通用型格密码算法库的设计与实现

摘　要：随着量子计算机的迅速发展和 Shor 等量子算法的提出，传统公钥密码的底层安全已受到严重威胁。应对方案是建立能够有效抵御量子计算攻击的后量子密码体制，其中，基于格的密码被广泛认为是最佳候选者。为了提升格密码方案的实现性能，本文设计并实现了一个通用型格密码算法库。在对现有格密码软硬件实现策略进行梳理的基础上，根据格代数结构和困难问题抽象出通用组件。采用固定某一乘数的 Montgomery 算法与 Barrett 算法实现模约简，利用快速数论变换算法与 Knuth-Yao 算法实现多项式乘法和离散高斯抽样。为了提升格密码算法库的运算速度，对部分代码进行 AVX2 并行化处理，并将 Knuth-Yao 的抽样时间常数化，以抵御计时侧信道攻击。基于所设计的组件构建底层数学层，为更上层的具体方案提供接口。实验结果表明，本文提出的通用型格密码算法库在运算效率上有了明显提升，尤其当模数q为 14-bit 时，多项式环$\mathbb{Z}_q[x]/(x^{128}+1)$上的乘法速度优于现有的 NTL 库 50 倍。

关键词：后量子密码　格密码　快速数论变换　离散高斯抽样　AVX2 指令集

ABSTRACT：With the remarkable development of quantum computers and the proposal of quantum algorithms such as Shor, the underlying security of traditional public key cryptosystems has been seriously threatened. The solution is to establish post-quantum cryptography (PQC) that can effectively resist quantum computing attacks. Among PQC, lattice-based cryptography is widely regarded as the best candidate. In order to improve the implementation performance of lattice-based cryptography scheme, this paper designs and implements a universal lattice-based cryptography algorithm library. On the basis of sorting out the existing software and hardware implementation strategies of lattice-based cryptography, universal components are abstracted according to lattice algebra structure and difficult problems. Modular reduction is implemented based on the Montgomery algorithm with one fixed multiplier and Barrett algorithm, and polynomial multiplication and discrete Gaussian sampling are implemented based on the number theoretic transform algorithm and Knuth-Yao algorithm. In order to further improve the implementation performance of the

algorithm library, we use AVX2 to parallelize part of the code, and constantize the sampling time of Knuth-Yao to defend against timing side-channel attacks. The components above all build the underlying mathematical layer which provides interfaces for higher-level specific schemes. The experimental results show that the universal lattice-based cryptography algorithm library proposed in this paper has significantly improved the efficiency. Especially when the modulus q is 14-bit, the multiplication speed on polynomial ring $\mathbb{Z}_q[x]/(x^{128}+1)$ is 50 times faster than the NTL library.

KEY WORD： Post-quantum cryptography　Lattice-based cryptography　Number theoretic transform　Discrete Gaussian sample　AVX2

引　言

量子计算技术是基于纠缠、叠加等量子力学现象实现的一种革命性计算技术，可在巨大的希尔伯特空间中进行计算，高效解决经典计算技术所对应问题的超集。1984 年，Peter Shor 提出运行于量子计算机上的 Shor 量子算法，能在多项式时间内求解大数分解难题和离散对数难题，对 RSA、ECC 等传统公钥密码体制的底层安全均造成严重威胁。对于密码行业，如果量子比特足够多的量子计算机成为现实，唯一可行的应对方案是设计能抵御量子计算攻击的后量子密码体制（Post-quantum Cryptosystem, PQC）。

美国国家标准与技术研究院（National Institute of Standards and Technology, NIST）于 2016 年公开征集后量子密码标准，至 2020 年 7 月已公布第三轮候选方案。其中特别地，基于格的密码体制占据主流，且格相关数学问题具有平均情况与最坏情况的困难等价性，因此被广泛认为是 PQC 中的最佳候选者。在保证底层问题安全性的基础上，对于具体实现的性能要求也同样至关重要，特别是在硬件或是时间周期受到严格限制的环境下，因此研究格密码体制通用核心组件的高效实现具有重大意义。

目前对于格密码方案高性能实现的研究路线可分为软件优化与硬件优化。在软件方面，2015 年，Liu 等人[1]基于 RLWE 问题的原始方案提出了一种高效软件实现方法，该实现中包含了对小整数模意义下运算算法、快速数论变换的数据地址转化结构以及高斯分布抽样器的精巧设计，并在 8 比特 AVR 单片机上成功取得了超越 RSA 加密算法的加解密效率。在硬件方面，2016 年，Howe 等人[2]提出了首个标准格上 LWE 型密码方案的纯硬件实现方法，采用错误分布抽样器并行工作的技巧在轻量级的 Spartan-6 平台上

取得了较为理想的实现性能。但由于格密码体系尚未制定完备的行业标准，目前提出的大部分优化策略不具备通用性，无法满足日益增多的上层方案带来的广泛需求。本文根据格代数结构和困难问题抽象出通用核心组件，基于软件优化，设计并实现了一个高效的通用型格密码算法库。该库具备较好的性能与安全性。

本文第 1 节详细阐述抽象出的模约简、多项式乘法与随机抽样三个通用组件的算法优化方案与技术路线，第 2 节介绍实现方法与实现细节，展示并对比分析算法库的性能结果，第 3 节对本文所完成的研究工作加以总结。

