Groth16方案分析

原理分析

Groth16: zk-snark工程实现的一种，主要在PGRH13的基础上进行了优化，实现了 3 个点（最少 2 个点）完成证明验证的过程.

验证过程

初始化设置

- 设置生成元g_1, g_2
- 设置g_1上点 $\alpha, \beta, \delta,\ \{x^i\}_{i=0}^{n-1}, \{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\gamma}\}_{i=0}^{l}, \{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}\}_{i=l+1}^{m}, \{\frac{x^i t(x)}{\delta}\}_{i=0}^{n-2}$
- 设置g_2上点 $\alpha, \beta, \delta, \gamma, \{x^i\}_{i=0}^{n-1}$

上述点需要采用MPC多方协同计算获得

证明者

- 选取随机数r和s
- 使用生成元g_1计算点A加密值,$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta$
- 使用生成元g_2计算点B加密值,$B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta$
- 使用生成元g_1计算点C加密值,$C = \frac{\sum_{i=l+1}^{m} a_i(\beta u_i(x)+\alpha v_i(x)+w_i(x))+h(x)t(x)}{\delta} + As + rB - rs\delta$

验证者

- 验证等式关系是否成立

  $[A]_1 \cdot [B]_2 = [\alpha]_1 \cdot [\beta]_2 + [\sum_{i=0}^{l} \frac{\sum_{i=0}^{l} a_i(\beta u_i(x)+\alpha v_i(x)+w_i(x))}{\gamma}]_1 [\gamma]_2 + [C]_1 \cdot [\delta]_2$

  利用双线性映射实现

  $e(g_1^A, g_2^B) = e(g_1^\alpha, g_2^\beta) \cdot e(g_1^{\sum_{i=0}^{l} \frac{\sum_{i=0}^{l} a_i(\beta u_i(x)+\alpha v_i(x)+w_i(x))}{\gamma}}, g_2^\gamma) \cdot e(g_1^C, g_2^\delta)$

完备性证明

$$
\begin{aligned}
AB &= (\alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta)(\beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta) \\
&= \alpha\beta + \alpha\sum_{i=0}^{m} a_i v_i(x) + \alpha s\delta + \beta\sum_{i=0}^{m} a_i u_i(x) + \sum_{i=0}^{m} a_i u_i(x) \sum_{i=0}^{m} a_i v_i(x) + s\delta\sum_{i=0}^{m} a_i u_i(x) + r\delta\beta + r\delta\sum_{i=0}^{m} a_i v_i(x) + rs\delta^2 \\
&= \alpha\beta + \sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + \sum_{i=l+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + h(x)t(x) + \alpha s\delta + s\delta\sum_{i=0}^{m} a_i u_i(x) + rs\delta^2 + r\delta\beta + r\delta\sum_{i=0}^{m} a_i v_i(x) + rs\delta^2 - rs\delta^2 \\
&= \alpha\beta + \sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + \sum_{i=l+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + h(x)t(x) + As\delta + rB\delta - rs\delta^2 \\
&= \alpha\beta + \frac{\sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\gamma}\gamma + C\delta
\end{aligned}
$$

安全性证明

Groth16方案是在PGRH13方案的基础上将一系列验证进行了合并

多项式插值定理

在PGRH13方案中零知识证明是基于拉格朗日插值定理实现，$L(x)R(x) - O(x) = h(x)t(x)$。

验证方法：$e(g^{L(x)}, g^{R(x)}) = e(g^{h(x)}, g^{t(x)})e(g^{O(x)}, g)$

KOE假设

在PGRH13方案中，使用KOE假设来完善安全性，

为保证证明人按照固定多项式生成规则，使用α对

验证方法：$e(g^{L(x)}, g^{\alpha}) = e(g^{aL(x)}, g)$

为保证保证L(x),R(x),O(x)使用相同输入，使用β对：

验证方法：$e(g^{L(x)+R(x)+O(x)}, g^{\beta}) = e(g^{\beta(L(x)+R(x)+O(x))}, g)$

为保证α对无法相互影响，使用不同的α对

验证方法：$e(g^{L(x)}, g^{\alpha_1}) = e(g^{aL(x)}, g)$

验证方法：$e(g^{R(x)}, g^{\alpha_2}) = e(g^{a_2 R(x)}, g)$

验证方法：$e(g^{O(x)}, g^{\alpha_3}) = e(g^{a_3 O(x)}, g)$

验证方法：$e(g^{\alpha_1 L(x)+\alpha_2 R(x)+\alpha_3 O(x)}, g) = e(g^{L(x)}, g^{\alpha_1})e(g^{R(x)}, g^{\alpha_2})e(g^{O(x)}, g^{\alpha_3})$

