# 密码学 zk 系列

# 第 12 课：zkSnark-Halo2 证明系统

**lynndell 博士**

新火科技 密码学专家 lynndell2010@gmail.com

**目录**

**密码学基础系列**

1. **对称加密与哈希函数**
2. **公钥加密与数字签名**
3. **RSA、环签名、同态加密**
4. **承诺、零知识证明、BulletProof 范围证明、Diffie-Hellman 密钥协商**

**门限签名系列**

5. **Li17 两方签名与密钥刷新**
6. **GG18 门限签名**
7. **GG20 门限签名**

**zk 系列**

8. **Groth16 证明系统**
9. **Plonk 证明系统**
10. **UltraPlonk 证明系统**
11. **SHA256 查找表**
12. **Halo2 证明系统**
13. **zkSTARK 证明系统**

**Plonk 证明系统 = 电路 + 多项式承诺**

**UltraPlonk 证明系统 = 电路 + 多项式承诺**

**Halo2 证明系统 = 电路 + 多项式承诺**

**电路：门约束、线约束、定制门、查找表**

**Plonk/UltraPlonk 多项式承诺：KZG 承诺、Dan Boneh 承诺**

**+Fflonk 优化**

**Halo2 多项式承诺：向量内积承诺**

# 1.承诺

**承诺三阶段**

**承诺：**消息为$x$，随机数为$r$，计算承诺$c = commit(x,r)$

**打开承诺：**消息为$x$，随机数为$r$，打开承诺$open(c) = (x,r)$

**验证：**$true / false \leftarrow verify(c,x,r)$

**两个性质：**

**隐藏性：**对消息进行承诺，攻击者不能通过承诺值提取任何有效消息。通常使用随机数 r 实现隐藏功能。

**绑定性：**对哪个消息承诺，则打开后仍然是那个消息，不能是其他消息。通常取决于算法本身，例如哈希函数、倍点运算都具有绑定性。

# 2.Pedersen 承诺

**初始化：**基域为$\mathbb{F}_p$，素数 $p$ 的范围是$2^{256}$。椭圆曲线$E(\mathbb{F}_p)$上的两个随机生成元为$G,H$。

**承诺：**消息为$x$，随机数为$r$，计算承诺$commit(x,r) = x \cdot G + r \cdot H$

**打开承诺：**消息为$x$，随机数为$r$，打开承诺$open(c) = (x,r)$

**验证：**$true / false \leftarrow verify(c,x,r)$

**Pedersen 承诺具有同态性**

$$\begin{aligned}A+B&=commit(x_a,r_a)+commit(x_b,r_b)\\&=x_a\cdot G+r_a\cdot H+x_b\cdot G+r_b\cdot H\\&=(x_a+x_b)\cdot G+(r_a+r_b)\cdot H\\&=commit(x_a+x_b,r_a+r_b)\end{aligned}$$

# 3.向量承诺

**初始化：**基域为$\mathbb{F}_p$，素数 $p$ 的范围是$2^{256}$。椭圆曲线$E(\mathbb{F}_p)$上的随机生成元为$\vec{G}=(G_1,...,G_n),H$。这些生成元**不是** KZG 承诺那样安全多方计算出来，而是基于公开的数据计算获得，例如$G_i=hash(G,i,SystemParameter),i=1,...,n$。

**哈希函数 hash，将输入数据映射为一个随机的椭圆曲线生成元$G_i$。**

不需要可信的初始化。

**KZG 承诺初始化**需要安全多方计算$\langle G_1,\alpha G_1,...,\alpha^{d-1}G_1;G_2,\alpha G_2\rangle$，是可信的初始化。使用安全多方计算，n 个用户参与，只要有一个参与方诚实删除保密随机数，则初始化安全。

**多项式承诺：**消息向量为$\vec{x}=(x_1,...,x_n)$，随机数为$r$，计算**多项式承诺**

$$commit(x,r)=x_1\cdot G_1+...+x_n\cdot G_n+r\cdot H=\vec{x}\cdot\vec{G}+r\cdot H$$

**打开承诺：**向量消息为$\vec{x}=x_1,...,x_n$，随机数为$r$，打开承诺$open(c)=(\vec{x},r)$

**验证：**$true/false\leftarrow verify(c,\vec{x},r)$

# 4.KZG 多项式承诺

**初始化：**椭圆曲线双线性群为$\mathcal{G}=(e,\mathbb{G},\mathbb{G}_T)$，【有毒废料】随机数$\alpha\in_R\mathbb{Z}_p^*$，$t+1$元组$\langle G,\alpha\cdot G,...,\alpha^t\cdot G\rangle\in\mathbb{G}^{t+1}$，令输出为

$$PK=\left(\mathcal{G},G,\alpha\cdot G,...,\alpha^t\cdot G\right)$$

将 setup 分为 2 个集合：**第一个集合为 PK，第二个集合为 VK**。

**专业术语：**

**CRS - Common Reference String；**

**SRS - Structured Reference String**

**多项式承诺：**对于**多项式**$\phi(x)=\sum_{j=0}^{\deg(\phi)}\phi_j\cdot x^j$，计算**多项式承诺**

$$C=\phi(\alpha)\cdot G=\sum_{j=0}^{\deg(\phi)}\left(\phi_j\cdot(\alpha^j G)\right)$$

**完全打开承诺：**发送多项式系数$\phi_j, j=...$

$$\phi(x)=\sum_{j=0}^{\deg(\phi)}\phi_j\cdot x^j$$

**验证打开承诺：**验证承诺$C$与公钥$PK$、多项式$\phi(x)$满足一致性。基于多项式$\phi(x)=\sum_{j=0}^{\deg(\phi)}\phi_j\cdot x^j$ **再次**计算承诺

$$C'=\phi(\alpha)\cdot G=\sum_{j=0}^{\deg(\phi)}\phi_j\cdot\alpha^j\cdot G$$

如果$C=C'$，则接受，否则拒绝。

**打开随机点承诺：**计算**商多项式**

$$\varphi_i(x)=\frac{\phi(x)-\phi(i)}{x-i}$$

基于多项式的系数和 PK，计算**商多项式承诺**

$$W_i=\varphi_i(\alpha)\cdot G$$

**输出**$(i,\phi(i),W_i)$。其中，横坐标 i 处的函数值为$\phi(i)$。

**校验：**如果以下等式成立，则接受，否则拒绝

$$e(C,G)=e(W_i,\alpha\cdot G-i\cdot G)\cdot e(G,G)^{\phi(i)}$$

**反之，如果双线性映射验证成功，则在索引**$i$**，多项式的值确实是**$\phi(i)$**。**

**注释：**

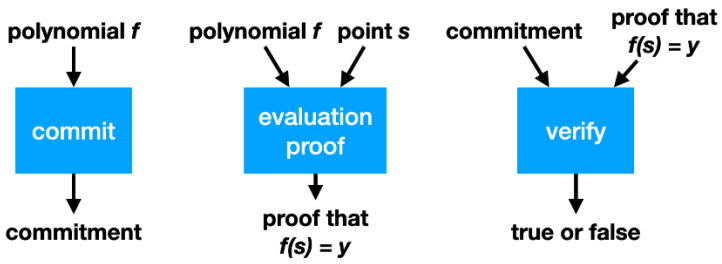
**商多项式**$\varphi_i(x)=\frac{\phi(x)-\phi(i)}{x-i}$**存在，则说明多项式在 i 处的函数值确实为**$\phi(i)$**。商多项式**$\varphi_i(x)$**的阶为**$n-1$**，最多有**$n-1$**个解。有限域**$\mathbb{F}_p$**空间为**$2^{256}$**。根据 Schwartz-Zippel 引理，攻击者碰撞成功概率可忽略。**

**Schwartz–Zippel 引理**

令$P$为有限域$\mathbb{F}$上的多项式$P=F(x_1,\ldots,x_n)$，其阶为$d$。令$S$为有限域$\mathbb{F}$的子集，从$S$中选择随机数$r_1,\ldots,r_n$，则多项式等于零的概率可忽略，即

$$\Pr[P(r_1,\ldots,r_n)=0]\leq\frac{d}{|S|}$$

在单变量情况下，多项式的阶为$d$，则最多有$d$个根。




**KZG 承诺、向量内积承诺，思路一样。**
**多项式承诺步骤：**

- 多项式承诺；
- **完全打开；**
- **验证方：一致性验证。**
- 在索引 i 计算函数值 $\phi(i)$；
- 生成函数值的**证明** $\pi=(i,W_i)$，是**商多项式的承诺；**
- **验证方：**一致性验证。作用：确保多项式正确、函数值正确。

# 5.向量内积实现多项式求值

秘密多项式 $f(x)=f_0+f_1\cdot x+...+f_n\cdot x^n$，将系数构造**秘密向量** $\vec{f}=(f_0,...,f_n)$。

**保密多项式** $f(x)$ **在随机索引 s 的函数值为** $f(s)$ **等价于秘密向量** $\vec{f}=(f_0,...,f_n)$

**与公开的随机点向量** $\vec{s}=(1,s,...,s^n)$ **的内积为** $\langle\vec{f},\vec{s}\rangle=\langle(f_0,...,f_n),(1,s,...,s^n)\rangle$

$$f(s)=f_0+f_1\cdot s+...+f_n\cdot s^n=\langle(f_0,...,f_n),(1,s,...,s^n)\rangle=\langle\vec{f},\vec{s}\rangle$$

因此，得出**关键结论**：**多项式求值等价于向量内积。**

如果向量内积是正确的，则多项式求值正确。

因此，如果验证方认可内积正确，则等价于认可多项式在一个随机点的求值正确；基于 **Schwartz–Zippel**，则等价于认可整个多项式的运算均正确，则等价于认可电路对所有数据运算均正确，则等价于认可 **Y=F(w)**运算正确。

**电路版就是 PK 和 VK。最关键在于 VK。**
**VK 存储到合约上或由验证方拥有，则证明方只能诚实使用 VK 对应的 PK 基于正确 witness 生成 proof，验证才会成功。**
**（1） 新电路 PK’，生成 proof’，验证方使用旧 VK 验证肯定失败。**
**（2） 使用错误 witness，生成的 proof’验证肯定也失败。**