1　算法库核心组件设计

对现有格密码软硬件实现策略的调研显示，基于格的密码方案普遍调用模约简、多项式乘法与随机抽样三个通用组件，本节也主要围绕其进行相应的算法库核心模块设计。

1.1　架构设计

本文中通用型格密码算法库的总体架构如图 1 所示，其中底层数学层（Layer-1、Layer-2）提供基本的运算模块，包括素性检测、素数生成和原根生成等用于预处理的算法，硬编码多项式环上的模数及对应原根。在硬编码数据的基础上，设计模约简、多项式环乘法和离散高斯抽样等组件，并使用 AVX2 指令集对部分代码进行优化。应用层（Layer-3）包括对基于 RLWE 的公钥加解密方案的测试，以及数据的编解码。

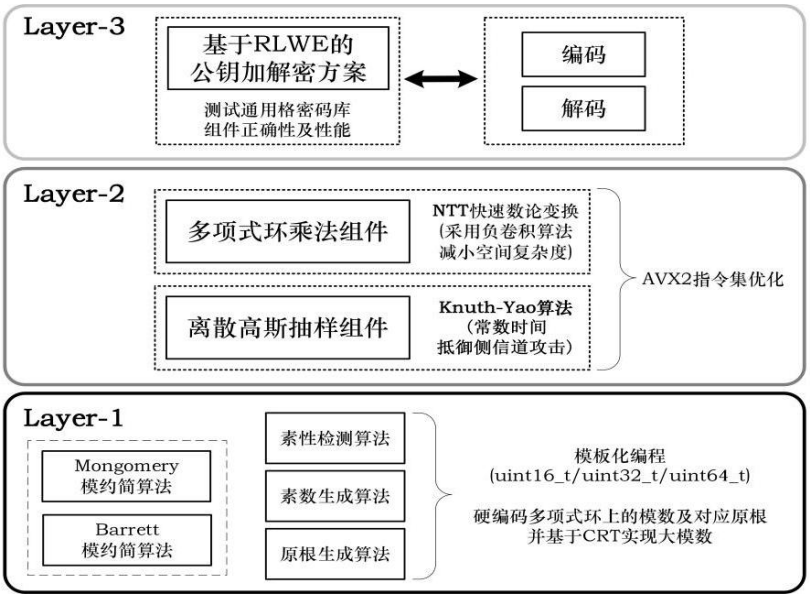

图 1　算法库总体架构

1.2　模约简组件

在 cpp 语言中，由于直接使用取模运算（%）将会调用底层的除法，时间开销较大。

本文使用合适的模约简算法，来进一步提升效率。

（1）Montgomery 模约简算法

选择与模数 N 互素的基数 R，且 $N<R$，一般为机器字长（$2^{16}/2^{32}/2^{64}$）。预计算R^{-1}，N'，满足 $0 < R^{-1} < N$，$0 < N' < R$，$R^{-1}R - N'N = 1$。Montgomery 算法的约简过程如算法 1 所示。

算法 1　Montgomery 模约简
输入：$0 \le x < RN$
输出：$t = xR^{-1} \bmod N$
1. $m \leftarrow (x \bmod R)N' \bmod R$
2. $t \leftarrow (x + mN)/R$　　// $0 \le t < 2N$
3. if $t \ge N$ then return $t - N$ else return t

以模数 N 类型为 uint32_t（无符号 32 位整型），$R = 2^{32}$，待约简的输入 x 类型为 uint64_t 为例，需要满足$N < 2^{31}$，否则第 2 行中的运算将会发生 uint64_t 的上溢出。

由于所选取的基数 R 为 2 的幂次，模 R 和除以 R 的运算可由效率更高的位运算（位与、移位）实现，相较于朴素的取模操作（%）提升了性能。

（2）Montgomery 模约简算法

给定模数 n，预计算正整数$k = \lceil log_2(n) \rceil$，$r = \lfloor 2^{2k}/n \rfloor$，即$2^{k-1} < n < 2^k$，$r < 2^{k+1}$。Barrett 算法输入 $0 \le x < 2^{2k}$，输出$t = x \bmod n$。Barrett 算法全过程中需要占用的最大位长度为$(2k + 2)$比特，因此以模数 n（需 k 比特的位长度）类型为 uint32_t 为例，需满足 $2k + 2 \le 64$，即$n < 2^{31}$，进而防止上溢出。

（3）应用场景分析

Montgomery 算法的输出并非直接的$t = x \bmod N$，而是$t = xR^{-1} \bmod N$。在一般的 Montgomery 模乘中，首先预计算$s = R^2 \bmod N$，再对乘数 a 或 b 计算$s' = Mont(s \cdot a)$或$s' = Mont(s \cdot b)$，最后得到$Mont(s' \cdot b)$或$Mont(s' \cdot a)$即为结果（其中 *Mont* 是约简函数）。单次基于 Montgomery 的模乘中调用了两次约简函数，均未 AVX2 优化的情况下时间开销略大于 Barrett 模乘。但其更适用于固定某一乘数的场景，特别是本文多项式环乘法组件中使用的快速数论变换（NTT），此时的时间性能反而优于 Barrett 算法。因此本文在计算模乘时，若两乘数均不固定，调用单次 Barrett 约简，但某一乘数固定时，仅调用单次 Montgomery 约简。