在Groth16方案中将插值定理和KOE假设验证方法合并，并且添加一些乘积项作为辅助项，则可以得到：

$L(x)R(x) + \alpha R(x) + \beta L(x) + \alpha\beta = (L(x) + \alpha)(R(x) + \beta) = \alpha\beta + \alpha R(x) + \beta L(x) + O(x) + h(x)t(x)$

假设$A = L(x) + \alpha, B = R(x) + \beta$

验证方法$e(g^{A}, g^{B}) = e(g^{\alpha}, g^{\beta})e(g^{\alpha R(x)+\beta L(x)+O(x)+h(x)t(x)}, g)$

既完成了插值定理的验证，也实现了KOE假设中α对β对的验证

随机偏置

在PGRH13方案中为了防止暴力破解，使用随机偏置

Suppose $L'(x) = L(x) + \delta_1 T(x), R'(x) = R(x) + \delta_2 T(x), O'(x) = O(x) + \delta_3 T(x)$

$$
\begin{aligned}
\text{Then } f'(x) &= L'(x) * R'(x) - O'(x) \\
&= (L(x) + \delta_1 T(x))(R(x) + \delta_2 T(x)) - (O(x) + \delta_3 T(x)) \\
&= (L(x)R(x) - O(x)) + L(x)\delta_2 T(x) + \delta_1 T(x)R(x) + \delta_1\delta_2 T(x)T(x) - \delta_3 T(x) \\
&= T(x)(H(x) + L(x)\delta_2 + R(x)\delta_1 + \delta_1\delta_2 T(x) - \delta_3) \\
H'(x) &= (H(x) + L(x)\delta_2 + R(x)\delta_1 + \delta_1\delta_2 T(x) - \delta_3)
\end{aligned}
$$

在Groth16方案中A，B中也添加类似的随机偏执$r\delta, s\delta$

$$
\begin{aligned}
&AB = (L(x) + \alpha + r\delta)(R(x) + \beta + s\delta) \\
&= \alpha\beta + \alpha R(x) + \beta L(x) + O(x) + h(x)t(x) + r\delta(R(x) + \beta + s\delta) + s\delta(L(x) + \alpha) \\
&= \alpha\beta + \alpha R(x) + \beta L(x) + O(x) + h(x)t(x) + r\delta(R(x) + \beta + s\delta) + s\delta(L(x) + \alpha + r\delta) - s\delta r\delta \\
&= \alpha\beta + \alpha R(x) + \beta L(x) + O(x) + h(x)t(x) + r\delta(R(x) + \beta + s\delta) + s\delta(L(x) + \alpha) \\
&= \alpha\beta + \alpha R(x) + \beta L(x) + O(x) + h(x)t(x) + s\delta A + r\delta B - s\delta r\delta
\end{aligned}
$$

额外的随机数

在PGRH13方案中为了防止多项式通过相同偏移进行变形使用了额外的随机数 γ

验证方法：$e(g^{\alpha_1 L(x)+\alpha_2 R(x)+\alpha_3 O(x)}, g^{\gamma}) = e(g^{L(x)}, g^{\alpha_1\gamma})e(g^{R(x)}, g^{\alpha_2\gamma})e(g^{O(x)}, g^{\alpha_3\gamma})$

在Groth16方案中利用随机数δ以及不同的生成元g1,g2来实现

假设$C = (\alpha R(x) + \beta L(x) + O(x) + h(x)t(x) + s\delta A + r\delta B - s\delta r\delta)/\delta$

所以$AB = \alpha\beta + C\delta$

$$
\begin{aligned}
&[A]_1[B_2] \\
&= [(L(x) + \alpha + r\delta)]_1[(R(x) + \beta + s\delta)]_2 \\
&= [\alpha]_1[\beta]_2 + [C]_1[\delta]_2
\end{aligned}
$$

