有 $\log(1-x) \simeq -x$，得到

$$k > (m+1)\log(2)c.$$

由此，对任意正整数 m，当满足以下条件时，QLAPOTI的失败概率小于 $2^{-(m+1)}$，从而保证成功：

$$n > (m+1) \cdot \frac{4\log(2) \cdot p \cdot c}{0.607927 \cdot 2^{e-2}}. \tag{9.1}$$

这里 c 表示Table 9.3中的 $\#\alpha$ (ac)。

依据 [BCRSE+26, Section 4] 的保守估计，QIMEN-PRISM 参数下QLAPOTI的失败概率分析如下。令 L 记Eq. (9.1)的右端，

$$P(\text{Qlapoti fails}) < P(n \le L) + P(\text{Qlapoti fails} \mid n > L) < F_\delta\left(\frac{L}{\sqrt{p}}\right) + 2^{-(m+1)}.$$

这里将所有满足 $n \le L$ 的情形均视为失败。对 $\delta = n/\sqrt{p}$，[BCRSE+26, Section 4] 中的经验估计给出 $F_\delta(x) \approx (6.46/2)x^2$。因此，

$$P(\text{Qlapoti fails}) < F_\delta\left(\frac{L}{\sqrt{p}}\right) + 2^{-(m+1)} \approx \frac{6.46}{2}\left(\frac{L}{\sqrt{p}}\right)^2 + 2^{-(m+1)}.$$

于是可通过寻找最大的 m 满足

$$2^{-(m+1)} > \frac{6.46}{2}(L/\sqrt{p})^2 = \frac{6.46}{2} \cdot \left((m+1) \cdot \frac{4\log(2) \cdot \sqrt{p} \cdot c}{0.607927 \cdot 2^{e-2}}\right)^2,$$

来获得失败率的上界，因为此时有

$$P(\text{Qlapoti fails}) < 2^{-(m+1)} + 2^{-(m+1)} = 2^{-m}.$$

具体而言，对 QIMEN-PRISM 参数集，从Table 9.3中读出 c 的取值，即可得到最终的失败概率。这些概率均确信地在安全级别之内（参见Table 9.4）。

需要指出，QLAPOTI的第一步调用了RANDOMEQUIVALENTPRIMEIDEAL，以上分析假设该步成功。若要纳入该步的失败率，可回顾Section 9.1中计算的RANDOMEQUIVALENTPRIMEIDEAL失败率，两者合并即可。

NGCC Level	p	$e-2$	c	failure rate
1	$69 \cdot 2^{313} - 1$	311	4510	2^{-255}
2	$27 \cdot 2^{500} - 1$	498	3715	2^{-442}
3	$15 \cdot 2^{1004} - 1$	1002	4734	2^{-944}

Table 9.4: 将Qlapoti应用于 QIMEN-PRISM 参数时的失败率上界。

9.4 Isogeny22Chain的失败概率分析 *

在 QIMEN-PRISM 的密钥生成、签名和验证过程中，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——尽管该阶段仍会捕获它们并重新开始流程。如果它们在签名过程中发生，则作为签名失败向上传播。如果它们在验证过程中发生，则以压倒性概率表明签名是恶意的，从而导致拒绝。

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。这种失败在验证诚实签名时永远不会发生。为估计密钥生成/签名中失败概率的上界，本节采用如下假设。

Heuristic 9.4.1. 在QIMEN-PRISM.KeyGen和QIMEN-PRISM.Sign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 9.4.2. 令 $E_1 \times E_2$ 为在QIMEN-PRISM.KeyGen、QIMEN-PRISM.Sign和QIMEN-PRISM.Verif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 $\mathbb{F}_{p^2}$ 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 $O(1)$ 个二维粘合同源；因此第二种类型失败的总失败概率为 $O(p^{-2})$。

需要指出，第二种失败也可能（以可忽略的概率）在验证诚实签名时发生，因为验证中的粘合计算方向与签名中相反。

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.

[BBC+25] Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM: Simple and compact identification and signatures from large prime degree isogenies. LNCS, pages 300–332. Springer, Cham, Switzerland, May 10–13, 2025.

[BBC+26] Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM with a pinch of salt: Simple, efficient and strongly unforgeable signatures from isogenies. Cryptology ePrint Archive, Paper 2026/443, 2026.

[BBS+26] Andrea Basso, Giacomo Borin, Maria Corte-Real Santos, Pierrick Dartois, Riccardo Invernizzi, Luciano Maino, Robi Pedersen, and Michel Seck. Isogeny-based signatures with randomizable keys. Cryptology ePrint Archive, Paper 2026/1169, 2026.

[BCL08] Reinier Bröker, Denis Charles, and Kristin Lauter. Evaluating large degree isogenies and applications to pairing based cryptography. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 100–112, Egham, UK, September 1–3, 2008. Springer, Berlin, Heidelberg, Germany.

[BCRSE+26] Giacomo Borin, Maria Corte-Real Santos, Jonathan Komada Eriksen, Riccardo Invernizzi, Marzio Mula, Sina Schaeffler, and Frederik Vercauteren. Qlapoti: Simple and efficient translation of quaternion ideals to isogenies. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology – ASIACRYPT 2025*, pages 174–205, Singapore, 2026. Springer Nature Singapore.