1.3 多项式环乘法组件

（1）基于 NTT 的多项式乘法

在目前的格密码方案中，多项式环上的乘法占据整个方案运算的较大比例，而教科书式多项式乘法的复杂度为$O(n^2)$，本文使用快速数论变换（NTT）将其降至$O(nlogn)$。基于 NTT 的多项式乘法过程可表示为$c = INTT(NTT(a) \circ NTT(b))$，其中 *NTT* 为是正向 NTT、$\circ$是逐点相乘、*INTT* 是逆向 NTT。实际上即等价于将多项式的系数表示转化为点值表示，再转回系数表示。设α为$\mathbb{Z}_q$上的 n 次单位根，正向 NTT 变换的定义如下所示：

$$\forall i \in \{0,1,...,n-1\},\ A_i = \sum_{j=1}^{n-1} a_j \alpha^{ij},\ A = \{A_0, A_1, ..., A_{n-1}\}$$

$$B_i = \sum_{j=1}^{n-1} b_j \alpha^{ij},\ B = \{B_0, B_1, ..., B_{n-1}\}$$

设$n = 2^l$，将 i 二进制展开为$(i_{l-1}, \ldots, i_1, i_0)_2$，$j$ 二进制展开为$(j_{l-1}, \ldots, j_1, j_0)_2$。并定义$a_0(j_{l-1}, \ldots, j_1, j_0) = a(j_{l-1}, \ldots, j_1, j_0)$，NTT 实现加速的蝶形结构如图 2 所示。

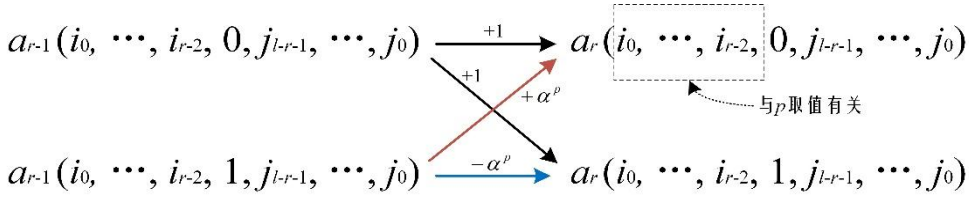

图 2 NTT 蝶形结构

通过蝶形结构的迭代运算，正向 NTT 变换得到$a_l(i_0, \ldots, i_{l-2}, i_{l-1}) = A(i_{l-1}, \ldots, i_1, i_0)$，即 $a(x)$在$\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$处的点值表示，$b(x)$同理。再逐点相乘即为 $c(x)$在相同处的点值表示。一般而言，逆向 NTT 变换的定义如下：$a_i = \frac{1}{n}\sum_{j=1}^{n-1} A_j\, \alpha^{-ij}$，但为进一步提升计算效率，本文不对正向变换后的结果做比特反序的处理，而是将其逐点相乘后直接作为逆向变换的输入，则最后得到的结果即为正序。INTT 的蝶形结构逆变换如图 3 所示。

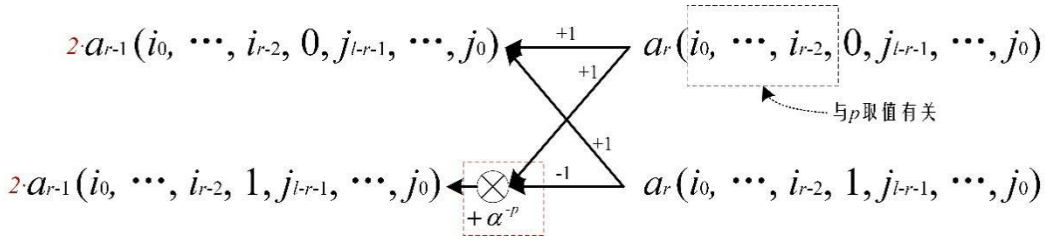

图 3 NTT 蝶形结构逆变换

由于蝶形结构逆变换的输出为真实值的 2 倍，最后的结果还需再乘上$n^{-1}\ mod\ q$。

（2）负卷积技术及其应用

NTT 适用条件为素数$q \equiv 1\ (mod\ n)$，以及环$S_q = \mathbb{Z}_q[x]/(x^n - 1)$，而需要加速环$R_q = \mathbb{Z}_q[x]/(x^n + 1)$上的多项式乘法时，一般将多项式 $a(x)$和 $b(x)$的系数向量增加到 $2n$维，即用 0 补齐，NTT 多项式乘后再显式模$x^n + 1$。但这导致了存储空间的倍增，以及计算效率的下降。本文使用负卷积技术[3]来解决上述问题，具体的定义如下：令η为$\mathbb{Z}_q$上的 $2n$次单位根，即$\eta^2 = \alpha$（这增加了对于素数 q 的约束，需要$q \equiv 1\ (mod\ 2n)$）。对于多项式$h(x)$，有$\tilde{h}(x) = h(\eta x)$，且特别地，当$h(x) = x^n + 1$ 时，$\tilde{h}(x) = -(x^n - 1)$。则一般地，设$h(x) = q(x)(x^n + 1) + r(x)$，有$\tilde{h}(x) = -\tilde{q}(x)(x^n - 1) + \tilde{r}(x)$，因此将多项式 $a(x)$和 $b(x)$预先处理为$\tilde{a}(x)$和$\tilde{b}(x)$，仍基于 n 维的 NTT 得到$\tilde{c}(x)$，最后将系数上η的影响消除即可。使用负卷积技术的 NTT 多项式乘法过程可表示为$c = (1, \eta^{-1}, \eta^{-2}, \ldots, \eta^{-(n-1)}) \circ INTT(NTT(\tilde{a}) \circ NTT(\tilde{b}))$。

