Nanjing University of Posts and Telecommunications

基于RISC V的后量子密码Classic McEliece优化实现设计

汇报人：胡啸宇

CONTENTS 目录

PART ONE

项目背景

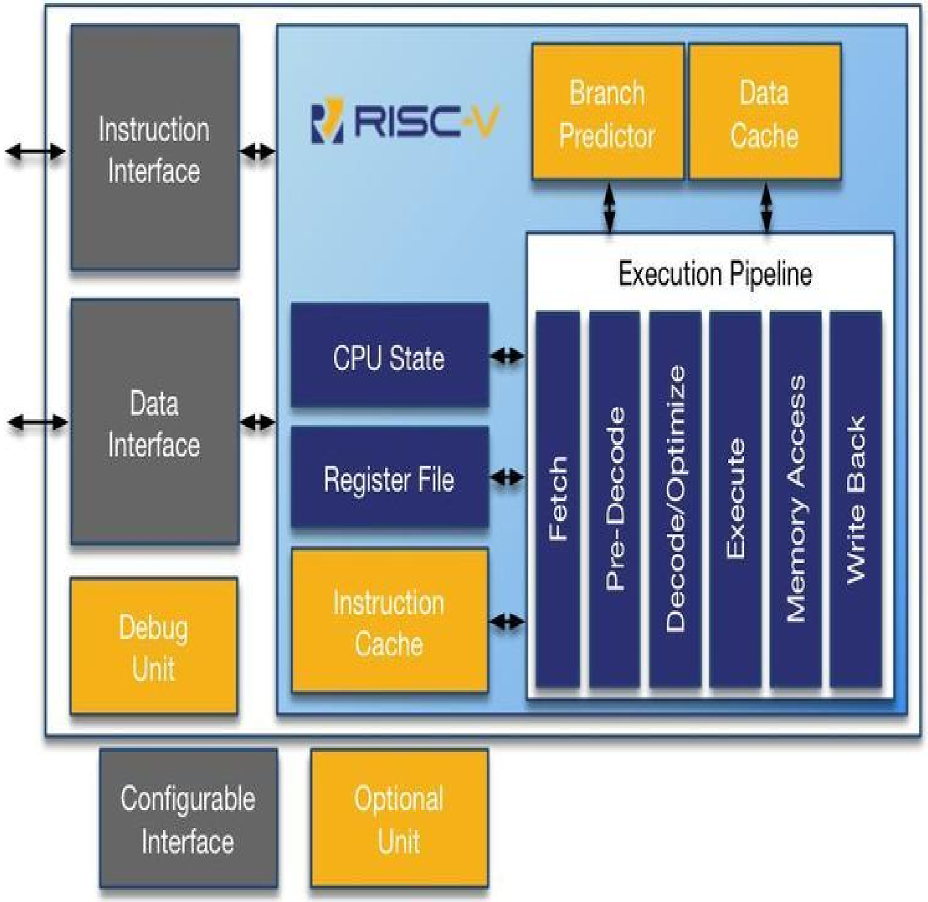
1.1 经典Mceliece

	Finalists	Alternates
KEMs/Encryption	Kyber NTRU SABER Classic McEliece	Bike FrodoKEM HQC NTRUprime SIKE
Signatures	Dilithium Falcon Rainbow	GeMSS Picnic SPHINCS+

NIST标准化进程第三轮PQC候选名单

经典McEliece是一种基于编码的量子密码算法，被广泛认可为安全可靠的加密方案之一。经典McEliece算法在NIST PQC（Post-Quantum Cryptography）竞赛中是唯一一个基于编码的密码算法，其安全性基于编码问题的困难性。这使得经典McEliece成为一种备受关注和研究的后量子密码方案，具有潜力在未来量子计算机兴起后仍然能够提供安全的数据加密和通信保护。