[BDD+24] Andrea Basso, Pierrick Dartois, Luca De Feo, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQIsign2D-West - the fast, the small, and the safer. In *ASIACRYPT 2024, Part III*, LNCS, pages 339–370. Springer, Singapore, Singapore, December 7–11, 2024.

[BDLS20] Daniel J Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *Open Book Series*, 4(1):39–55, 2020.

[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.

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

[CDF+21] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In *2021 IEEE Symposium on Security and Privacy*, pages 1696–1714, San Francisco, CA, USA, May 24–27, 2021. IEEE Computer Society Press.

[CEMR24] Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Krijn Reijnders. AprèsSQI: Extra fast verification for SQIsign using extension-field signing. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 63–93, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[CGMT26] Jolijn Cottaar, Steven D. Galbraith, Luciano Maino, and Monika Trimoska. An analysis of a weakened version of PRISM. Cryptology ePrint Archive, Paper 2026/906, 2026.

[Coh93] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer, New York, NY, USA, 1993.

[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.

[Cou96] Jean-Marc Couveignes. Computing ℓ-isogenies using the p-torsion. In *Algorithmic Number Theory*. Springer, 1996.

[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.

[DFH+24] Jelle Don, Serge Fehr, Yu-Hsuan Huang, Jyun-Jie Liao, and Patrick Struck. Hide-and-seek and the non-resignability of the BUFF transform. In *TCC 2024, Part III*, LNCS, pages 347–370. Springer, Cham, Switzerland, November 2024.

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

[DKL+20] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 64–93, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland.

[DLLW23] Luca De Feo, Antonin Leroux, Patrick Longa, and Benjamin Wesolowski. New algorithms for the Deuring correspondence - towards practical and secure SQISign signatures. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 659–690, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[DLP93] Ivan Damgård, Peter Landrock, and Carl Pomerance. Average case error estimates for the strong probable prime test. *Mathematics of computation*, 61(203):177–194, 1993.

[DLRW24] Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQIsignHD: New dimensions in cryptography. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 3–32, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[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.

[Ebe07] David Eberly. The laplace expansion theorem: Computing the determinants and inverses of matrices. *Geometric Tools, LLC, Scottsdale, Ariz, USA*, 2007.

[Elk98] Noam D. Elkies. Elliptic and modular curves over finite fields and related computational issues. In *Computational Perspectives on Number Theory*. American Mathematical Society, 1998.

[FHK25] Serge Fehr, Yu-Hsuan Huang, and Julia Kastner. Sandwich BUFF: Achieving non-resignability using iterative hash functions. In *TCC 2025, Part III*, LNCS, pages 235–265. Springer, Cham, Switzerland, November 2025.

[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.

[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.

[Koh96] David R. Kohel. *Endomorphism Rings of Elliptic Curves over Finite Fields*. PhD thesis, University of California, Berkeley, 1996.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

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

[LLL82] A.K. Lenstra, H.W.jun. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[LM00] Reynald Lercier and François Morain. Computing isogenies between elliptic curves over F_{p^n} using couveignes' s algorithm. *Mathematics of Computation*, 69(229):351–370, 2000.

[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.

[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.

[NO25] Kohei Nakagawa and Hiroshi Onuki. Attacks on PRISM-id via torsion over small extension fields. Cryptology ePrint Archive, Report 2025/1602, 2025.

[NOC+24] Kohei Nakagawa, Hiroshi Onuki, Wouter Castryck, Mingjie Chen, Riccardo Invernizzi, Gioella Lorenzon, and Frederik Vercauteren. SQIsign2D-East: A new signature scheme using 2-dimensional isogenies. In *ASIACRYPT 2024, Part III*, LNCS, pages 272–303. Springer, Singapore, Singapore, December 7–11, 2024.

[NS09] Phong Q. Nguyen and Damien Stehlé. An lll algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009.

[NV10] Phong Q Nguyen and Brigitte Vallée. *The LLL algorithm*. Springer, 2010.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[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.

[PZ24] Patrick Pelissier and Paul Zimmerman. The DPE library, 2024.

[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.

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

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

[Thr] Prism: Compact threshold signatures from pushforwards of large-degree isogenies. https://csrc.nist.gov/presentations/2026/mpts2026-4a2. Accessed: 2026-06-17.

[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.

[ZSP+18] Gustavo Zanon, Marcos A. Simplício, Jr., Geovandro C. C. F. Pereira, Javad Doliskani, and Paulo S. L. M. Barreto. Faster isogeny-based compressed key agreement. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 248–268, Fort Lauderdale, Florida, United States, April 9–11, 2018. Springer, Cham, Switzerland.