1.4 离散高斯抽样组件

（1）基于 Knuth-Yao 的抽样

Knuth-Yao 抽样算法[4]基于离散分布生成树（DDG 树）进行随机游走，可用于高效地在$\mathbb{Z}$上实现离散高斯抽样。给定分布概率$(P_0, P_2, \ldots, P_{n-1})$，离散变量样本点的概率矩阵 p 的第 i 行即为$\mathrm{P_i}$小数部分的二进制展开（展开长度等于精度λ），且满足$\sum_{i=0}^{n-1} P_i = 1$。DDG 树则根据概率矩阵生成（树的第$j + 1$ 层（第一层为根）对应二进制小数点后的第 j 位），包含内部节点和末端节点，其中内部节点对应概率矩阵中的零元素，末端节点对应非零元素。DDG 树与对应概率矩阵的示例如图 4 所示（末端节点编号为抽样结果）。

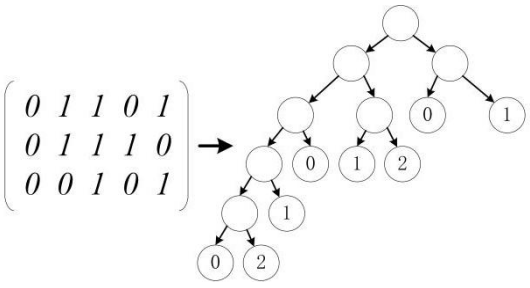

图 4 DDG 树与对应概率矩阵

进行抽样时，从 DDG 树的根节点开始向下随机游走（游走方向服从$p = 0.5$ 的伯努利分布），当遇到内部节点时继续随机游走，直至遇到末端节点，并输出抽样结果。

（2）常数运行时间实现

基于标准 Knuth-Yao 算法的离散高斯抽样存在时钟循环消耗不固定的问题，因此易受到计时侧信道攻击。本文为保证抽样器安全，在已得到抽样结果后，仍游走至 DDG 树的最后一层（不影响输出），使得运行时间常数化，抵御潜在的侧信道攻击。

（3）特殊场景优选方案

假设环上需要采样的噪声多项式服从误差分布χ，且满足$\| \chi \|_\infty = B$。由于$\mathbb{Z}$上的离散高斯抽样值$x \in \{-\tau\sigma, \ldots, \tau\sigma\}$，而预处理的 AVX2 实现中约束了$\tau\sigma < 128$，当$B \geq 128$时，本文基于高斯分布的加和性质，即若$X \sim D_{\mathbb{Z},\sigma_1}$，$Y \sim D_{\mathbb{Z},\sigma_2}$，有$U = X + Y \sim D_{\mathbb{Z},\sqrt{\sigma_1^2+\sigma_2^2}}$，将该场景下的抽样器拆分为多个 Knuth-Yao 抽样器，其中每个标准差σ_i均满足$\tau\sigma_i < 128$。

当抽样器的效率要求较苛刻时，本文使用中心二项分布代替离散高斯分布，这在部分格密码方案中已有采用，且在 2020 年张江[5]等人的论文中被证明了其安全性基础。

1.5 AVX2 高效实现

（1）模约简优化

对于类型为 uint16_t 的模数 N，本文基于 AVX2 指令集实现并行优化。Montgomery 模约简算法中，x 可分为高 16 比特x_{hi}和低 16 比特x_{lo}两部分，因此可并行 16 组 Montgomery 约简，具体流程如图 5 所示。

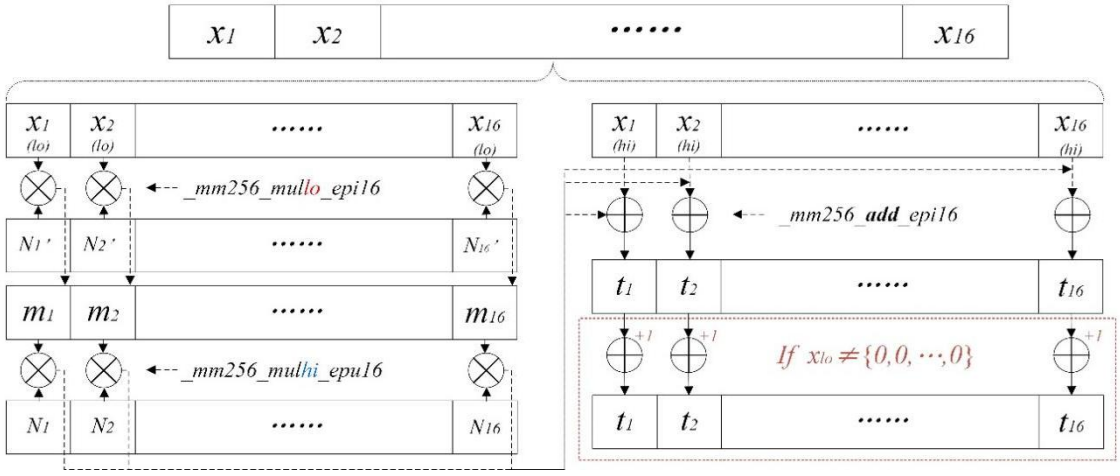

图 5 Montgomery 约简（AVX2 实现）

由于$R = 2^{16}$，$(xN') \bmod R$可交予$_mm256_mullo_epi16$完成，即只处理低 16 位。$(mN)/R$可交予$_mm256_mulhi_epu16$完成，即只处理高 16 位，再与各组输入的高 16 位组成的向量执行$_mm256_add_epi16$，得到约简结果。需要注意的是，与 2021 年杨昊[6]等人对 Montgomery 约简的 AVX2 实现不同，此时的结果仅进行高 16 位的加法（未考虑进位情况），而算法 1 中有$R|(x + mN)$，等价于加法结果的低 16 位为 0，因此当x