**电路包含 2 个核心算法：EdDSA 验证算法和哈希函数。**
**只有用户拥有 EdDSA 签名私钥，只有用户能生成正确的签名。因此，只有用户提交正确的交易单，证明方才能构造正确的 witness。因此，证明方不能盗用用户资产。**

**本质上：EdDSA 验证算法和哈希函数使得证明方构造 witness 具有 NP 困难，从而保护用户资产安全。**

# 6.向量内积承诺

## 6.1 多项式的值与证明

证明方知道 n 维向量秘密 $\vec{a}$，基于 *SystemParameter* 计算 $s := hash(SystemParameter)$，计算 $\vec{b} = (1, s, ..., s^n)$

**随机打开点承诺：**证明方**发送多项式值** $z = \langle \vec{a}, \vec{b} \rangle$

**完全打开承诺：**证明方**发送** $\vec{a}$；

**发送数据为：** $z, \vec{a}$，长度为 $n+1$。

**验证：**验证方计算 $s := hash(SystemParameter)$ 和 $\vec{b}$，则能够校验 $z == \langle \vec{a}, \vec{b} \rangle$

**优化版：**

证明方知道 n 维向量秘密 $\vec{a}$，基于 *SystemParameter* 计算 $s := hash(SystemParameter)$，计算 $\vec{b} = (1, s, ..., s^n)$

**随机打开点承诺：**证明方**发送多项式值** $z = \langle \vec{a}, \vec{b} \rangle$；

**挑战：**证明方基于承诺基于 $z$ 计算 $x := hash(SystemParameter, z)$

**折半响应：**证明方将 n 为向量**折半为**

$$\vec{a}' = x^{-1}(a_0, ..., a_{n/2}) + x(a_{n/2+1}, ..., a_n), \vec{b}' = x^{-1}(b_0, ..., b_{n/2}) + x(b_{n/2+1}, ..., b_n)$$

再计算

$$\begin{aligned}
\langle \vec{a}', \vec{b}' \rangle &= \left\langle \left(x^{-1}a_1 + xa_3, x^{-1}a_2 + xa_4\right), \left(xb_1 + x^{-1}b_3, xb_2 + x^{-1}b_4\right) \right\rangle \\
&= \left(a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4\right) + x^{-2}\left(a_1b_3 + a_2b_4\right) + x^2\left(a_3b_1 + a_4b_2\right) \\
&= z + x^{-2}l_z + x^2 r_z
\end{aligned}$$

**发送数据为：** $z,\vec{a}',l_z,r_z$，长度为$\frac{n}{2}+3$，向量减半

**验证：**验证方基于 *SystemParameter* 计算 $s:=hash(SystemParameter)$，计算

$\vec{b}=(1,s,...,s^n)$；

基于承诺 $z$ 和 *SystemParameter*，计算 $x:=hash(SystemParameter,z)$ 和

$\vec{b}'=(b_0,...,b_{n/2})+x(b_{n/2+1},...,b_n)$；

基于 $\vec{a}',\vec{b}'$ 校验等式

$$\left\langle \vec{a}',\vec{b}' \right\rangle = z + x^{-2}l_z + x^2 r_z$$

假如 z 正确，错误打开 $\vec{a}'$，验证成功的概率可忽略。

**根据 Schwartz–Zippel 引理，证明方作弊成功概率可忽略，只有诚实打开承诺 $\vec{a}'$。**

$$com\left(z=\left\langle \vec{a},\vec{b} \right\rangle\right)_{open(\vec{a})} \xleftarrow{Schwartz-Zippel-Lemma} com\left(z'=\left\langle \vec{a}',\vec{b}' \right\rangle\right)_{open(\vec{a}')} \xleftarrow{Schwartz-Zippel-Lemma} com\left(z''=\left\langle \vec{a}'',\vec{b}'' \right\rangle\right)_{open(\vec{a}'')}$$

## 6.2 方案 1（完全打开）

证明方知道秘密 $\vec{a}=(a_1,a_2,a_3,a_4)$

**初始化：**系统参数为 $\vec{G}=(G_1,G_2,G_3,G_4)$

**多项式承诺：**证明方计算**多项式承诺** $A:=\left\langle \vec{a},\vec{G} \right\rangle=\sum_{i=1}^{4}a_iG_i$；

**挑战：**证明方计算 $s:=hash(A,\vec{G})$；

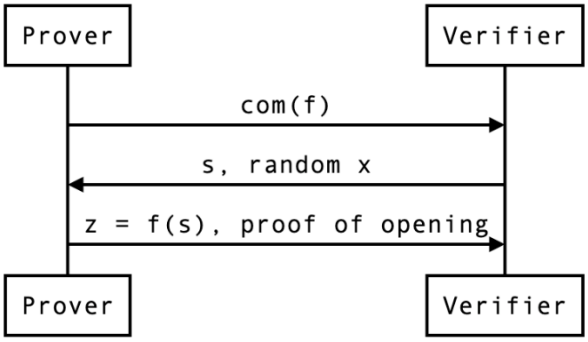
**响应：**证明方计算 $\vec{b}=(b_1,b_2,b_3,b_4):=(1,s,s^2,s^3)$，计算**多项式的值** $z:=\left\langle \vec{a},\vec{b} \right\rangle$；

发送数据：$(A,z,\vec{a})$，长度为 $n+2$ 数据量很大。

**验证：**验证方基于 $A,\vec{G}$ 计算 $s$ 和 $\vec{b}$；

- 基于基于 $\vec{a}$、$\vec{G}$ 和 $A$，校验 $A=\left\langle \vec{a},\vec{G} \right\rangle$，**多项式承诺正确；**
- 基于 $\vec{a}$、$\vec{b}$ 和 $z$，校验 $z=\left\langle \vec{a},\vec{b} \right\rangle$，**多项式的值正确。**

**因此，验证方认可：多项式的值确实是 z。**




## 6.3 方案 2（折半打开）

证明方知道秘密 4 维向量 $\vec{a}=(a_1,a_2,a_3,a_4)$

**初始化：**系统参数为 $\vec{G}=(G_1,G_2,G_3,G_4)$

**多项式承诺：**证明方计算**多项式承诺** $A:=\langle\vec{a},\vec{G}\rangle=\sum_{i=1}^{4}a_iG_i$，发送 $A$；

**挑战：**证明方计算 $s:=hash(A,\vec{G}),x:=hash(s)$；

**折半响应：**证明方计算 4 维向量 $\vec{b}=(b_1,b_2,b_3,b_4):=(1,s,s^2,s^3)$

**计算多项式的值：** $z:=\langle\vec{a},\vec{b}\rangle$

**发送** $z$；

**折半响应：**计算 2 维向量

$$\vec{a}':=x^{-1}(a_1,a_2)+x(a_3,a_4)=\left(x^{-1}a_1+xa_3,x^{-1}a_2+xa_4\right)$$
$$\vec{b}':=x(b_1,b_2)+x^{-1}(b_3,b_4)=\left(xb_1+x^{-1}b_3,xb_2+x^{-1}b_4\right)$$
$$\vec{G}':=x(G_1,G_2)+x^{-1}(G_3,G_4)=\left(xG_1+x^{-1}G_3,xG_2+x^{-1}G_4\right)$$

**发送** $\vec{a}'$；

证明方计算

$$\langle\vec{a}',\vec{G}'\rangle=(x^{-1}a_1+xa_3)(xG_1+x^{-1}G_3)+(x^{-1}a_2+xa_4)(xG_2+x^{-1}G_4)$$
$$=A+x^{-2}(a_1G_3+a_2G_4)+x^2(a_3G_1+a_4G_2)$$
$$=A+x^{-2}L_a+x^2R_a$$

**发送** $(L_a,R_a)$；

计算

$$\begin{aligned}\left\langle \vec{a}', \vec{b}' \right\rangle &= \left\langle \left(x^{-1}a_1 + xa_3, x^{-1}a_2 + xa_4\right), \left(xb_1 + x^{-1}b_3, xb_2 + x^{-1}b_4\right) \right\rangle \\ &= \left(a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4\right) + x^{-2}\left(a_1b_3 + a_2b_4\right) + x^2\left(a_3b_1 + a_4b_2\right) \\ &= z + x^{-2}l_z + x^2r_z\end{aligned}$$

**发送**$(l_z, r_z)$；

**发送数据：**$\left(A, z, \vec{a}', (L_a, R_a), (l_z, r_z)\right)$，长度为$\frac{n}{2}+6$。

**优点：**$\vec{a}'$数据量折半，然后递归；**缺点：**引入$(L_a, R_a), (l_z, r_z)$数据量不小。

**验证：**验证方基于$A, \vec{G}$，计算$x, s, \vec{b}$；

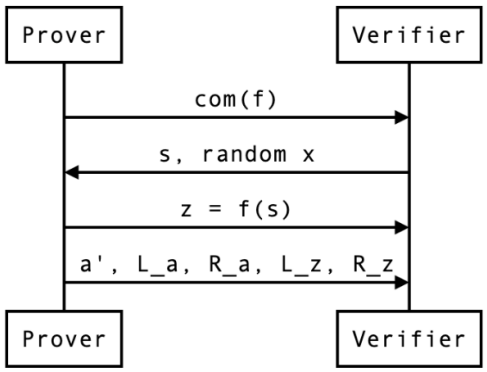
- 基于$(L_a, R_a)$计算出$A + x^{-2}L_a + x^2R_a$，校验等式$A + x^{-2}L_a + x^2R_a == \left\langle \vec{a}', \vec{G}' \right\rangle$，**多项式承诺正确；**
- 基于$(z, l_z, r_z)$计算出$z + x^{-2}l_z + x^2r_z$，校验等式$z + x^{-2}l_z + x^2r_z == \left\langle \vec{a}', \vec{b}' \right\rangle$，**多项式的值正确；**