CHAPTER A
实现概览

本章简要介绍 QIMEN-PRISM 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。

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-PRISM.KeyGen, QIMEN-PRISM.Sign, QIMEN-PRISM.Verif ）调用。本模块的完整描述见Chapter D。

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 E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。本模块的完整描述见Chapter F。

qlapoti: 提供理想与同源之间相互翻译的算法，用于 hd 和核心方案功能（QIMEN-PRISM.KeyGen, QIMEN-PRISM.Sign, QIMEN-PRISM.Verif ）。

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-PRISM.KeyGen, QIMEN-PRISM.Sign, QIMEN-PRISM.Verif 。

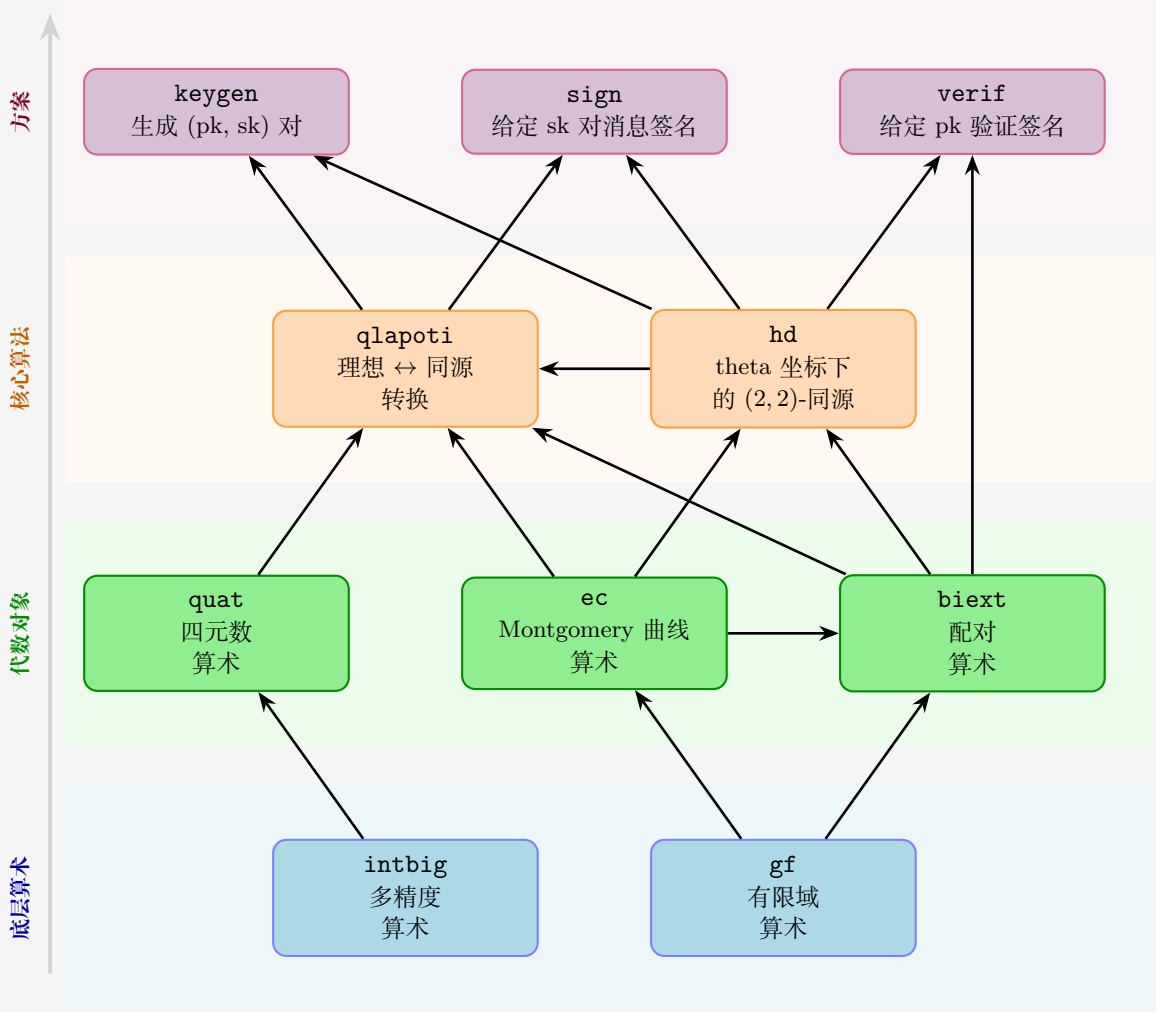

Figure A.1: QIMEN-PRISM 参考实现的模块结构，垂直方向表示抽象层级的递升。

CHAPTER B

有限域算术（实现细节）*

QIMEN-PRISM 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。

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 22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 23，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.1中总结了这些开销，供后续各节使用。

Table B.1: 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 22 Fp2Mul(a, b)

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

1: $t_0 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1)$
2: $t_1 \leftarrow a_1 \cdot b_1$
3: $c_0 \leftarrow a_0 \cdot b_0$
4: $c_1 \leftarrow t_0 - t_1 - c_0$ ▷ $= a_0 b_1 + a_1 b_0$
5: $c_0 \leftarrow c_0 - t_1$ ▷ $= a_0 b_0 - a_1 b_1$
6: **return** $c = c_0 + c_1\, i$