的低 16 位不为全 0 时，对上述约简结果向量中的所有元素加一。最后得到的结果落在$[0,2N)$上，本文通过进一步限制 R 与 N 之间的关系（$N<\frac{R}{4}$），来保证乘法结果不超过算法输入的上界RN，进而将每一次的约简结果范围放宽至$[0,2N)$，仅当遇到严格的$[0,N)$区间要求时，才进行单次修正处理，减少时间开销。

（2）多项式环乘法优化

本文同样基于 AVX2 对 NTT 进行加速，以$n=2^5$，$q<2^{16}$为例，设计如图 6 所示（其中同颜色的 16 组 16 比特的数据被载入同一个 256 比特的向量）。

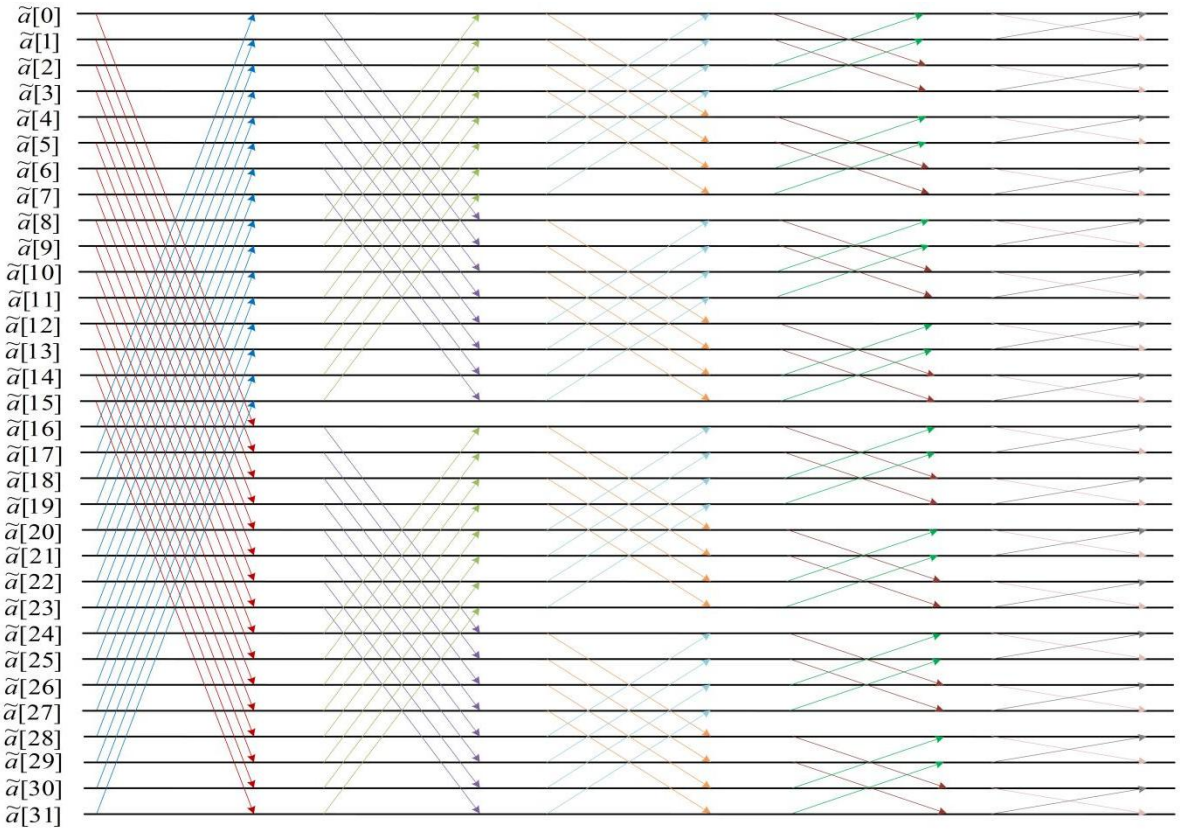

图 6 AVX2 并行化 NTT

执行 NTT 前已预先计算蝶形运算中所需的α^p（组成数组 w），并将数组$\tilde{a}$的指针记作 a。将蝶形结构的跨度记作NP，第一层（$NP=16$）的蝶形运算优化过程如下：调用$_mm256_load_si256(a)$载入向量$vec1$，$_mm256_load_si256(a+NP)$载入向量$vec2$，$_mm256_set1_epi16((w[0]\cdot R)\%q)$载入向量$vec3$（$w[0]\cdot R$同样为预计算数据，作为 Montgomery 模乘的固定乘数）。再将 $\_mm256\_mullo\_epi16(vec2,vec3)$ 与 $_mm256_mulhi_epu16(vec2,vec3)$的结果作为 AVX2 实现的 Montgomery 模约简输入，得到模 q 下$w[0]\cdot vec2$ 的向量表示$vec4$。最后调用$_mm256_add_epi16(vec1,vec4)$与 $\_mm256\_sub\_epi16(vec1,vec4)$，并退出向量化，转换回一般数组表示。由于约简结果