验证等式分别使用$A$和$z$；

**根据 Schwartz–Zippel 引理，**（1）对于$A$，要求证明方老实进行向量内积承诺，计算出$A$，否则发生碰撞成功的概率可忽略；（2）对于$z$，要求证明方老实计算多项式的值为$z$，否则发生碰撞成功的概率可忽略。

**因此，验证方认可：多项式的值确实是 z。**

**对应于 KZG 承诺验证方认可多项式在 index 为 i 处的值确实是**$\phi(i)$**。**




## 6.4 方案 3（折半打开，并行承诺）

**承诺：**证明方基于 $SystemParameter$ 计算 $s := hash(SystemParameter)$，计算 4 维向量 $\vec{b} = (b_1, b_2, b_3, b_4) := (1, s, s^2, s^3)$，计算**多项式承诺与多项式的值**

$$C := A + zU = \langle \vec{a}, \vec{G} \rangle + \langle \vec{a}, \vec{b} \rangle U$$

发送 $C$；

**挑战：**证明方计算 $x := hash(SystemParameter, C, A, U)$

**折半响应：**证明方计算

$$\vec{a}' := x^{-1}(a_1, a_2) + x(a_3, a_4) = \left(x^{-1}a_1 + xa_3, x^{-1}a_2 + xa_4\right)$$
$$\vec{b}' := x(b_1, b_2) + x^{-1}(b_3, b_4) = \left(xb_1 + x^{-1}b_3, xb_2 + x^{-1}b_4\right)$$
$$\vec{G}' := x(G_1, G_2) + x^{-1}(G_3, G_4) = \left(xG_1 + x^{-1}G_3, xG_2 + x^{-1}G_4\right)$$

发送 $\vec{a}'$；
计算

$$\begin{aligned}
&\langle \vec{a}', \vec{G}' \rangle + \langle \vec{a}', \vec{b}' \rangle U \\
&= \left[A + x^{-2}L_a + x^2 R_a\right] + \left[z + x^{-2}l_z + x^2 r_z\right]U \\
&= C + x^{-2}\left(L_a + l_z U\right) + x^2\left(R_a + r_z U\right) \\
&= C + x^{-2}L + x^2 R
\end{aligned}$$

其中 $L = L_a + l_z U, R = R_a + r_z U$

发送 $L, R$；

**发送数据为：**$\left(C, \vec{a}', L, R\right)$，长度为 $\frac{n}{2} + 3$。

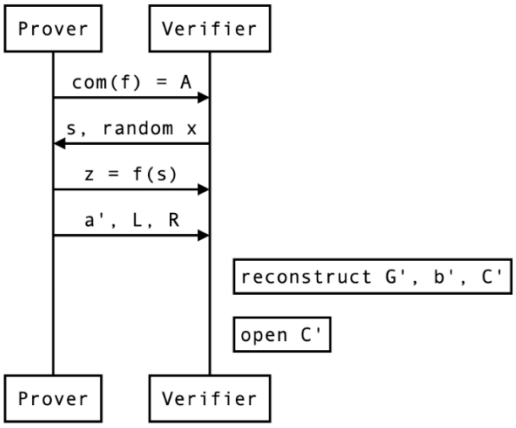
**优点：N 次递归后，总共发送**$\left(C, a, L_1, R_1, ..., L_{\log_2 n}, R_{\log_2 n}\right)$，数据总量为 $2(1 + \log_2 n)$。

**验证：**验证方基于 $C$ 和 $\vec{G}$ 计算 $s, x, \vec{b}'$，校验

$$C + x^{-2}L + x^{-2}R = \langle \vec{a}', \vec{G}' \rangle + \langle \vec{a}', \vec{b}' \rangle U$$

多项式承诺与多项式的值均正确。

**N 次之后，校验等式为** $P = C + \sum_{i=1}^{n}\left(x_i^{-2}L_i + x_i^2 R_i\right) = aG + abU = a\left(G + bU\right) = aH$




## 6.5 方案 4（添加零知识）

**添加随机项** $rH=\left\langle \vec{r},\vec{1}^4 \right\rangle H$ **，实现零知识**

**承诺：**证明方计算 $s:=hash(SystemParameter)$，计算 4 维向量

$$\vec{b}=(b_1,b_2,b_3,b_4):=(1,s,s^2,s^3)$$

计算随机向量

$$\begin{aligned}
&r=r_1+r_2+r_3+r_4\\
&\vec{r}=(r_1,r_2,r_3,r_4)\\
&\vec{1}^4=(1,1,1,1)\\
&r=\left\langle \vec{r},\vec{1}^4 \right\rangle\\
&\vec{r}'=\left(x^{-1}r_1+xr_3,x^{-1}r_2+xr_4\right)\\
&\vec{1}'=\left(x+x^{-1},x+x^{-1}\right)\\
&r':=\left\langle \vec{r}',\vec{1}' \right\rangle\\
&=\left\langle \left(x^{-1}r_1+xr_3,x^{-1}r_2+xr_4\right),\left(x+x^{-1},x+x^{-1}\right) \right\rangle\\
&=\left(r_1+r_2+r_3+r_4\right)+x^{-2}\left(r_1+r_2\right)+x^2\left(r_3+r_4\right)\\
&=r+x^{-2}r_L+x^2r_R
\end{aligned}$$

**并行**承诺 $\vec{a}$ 和 $z$，且添加随机向量，计算

$$C:=A+zU+rH=\left\langle \vec{a},\vec{G} \right\rangle+\left\langle \vec{a},\vec{b} \right\rangle U+\left\langle \vec{r},\vec{1}^4 \right\rangle H$$

发送 $C$；

**挑战：**证明方计算 $x:=hash(SystemParameter,C,A,U)$

**折半响应：**证明方计算

$$\vec{a}' := x^{-1}(a_1, a_2) + x(a_3, a_4) = \left(x^{-1}a_1 + xa_3, x^{-1}a_2 + xa_4\right)$$
$$\vec{b}' := x(b_1, b_2) + x^{-1}(b_3, b_4) = \left(xb_1 + x^{-1}b_3, xb_2 + x^{-1}b_4\right)$$
$$\vec{G}' := x(G_1, G_2) + x^{-1}(G_3, G_4) = \left(xG_1 + x^{-1}G_3, xG_2 + x^{-1}G_4\right)$$

发送 $\vec{a}'$；
计算

$$C' := A' + z'U + r'H = \left\langle \vec{a}', \vec{G}' \right\rangle + \left\langle \vec{a}', \vec{b}' \right\rangle U + r'H$$
$$= \left[A + x^{-2}L_a + x^2 R_a\right] + \left[z + x^{-2}l_z + x^2 r_z\right]U + \left(r + x^{-2}r_L + x^2 r_R\right)H$$
$$= \left(A + zU + rH\right) + x^{-2}\left(L_a + l_z U + r_L H\right) + x^2\left(R_a + r_z U + r_R H\right)$$
$$= C + x^{-2}\left(L_a + l_z U + r_L H\right) + x^2\left(R_a + r_z U + r_R H\right)$$
$$= C + x^{-2}L + x^{-2}R$$

其中，$L = L_a + l_z U + r_L H, R = R_a + r_z U + r_R H$ 。

发送 $\vec{a}', L, R$；

**发送数据：** $\left(C, \vec{a}', L, R\right)$，数据量与方案 3 相同

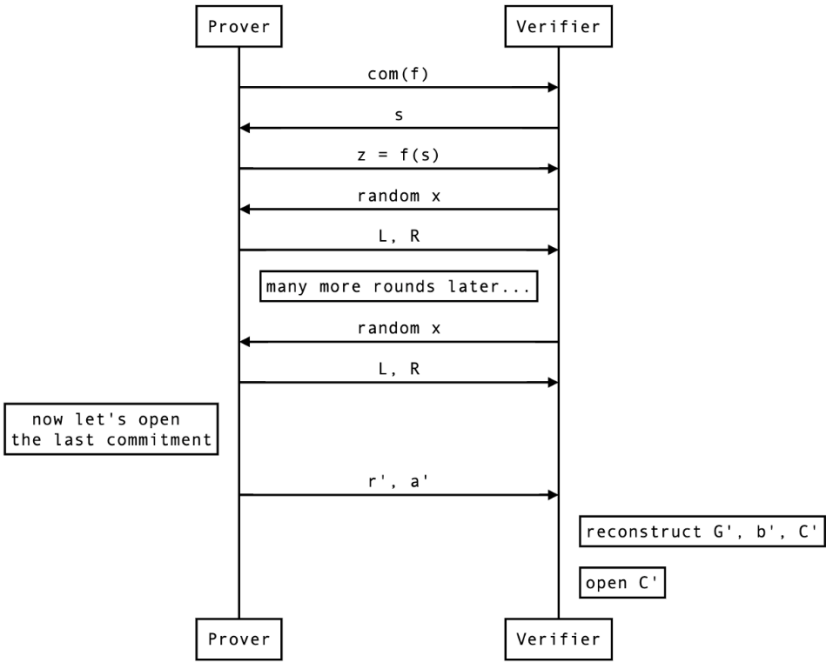
**N 次递归后，总共发送** $\left(C, a, r, L_1, R_1, ..., L_{\log_2 n}, R_{\log_2 n}\right)$

**验证：** 验证方基于 $C$ 和 $\vec{G}$ 计算 $s, x, \vec{b}'$，校验

$$C + x^{-2}L + x^2 R = \left\langle \vec{a}', \vec{G}' \right\rangle + \left\langle \vec{a}', \vec{b}' \right\rangle U + r'H$$

**N 次之后，校验等式为**

$$\sum_{i=1}^{n} C + x_i^{-2}L_i + x_i^2 R_i = aG + abU + rH$$
$$= a\left(G + bU\right) + rH$$




对于最后一次 $C == aG + abU + rH$，使用经典的 sigma 协议即可证明其知道 $a$，或直接打开 $a$。**不打开 $a$，而是使用承诺、挑战、响应证明知道 $a$。**

## 6.6Sigma 协议

**承诺：**择随机数 $d, s$，计算承诺 $R := d(G + bU) + sH$

**挑战：** $c := Hash(C, R)$

**响应：** $z_1 := ac + d, z_2 := cr + s$

**校验：** $c \cdot C + R == z_1 G + z_1 b \cdot U + z_2 \cdot H$

$$c \cdot C + R = acG + abcU + rcH + dG + dbU + sH = z_1 G + z_1 b \cdot U + z_2 \cdot H$$