1.2 RISC-V



- ✓ **RISC-V指令集架构（ISA）是由安德鲁•沃特曼和加州大学伯克利分校的其他人在2010年提出的，其特点是CPU指令种类少、功能单纯,相对于成熟的Intel X86架构,RISC-V架构的最大优势在于精简的指令集使其适用于高速运行的场景。RISC-V是一种高质量、开放架构，低功耗的指令集架构，由RISC-V基金会负责维护和推动。其设计目标是实现代码开源和指令集开放。使用RISC-V架构设计的芯片具有较低的面积功耗、更简洁的结构，并且在面积功耗和性能之间取得更好的平衡，使其在实际应用中具备更高的性价比。**

1.3 课题意义

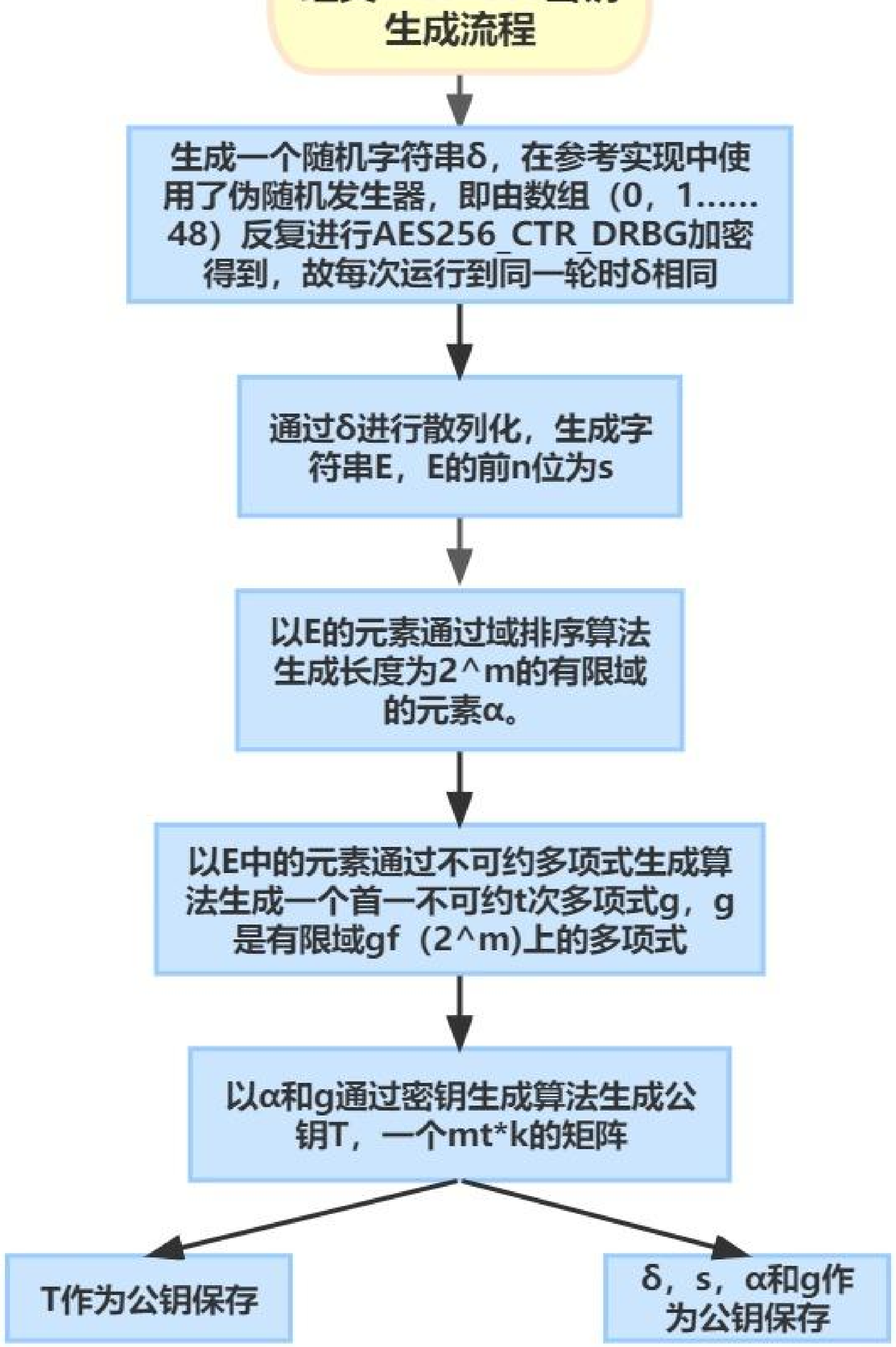
- ✓ 将经典McEliece算法应用于RISC-V指令集架构中，充分利用RISC-V架构的优势，如精简而规范的指令集和流水线处理,不仅能够保持算法的安全性，还能大幅提高在硬件层面上的执行效率和性能。
- ✓ 算法的实现和优化对于数据安全具有重要意义
- ✓ 促进算法在硬件上的广泛应用，进一步推动密码学与计算机体系结构的融合，推动密码学领域的研究和发展