落在$[0,2q)$上，上述数组中的元素也均落在$[0,4q) \cup (R-2q,R)$上，其中后者是由于无符号类型发生下溢出。数组中的元素至多只需减去或加上（发生上溢出）1 次 $2q$即可。

倒数四层的蝶形运算（$NP=8,4,2,1$）需要额外在载入向量阶段做排列处理。以倒数第四层为例，调用$_mm256_load_si256(a)$载入向量$vec1$，$_mm256_load_si256(a+16)$载入向量$vec2$ 后，排列处理如图 7 所示。

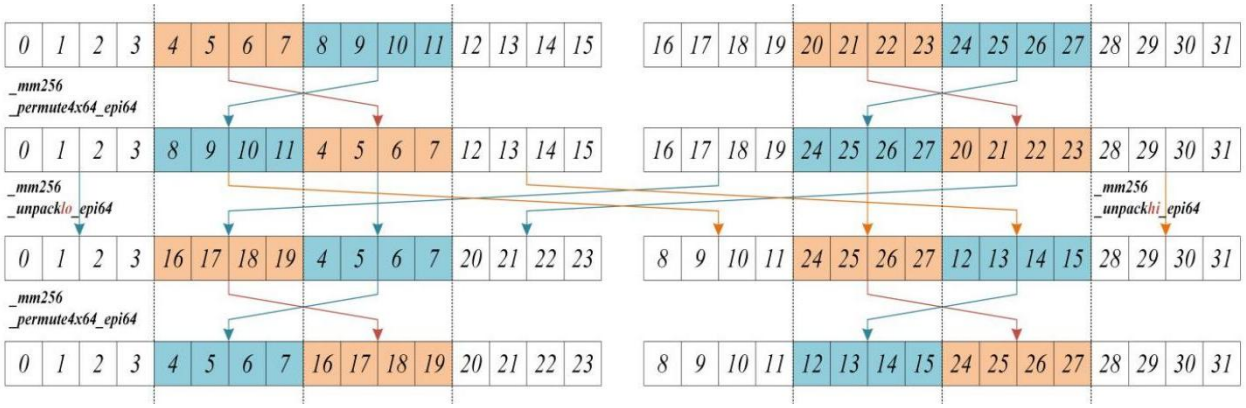

图 7 蝶形运算排列处理（倒数第四层）

（3）抽样器优化

本文预计算概率矩阵中各列的汉明重量$weight[\lambda]$（其中λ为精度），实现抽样结果所在列的快速定位。预处理过程使用 AVX2 做并行优化，由于 AVX2 向量要求地址以 32 字节对齐，本文将λ也约束为 32 的倍数。除此之外，约束$n \le 256$，使得概率矩阵$p[n][\lambda]$和各列汉明重量$weight[\lambda]$中元素的类型统一为 uint8_t，即并行预处理概率矩阵的 32 列。

2 算法库实现与结果分析

2.1 模约简

本文对非 AVX2 版本的模约简进行性能测试，发现 Montgomery 模乘的运算速度略低于 Barrett 模乘，但固定某一乘数后，后者单次模乘时间又长于前者。由于测试数据位长在 16 比特时的除法开销差异不明显，cpp 自带的取模运算（%）效率反而优于约简算法，本文选择在 16 比特时基于 AVX2 实现并行优化，最终方案的性能测试结果如图 8 所示，可以看出固定某一乘数时，Montgomery 模乘效率均大幅优于其他方法。

2.2 多项式环乘法

本文对多项式环乘法组件中逐点相乘的部分也进行了优化：非 AVX2 模式下，使用 Barrett 模乘（因为此时无固定乘数，Barrett 效率更优于一般的 Montgomery 模乘）；AVX2 模式下，仍基于 Montgomery 模乘，且约束多项式环的度数为 16 的倍数。