此时如果产生相同偏移，因为根据乘法法则，会产生许多新的元素，但是由于$[C]_1[\delta]_2$是2个点的乘，使得C无法逆向运算从而做出相应的变更，这使得偏移变形的范围很有限，只能是随机偏置的相关倍数，所以攻击方式比较单一，可以通过其他方式避免，这种攻击方式被称为可加工性攻击。

其他

在实际使用过程中输入也区分公开输入和私密输入，具体过程已经在完备性证明中证明，此时我们再额外使用一个随机数利以及乘法性质将其保护起来，

$$
[A]_1 \cdot [B]_2 = [\alpha]_1 \cdot [\beta]_2 + [\sum_{i=0}^{l} \frac{\sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\gamma}]_1[\gamma]_2 + [C]_1 \cdot [\delta]_2
$$

最终可以通过3个点来完成验证，减少了空间

攻击

利用已经生成的证明进行伪造

方法

$$
\begin{aligned}
A' &= A \\
B' &= B + \delta \\
C' &= C + A
\end{aligned}
$$

或者

$$
\begin{aligned}
A' &= xA \\
B' &= x^{-1}B \\
C' &= C
\end{aligned}
$$

防护

- 对证明进行签名，即使复制了证明也无法复制签名
- 将公共输入中的一部分或者部分作为只能使用一次的变量
- 将证明发送人也作为公共输入的一部分
- 使用其他证明方法

可信初始化设置

zk-snark的Groth16方案针对每个项目都需要一个可信初始化设置来生成CRS公共字符串。可信初始化设置过程中产生的随机数被称为**有毒废料**需要丢弃，否则存在伪造证明的安全风险。

该过程需要采用MPC多方协同计算来实现，被称作 **powers_of_tau**

其原理是采用ecdh类似的公共密钥生成技术，使得单一参与方无法操控

【多方计算原理】

假设现需要通过MPC仪式生成公共字符串M。一共有N个参与者，对每个参与者进行编号，表示为$\{P_i|i\in[1,N]\}$。每个参与者按照固定时隙提交数据，表示为$\{J_i|i\in[1,N]\}$，数据被追加记录在公共的数据表内，每个时隙公共数据表的状态用$\{Table_i|i\in[1,N]\}$表示，加密数据用$\{[x]_i\}$表示。

简易计算协议

生成过程

假设本轮参与者是P_i

- 生成随机数随机数x_i,其加密值$E(x_i)=[x_i]_1$

a) 如果$i=0$,生成 $M=E(x_i)=[x_i]_1$

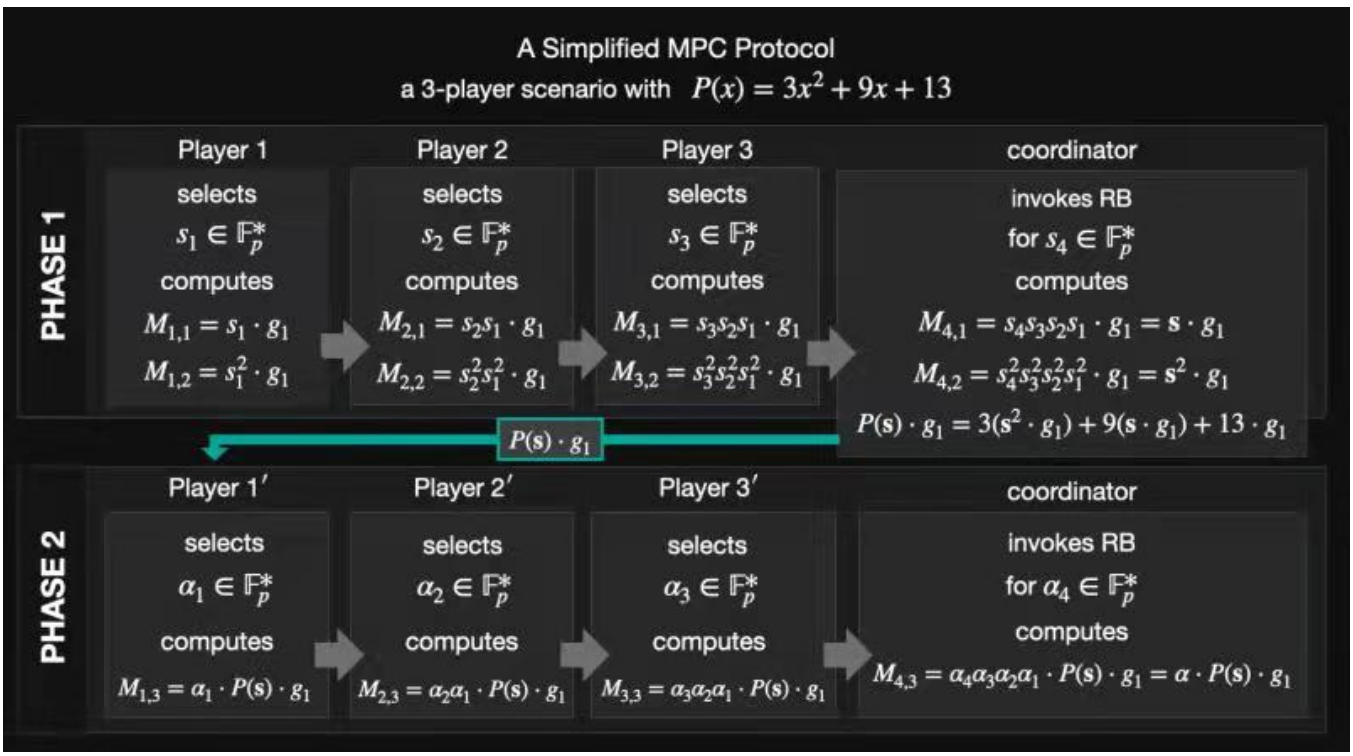
b) 如果$i!=0$,从上一轮公共数据表状态$Table_{i-1}$中获取公共字符串M',并生成$M=M'x_i$