PART TWO

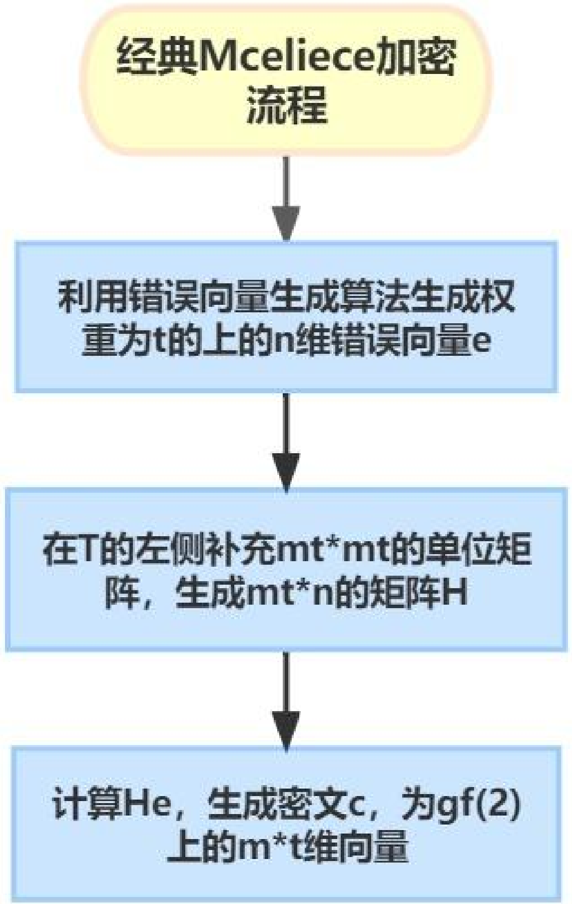
2. 相关知识

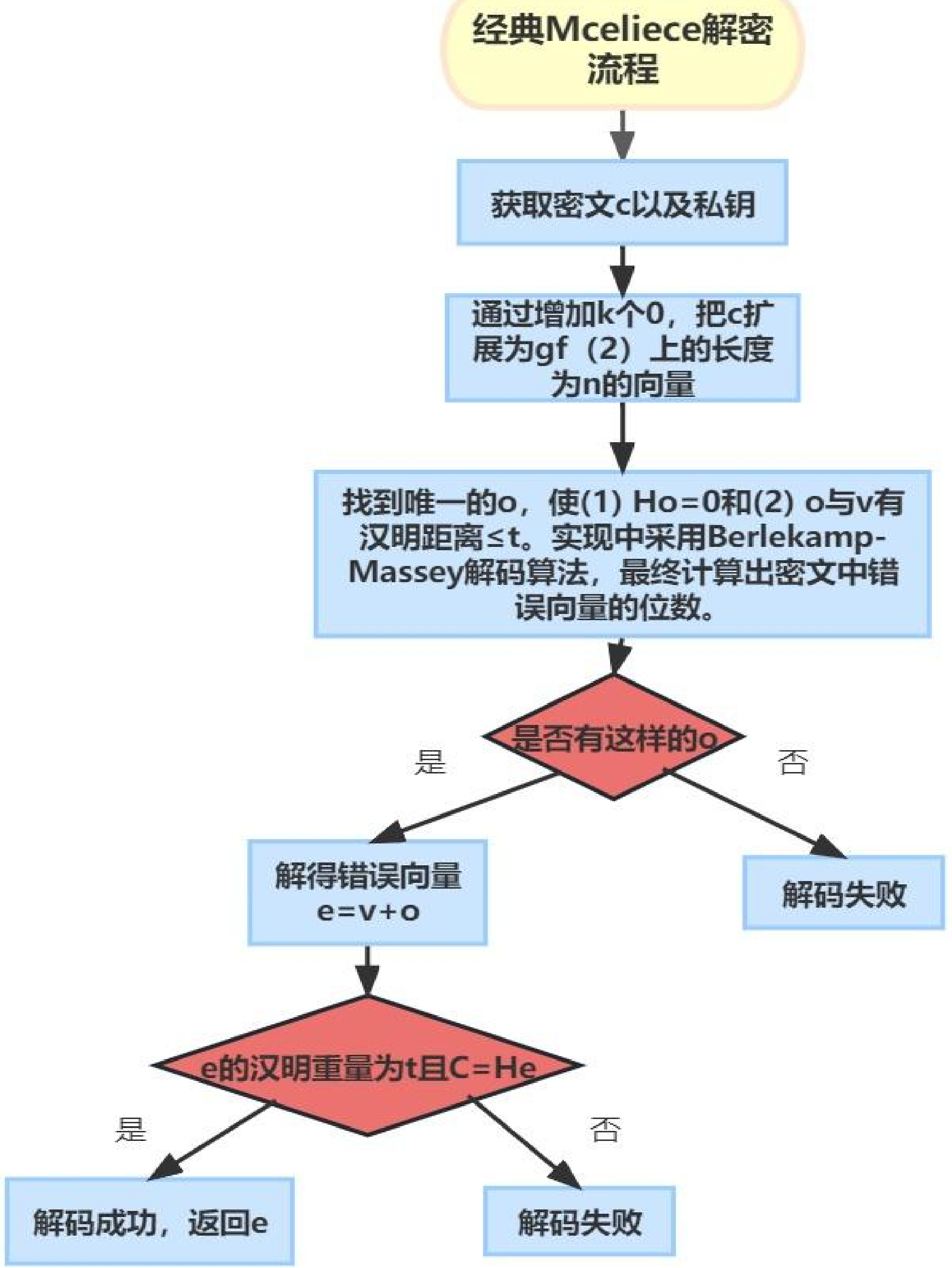
2.1 Mceliece运行流程

- ✓ **McEliece算法基于线性码的理论，利用Goppa码的纠错能力和困难编码问题的难解性构建了一个安全的公钥密码系统。**
- ✓ **n,m,t,k,q为算法中使用的参数，其中n为编码长度，k为编码维度，t为纠错能力，q为所使用的字段的大小，有$m=2^q$,$n=mt+k$。**