Algorithm 23 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。QIMEN-PRISM 仅使用 2-adic 情形。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 24 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
    $b \leftarrow 1$
$\eta' \leftarrow \eta \cdot \zeta^{-b}$
$k' \leftarrow \mathrm{DLogPowerOfTwo}(\zeta^2, \eta', e-1)$
return $b + 2k' \bmod 2^e$

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 25）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 26）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 27）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 25 $\mathrm{xDBL}(P, (A_{24} : C_{24}))$

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 26 $\mathrm{xADD}(P, Q, P - Q)$

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 27 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 28 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 29 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 30 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 $[m]P+[n]Q=(X_{[m]P+[n]Q}:Z_{[m]P+[n]Q})$ ▷ $(6k+2)\mathrm{S}+(12k+4)\mathrm{M}+(16k+6)\mathrm{a}$

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。
- 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 31 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 32 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 33 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-PRISM 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 35）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 34）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 36）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 34 $\mathrm{ADD}(P, Q, (A : C))$

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 35 $\mathrm{DBL}(P, (A : C))$

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 36 $\mathrm{ADDCOMPONENTS}(P, Q, (A : C))$

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

CHAPTER D

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 D.1 引入 level-2 theta 坐标，
- Section D.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 D.4；
2. 在 theta 坐标中迭代一般 $(2,2)$-同源，参见 Section D.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 D.5。

D.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。

D.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 37。

Algorithm 37 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: $2\mathrm{M} + 2\mathrm{a}$

D.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。

D.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。

D.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 38 ThetaPrecomp(0_A)

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

$(A, B, C, D) \leftarrow \mathcal{H} \circ \mathcal{S}(a, b, c, d)$　$t_1 \leftarrow A \cdot B$　$t_2 \leftarrow C \cdot D$
$ABC \leftarrow t_1 \cdot C$　$ABD \leftarrow t_1 \cdot D$　$ACD \leftarrow t_2 \cdot A$
$BCD \leftarrow t_2 \cdot B$　$t_1 \leftarrow a \cdot b$　$t_2 \leftarrow c \cdot d$
$abc \leftarrow t_1 \cdot c$　$abd \leftarrow t_1 \cdot d$　$acd \leftarrow t_2 \cdot a$
$bcd \leftarrow t_2 \cdot b$　.　.
consts $\leftarrow \{abc, abd, acd, bcd, ABC, ABD, ACD, BCD\}$
return consts　▷ 代价: 4S + 12M (+8a)

Algorithm 39 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

$(x_P, y_P, z_P, w_P) \leftarrow P$　$(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8) \leftarrow$ consts
$(X_{2P}, Y_{2P}, Z_{2P}, W_{2P}) \leftarrow \mathcal{S} \circ \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, w_P)$　$X_{2P} \leftarrow X_{2P} \cdot c_8$
$Y_{2P} \leftarrow Y_{2P} \cdot c_7$　$Z_{2P} \leftarrow Z_{2P} \cdot c_6$
$W_{2P} \leftarrow W_{2P} \cdot c_5$　$(X_{2P}, Y_{2P}, Z_{2P}, W_{2P}) \leftarrow \mathcal{H}(X_{2P}, Y_{2P}, Z_{2P}, W_{2P})$
$X_{2P} \leftarrow X_{2P} \cdot c_4$　$Y_{2P} \leftarrow Y_{2P} \cdot c_3$
$Z_{2P} \leftarrow Z_{2P} \cdot c_2$　$W_{2P} \leftarrow W_{2P} \cdot c_1$
return $(X_{2P} : Y_{2P} : Z_{2P} : W_{2P})$　▷ 代价: 8S + 8M + 16a

D.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 D.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 D.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 D.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 D.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。

D.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 40 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 41 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

D.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 42 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

D.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 43 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

D.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 44 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 45 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

D.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。

D.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 46 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 47 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 48 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

D.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-PRISM 很有用。

Algorithm 49 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 50 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 $(x_{\Phi(P)}:y_{\Phi(P)}:z_{\Phi(P)}:w_{\Phi(P)})$

D.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 51 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 52 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 53 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 54 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** $(X_1 : Z_1), (X_2 : Z_2)$

D.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 55 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 56 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 57 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 58 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 59 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 E

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]

E.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 60 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 61 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 62 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 63 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** $X(P_2)/X(P_1)$

E.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 64 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 65 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 $\lambda^{(p^2-1)/2^e}$

CHAPTER F

四元数算法（实现细节）*

本附录给出实现所用的四元数算法。相关代数背景参见Sections 3.2.2 and 3.2.3。正文仅描述实现接口及其子程序间的依赖关系。本章分为三个部分。第一部分为格层，包括四元数元素与格的数据表示、实现的规范化约定以及格上基本操作。第二部分为理想层，基于上述格程序构建，涵盖左理想的表示、阶的计算以及高层四元数算法所需的理想构造。

F.1 格算法

在 QIMEN-PRISM 实现中，四元数代数元素 α 存储为一对 $(\boldsymbol{a}, e)$，其中 $\boldsymbol{a} = (a_0, a_1, a_2, a_3)^T \in \mathbb{Z}^4$ 且 $e \in \mathbb{Z} \setminus \{0\}$，即

$$\alpha = \frac{a_0 + a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k}}{e},$$