- 更新本轮公共数据表状态$Table_i$，即记录M和 $[x_i]_1$

验证过程

每轮公共数据表状态$Table_i$被更新前需要通过验证

验证方法：$e(g^M,g^1)=e(g^{M'},g^{x_i})$



完整计算协议

整个过程会分成3阶段：**初始化**，**通用计算**和**特殊计算**。其中初始化和通用计算可以适用于任何项目只需要执行一次，特殊计算需要针对每个项目单独运行。

整个过程最终需要生成以下点：

- 生成元g_1上的点：$\alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\gamma}\}_{i=0}^{l}, \{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}\}_{i=l+1}^{m}, \{\frac{x^i t(x)}{\delta}\}_{i=0}^{n-2}$
- 生成元g_2上的点：$\beta, \delta, \gamma, \{x^i\}_{i=0}^{n-1}$

初始化

设置生成元g1和g2

通用计算

该阶段需要通过多方计算生成g_1上点 $\alpha, \beta, \{x^i\}_{i=0}^{2n-2}, \{\alpha x^i\}_{i=0}^{n-1}, \{\beta x^i\}_{i=0}^{n-1}$和$g_2$上点$\beta, \{x^i\}_{i=0}^{n-1}$

该阶段每一轮的参与者P_j需要：

生成随机数α_j, β_j, x_j

利用生成元g_1生成$[\alpha_j]_1, [\beta_j]_1, [x_j]_1$

利用上一轮的数据生成本轮的$[\alpha]_1 = [\alpha' \alpha_j]_1$

利用上一轮的数据生成本轮的$[\beta]_1 = [\beta' \beta_j]_1$

利用上一轮的数据生成本轮的$[x]_1 = [x' x_j]_1$

利用上一轮的数据生成本轮的$[x^i]_1 = [(x^i)' x_j^i]_1, i \in [0, 2n-2]$

利用上一轮的数据生成本轮的$[\alpha x^i]_1 = [(\alpha)'(x^i)' \alpha_j x_j^i]_1, i \in [0, n-1]$

利用上一轮的数据生成本轮的$[\beta x^i]_1 = [(\beta)'(x^i)' \beta_j x_j^i]_1, i \in [0, n-1]$

验证

验证本轮的$[\alpha]_1$ 的正确性， $e([\alpha]_1, g_1) = e([\alpha']_1, [\alpha_j]_1)$

验证本轮的$[\beta]_1$ 的正确性， $e([\beta]_1, g_1) = e([\beta']_1, [\beta_j]_1)$

验证本轮的$[x]_1$ 的正确性， $e([x]_1, g_1) = e([x']_1, [x_j]_1)$

验证本轮的$[x^i]_1$ 的正确性， $e([x^i]_1, g_1) = e([x]_1, [x^{i-1}]_1), i \in [1, 2n-2]$

验证本轮的$[\alpha x^i]_1$ 的正确性， $e([\alpha x^i]_1, g_1) = e([\alpha]_1, [x^i]_1), i \in [1, n-1]$