四大优点

1. 使用 **plonk** 的门约束、线约束、定制门、查找表，构造多项式
2. 对多项式使用向量内积承诺：不需要基于有毒废料计算 **CRS**
3. 使用 **Tweedledum/Tweedledee** 曲线，实现递归零知识证明
4. 使用累积递归的方式，减少验证复杂度；做均摊

## 6.7 计算均摊(Amortization)

**方法 1：使用递归证明：验证方的计算复杂度较高的地方，均修改为证明方计算；**

**$\vec{G}$ 折叠验证电路，证明方生成 proof 和计算结果。**

验证方校验 proof 后，认可该计算结果，然后基于计算结果实现低复杂度验证。这个额外的 proof 和计算结果，也使用随机数与之前的多项式进行线性组合。

$$C+\sum_{i=1}^{n}\left(x_i^{-2}L_i+x_i^{2}R_i\right)=a\left(G+bU\right)+rH$$

**方法 2：均摊成本**

根据折半定理，需要 n=$\log_2$N 轮，所以等式左边需要 O($\log_2$N)时间计算。

每轮都需要折半$\vec{G}$，**验证方也需要具体计算折半过程，所以等式右边计算 G 需要 O(N)时间，**其他项需要 O(1)时间。

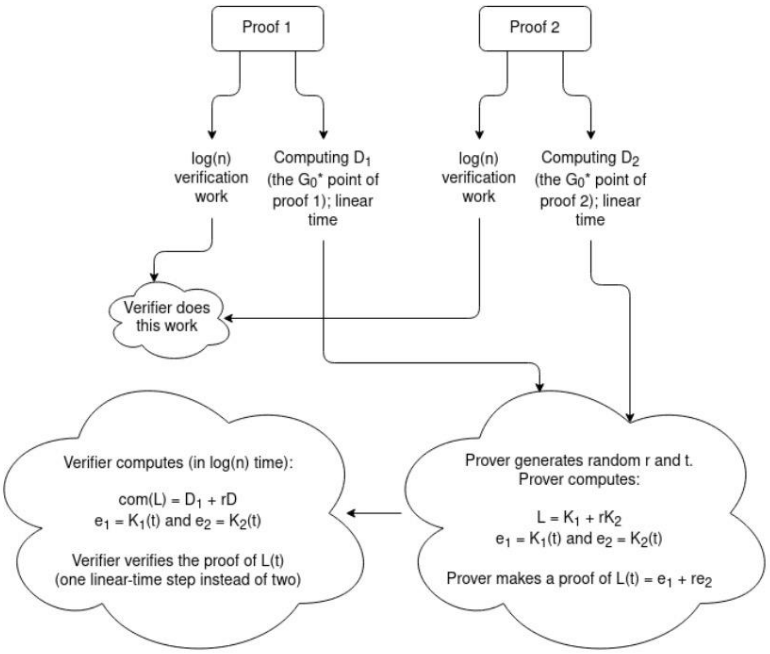
每增加一个证明 i，会增加一次左边的验证时间；

**右边 Gi 的计算可以利用随机挑战的线性组合，且让证明方计算该线性组合；**

**验证者只需在最后验证最新的 G'等于前面 Gi 的组合。**

因此，验证者最终只需要计算一次 G'，即：增加一个内积证明的验证，总验证时间只边际增加了 O(log(n))，只需要左边。

根据 Schwartz–Zippel，线性组合后发生碰撞的概率可忽略。




# 7.UltraPlonk 证明系统

**符号表达：** $G_1, G_2$ 分别是群 $\mathbb{G}_1, \mathbb{G}_2$ 的生成元， $\chi$ $\chi

$$[g]_1 = [g(\chi)]_1 = [g(X)] \cdot G_1 \in \mathbb{G}_1,$$
$$[g]_2 = [g(\chi)]_2 = [g(X)] \cdot G_2 \in \mathbb{G}_2$$

**系统初始化：**
**Plonk 门数为** $n$ ，

**（1）基于有毒废料生成 KZG 的 PK：** $(\chi \cdot [1]_1, ..., \chi^{n+5} \cdot [1]_1)$

**（2）公开且正确的表格多项式** $T_{1,i}, T_{2,i}, T_{3,i}, i = 1, ..., n$

**（3）门选择多项式：**

$$(q_{M_i}, q_{A_i}, q_{B_i}, q_{C_i}, q_{Const_i})_{i=1}^{n}, \sigma(X)$$

$$q_M(X) = \sum_{i=1}^{n} q_{M_i} L_i(X)$$

$$q_A(X) = \sum_{i=1}^{n} q_{A_i} L_i(X)$$

$$q_B(X) = \sum_{i=1}^{n} q_{B_i} L_i(X)$$

$$q_C(X) = \sum_{i=1}^{n} q_{C_i} L_i(X)$$

$$q_{Const}(X) = \sum_{i=1}^{n} q_{Const_i} L_i(X)$$

**（4）线多项式（置换多项式）：**

$$S_{\sigma_1}(X) = \sum_{i=1}^{n} \sigma(i) L_i(X)$$

$$S_{\sigma_2}(X) = \sum_{i=1}^{n} \sigma(n+i) L_i(X)$$

$$S_{\sigma_3}(X) = \sum_{i=1}^{n} \sigma(2n+i) L_i(X)$$

**共同构造 UltraPlonk 的 PK。**

**Public input:** $l, (w_i)_{i\in[l]}$

**Prover input:** $(w_i)_{i\in[3n]}$，**witness**

## 7.1 对电路赋值进行承诺

**Round 1:【不变】【电路门管脚数据多项式承诺】**电路门的输入三个信号多项式

生成随机数 $b_1, ..., b_6 \in \mathbb{F}$

$$a(X) = (b_1 X + b_2) Z_H(X) + \sum_{i=1}^{n} w_i L_i(X)$$

$$b(X) = (b_3 X + b_4) Z_H(X) + \sum_{i=1}^{n} w_{n+i} L_i(X)$$

$$c(X) = (b_5 X + b_6) Z_H(X) + \sum_{i=1}^{n} w_{2n+i} L_i(X)$$

输出**管脚数据三个多项式承诺** $[a]_1, [b]_1, [c]_1$

**注意：**

Rollup 功能，可以去掉随机项，减少计算复杂度，**仅实现数据压缩、计算压缩的功能，缺少零知识。**

zkRollup 功能，则不能去掉随机项。

Zcash 要实现零知识，则不能去掉随机项。

## 7.2 对查表数据进行承诺

**Round 2:**

基于承诺计算随机数 $\varsigma \in \mathbb{F}_p$；

保密数据向量 $\vec{f}=(f_0,...,f_{n-1})$

公开且正确的表格向量$\vec{t}=(t_0,...,t_{n-1})$表达如下：

$$f_i=\begin{cases}a(\omega^i)+\varsigma\cdot b(\omega^i)+\varsigma\cdot c(\omega^i), \text{if-the-ith-gate-is-a-lookup-gate}\\ \mathrm{T}_{1,i}+\varsigma\cdot\mathrm{T}_{2,i}+\varsigma^2\cdot\mathrm{T}_{3,i}, \text{otherwise}\end{cases}$$

$$t_i=\mathrm{T}_{1,i}+\varsigma\cdot\mathrm{T}_{2,i}+\varsigma^2\cdot\mathrm{T}_{3,i}, i=1,...,n$$

生成随机数$b_7,...,b_{13}\in\mathbb{F}$

计算**保密数据多项式** $f(X)$**和公开且正确的表格数据多项式**$t(X)$

$$f(X)=(b_7X+b_8)Z_H(X)+\sum_{i=1}^{n}f_iL_i(X)$$

$$t(X)=\mathrm{T}_{1,i}(X)+\varsigma\cdot\mathrm{T}_{2,i}(X)+\varsigma^2\cdot\mathrm{T}_{3,i}(X)$$

令$\vec{s}=(\vec{f},\vec{t})$由$\vec{t}$划分，将$\vec{s}$拆为$\vec{h}_1,\vec{h}_2$，

$$\vec{h}_1=(s_1,s_3,...,s_{2n-1})$$

$$\vec{h}_2=(s_2,s_4,...,s_{2n})$$

计算对应的多项式

$$h_1(X)=(b_9X^2+b_{10}X+b_{11})Z_H(X)+\sum_{i=1}^{n}s_{2i-1}L_i(X)$$

$$h_2(X)=(b_{12}X^2+b_{13})Z_H(X)+\sum_{i=1}^{n}s_{2i}L_i(X)$$

**输出三个多项式承诺**$[f(x)]_1,[h_1(x)]_1,[h_2(x)]_1$。

## 7.3 对相等约束置换进行承诺

**Round 3:【线数据多项式承诺】**基于承诺计算随机数$\beta,\gamma,\delta,\varepsilon\in\mathbb{F}_p$；

生成随机数$b_{14},...,b_{19}\in\mathbb{F}$，

（1）计算置换多项式（线约束）

$$z_1(X)=(b_{14}X^2+b_{15}X+b_{16})Z_H(X)+$$
$$L_1(X)+$$
$$\sum_{i=1}^{n-1}L_{i+1}(X)\prod_{j=1}^{i}\frac{w_j+\beta\omega^{j-1}+\gamma}{w_j+\sigma(j)\beta+\gamma}\frac{w_{n+j}+\beta k_1\omega^{j-1}+\gamma}{w_{n+j}+\sigma(n+j)\beta+\gamma}\frac{w_{2n+j}+\beta k_2\omega^{j-1}+\gamma}{w_{2n+j}+\sigma(2n+j)\beta+\gamma}$$