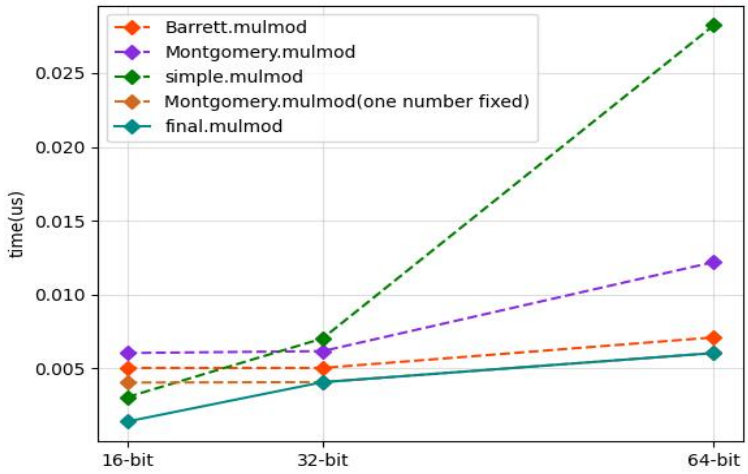

图 8 模约简算法性能测试（16-bit 基于 AVX2 优化）

对本算法库的多项式环乘法组件进行性能测试，并与现有 NTL 库（版本为 11.4.3）在ZZ_pX上的$MulMod$方法对比，结果如图 9 和图 10 所示。

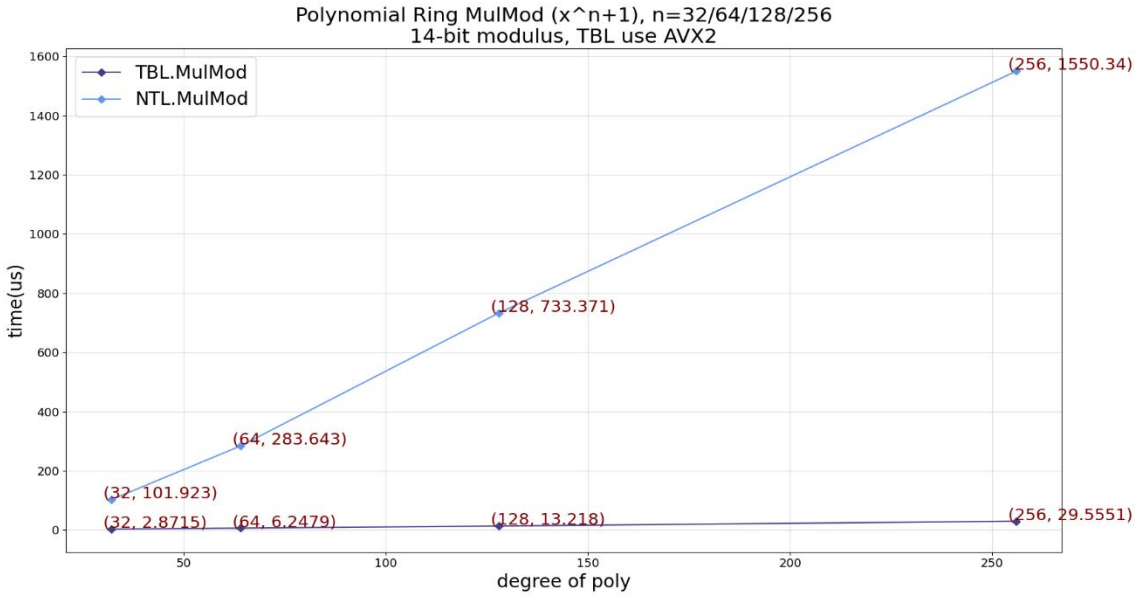

图 9 多项式环乘法组件性能测试（16-bit 基于 AVX2 优化）

其中 16 比特数据表示下，基于 AVX2 优化。实验测试范围包括 32/64/128/256 比特的多项式环度数，可以看到 NTL 做单次乘法的耗时分别是本文的 35/45/55/52 倍。

32 比特数据表示下，不基于 AVX2 实现。实验测试范围仍然包括 32/64/128/256 比特的多项式环度数，此时 NTL 做单次乘法的耗时分别是本文的 23/28/40/41 倍。

2.3 应用层测试

算法库的最上层为具体的基于 RLWE 的公钥加解密方案，如下所示：

设$f(x) = x^n + 1 \in Z[x]$，对应的多项式环$R_q = \mathbb{Z}_q[x]/(f(x))$。

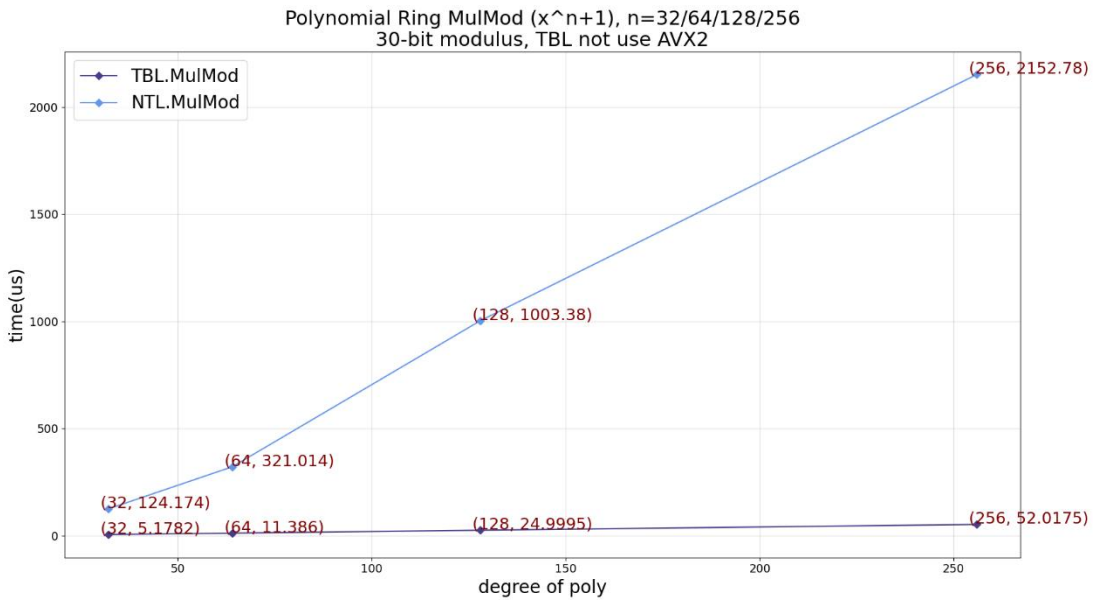

图 10 多项式环乘法组件性能测试（32-bit 无 AVX2）

（1）密钥生成：在$\mathbb{Z}_q^n$上均匀随机采样环R_q中的多项式 g，在误差分布χ上采样小系数多项式 s 和 e。则私钥$sk = s$，公钥$pk = (g, t = gs + e) \in R_q^2$。