验证本轮的$[\beta x^i]_1$ 的正确性， $e([\beta x^i]_1, g_1) = e([\beta]_1, [x^i]_1), i \in [1, n-1]$

同理g2上的点也如此计算

另外还需验证生成元g1,g2上共同点参数一致，$e([\alpha]_1, g_2) = e([\alpha]_2, g_1)$

非通用计算

该阶段需要通过多方计算计算每个项目所使用的特殊点，

生成元g_1上点 $\delta, \{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\gamma}\}_{i=0}^{l}, \{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}\}_{i=l+1}^{m}, \{\frac{x^i t(x)}{\delta}\}_{i=0}^{n-2}$

生成元g_2上点 δ, γ

利用通用计算阶段的结果可以生成一些线性关系

$\{K_{in_i} = \frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\gamma}\}_{i=0}^{l}$

$\{K_{out_i} = \frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}\}_{i=l+1}^{m}$

$\{H_i = \frac{x^i t(x)}{\delta}\}_{i=0}^{n-2}\}$

$u_i(x), v_i(x), w_i(x)$多项式系数需要采用FFT求得

$t(x) = (x-w)(x-w^1)\dots(x-\omega^{n-1}) = x^n - 1$

对于该阶段每一轮的参与者P_j

利用生成元g_1生成本轮的随机信标$[\delta_j]_1, [\gamma_j]_1$

利用上一轮的数据生成本轮的$[\delta]_1 = [\delta' \delta_j]_1$

利用上一轮的数据生成本轮的$[K_{in_i}]_1 = [K'_{in_i}/\delta_j]_1$

利用上一轮的数据生成本轮的$[K_{out_i}]_1 = [K'_{out_i}/\gamma_j]_1$

利用上一轮的数据生成本轮的$[H_i]_1 = [H'_i/\delta_j]_1$

验证

验证本轮的$[\delta]_1$ 的正确性， $e([\delta]_1, g_1) = e([\delta']_1, [\delta_j]_1)$

验证本轮的$[K_{in_i}]_1$ 的正确性， $e([K_{in_i}]_1, g_1) = e([K'_{in_i}]_1, [1/\delta_j]_1)$

验证本轮的$[K_{out_i}]_1$ 的正确性， $e([K_{out_i}]_1, g_1) = e([K_{out_i}]_1, [1/\gamma_j]_1)$

验证本轮的$[K_{in_i}]_1$ 的正确性， $e([H_i]_1, g_1) = e([H'_i]_1, [1/\delta_j]_1)$

同理g2上的点也如此计算

另外还需验证生成元g1,g2上共同点参数一致，$e([\alpha]_1, g_2) = e([\alpha]_2, g_1)$

补充

费马小定理

假如p是质数，且a和p互质，那么$a^{p-1}(modp) = 1$。 因此，在有限域中，$b^{-1} = b^{p-2}$

FFT快速傅里叶变换

【多项式】

一个n-1次多项式$f(x) = \sum_{i=0}^{n-1} a_i x^i$,有两种表示法：系数表示法和点值表示法

系数表示法

用每一项的系数来表示，$f(x) = \{a_0, a_1, a_2, \dots, a_{n-1}\}$

点值表示法

使用n个点来代表经过这些点的唯一曲线

$f(x) = \{\{x0, f(x0)\}, \{x1, f(x1)\}, \dots, \{x_{n-1}, f(x_{n-1})\}\}$

用n个不同的x带入多项式，并计算相应多项式的计算值，时间复杂度为$O(n^2)$

$O(n^2) =$ 多项式阶为$n * x$个数n

多项式乘法

假设有2个n阶多项式相乘

如果采用系数表示法，则使用朴素(普通)矩阵乘法的时间复杂度为$O(n^2)$

$$[\,b_0 \quad b_1 \quad \dots b_n\,]\begin{bmatrix} a_0 & a_1 & \dots a_n \\ a_0 & a_1 & \dots a_n \\ \dots & & \\ a_0 & a_1 & \dots a_n \end{bmatrix} = \begin{bmatrix} a_0b_0 & a_1b_0 & \dots a_nb_0 \\ a_0b_1 & a_1b_1 & \dots a_nb_1 \\ \dots & & \\ a_0b_n & a_1b_n & \dots a_nb_n \end{bmatrix}$$