其中分母 e 以固定的四元数基 $(1, \mathbf{i}, \mathbf{j}, \mathbf{k})$ 为参考。

格 L 存储为一对 (B, d)，其中 $d \in \mathbb{Z} \setminus \{0\}$ 且

$$B = (\boldsymbol{b}_0\ \boldsymbol{b}_1\ \boldsymbol{b}_2\ \boldsymbol{b}_3) = (b_{ir})_{0 \le i,r \le 3} \in M_4(\mathbb{Z}).$$

其中 $\boldsymbol{b}_r = (b_{0r}, b_{1r}, b_{2r}, b_{3r})^T$ 是 B 的第 r 列。因此第 r 个格基元素的坐标向量为 $\boldsymbol{b}_r/d$。格操作前，实现先将整数基矩阵 B 化为 HNF；构造新格后，实现再将 B 与 d 除以二者的公因子（即 B 所有元素与 d 的最大公因数），将分母化为约化形式。

Algorithm 66 HNF(M)

Input: An integer matrix M with d rows and $c \geq d$ columns with rank d. Its columns are denoted by M_i^ℓ to M_c^ℓ, and the coefficient in row r and column i is denoted by $M_{r,i}$.
Output: The HNF of M.

for i from d down to 1 **do**
  for j from $i-1$ down to 1 **do**
    if $M_{i,j}$ and $M_{i,i}$ are both 0 **then**
      $g, u, v \leftarrow 1, 1, 0$
    else
      $g, u, v \leftarrow \mathsf{XGCD}(M_{i,i}, M_{i,j})$
    $M_i^\ell \leftarrow uM_i^\ell + vM_j^\ell$
    for j from $i-1$ down to 1 **do**
      $g \leftarrow M_{i,j}/M_{i,i}$
      $M_j^\ell \leftarrow M_j^\ell - gM_i^\ell$
    for j from $i+1$ up to c **do**
      $r \leftarrow M_{i,j} \bmod M_{i,i}$
      $g \leftarrow (M_{i,j} - r)/M_{i,i}$
      $M_j^\ell \leftarrow M_j^\ell - gM_i^\ell$
return M

Algorithm 67 LatticeEquality(L_1, L_2)

Input: Lattices L_1 and L_2 in HNF.
Output: true if $L_1 = L_2$, and false otherwise.

return $(|d_2|B_1 = |d_1|B_2)$

Algorithm 68 LatticeSum(L_1, L_2)

Input: Lattices L_1 and L_2 in HNF.
Output: $L_1 + L_2$ in HNF.

$M \leftarrow [\, d_1 B_2 \mid d_2 B_1 \,]$
$B \leftarrow \mathrm{HNF}(M)$
$d \leftarrow d_1 d_2$
Divide B and d by their common content.
return (B, d)

Algorithm 69 LatticeDual(L)

Input: A lattice $L = (B, d)$.
Output: The dual lattice L^*.

$M \leftarrow B^T$
$\Delta \leftarrow \det(M)$
$B' \leftarrow d \cdot \Delta \cdot M^{-1}$
$d' \leftarrow \Delta$
return (B', d') ▷ without HNF

Algorithm 70 LatticeIntersection(L_1, L_2)

Input: Lattices L_1 and L_2 in HNF.
Output: $L_1 \cap L_2$ in HNF.

$D_1 \leftarrow \text{LatticeDual}(L_1)$ and $D_2 \leftarrow \text{LatticeDual}(L_2)$
$S \leftarrow \text{LatticeSum}(D_1, D_2)$
$L \leftarrow \text{LatticeDual}(S)$
$L \leftarrow \text{HNF}(L)$
Reduce the denominator of L.
return L

Algorithm 71 LatticeMultiplication(L_1, L_2)

Input: Two lattices L_1 and L_2.
Output: $L_1 L_2$ in HNF.

$((\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3), d_1) \leftarrow L_1$
$((\boldsymbol{b}'_0, \boldsymbol{b}'_1, \boldsymbol{b}'_2, \boldsymbol{b}'_3), d_2) \leftarrow L_2$
$d \leftarrow d_1 d_2$
$S_0 \leftarrow (\boldsymbol{b}_0\boldsymbol{b}'_0, \ldots, \boldsymbol{b}_0\boldsymbol{b}'_3, \boldsymbol{b}_1\boldsymbol{b}'_0, \ldots, \boldsymbol{b}_1\boldsymbol{b}'_3)$
$H_0 \leftarrow \text{HNF}(S_0)$
$S_1 \leftarrow (\boldsymbol{b}_2\boldsymbol{b}'_0, \ldots, \boldsymbol{b}_2\boldsymbol{b}'_3, \boldsymbol{b}_3\boldsymbol{b}'_0, \ldots, \boldsymbol{b}_3\boldsymbol{b}'_3)$
$H_1 \leftarrow \text{HNF}(S_1)$
$B \leftarrow \text{HNF}([\, H_0 \mid H_1 \,])$
Divide B and d by their common content.
return (B, d)

Algorithm 72 LatticeContainment(α, L)

Input: A lattice L in HNF and a quaternion element $\alpha = \boldsymbol{a}/e$.
Output: A pair $(\mathsf{res}, \boldsymbol{c})$ where res states whether $\alpha \in L$, and $\boldsymbol{c}$ gives coordinates if membership holds.