（2）加密：n 比特的明文$z \in \{0,1\}^n$即对应环R_q上的一个元素。在误差分布χ上采样小系数多项式r, e_1, e_2，计算$u = g \cdot r + e_1 \ (mod\ q)$，$v = t \cdot r + e_2 + \lfloor q/2 \rfloor \cdot z \ (mod\ q)$。则密文为$(u, v) \in R_q^2$。

（3）解密：对于密文(u, v)，计算

$$\begin{aligned} v - u \cdot s &= g \cdot s \cdot r + r \cdot e + e_2 + \lfloor q/2 \rfloor \cdot z - g \cdot s \cdot r - s \cdot e_1 \\ &= (r \cdot e + e_2 - s \cdot e_1) + \lfloor q/2 \rfloor \cdot z \end{aligned}$$

需要设置合适的参数，使得$(r \cdot e + e_2 - s \cdot e_1)$的各系数模 q 落在$(-\lfloor q/4 \rfloor, \lfloor q/4 \rfloor)$上，即可通过判断上述计算结果落在$(0, \lfloor q/4 \rfloor) \cup (\lfloor 3q/4 \rfloor, q)$上还是$(\lfloor q/4 \rfloor, \lfloor 3q/4 \rfloor)$上，确定 z 的各系数为 0 还是 1。

定义 1[7] 设环$R = Z[x]/(f(x))$，称$\delta_R = max\{\frac{\|a(x) \cdot b(x)\|_\infty}{\|a(x)\|_\infty \cdot \|b(x)\|_\infty} : a(x), b(x) \in R\}$为环 R 的扩张因子，其中无穷范数$\| a(x) \|_\infty = max_i |a_i|$。

定理 1[7] 设环$R = Z[x]/(f(x))$，若$f(x) = x^n + 1$，则环上的扩张因子$\delta_R = n$。

假设误差分布χ满足$\| \chi \|_\infty = B$，则$(r \cdot e + e_2 - s \cdot e_1)$系数的最大绝对值小等于$2\delta_R \cdot B^2 + B$。因此参数设置满足 $2\delta_R \cdot B^2 + B < q/4$ 时，该方案加解密正确性成立。

本文通过其对上述所有通用核心组件的正确性与性能进行整体测试。具体实验中，

使噪声多项式服从的误差分布χ满足$\| \chi \|_\infty = 16$，多项式环度数$n = 128$，模数$q = 1073479681$，因此参数设置满足要求。将明文按 16 字节一组进行加密（不足则用 0 补齐），对应的每组密文大小为 1KB。加解密的正确性验证如图 11 所示，可以看出明文与解密后的结果一致。

图 11 RLWE 公钥加密正确性验证

3 结束语

本文对通用型格密码算法库的高效实现进行了探索，并构建了一套基于软件优化的多层架构的算法库，为具体格密码方案提供底层接口，具体包括：

（1）设计了通用型格密码算法库的总体架构，并基于格密码的数学基础抽象出算法库需实现的通用组件。

（2）设计并实现了基于 Montgomery 的模约简组件（相较于原始 Montgomery 算法放宽数据区间，减少一定开销），并在 16 比特模数下基于 AVX2 实现并行优化。

（3）设计并实现了多项式环$\mathbb{Z}_q[x]/(x^n+1)$上基于 NTT 的多项式乘法（其中包括更为高效的逆向 NTT 变换，以及融合负卷积技术的有效降低内存开销），并在 16 比特模数下基于 AVX2 实现并行优化。

（4）设计并实现了基于 Knuth-Yao 的离散高斯分布抽样器，将其运行时间常数化，以抵御可能的计时侧信道攻击，并对预处理环节进行 AVX2 并行优化。最后基于高斯分布的加和性质，对抽样范围较大时的场景也提出了多个抽样器组合的方案。

（5）通过基于 RLWE 的具体公钥加解密方案，对本文的算法库进行正确性校验，并与 NTL 库进行性能对比测试，结果显示多项式环$\mathbb{Z}_q[x]/(x^n+1)$上乘法的整体速度优于 NTL 库的 MulMod 方法 30 倍以上。

参考文献

[1] Liu Z, Seo H, Sinha Roy S, et al. Efficient Ring-LWE Encryption on 8-Bit AVR Processors[C]. Cryptographic Hardware and Embedded Systems (CHES) 2015, Lecture Notes in Computer Science (LNCS), 2015, 9293:663-682.

[2] James Howe, Ciara Moore, M. O'Neill, et al. Standard lattices in hardware[C]. 53rd Annual Design Automation Conference, 2016: 1-6.

[3] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, et al. SWIFFT: A modest proposal for FFT hashing[C]. FSE 2008, Lecture Notes in Computer Science, 2008, 5086: 54-72.

[4] DWARAKANATH N C, GALBRAITH S D. Sampling from discrete gaussians for lattice-based cryptography on a constrained device[J]. Communication and Computing, 2014, 25(3): 159-180.

[5] 张江, 范淑琴. 关于非对称含错学习问题的困难性研究[J]. 电子与信息学报, 2020, 42(2): 327-332.

[6] 杨昊, 刘哲, 黄军浩, 等. AKCN-MLWE算法AVX2高效实现[J]. 计算机学报, 2021, 44(12): 2560-2572.

[7] 王爱兰,宋巍涛,赵秀凤. 剩余类环上扩张因子的性质[J]. 山东大学学报 (理学版), 2018, 53(11): 78-84.