$O(n^2) = $多项式$A(x)$系数个数$n$ * 多项式$B(x)$系数个数n

如果采用点值表示法,则时间复杂度为$O(n)$,（求卷积）

$h(x) = f_1(x)f_2(x) = \{\{x_0, f_1(x_0)f_2(x_0)\}, \{x_1, f_1(x_1)f_2(x_1)\}, \dots\dots, \{x_{n-1}, f_1(x_{n-1})f_2(x_{n-1})\}\}$

采用点值表示法的多项式在计算乘法时效率会高于使用系数表示法的多项式

系数转点值的算法叫DFT（离散傅里叶变换），点值转系数的算法叫IDFT（离散傅里叶逆变换）

【复数】

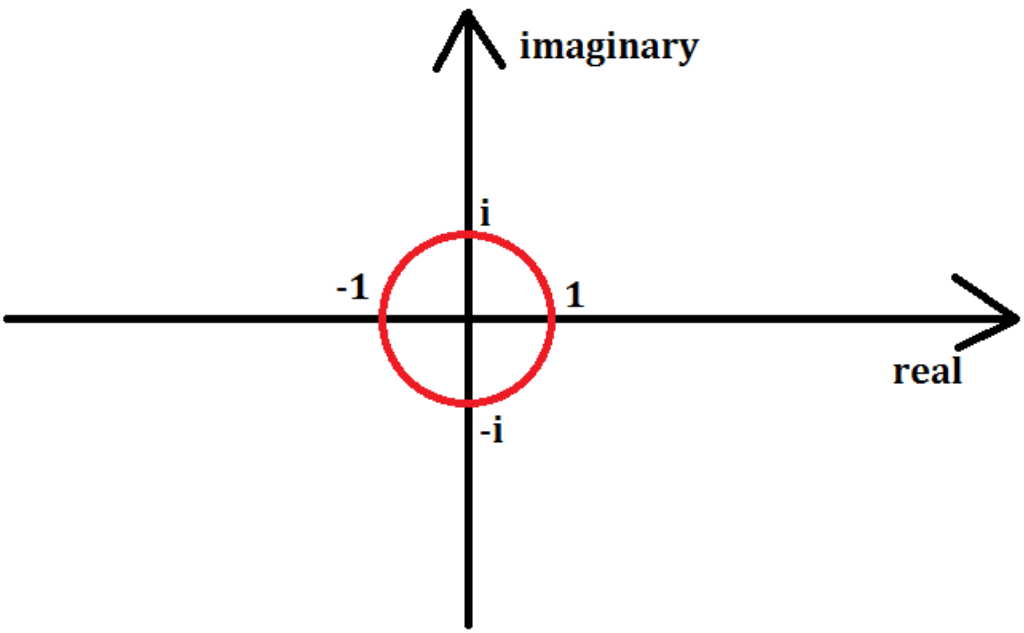
复数表示法：a+bi

复数加法:$z1 + z2 = (a + c) + (b + d)i$

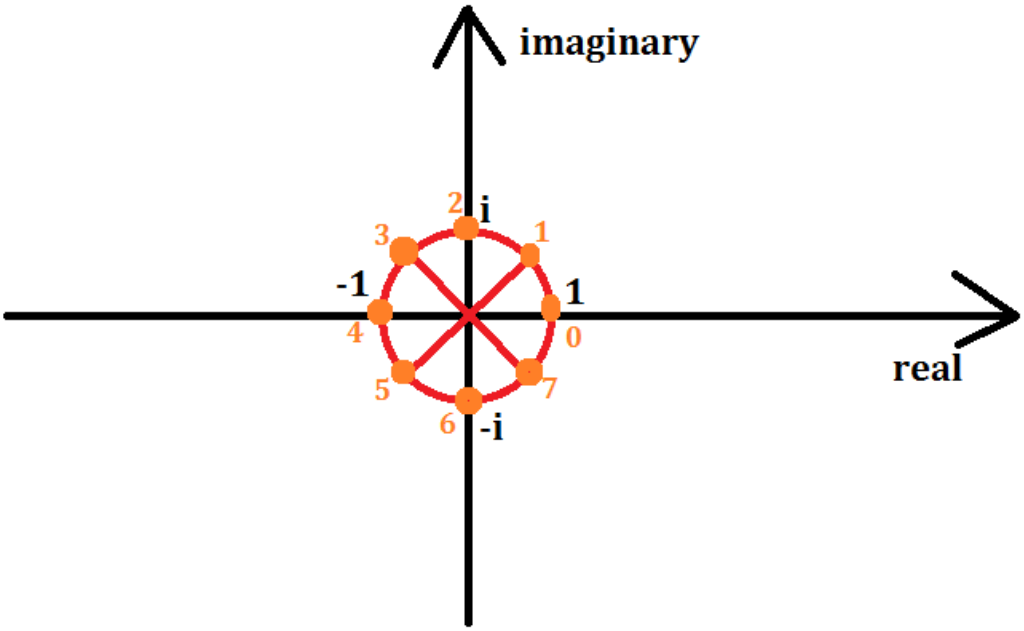
复数乘法:$z1z2 = (ac - bd) + (ad + bc)i = (a1, \theta1) * (a2, \theta2) = (a1a2, \theta1 + \theta2)$,模长相乘，极角相加