$(B, d) \leftarrow L$
$\boldsymbol{w} \leftarrow d\boldsymbol{a}$
for $j = 3, 2, 1, 0$ **do**
    if eb_{jj} does not divide w_j **then**
        return (false, _)
    $c_j \leftarrow w_j/(eb_{jj})$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - c_j e\boldsymbol{b}_j$
if $\boldsymbol{w} = 0$ **then**
    return $(\mathsf{true}, \boldsymbol{c})$
else
    return (false, _)

Algorithm 73 LatticeInclusion(L_1, L_2)

Input: Two lattices L_1, L_2
Output: A boolean value: **true** if $L_1 \subset L_2$, **false** otherwise

return LatticeEquality(LatticeSum(L_1, L_2),L_2)

Algorithm 74 LatticeIndex(L_1, L_2)

Input: HNF lattices $L_1 \subseteq L_2$.
Output: The index $[L_2 : L_1]$.

$(B, d) \leftarrow L_1$
$(B', d') \leftarrow L_2$
for $i = 0, 1, 2, 3$ **do**
    $n \leftarrow d'^4 \prod_{i=0}^{3} b_{ii}$
    $m \leftarrow d^4 \prod_{i=0}^{3} b'_{ii}$
return $|n/m|$

格约化. L2$_{\eta,\delta}$算法源自先前的工作 [NS09]。该算法以任意格基及其关联的 Gram 矩阵为输入，返回原格的 (η, δ)-约化基及更新后的 Gram 矩阵。外层循环的高层操作均通过多精度整数运算实现，操作对象为格基元素和 Gram 矩阵元素。内层中间计算则依赖浮点算术，负责求值、存储和更新 Gram-Schmidt 正交化（GSO）系数 $r_{i,j}$ 和 $\mu_{i,j}$（其中 $i \geq j$）。在 [NS09] 中，这组正交化参数称为 GSO 族；更多背景参见 [NV10] 第 5 章。在下文的伪代码中，所有以浮点格式存储的数值标量、向量和矩阵均标注类型:: FLOAT，每次从整数到浮点的显式转换记作 FLOAT()。

在所考虑的二次空间中，所有向量具有整数坐标，格基表示为整数矩阵。子程序ExtendGSOFamily从 Gram 矩阵和不完整的 GSO 族出发，逐步构造 Gram-Schmidt 正交化系数。子程序SizeReduce约化基向量 $\boldsymbol{b}_k$；若任一投影系数超过阈值 $\bar{\eta}$，则更新 GSO 族和 Gram 矩阵。子程序InsertBefore将 $\boldsymbol{b}_k$ 插入基中 $\boldsymbol{b}_s$ 之前，随后更新继承的 GSO 系数及对应的对角正交化值。

Algorithm 75 ExtendGSOFamily$(\boldsymbol{G}, k, r :: \text{Float}, \mu :: \text{Float})$

Input: $\boldsymbol{G}$ a $d \times d$ Gram matrix, index $1 \leq k < d$, the GSO family r, μ up to row $k-1$.
Output: The GSO family r, μ up to row k.

for j from 0 up to k **do**
    $r_{k,j} \leftarrow \text{Float}(\boldsymbol{G}_{k,j})$
    for l from 0 up to $j-1$ **do**
        $r_{k,j} \leftarrow r_{k,j} - r_{k,l}\mu_{j,l}$
    if $j < k$ **then**
        $\mu_{k,j} \leftarrow \dfrac{r_{k,j}}{r_{j,j}}$
return r, μ

Algorithm 76 SizeReduce$((\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G}, k, r :: \text{Float}, \mu :: \text{Float}, \bar{\eta} :: \text{Float})$

Input: A basis $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1})$ of a quadratic space, its $d \times d$ Gram matrix $\boldsymbol{G}$, index $1 \leq k < d$, the GSO family r, μ up to row $k-1$, parameter $\frac{1}{2} < \bar{\eta} < 1$.

Output: $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1})$ size-reduced basis of the same lattice, its Gram matrix $\boldsymbol{G}$, the GSO family r, μ up to row k.

done ← false
while not done **do**
  $r, \mu \leftarrow$ ExtendGSOFamily$(\boldsymbol{G}, k, r, \mu)$
  done ← true
  for i from $k-1$ down to 0 **do**
    if $|\mu_{k,i}| > \bar{\eta}$ **then**
      done ← false
      $X \leftarrow \lfloor \mu_{k,i} \rceil$ ▷ Round to the closest integer
      $\boldsymbol{b}_k \leftarrow \boldsymbol{b}_k - X\boldsymbol{b}_i$ ▷ Update basis
      for j from 0 up to $d-1$ **do**
        $\boldsymbol{G}_{k,j} \leftarrow \boldsymbol{G}_{k,j} - X\boldsymbol{G}_{i,j}$ ▷ Update Gram matrix
      for j from 0 up to $d-1$ **do**
        $\boldsymbol{G}_{j,k} \leftarrow \boldsymbol{G}_{j,k} - X\boldsymbol{G}_{j,i}$ ▷ Update Gram matrix
      for j from 0 up to $i-1$ **do**
        $\mu_{k,j} \leftarrow \mu_{k,j} - \text{Float}(X)\mu_{i,j}$ ▷ Update μ
return $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G}, r, \mu$