输出**线数据多项式承诺**$[z]_1$。

添加零知识的**其他方法**，在**置换集合**中添加**随机数函数值**。

## 7.4 对查表置换进行承诺

**【公开且正确的 Table 多项式承诺】**

$$z_2(X)=(b_{17}X^2+b_{18}X+b_{19})Z_H(X)+\sum_{i=1}^{n-1}\left(L_{i+1}(X)\prod_{j=1}^{i}\frac{(1+\delta)(\varepsilon+f_j)(\varepsilon(1+\delta)+t_j+\delta t_{j+1})}{(\varepsilon(1+\delta)+s_{2j-1}+\delta s_{2j})(\varepsilon(1+\delta)+s_{2j}+\delta s_{2j+1})}\right)$$

$$\Leftrightarrow F(\beta,\gamma)\equiv G(\beta,\gamma)\Leftrightarrow f\subset t$$

输出**公开且正确的表格多项式承诺**$[z]_2$。

添加零知识的**其他方法**，在**置换集合**中添加**随机数函数值**。

## 7.5 消退证明

**Round 4:【保密数据满足门约束与线约束、查找表】**基于上述承诺计算随机数$\alpha\in\mathbb{F}_p$。

**商多项式存在，则确保上述门约束与线约束成立、查找表**

$$q(X)=\frac{1}{Z_H(X)}\begin{cases}Gate:(q_A(X)\cdot a(X)+q_B(X)\cdot b(X)+q_C(X)\cdot c(X)+q_M(X)\cdot a(X)\cdot b(X)+q_{Const}(X))\cdot 1\\ Line:+\big((a(X)+\beta X+\gamma)(b(X)+\beta k_1X+\gamma)(c(X)+\beta k_2X+\gamma)z(X)\big)\cdot\alpha\\ Line:-\big((a(X)+\beta S_{\sigma_1}(X)+\gamma)(b(X)+\beta k_1S_{\sigma_2}(X)+\gamma)(c(X)+\beta k_2S_{\sigma_3}(X)+\gamma)z(\omega X)\big)\cdot\alpha\\ Line:+(z(X)-1)L_1(X)\cdot\alpha^2\\ +q_K(X)(a(X)+\varsigma\cdot b(X)+\varsigma^2\cdot b(X)-f(X))\cdot\alpha^3\\ +z_2(X)(1+\delta)(\varepsilon+f(X))(\varepsilon(1+\delta)+t(X)+\delta t(X\omega))\cdot\alpha^4\\ -z_2(X\omega)(\varepsilon(1+\delta)+h_1(X)+\delta h_1(X\omega))(\varepsilon(1+\delta)+h_2(X)+\delta h_2(X\omega))\cdot\alpha^4\\ +(z_2(X)-1)L_1(X)\cdot\alpha^5\end{cases}$$

生成随机数$b_{20},\ldots,b_{22}\in\mathbb{F}$，

将$q(X)$分解为 3 个多项式

$$q(X)=\left(q_{low}(X)+b_{10}X^n\right)+\left(X^{n+1}q_{mid}(X)-b_{10}+b_{11}X^n\right)+\left(X^{2n+4}q_{high}(X)-b_{11}\right)$$

**红色部分是随机项，缺少这部分，则无法实现零知识。**

输出**这三个多项式承诺**$\left([q_{low}]_1,[q_{mid}]_1,[q_{high}]_1\right)$。

这个商多项式确保门约束和线约束、查找表约束正确。

## 7.6 多项式求值

**Round 5:** 【多项式随机打开点】基于上述承诺计算随机数 $\mathfrak{I} \in \mathbb{F}_p$。计算**多项式的值**

$$
\begin{aligned}
&a(\mathfrak{I}), b(\mathfrak{I}), c(\mathfrak{I}),\\
&S_{\sigma_1}(\mathfrak{I}), S_{\sigma_2}(\mathfrak{I}),\\
&f(\mathfrak{I}), t(\mathfrak{I}), t(\omega\mathfrak{I}),\\
&z_1(\omega\mathfrak{I}), z_2(\omega\mathfrak{I}),\\
&h_1(\omega\mathfrak{I}), h_2(\omega\mathfrak{I})
\end{aligned}
$$

输出上述**函数值**。

## 7.7 多点打开证明

**Round 6:** 基于上述承诺计算随机数 $v \in \mathbb{F}_p$。

计算一个辅助的**线性多项式**

$$
r(X) = \begin{cases}
Gate: \left(a(\mathfrak{I})b(\mathfrak{I})q_M(X) + a(\mathfrak{I})q_A(X) + b(\mathfrak{I})q_B(X) + c(\mathfrak{I})q_C(X) + PI(\mathfrak{I}) + q_{Const}(X)\right)\cdot 1\\
Line: +\alpha\cdot[\left(\left(a(\mathfrak{I}) + \beta\mathfrak{I} + \gamma\right)\left(b(\mathfrak{I}) + \beta k_1\mathfrak{I} + \gamma\right)\left(c(\mathfrak{I}) + \beta k_2\mathfrak{I} + \gamma\right)z_1(X)\right)\\
Line: -\left(\left(a(\mathfrak{I}) + \beta s_{\sigma_1}(\mathfrak{I}) + \gamma\right)\left(b(\mathfrak{I}) + \beta k_1 s_{\sigma_2}(\mathfrak{I}) + \gamma\right)\left(c(\mathfrak{I}) + \beta\cdot S_{\sigma_3}(\mathfrak{I}) + \gamma\right)\right)z_1(\mathfrak{I}\omega)]\\
Line: +\alpha^2\cdot(z_1(X) - 1)L_1(\mathfrak{I})\\
+\alpha^3\cdot q_K(X)(a(\mathfrak{I}) + \varsigma\cdot b(\mathfrak{I}) + \varsigma^2\cdot c(\mathfrak{I}) - f(\mathfrak{I}))\\
+\alpha^4\cdot[z_2(X)(1+\delta)(\varepsilon + f(\mathfrak{I}))(\varepsilon(1+\delta) + t(\mathfrak{I}) + \delta t(\mathfrak{I}\omega))\\
-z_2(\mathfrak{I}\omega)(\varepsilon(1+\delta) + h_1(\mathfrak{I}) + \delta h_1(\mathfrak{I}\omega))(\varepsilon(1+\delta) + h_2(\mathfrak{I}) + \delta h_2(\mathfrak{I}\omega))]\\
+\alpha^5\cdot(z_2(X) - 1)L_1(\mathfrak{I})\\
-\alpha^6\cdot Z_H(\mathfrak{I})\left((q_{low}(\mathfrak{I}) + b_{10}\mathfrak{I}^n) + (\mathfrak{I}^{n+1}q_{mid}(\mathfrak{I}) - b_{10} + b_{11}\mathfrak{I}^n) + (\mathfrak{I}^{2n+4}q_{high}(\mathfrak{I}) - b_{11})\right)
\end{cases}
$$

**计算 KZG 承诺中的商多项式，确保上述所有多项式正确**

$$W_{\mathfrak{I}}(X)=\frac{1}{X-\mathfrak{I}}\begin{pmatrix} r(X) \\ +v\cdot\big(a(X)-a(\mathfrak{I})\big) \\ +v^2\cdot\big(b(X)-b(\mathfrak{I})\big) \\ +v^3\cdot\big(c(X)-c(\mathfrak{I})\big) \\ +v^4\cdot\big(S_{\sigma_1}(X)-s_{\sigma_1}(\mathfrak{I})\big) \\ +v^5\cdot\big(S_{\sigma_2}(X)-s_{\sigma_2}(\mathfrak{I})\big) \\ +v^6\cdot\big(f(X)-f(\mathfrak{I})\big) \\ +v^7\cdot\big(h_2(X)-h_2(\mathfrak{I})\big) \\ +v^8\cdot\big(t(X)-t(\mathfrak{I})\big) \end{pmatrix}$$

$$W_{\omega\mathfrak{I}}(X)=\frac{1}{X-\omega\mathfrak{I}}\begin{pmatrix} z_1(X)-z_1(\mathfrak{I}\omega) \\ +v\cdot\big(t(X)-t(\mathfrak{I}\omega)\big) \\ +v^2\cdot\big(z_2(X)-z_2(\mathfrak{I}\omega)\big) \\ +v^3\cdot\big(h_1(X)-h_1(\mathfrak{I}\omega)\big) \end{pmatrix}$$

计算并输出**商多项式**承诺$[W_{\mathfrak{I}}]_1,[W_{\omega\mathfrak{I}}]_1$。

**商多项式存在，则确保多项式的随机打开点正确，包括：门多项式、线多项式、门约束与线性约束多项式、Table 多项式等均正确。**

**最终证明为**

$$\pi_{SNARK}=\begin{pmatrix} Commit:[a(x)]_1,[b(x)]_1,[c(x)]_1, \\ Commit:[f(x)]_1,[h_1(x)]_1,[h_2(x)]_1, \\ Commit:[z_1(x)]_1,[z_2(x)]_1, \\ Commit:[q_{low}]_1,[q_{mid}]_1,[q_{high}]_1, \\ Commit:[W_{\mathfrak{I}}]_1,[W_{\omega\mathfrak{I}}]_1, \\ Value:a(\mathfrak{I}),b(\mathfrak{I}),c(\mathfrak{I}), \\ Value:s_{\sigma_1}(\mathfrak{I}),s_{\sigma_2}(\mathfrak{I}), \\ Value:f(\mathfrak{I}),t(\mathfrak{I}),t(\mathfrak{I}\omega), \\ Value:z_1(\mathfrak{I}\omega),z_2(\mathfrak{I}\omega),h_1(\mathfrak{I}\omega),h_2(\mathfrak{I}) \end{pmatrix}$$



# 8.Halo2 证明系统 1

**Halo2 的证明系统与 UltraPlonk 严格的一一对应关系。**

为辅助解释，经常如下的约束系统：

- 四个 advice 列 a,b,c,d
- 一个 fixed 列 f

$$a \cdot b \cdot c_{-1} - d = 0$$

- 三个自定义门： $f_{-1} \cdot c = 0$

$$f \cdot d \cdot a = 0$$

# 8.1 查表证明系统简介

## 8.1.1 查表证明

查找表数据一共 $2^k$ 行，编号从 0 开始。A 和 S 列分别是“子集”和“全集”。

**查找表证明：A 列中的元素都在 S 列中** $f \subset t$ **。**具体而言，A 列中多个位置的元素对应 S 列中同一位置的元素，且 S 列中的某些元素可以不出现在 A 列的任何位置上。

- S 列并不一定是固定的。可以支持固定元素的查找或动态查找（后者采用 advice 列）。
- A 和 S 列中的元素可以重复。如果 A 和 S 列中的元素个数不正好是 $2^k$，可以用 A 和 S 列中任意元素进行扩展。