单位根

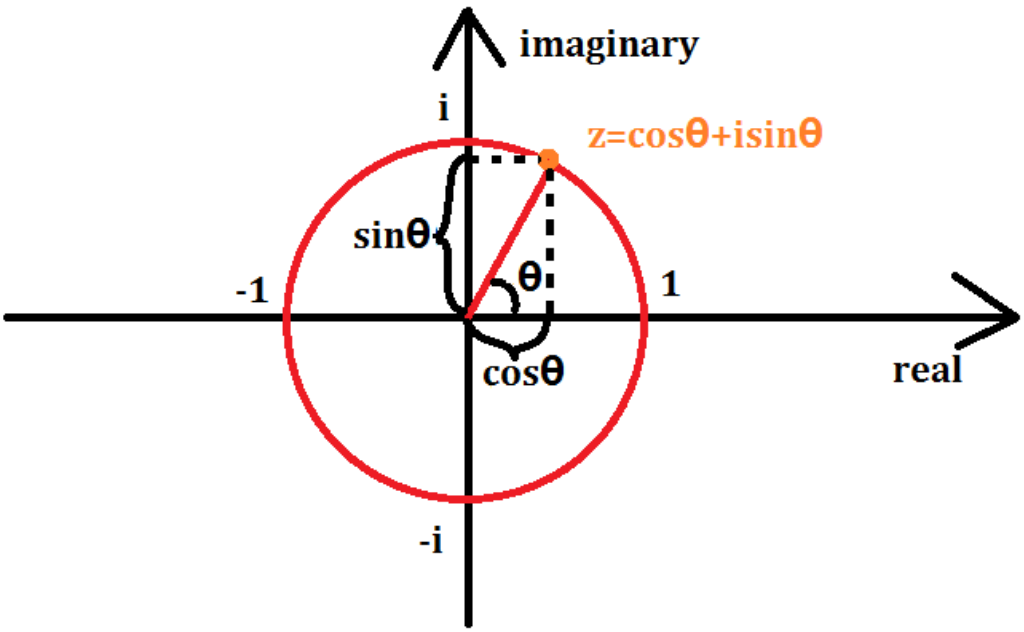
在实轴和虚轴建立的坐标系中，以原点为中心构建的单位元上的点所表示的复数经过若干次次方都可以得到1



假设将该单位元n等分，并从(1,0)点按逆时针开始编号，则编号为k的点对应的复数是w_n^k



按照复数乘法的特性,则有$(w_n^1)^k = w_n^k$，将w_n^1称为n次单位根



对于每一个等分点则有

$w_n^k = cos\frac{n}{k}2\pi + sin\frac{n}{k}2\pi i$

因此多项式点值表示法中所用到的$x_0, x_1, \ldots . x_{n-1}$都可以用$w_n^0, w_n^1, \ldots . w_n^{n-1}$表示

另外还有些其他性质

$w_n^k = cos\frac{n}{k}2\pi + sin\frac{n}{k}2\pi i = cos\frac{2n}{2k}2\pi + sin\frac{2n}{2k}2\pi i = w_{2n}^{2k}$

$w_n^{k+\frac{n}{2}} = -w_n^k$，因为$w_n^{\frac{n}{2}} = cos\frac{n/2}{n}2\pi + sin\frac{n}{k}2\pi i = cos\pi + sin\pi i = -1$