Algorithm 77 InsertBefore$((\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G}, k, s :: \text{Float}, \mu :: \text{Float})$

Input: A basis $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_s, \ldots, \boldsymbol{b}_k, \ldots, \boldsymbol{b}_{d-1})$ of a quadratic space, its $d \times d$ Gram matrix $\boldsymbol{G}$, indices $0 \leq s < k < d$, the GSO family r, μ up to row k.

Output: The basis $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_k, \boldsymbol{b}_s, \ldots, \boldsymbol{b}_{d-1})$ where $\boldsymbol{b}_k$ has been inserted before $\boldsymbol{b}_s$, the associated Gram matrix, the GSO family r, μ up to row s.

for j from k down to $s+1$ **do**
  swap $\boldsymbol{b}_j$ and $\boldsymbol{b}_{j-1}$
  for i from 0 up to $d-1$ **do**
    swap $\boldsymbol{G}_{i,j}$ and $\boldsymbol{G}_{i,j-1}$
  for i from 0 up to $d-1$ **do**
    swap $\boldsymbol{G}_{j,i}$ and $\boldsymbol{G}_{j-1,i}$
$r_{s,s} \leftarrow \text{Float}(\boldsymbol{G}_{s,s})$
for i from 0 up to $s-1$ **do**
  $\mu_{s,i} \leftarrow \mu_{k,i}$
  $r_{s,i} \leftarrow r_{k,i}$
  $r_{s,s} \leftarrow r_{s,s} - \mu_{s,i} r_{s,i}$
return $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G}, r, \mu$

Algorithm 79 IdealGenerator(I)

Input: A left $\mathcal{O}_0$-ideal I.
Output: α primitive in I such that $I = \mathcal{O}(\alpha, \mathrm{nrd}(I))$.

$N_I \leftarrow \mathrm{nrd}(I)$
$n \leftarrow 0$
while true **do**
  $n \leftarrow n + 1$
  for a from $-n$ up to n **do**
    for b from $-n + |a|$ up to $n - |a|$ **do**
      for c from $-n + |a| + |b|$ up to $n - |a| - |b|$ **do**
        $d \leftarrow n - |a| - |b| - |c|$
        if $\gcd(a, b, c, d) = 1$ **then**
          $(B, \_, \_, \_) \leftarrow I$
          $\alpha \leftarrow a\boldsymbol{b}_0 + b\boldsymbol{b}_1 + c\boldsymbol{b}_2 + d\boldsymbol{b}_3$
          $q \leftarrow \mathrm{nrd}(\alpha)/N_I$
          if $\gcd(q, N_I) = 1$ **then return** α

Algorithm 78 $\mathrm{L2}_{\eta,\delta}((\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G})$

Input: A basis $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1})$ of a d-dimensional quadratic space, the associated $d \times d$ Gram matrix $\boldsymbol{G}$.
Output: A (η, δ)-reduced basis of the same lattice, its associated Gram matrix.

$\bar{\delta} \leftarrow \textsc{Float}\left(\frac{\delta+1}{2}\right)$, $\bar{\eta} \leftarrow \textsc{Float}\left(\frac{\eta+0.5}{2}\right)$
$r_{0,0} \leftarrow \textsc{Float}(\boldsymbol{G}_{0,0})$, $\mu_{0,0} \leftarrow \textsc{Float}(1)$,
$\boldsymbol{T} \leftarrow [\textsc{Float}(0), \textsc{Float}(0), \textsc{Float}(0), \textsc{Float}(0)]$
$k = 1$
while $k < d$ **do**
  $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G}, r, \mu \leftarrow \textsc{SizeReduce}((\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G}, k, r, \mu, \bar{\eta})$
  $T_0 \leftarrow \textsc{Float}(\boldsymbol{G}_{k,k})$
  for i from 1 up to $k - 1$ **do**
    $T_i \leftarrow T_{i-1} - \mu_{k,(i-1)} r_{k,(i-1)}$
  $s \leftarrow \min\{0 \le i \le k \mid \text{such that } T_j < \bar{\delta} r_{j,j} \text{ for all } i \le j < k\}$
  if $k \neq s$ **then**
    $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G}, r, \mu \leftarrow \textsc{InsertBefore}((\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G}, k, s, r, \mu)$
    $k \leftarrow s$
  $k \leftarrow k + 1$
return $(\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{d-1}), \boldsymbol{G}$

F.2 理想算法

在 QIMEN-PRISM 实现中，理想的数据结构基于格的存储结构。左 $\mathcal{O}$-理想 I 存储以下信息：表示 I 的格 L、I 的约化范数 $\mathrm{nrd}(I)$ 及其父极大序 $\mathcal{O}$（该序也存储为格）。下文的算法中，理想 I 记作 $(L_I, \mathrm{nrd}(I), \mathcal{O}_I)$ 或 $(B_I, d_I, \mathrm{nrd}(I), B_{\mathcal{O}}, d_{\mathcal{O}})$。

Algorithm 80 IdealSum(I_1, I_2)

Input: Two left $\mathcal{O}$-ideals I_1, I_2.
Output: Left $\mathcal{O}$-ideal $I_1 + I_2$.