或可以增加一个“查找选择多项式”，控制在 A 列中的哪些元素参与查找。采用这种方法可以替换下述公式中的 A(X)。如果一个元素不需要查找，用 S0 替换 A。
假设 $l_i$ 是拉格朗日基多项式，在 i 行多项式为 1，其他行为 0。
从 A 和 S 列的置换开始，置换分别为 A’和 S’列。
可以通过**置换证明** Z 约束它们之间的**置换关系**：

$$Z(\omega X) \cdot (A'(X) + \beta) \cdot (S'(X) + \gamma) - Z(X) \cdot (A(X) + \beta) \cdot (S(X) + \gamma) = 0$$
$$l_0(X) \cdot (1 - Z(X)) = 0$$

注释：该等式为**累加器的递归表达**和起始约束。

**等价于，**在除以 0 不发生的情况下，对所有的 i∈[0,2^k)满足：

$$Z_{i+1} = Z_i \cdot \frac{(A_i + \beta) \cdot (S_i + \gamma)}{(A_i' + \beta) \cdot (S_i' + \gamma)}$$
$$Z_{2^k} = Z_0 = 1$$

注释：该等式为**累加器的**递归表达和起始约束。

这证明 A’和 S’列分别是 A 和 S 列的置换，但是并没有指明具体的置换关系。
β 和 γ 是独立因子，**采用这两个因子，可以将两个置换论据组合在一起。**

这些置换的目的是让证明者提供的 A’和 S’列满足一定的条件：

[1] A’列中相同的元素位置靠在一起。这可以通过某种排序算法实现。相同的元素在 A’列中挨在一起，并且 A’列是 A 列的置换。

[2] **A’列中挨在一起的相同元素的第一个元素存在于 S’列中。**除去这个限制外，S’列是 S 列的一个任意置换。

通过如下的规则限制 $A_i' = S_i'$ 或 $A_i' = A_{i-1}'$

$$\left(A'(X)-S'(X)\right)\cdot\left(A'(X)-A'(\omega X)\right)=0$$

注释：左边对应条件 2，中间对应条件 1.

通过如下的规则限制 $A_0'=S_0'$

$$l_0(X)\cdot\left(A'(X)-S'(X)\right)=0$$

由于第二个规则，第一个规则中的$\left(A'(X)-A'(\omega X)\right)$这一项在 0 行没有效果，尽管 $\omega^{-1}X$ “能反转”。

因此，**证明了 A 列中的元素都在 S 列中，A’列中的元素都在 S’列中。**

### 8.1.2 添加零知识

在 PLONK 算法为基础的证明系统中加入零知识，**需要在每列的最后加入 t 个随机元素。**这些需要对查找证明进行调整，因为这些随机的元素并不满足之前的约束关系。

限制有效的行数为$u=2^k-t-1$。增加两个**选择多项式**：

- $q_{blind}(X)$ 在最后 t 行设置为 1，其他行设置为 0；
- $q_{last}(X)$ 只在 u 行设置为 1，其他行设置为 0（它设置在有效行和盲化行的交界处）。

将之前的规则限制在有效行上：

$$\left(1-\left(q_{last}(X)+q_{blind}(X)\right)\right)\cdot Z(\omega X)\cdot(A'(X)+\beta)\cdot(S'(X)+\gamma)-Z(X)\cdot(A(X)+\beta)\cdot(S(X)+\gamma)=0$$

$$\left(1-\left(q_{last}(X)+q_{blind}(X)\right)\right)\cdot\left(A'(X)-S'(X)\right)\cdot\left(A'(X)-A'(\omega X)\right)=0$$

**注释 1：**第一个公式是**累加器**的递归表达**且添加随机多项式**，第二个公式确保 A’列中挨在一起的相同元素的第一个元素存在于 S’列中，且添加随机多项式。

**注释 2：**添加的随机项与 plonk 论文中 round1/2 添加的红色部分随机项效果一样的 $a(X)=(b_1X+b_2)Z_H(X)+\sum_{i=1}^{n}w_iL_i(X)$

在 0 行的限制规则保持不变：

$$l_0(X)\cdot\left(A'(X)-S'(X)\right)=0$$

$$l_0(X)\cdot(1-Z(X))=0$$

因为不能再依赖在 $\omega^{2^k}$ 点的环绕保证 Z 为 1，相反要约束 $Z(\omega^u)$ 为 1。这里有个难点：如果在任意 i∈[0,u)， $A_i+\beta$ 或 $S_i+\gamma$ 为 0 的话，置换证明可能不成立。虽

然这种情况在 β 和 γ 的取值下概率可以忽略，但是，对于完美零知识和完备性来说是个障碍（攻击者可以构造这样的情况）。

确保完美的**完备性**和**零知识**，允许 $Z(\omega^u)$ 为 0 或 1:

$$q_{last}(X)\cdot\big(Z(X)\cdot(1-Z(X))\big)=0$$

如果 $A_i+\beta$ 或 $S_i+\gamma$ 在某些 i 上为 0，则可以在 i<j≤u 范围设置 $Z_j=0$，满足上述的约束。注释：**添加一个选择多项式，使得在索引为 $\omega^u$ 处满足约束。**

注意的是，挑战因子 β 和 γ 是在 A 和 S 列（以及 A'和 S'列）承诺后生成的，证明者无法伪造 $A_i+\beta$ 或 $S_i+\gamma$ 为 0 的情况。因为这种情况的概率可以忽略，可行性不受影响。

### 8.1.3 开销

- 原始 A 列和固定 S 列。
- 置换函数 Z。
- 两个置换 A'和 S'列。
- 门约束方程的阶不高。

### 8.1.4 一般化

halo2 的查找证明实现实现了上述技术的一般化：

- **A 列和 S 列扩展为多列**，通过**随机数线性组合**。A'和 S'列可以是单列。
  - S 列的各列承诺可以**提前计算**。这样在挑战因子确定后，利用 Pedersen 承诺的同态性质，可以很简便的组合成 S 列的承诺。
  - A 列可以是采用相对引用的任意多项式表达式。这些可以替换到约束规则中去，受制于最大的阶。这样可能可以省去一个或者多个 advice 列。
- 查找论据可以通过子集论据实现任意长度的关系。为了约束 R(x,y,...)，将 R 看成是 S（通过前面的方法），并且检查关系(x,y,...)∈R。
  - 如果 R 代表一个函数，同样需要检查输入是否在域中。这是想要的，经常会省去额外的范围检查。
- 可以在同一个电路中支持多表。利用标示列将多个表组合成一张表。
  - 标示列可以和之前提到的"查找选择多项式"合并。

这些泛化和 Plookup 论文的第 4 和第 5 节中的技术类似。与 Plookup 的区别是子集论据。相关技术，子集论据也适用；Plookup 论文的第 5 节中的优化范围检查技术同样可以用在子集论据中。

## 8.2 置换证明系统简介

halo2 电路中的门要“本地”操作（就是在当前行或者事先定义好的相关行进行操作）**通常需要将某个任意单元格的值拷贝到当前行，以便在门中使用。该工作就由线约束来描述，该约束就强制源和目的单元格中包含相同的值。**

通过构造一个代表这些约束的置换来实现这些相等约束，进而在最终的证明中包含一个置换证明来确保约束成立。

### 8.2.1 相关记号

置换就是一个集合以一对一的方式自己到自己的一个映射。一个置换可以唯一分解成多个轮换的组合（每个轮换都是首尾相接的，轮换和轮换之间排序）。

通常，使用轮换记号来表示置换。令(a b c)代表如下轮换：a 映射到 b，b 映射到 c，c 映射到 a（该法则可以推广到任意大小的轮换）。把两个或更多的轮换一个接一个的写在一起就代表了一个相关置换的组合。比如，(a b)(c d)就代表了 a 映射到 b，b 映射到 a，c 映射到 d，d 映射到 c 的置换。

### 8.2.2 构造置换

#### 目标

构造这样一个置换，在同一个相等约束下的这些变量形成一个轮换。
举例，定义如下相等约束的电路：

a≡b
a≡c
d≡e

由上可得相等约束集合{a,b,c}和{d,e}，构造如下的置换：

(a b c), (d e)

该置换就定义由[a,b,c,d,e]到[b,c,a,e,d]的一个映射。

#### 算法

需要记录这些轮换的集合，**实际上也就是一些不相交集合的集合。**
采取如下方法来表示当前的状态：

- 数组 mapping 表示**置换本身**；
- 辅助数组 aux 记录每个轮换中代表**元素**；
- 数组 sizes 记录每个轮换的**元素个数**。

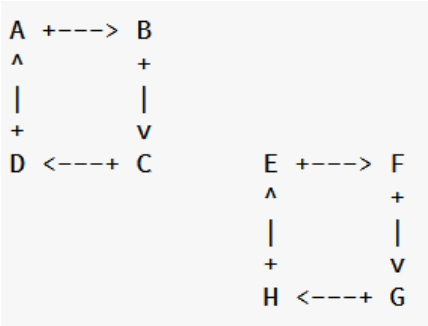
一个给定的轮换 C 中选取一个不变的值 c，对 C 中的每个元素 x，aux(x)都得到相同的值，即 c∈C。有了这个值，对于给定的两个给定的元素 x 和 y，可以通过检查 aux(x)=aux(y)是否成立，快速判断它们是否在同一个轮换中。sizes(aux(x))也代表了包含 x 的轮换的大小。（只有 sizes(aux(x))是有保证的，而不是 sizes(x)。）