$w_n^0 = w_n^n$

【快速傅里叶变换】

多项式系数表示法转换成点值表示法

原理

将多项式$A(x)$按照奇偶项分割

$A(x) = \sum_{i=0}^{n-1} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} = (a_0 + a_2 x^2 + \ldots + a_{n-2} x^{n-2}) + x(a_1 + a_3 x^2 + \ldots + a_{n-1} x^{n-2})$

再设置另外两个多项式

$A_1(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{n/2-1}$

$A_2(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{n/2-1}$

则$A(x) = A_1(x^2) + xA_2(x^2)$

假设$x = w_n^k$，$k < n/2$

$A(w_n^k) = A_1((w_n^k))^2) + w_n^k A_2((w_n^k))^2) = A_1(w_n^{2k}) + w_n^k A_2(w_n^{2k}) = A_1(w_{n/2}^k) + w_n^k A_2(w_{n/2}^k)$

再假设$x = w_n^{k+n/2}, k < n/2$

$A(w_n^{k+n/2}) = A_1((w_n^{k+n/2}))^2) + w_n^{k+n/2} A_2((w_n^{k+n/2}))^2) = A_1(w_n^{2k} w_n^n) - w_n^k A_2(w_n^{2k} w_n^n) = A_1(w_n^{2k}) - w_n^k A_2(w_n^{2k}) = A_1(w_{n/2}^k) - w_n^k A_2(w_{n/2}^k)$

两者只有元素项的加减关系

因此如果知道了

$A_1(x)$和$A_2(x)$在$\{w_{n/2}^0, w_{n/2}^1 \ldots\ldots\ldots w_{n/2}^{n/2}\}$上的取值，则可以在$O(n)$时间内求出$A(x)$在$\{w_n^0, w_n^1 \ldots\ldots\ldots w_n^n\}$上的取值（求一次加法，求一次减法）

而$A_1(x)$和$A_2(x)$又可以使用递归不断分割，直到n=1

因此快速傅里叶变换（多项式系数表示法转换成点值表示法）的时间复杂度:

$T(n) = 2T(n/2) + O(n) = O(nlogn)$

【快速傅里叶逆变换】

多项式点值表示法转换成系数表示法

原理

把多项式A(x)的快速傅里叶变换结果作为另一个多项式B(x)的系数，并取单位根的倒数即$w_n^0, w_n^{-1} \ldots\ldots, w_n^{1-n}$

作为x带入B(x),得到的每个数再除以n就是A(x)的各项系数。这就是快速傅里叶变换逆变换，相当于在FFT基础上再搞一次FFT。

推导

假设$\{(w_n^0, y_0), (w_n^1, y_1), \ldots. (w_n^{n-1}, y_{n-1})\}$是多项式$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots a_{n-1} x^{n-1}$的快速傅里叶变换结果。多项式$B(x) = y_0 + y_1 x + \ldots\ldots y_{n-1} x^{n-1}$,取单位根的倒数即$w_n^0, w_n^{-1} \ldots\ldots, w_n^{1-n}$

作为x带入B(x),得到一个新的快速傅里叶变换结果$\{(w_n^0, z_0), (w_n^{-1}, z_1), \ldots. (w_n^{-(n-1)}, z_{n-1})\}$

$z_k = \sum_{i=0}^{n-1} y_i (w_n^{-k})^i$
$= \sum_{i=0}^{n-1} (\sum_{j=0}^{n-1} a_j (w_n^i)^j)(w_n^{-k})^i$
$= \sum_{j=0}^{n-1} a_j (\sum_{i=0}^{n-1} (w_n^{j-k})^i)$

当$j-k=0$时，$\sum_{i=0}^{n-1}(w_n^{j-k})^i = n$，其余通过等比例求和可知等于$0$，所以$z_k = na_k$

因此快速傅里叶变换逆变换（多项式点值表示法转换成系数表示法）的时间复杂度:

$T(n) = 2T(n/2) + O(n) = O(nlogn)$

参考资料

FFT变换 https://www.cnblogs.com/YjmStr/p/15078683.html

FFT变换 https://www.cnblogs.com/RabbitHu/p/FFT.html

FFT变换 https://blog.csdn.net/enjoy_pascal/article/details/81478582/

Groth16 https://eprint.iacr.org/2016/260.pdf

MPC https://eprint.iacr.org/2017/1050.pdf