$L \leftarrow \text{LatticeSum}(L_1, L_2)$
$N \leftarrow \text{LatticeIndex}(L, \mathcal{O})$
$N \leftarrow \sqrt{N}$
return $(L, N, \mathcal{O})$

Algorithm 81 IdealMul(I, J)

Input: Two ideals I and J such that $\mathcal{O}_R(I) = \mathcal{O}_L(J)$.
Output: IJ.

$N \leftarrow \text{nrd}(I)\,\text{nrd}(J)$
$(L_1, \_, \mathcal{O}) \leftarrow I$
$(L_2, \_, \_) \leftarrow J$
return $(\text{LatticeMultiplication}(L_1, L_2), N, \mathcal{O})$

Algorithm 82 IdealIntersection(I_1, I_2)

Input: Two left $\mathcal{O}$-ideals I_1 and I_2.
Output: Left $\mathcal{O}$-ideal $I_1 \cap I_2$.

$L \leftarrow \text{LatticeIntersection}(L_1, L_2)$
$N \leftarrow \text{LatticeIndex}(L, \mathcal{O})$
$N \leftarrow \sqrt{N}$
return $(L, N, \mathcal{O})$

Algorithm 83 IdealInverse(I)

Input: A left $\mathcal{O}$-ideal I.
Output: The inverse of I (stored as lattice)

$(B, d, N, \_) \leftarrow I$
$C \leftarrow \text{diag}(1, -1, -1, -1)$
$B' \leftarrow CB$
$d' \leftarrow d \cdot N$
return (B', d')

Algorithm 84 RightOrderOfAnIdeal(I)

Input: A left $\mathcal{O}$-ideal I.
Output: Right order of I.

$(L, \_, \_) \leftarrow I$
return $\text{LatticeMultiplication}(\text{IdealInverse}(I), L)$

Algorithm 85 IdealCreatePrincipal$(x, \mathcal{O})$

Input: An element $x = \boldsymbol{a}/d_x$ and an order $\mathcal{O} = (B_{\mathcal{O}}, d_{\mathcal{O}})$.
Output: Left-$\mathcal{O}$ ideal $\mathcal{O}x$.

$((\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3), d_{\mathcal{O}}) \leftarrow \mathcal{O}$
for $i = 0, 1, 2, 3$ **do**
    $\boldsymbol{m}_i \leftarrow \boldsymbol{b}_i \cdot x$
$B \leftarrow (\boldsymbol{m}_0, \boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3)$ ▷ The columns of B are the coordinates of $\boldsymbol{b}_i x$
$d \leftarrow d_x \cdot d_{\mathcal{O}}$
Divide B and d by their common content.
$B \leftarrow \mathrm{HNF}(B)$
$N \leftarrow \mathrm{nrd}(x)$
return $(B, d, N, \mathcal{O})$

Algorithm 86 IdealCreatePrimitive$(x, \mathcal{O}, N)$

Input: A primitive element $x \in \mathcal{O}$ and an integer N.
Output: $\mathcal{O}x + \mathcal{O}N$

$I \leftarrow$ IdealCreatePrincipal$(x, \mathcal{O})$
$B \leftarrow N \cdot B_{\mathcal{O}}$ ▷ Matrix by integer multiplication
$d \leftarrow d_{\mathcal{O}}$
$(L_1, \_, \_) \leftarrow I$
$L_2 \leftarrow (B, d)$
$L \leftarrow$ LatticeSum(L_1, L_2)
$N' \leftarrow$ LatticeIndex$(L, \mathcal{O})$
$N' \leftarrow \sqrt{N'}$
return $(L, N', \mathcal{O})$

Algorithm 87 ConnectingIdeal$(\mathcal{O}_L, \mathcal{O}_R)$

Input: Two orders $\mathcal{O}_L$ and $\mathcal{O}_R$.
Output: $N\mathcal{O}_L\mathcal{O}_R$, where N is the square root of the index of $\mathcal{O}_L \cap \mathcal{O}_R$ in $\mathcal{O}_L$.

$I \leftarrow$ LatticeIntersection$(\mathcal{O}_L, \mathcal{O}_R)$
$N \leftarrow$ LatticeIndex$(I, \mathcal{O}_L)$
$N \leftarrow \sqrt{N}$
$(B, d) \leftarrow$ LatticeMultiplication$(\mathcal{O}_L, \mathcal{O}_R)$
$B \leftarrow N \cdot B$
Divide B and d by their common content
return (B, d)