该算法以表示一个单位置换开始：
对所有 x，令 mapping(x)=x，aux(x)=x，和 sizes(x)=1。

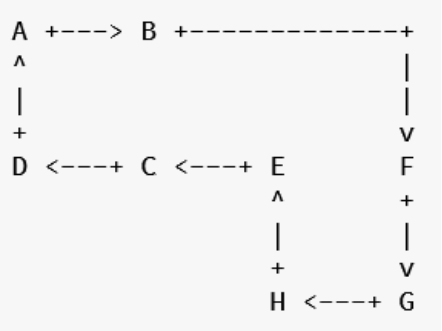
按如下步骤**增加一条相等约束** left ≡ right：

1. **检查 left 和 right 是否已经在同一个轮换中，即检查 aux(left)=aux(right)是否成立。如果是此种情况，那么不许做任何事情。**
2. 否则，left 和 right 就一定分属不同的轮换。令 left 是相对大的轮换，而 right 则为相对小的那个。如果 sizes(aux(left))<sizes(aux(right))，则交换一下使其满足需求。
3. 令 sizes(aux(left)):=sizes(aux(left))+sizes(aux(right))。
4. 下一步对 right（较小的）轮换中每一个元素 x 令 aux(x):=aux(left)。
5. **通过交换 mapping(left)和 mapping(right)中的元素（的轮换），将较小的轮换接入到较大的轮换中去。**

举例，两个不相交的**轮换**(A B C D)和(E F G H)：




**在增加约束 B≡E 后，上文算法将得出如下的轮换：**




**3 个线约束，3 个线约束线性组合**

**一个线约束**

如果不检查 left 和 right 是否在同一个轮换中，**那么可能就会丢掉某些相等约束。**举个例子，如果有如下的约束:

- a≡b
- b≡c
- c≡d
- b≡d

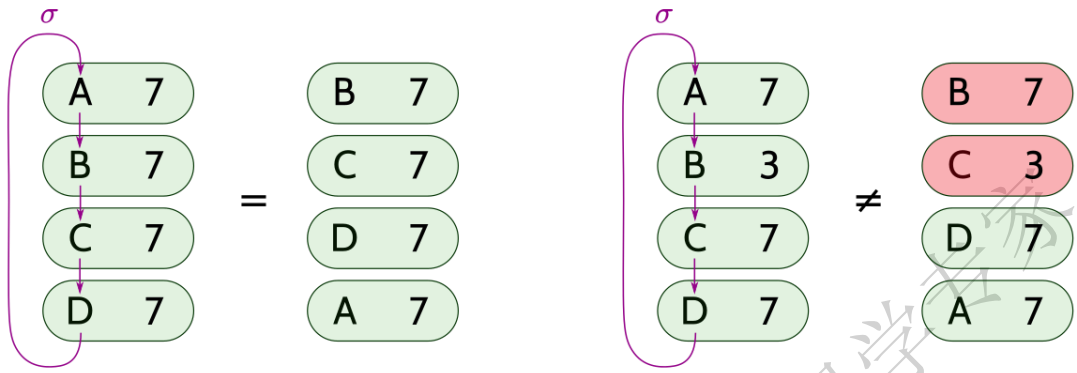
如果在处理一条新的相等约束时仅只执行上述算法中的第五步，那么得到的最终结果将是(a b)(c d)，而不是正确的结果(a b c d)。

## 8.2.3 证明系统说明

注释：就是**累加器**系统说明。

需要检查 m 列中所有单元格的置换，该 m 列分别由其拉格朗日基的多项式 $v_0,\ldots,v_{m-1}$ 来表示。

用 F×中不同的元素来标记 m 列中每个单元格。
假设基于这些标记的一个置换：σ(column:i,row:j)=(column:i',row:j')。其中，这些轮换就对应了相等约束。
考虑对{(label,value)}的集合，当且仅当对每对的 label 都按照 σ 进行置换后，得到了与原集合相同的集合，才能证明集合中的每个对是相等的。




因为所有的标记都是不同的，最终就可以用一个代表相等约束的集合来代替多个代表相等关系的集合，而这进一步可以用**累加器**来检查置换的正确性。

令 ω 是 $2^k$ 次单位根，δ 是 T 次单位根，满足 $T\cdot 2^S+1=p$，其中 T 是奇数并且 k≤S。在置换证明中，用 $\delta^i\cdot\omega^j\in F^\times$作为第 j 行、第 i 列的单元格的标记。

σ 由 m 个多项式组成的一个向量，其中对每一项 si(X)满足 $s_i(\omega^j)=\delta^{i'}\cdot\omega^{j'}$。

用一个包含 m 个多项式的向量来标识一个置换，其中每一项 $ID_i(\omega^j)$ 都满足

$ID_i(\omega^j)=\delta^i\cdot\omega^j$。

用一个挑战 β 将每一个(label, value)对压缩成 value+β·label。与在查找表的积证明中所做的一样，也使用一个挑战 γ 来盲化每一个积项。

有一个在 m 列，即：$v_0,\ldots,v_{m-1}$ 上的一个置换，表示为 $s_0,\ldots,s_{m-1}$，想要确保如下等式成立：

$$\prod_{i=0}^{m-1}\prod_{j=0}^{n-1}\frac{\left(v_i(\omega^j)+\beta\cdot\delta^i\cdot\omega^j+\gamma\right)}{\left(v_i(\omega^j)+\beta\cdot s_i(\omega^j)+\gamma\right)}=1$$

- $v_i(\omega^j)+\beta\cdot\delta^i\cdot\omega^j$ 代表置换**前**的对(label, value)
- $v_i(\omega^j)+\beta\cdot s_i(\omega^j)$ 代表置换**后**的对(σ(label), value)

注释：就是**累加器**的最终表达。

令多项式 $Z_P$ 满足

$$Z_P(\omega^0)=Z_P(\omega^n)=1$$

并且对 0≤j<n:

$$Z_P(\omega^{j+1}) = \prod_{h=0}^{j}\prod_{i=0}^{m-1}\frac{\left(v_i(\omega^h)+\beta\cdot\delta^i\cdot\omega^h+\gamma\right)}{\left(v_i(\omega^h)+\beta\cdot s_i(\omega^h)+\gamma\right)} = Z_P(\omega^j)\cdot\prod_{i=0}^{m-1}\frac{\left(v_i(\omega^j)+\beta\cdot\delta^i\cdot\omega^j+\gamma\right)}{\left(v_i(\omega^j)+\beta\cdot s_i(\omega^j)+\gamma\right)}$$

注释：分别为坐标累机器的**最终表达**与**递归表达**。

接下来只需要强制满足如下的规则：

$$Z_P(\omega X)\cdot\prod_{i=0}^{m-1}\left(v_i(X)+\beta\cdot s_i(X)+\gamma\right)-Z_P(X)\cdot\prod_{i=0}^{m-1}\left(v_i(X)+\beta\cdot\delta^i\cdot X+\gamma\right)=0$$

$$l_0\cdot(1-Z_P(X))=0$$

注意，上述第一条规则对其处理的多项式的假设其约束的列数 m 要符合 PLONK 设置的次数界限。接下来，将在下文阐述如何处理列数超出边界的情况。

### 8.2.4 添加零知识

与查找表证明类似，也需要调整上述的证明来处理每列的最后 t 行，因为它们被填入了随机值（也就是这些行不满足上面的积证明）。

限定有用的行数是 $u=2^k-t-1$，同时增加两个选择多项式，这两个**选择多项式**与查找表证明中定义的是一样的：

- $q_{blind}$ 只在最后 **t 行设为 1**，而在其他地方则为 0。
- $q_{last}$ 则旨在 u 行设为 1，其他行则为 0（只在有用行和盲化行之间的那一行起作用）。

（1）在那些有用的行真正应用累加器原理：

$$(1-q_{last}(X)-q_{blind}(X))\cdot\left(Z_P(\omega X)\cdot\prod_{i=0}^{m-1}\left(v_i(X)+\beta\cdot s_i(X)+\gamma\right)-Z_P(X)\cdot\prod_{i=0}^{m-1}\left(v_i(X)+\beta\cdot\delta^i\cdot X+\gamma\right)\right)=0$$

（2）在 0 行的的规则仍然是相同

$$l_0(X)\cdot(1-Z_P(X))=0$$

（3）由于不能依赖全景来确保每一个积证明 $Z_P$ 都在 $\omega^{2^k}$ 等于 1，那么实际上应该约束 $Z(\omega^u)=1$。这在查找证明中同样带来问题，需要允许 $Z(\omega^u)$ 等于 0 或者 1：

$$q_{last}(X)\cdot(Z_P(X)^2-Z_P(X))=0$$

### 8.2.5 支持大列

**在 halo2 的实现中，实际上并不限制相等约束可以占用的行数。**因此，就必须解决上述方法中可能导致的积规则中的**多项式可能超出 CRS 设置的次数边界的问题。**一个简单办法是直接升高次数边界，但是在没有其他规则需要用到如此大的次数的情况下，这一办法是低效的。

另一种方法，把积证明**分割成 b 个**，每个都是其中 m 列的证明，即：$Z_{P,0},\ldots Z_{P,b-1}$，同时增加一条规则，就是每个积（证明）最终的积被设置成下一个积（证明）的起始值。
这就是说，对于 0≤a<b，**累加器**多项式**分为多份**：

$$\left(1-(q_{last}(X)+q_{blind}(X))\right)\cdot$$
$$\left(Z_{P,a}(\omega X)\cdot\prod_{i=am}^{(a+1)m-1}\left(v_i(X)+\beta\cdot s_i(X)+\gamma\right)-Z_P(X)\cdot\prod_{i=am}^{(a+1)m-1}\left(v_i(X)+\beta\cdot\delta^i\cdot X+\gamma\right)\right)=0$$