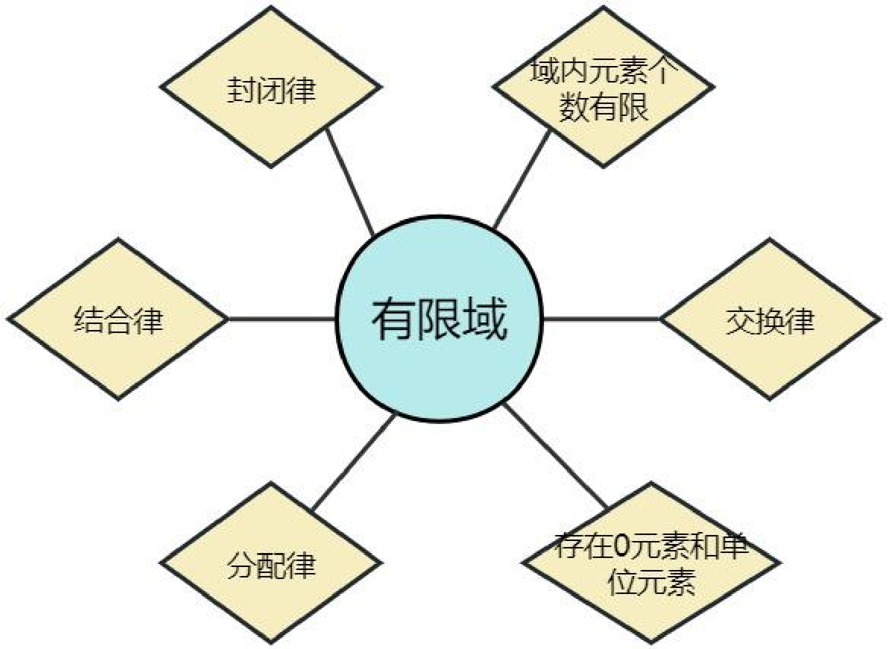
2. Mceliece运行流程





2.2 数论 有限域

有限域:仅含有限多个元素的域



加法运算是模q的求和运算，乘法运算是模q的乘法运算

2.2 数论 有限域

通过多项式环的商环构造有限域的步骤:

a.选择一个素数p作为有限域的特征，通常p = 2。

b.构造一个p次多项式环$F_p[x]$，即多项式的系数取自特征为p的有限域F_p。

c.选择一个正整数n作为有限域的扩展次数，假设$q=p^n$，选择一个n次的不可约多项式f(x)作为$F_p[x]$中的不可约多项式，确保它没有任何根。

d.定义一个等价关系，它表示在Fp[x]中两个多项式在f(x)下的模等价。

e.构造商环$F_p[x]/(f(x))$，即将Fp[x]中所有多项式除以f(x)得到的多项式的等价类。

商环$F_p[x]/f(x)$是一个元素个数为q的有限域，其中的元素表示为多项式的等价类。这里的有限域中的加法和乘法运算定义为多项式的加法和乘法，并对结果取模f(x)。

简单来说就是一个元素个数为2^q次方的有限域可以用有q项的多项式表示，这个多项式的系数就是元素个数为2的有限域的元素，加法和乘法就是多项式的加法和乘法再对f(x)取模。

PART THREE

3 m位的有限域的计算优化

3.1 实验设计

在参数m= 12，n = 3488，t= 6的经典的Mceliece加解密算法中，12位的有限域是非常重要的数据结构，许多重要的计算步骤都围绕其展开和衍生。在经典的Mceliece实现中，先是以F生成多项式域$F_{2^m}(F_2[z]/z12+z3+1)$，**域元素以F_2上的m次多项式表示，系数即F_2的元素**，即0，1，域内的乘，加计算即多项式的乘，加。在有限域F_{2^m}内，加法和减法都可以以异或操作表示，乘法操作可以以相与操作表示。

此处数据结构：gf(short int)，其中每一位对应多项式的一项

010000011000 $\xrightarrow{\text{代表}}$ x^3+x^4+x^10

3.1 实验设计

```
asm volatile(
"mv x5,%0\n"
"li x6,0x7FC \n"
"slli x6,x6,12\n"
"and x6, x5, x6 \n"
"srli x6, x6, 9 \n"
"xor x5, x5, x6\n"
"srli x6, x6, 3 \n"
" xor x5, x5, x6\n"
"li x6,0x3 \n"
"slli x6,x6,12 \n"
"and x6, x5, x6\n"
"srli x6, x6, 9 \n"
"xor x5, x5, x6\n"
"srli x6, x6, 3 \n"
" xor x5, x5, x6\n"
"li x6,1\n"
"slli x6, x6, 12 \n"
"addi x6, x6, -1 \n"
"and %0, x5, x6 \n"
:"+r"(tmp)
:
:"x5","x6"
);
```

- ✓ 优化策略：采用RISC-V架构的高效指令 完成算法中的步骤，以便合理利用寄存器，充分利用RISC-V指令集的优势，达成性能的提升。
- ✓ 实现方法：利用asm volatile 嵌套汇编指令

3.2 乘法部分

代码 1：乘法部分实现指令代码	
输入：t0(%1)，t1(%2)	5：sll x5,x5,x7
输出：tmp(%0)	6：and x5,%1,x5
1：li x6,12	7：mul x5,x5,%2
2：li x7,1"	8：xor %0,x5,%0
3：loop%=:	9：addi x7,x7,1
4：li x5,1	10：blt x7,x6,loop%=

乘法部分使用schoolbook的多项式乘法。如第5行所示，移位操作由1向左执行，并执行and操作。这个移位操作是一个将值增加1到*m*，并以同样的方式执行从第5行到第7行的操作的过程。最后，可以看到*m*值为12。经过这些操作，有F2^12上的乘法的结果。在此之中，输入为t0和t1，输出结果至tmp。在此代码中，共进行12轮循环，每一轮循环都将t1的对应位数乘以t0，并加到tmp中，通过此操作完成多项式乘法的计算。

PART
FOUR

4 多项式乘法优化

4.1 相关设计

在参数m=12，n = 3488,t = 64的经典的 Mceliece加解密算法实现中，$F_{(2^m)t}$是以F2^m生成的扩域$F_{2^m}[y]/y^6+ + y^3+y+z$，**域元素以$F_{2^m}$上的t次多项式表示,系数即$F_{2^m}$内的元素**，域内的乘，加计算即多项式的乘，加，并且依赖于F_{2^m}内的操作。多项式乘法的优化主要针对这一层的计算。

此处的数据结构：长度为t的gf数组，组内每一元素对应多项式的一个系数。

4.2 schoolbook多项式乘法

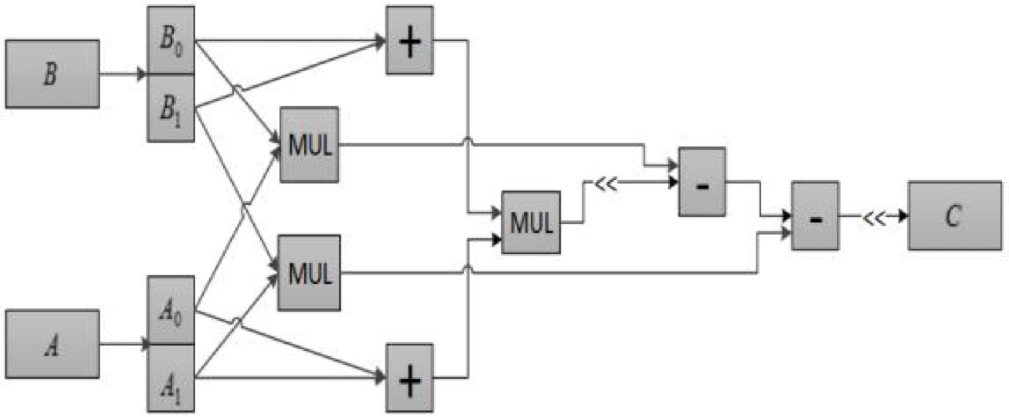
Schoolbook多项式乘法是一种基本的多项式乘法算法。它通过逐项相乘并累加的方式实现乘法操作。

- ✓ 简单直接，泛用性高。

- ✓ 时间复杂度较高，随着多项式的阶数增加，计算时间呈现出二次增长的趋势。
- ✓ 没有充分利用多项式乘法的结构特点，导致计算过程中存在冗余的乘法和加法操作。

4.3 karatsuba多项式乘法

Kartasuba算法采用了分治策略，将原始的多项式乘法问题分解为较小规模的子问题，通过递归地求解这些子问题，可以降低计算复杂度。
Karatsuba 算法将输入的乘数和被乘数拆分成两个二项多项式，再通过数学合并的方式只使用三个小的乘法进行计算，节省一个乘法操作。



4.3 karatsuba多项式乘法

$$A = a_1 * 2^{n/2} + a_0$$

$$B = b_1 * 2^{n/2} + b_0$$

⇩

$$C = A * B = a_1 * b_1 * 2^{2*(n/2)} + (a_1 * b_0 + a_0 * b_1) * 2^{n/2} + a_0 * b_0$$

⇩

$$\begin{aligned} C = A * B = {} & a_1 * b_1 * 2^{2*(n/2)} \\ & + \{(a_1 + a_0) * (b_1 + b_0) - a_1 * b_1 - a_0 * b_0\} * 2^{n/2} \\ & + a_0 * b_0 \end{aligned}$$

算法 2：*karatsuba(F,G,n)*

输入： 两个 n 位的多项式 F，G，其中$F = f_0 + f_1 t + ... + f_{2k-1} t^{2k-1}$，$G = g_0 + g_1 t + ... + g_{2k-1} t^{2k-1}$，$n$ 为多项式的长度，即 $n=2k$

输出： $R = F * G$

1. 如果 n 为 1，直接计算返回结果。
2. 设$F0 = f_0 + f_1 t + ... + f_{k-1} t^{k-1}$，$F1 = f_k + f_{k+1} t + ... + f_{2k-1} t^{k-1}$，$G0 = g_0 + g_1 t + ... + g_{k-1} t^{k-1}$，$G1 = g_k + g_{k+1} t + ... + g_{2k-1} t^{k-1}$，则$F = F0 + F1 t^k$，$G = G0 + G1 t^k$。
3. 计算$R0 = karatsuba$（$F0$，$G0$，$n/2$）
4. 计算$R1 = karatsuba$（$F1$，$G1$，$n/2$）
5. 计算$R2 = karatsuba$（$F0 + F1$，$G0 + G1$，$n/2$）
6. 计算结果$R = R0 +$（$R0 + R1 + R2$）$t^{n/2} + R1 t^n$

4.3 karatsuba多项式乘法

类似地，有分割为3项和分割为5项的karatsuba算法，每一轮次递归时能更多地减少乘法次数。

由于递归过程中的性能损耗较多，实际测试中karat-3和karat-5并未比karat-2有较大提升。

算法 3：*Karat3(F,G,n)*

输入： 两个 n 位的多项式 F，G，其中$F=f_0+f_1t+...+f_{3k-1}t^{3k-1}$，$G=g_0+g_1t+...+g_{3k-1}t^{3k-1}$，$n$ 为多项式的次数，即 $n=3k$

输出： $R=F*G$

1. 如果 n 为 1，直接计算返回结果。
2. 设$F0=f_0+f_1t+...+f_{k-1}t^{k-1}$，$F1=f_k+f_{k+1}t+...+f_{2k-1}t^{k-1}$，$F2=f_{2k}+f_{2k+1}t+...+f_{3k-1}t^{k-1}$,$G0=g_0+g_1t+...+g_{k-1}t^{k-1}$,$G1=g_k+g_{k+1}t+...+g_{2k-1}t^{k-1}$，$G2=g_{2k}+g_{2k+1}t+...+g_{3k-1}t^{k-1}$则$F=F0+F1t^k+F2t^{2k}$ ，$G=G0+G1t^k+G2t^{2k}$
3. 计算$R0=karat3$（$F0$，$G0$，$n/3$)
4. 计算$R1=karat3$（$F1$，$G1$，$n/3$)
5. 计算$R2=karat3$（$F2$，$G2$，$n/3$)
6. 计算$R3=karat3$（$F0+F1$，$G0+G1$，$n/3$）
7. 计算$R4=karat3$（$F0+F2, G0+G2, n/3$)
8. 计算$R5=karat3$（$F1+F2$，$G1+G2$，$n/3$）
9. 计算结果$R=R0+(R0+R1+R3)t^{n/3}+(R0+R1+R2+R4)t^{n*2/3}+$（$R1+R2+R5$）$t^n+R2t^{n*4/3}$。

PART
FIVE

5 展望

5.2 未来展望

- ✓ **关注矩阵约减部分的优化，将高斯约减优化为LUP约减。**
- ✓ **进一步探究RISC-V的高效指令，利用RISC-V指令完成更多优化。**
- ✓ **尝试更多的多项式算法，包括Toomcook等。**
- ✓ **……**

THANKS

请老师批评指导