上述规则假设总列数是 m 的整数倍；如果不是，那么就让最后一个列集合少于 m 个项。
（1）对第一个列集合

$$l_0(X)\cdot(1-Z_{P,0}(X))=0$$

（2）对于剩余的每一个列集合 0<a<b，使用如下规则将 $Z_{P,a-1}(\omega^u)$ 拷贝到下一个列集合的最开始 $Z_{P,a}(\omega^0)$

$$q_0(X)\cdot(Z_{P,a}(X)-Z_{P,a-1}(\omega^u X))=0$$

（3）对最后一个列集合，约束 $Z_{P,b-1}(\omega^0)$ 为 0 或者 1

$$q_{last}(X)\cdot(Z_{P,b-1}(X)^2-Z_{P,b-1}(X))=0$$

因此，提供**完备性**和**零知识**。

# 8.3 电路承诺

## 8.3.1 对电路赋值进行承诺（对应 7.1 节）

### 注释：对应 UltraPlonk 第 7.1 节

在生成证明之前，**证明者已经有一张能够满足约束条件的单元格赋值表。**赋值表有 $n=2^k$ 行；其列被划分为三类：advice,instance,fixed。
定义 $F_{i,j}$ 表示第 j 行的第 i 个 fixed 列，由验证者提供的；
定义 $A_{i,j}$ 为第 j 行的第 i 个 advice 或 instance 列，由证明者提供的；
实际上，证明者和验证者都需要计算 instance 列和 fixed 列的承诺，**只有对 advice 列的承诺在证明中。**

为对表中的各个赋值进行承诺，为每列构造一个次数为 n−1 的拉格朗日多项式，其取值域大小为 n（令 ω 为 n 次单位根）。

- 【advice/instance 数据多项式承诺】令 $a_i(\omega^j) = A_{i,j}$，得到 $a_i(X)$ 多项式
- 【fixed 列多项式承诺】令 $f_i(\omega^j) = F_{i,j}$，得到 $f_i(X)$ 多项式

然后对每列的多项式进行承诺：

$$A = [Commit(a_0(X)), \ldots, Commit(a_i(X))]$$
$$F = [Commit(f_0(X)), \ldots, Commit(f_i(X))]$$

其中，F 在生成密钥的时候被创建，使用 1 作为盲因子。A 由证明者计算并发送给验证者。

## 8.3.2 对查表数据进行承诺（对应 7.2 节）

验证者随机选择一个 θ，用于确保同一个查找表内不同的列线性无关。接着证明者对每个查找表的置换进行承诺

- 给定一个查找表，输入列多项式为[$A_0(X),\ldots,A_{m-1}(X)$]，真值表列的多项式为[$S_0(X),\ldots,S_{m-1}(X)$]，证明者构造**两个压缩后的多项式**

$$A_{compressed}(X) = \theta^{m-1}A_0(X) + \theta^{m-2}A_1(X) + \cdots + \theta A_{m-2}(X) + A_{m-1}(X)$$
$$S_{compressed}(X) = \theta^{m-1}S_0(X) + \theta^{m-2}S_1(X) + \cdots + \theta S_{m-2}(X) + S_{m-1}(X)$$

- 然后证明者对 $A_{compressed}(X)$和 $S_{compressed}(X)$依照**查找表证明**的规则进行排列，得到 A’(X)和 S’(X).

证明者为所有的查找表进行承诺：

$$L = [(Commit(A'(X))), Commit(S'(X))), \ldots]$$

并发送给验证者。

当验证者接收到 A，F，和 L 后，随机采样 β 和 γ 作为挑战，用于后续的置换和查找表证明验证。（随机采样是可以重复利用的，因为证明是相互独立的）。

## 8.3.3 对相等约束置换进行承诺（对应 7.3 节）

令 c 为相等约束所涉及的列数。
令 m 为能容纳的最大列数，m 不能超过 PLONK 配置中多项式的次数上限。
令 u 为置换证明章节中定义的可以“使用”的行数。
令 b=ceiling(c/m).
证明者构造一个长度为 bu 的向量 P，对于每个列集合，有，0≤a<b，对于每一行有 0≤j<u

$$P_{au+j} = \prod_{i=am}^{\min(c,(a+1)m)-1} \frac{\left(v_i(\omega^i) + \beta \cdot \delta^i \cdot \omega^j + \gamma\right)}{\left(v_i(\omega^j) + \beta \cdot s_i(\omega^j) + \gamma\right)}$$

证明者计算 P 的**坐标累机器**，从 1 开始，并且多项式向量 $Z_{P,0..b-1}$ 每个都有拉格朗日基表示，基于坐标累机器的长度为 u 的切片，如置换证明章节中所描述的。

最后证明者为每个 $Z_{P,a}$ 多项式进行承诺：

$$Z_P=[Commit(Z_{P,0}(X)),\ldots,Commit(Z_{P,b-1}(X))]$$

并发送给验证者。

### 8.3.4 对查找置换进行承诺（对应 7.4 节）

除了需要对单独的相等约束进行承诺外，对每一个查找表，证明者也需要对置换进行承诺。
证明者构造一个向量 P:

$$P_j=\frac{\left(A_{compressed}(\omega^i)+\beta\right)\left(S_{compressed}(\omega^i)+\gamma\right)}{\left(A'(\omega^i)+\beta\right)\left(S'(\omega^i)+\gamma\right)}$$

证明者构造多项式 $Z_L$，拉格朗日基表示为 P 多项式的**累加器**，从 $Z_L(1)=1$ 开始。β 和 γ 用于在组合 A'(X)和 S'(X)的同时保持这两者无关。β 和 γ 是验证者在证明者创建多项式 A,F,和 L 之后采样的（因此，承诺在查找表列中用到的单元格的值，以及每个查找表的 A'(X)和 S'(X)）。

如之前一样，证明者对每个 ZL 多项式进行承诺：

$$Z_L=[Commit(Z_L(X)),\ldots]$$

## 8.4 消退证明（对应 7.5 节）

### 8.4.1 商多项式

**满足所有的约束关系：门约束、线约束、查表约束**

在电路的所有赋值都已经承诺之后，证明者现在需要证明各种各样的电路关系都是满足的：

- 标准门，
- 定制门，用多项式 $gate_i(X)$表示。
- 查找证明约束
- 相等约束

从电路的列的角度看，每个关系都由一个 d 次多项式表示（在所有关系中最大的次数。假设对应一列的赋值多项式的次数是 n−1，那么从 X 的角度看，关系多项式的次数就是 d(n−1)

在例子中，门多项式的次数是 3n−3。

$$gate_0(X) = a_0(X) \cdot a_1(X) \cdot a_2(X\omega^{-1}) - a_3(X)$$
$$gate_1(X) = f_0(X\omega^{-1}) \cdot a_2(X)$$
$$gate_2(X) = f_0(X) \cdot a_3(X) \cdot a_0(X)$$

如果描述关系的多项式为 0，则关系是满足的。一种表示这种此种约束的方法，就是将每个多项式关系都除以**消失多项式** $t(X)=(X^n-1)$，消失多项式是以 $\omega^i$ 作为根的最低次数的单项式。如果关系多项式能被 t(X)整除，那么在域上这个关系多项式就等于 0。

这种朴素的构造方式的弊端在于，其需要对每个关系都需要有一个多项式承诺。相反，可以对电路中**所有关系同时进行承诺**：验证者取一个 y，然后证明者构造一个商多项式，

$$h(X) = \frac{gate_0(X) + y \cdot gate_1(X) + \cdots + y^i \cdot gate_i(X) + \ldots y^j wire(X) + \ldots + y^k table(X)}{t(X)}$$

其中，分子是电路关系的随机的线性组合（证明者需要在验证者给出 y 之前先承诺单元格赋值）。

- 如果分子多项式（以 X 为自变量）能够被**消失多项式** t(X)整除，那么所有的关系都满足的概率就是非常高的
- 相反地，根据 Schwartz-Zippel 引理，攻击者碰撞成功概率可忽略。在点 x 处，h(x)·t(x)与分子多项式在该处的值就几乎不可能相等。也就是说，在此种情况下，分子多项式不能整除 t(X)。

### 8.4.2 商多项式承诺

h(X)的次数是 (d−1)n−d（因为分母 t(X)的次数是 n）。但是，在 Halo2 中的多项式承诺机制仅支持次数 n−1 次的多项式的承诺（这是因为除了关系多项式的承诺，协议其他部分所需承诺的多项式的最大次数就是 n−1）。为了不给多项式承诺机制增加额外的成本，验证者就需要把 h(X)**分割成多个块**，每个只有 n−1 次的多项式的和，

$$h_0(X) + X^n h_1(X) + \cdots + X^{n(d-1)} h_{d-1}(X)$$

并对每一部分均进行承诺。

$$H = [Commit(h_0(X)), Commit(h_1(X)), \ldots, Commit(h_{d-1}(X))].$$

## 8.5 多项式求值（对应 7.6 节）

电路所有的特点都已经被承诺了。现在，验证者想要验证一下证明者提供的承诺是不是正确的 h(X)。于是，验证者选取一个 x，证明者需要提供其所声称的所有多项式在 x 处的值，包括电路中使用的所有相对偏移和 h(X)的值。
在举例中，这就是：

$$a_0(x)$$
$$a_1(x)$$
$$a_2(x), a_2(x\omega^{-1})$$
$$a_3(x)$$
$$f_0(x),\ f_0(x\omega^{-1})$$
$$h_0(x),\ ...,h_{d-1}(x)$$

验证者现在要验证这些值是否满足 h(X)的形式：

$$\frac{y^0 \cdot gate_0(x)+\cdots+y^i \cdot gate_i(x)+\ldots y^j wire(x)+...+y^k table(x)}{t(x)} = h_0(x)+\cdots+x^{n(d-1)}h_{d-1}(x)$$

如果这些值确实满足的门约束，则验证者接下来就需要验证这些值本身与最初的电路承诺以及承诺 H 是否一致。为高效实现此一目的，使用多点打开证明。

## 8.6 多点打开证明（对应 7.7 节）

$A,B,C,D$ 分别是多项式 $a(X),b(X),c(X),d(X)$ 的多项式承诺值。

如果要打开多项式 a, b 在点 x 的取值，
与此同时，要打开 c,d 在两个点 x, ωx 的取值。
（此处的 ω 是乘法子群的本原单位根）。
可以按打开点的集合，将**多项式承诺值线性组合**：

| $\{x\}$ | $\{x,\omega x\}$ |
|---|---|
| $A$ | $C$ |
| $B$ | $D$ |

对于每一个组，将这些**多项式合并成一个多项式线性组合**，然后构造一个多项式 Q，并在**该点打开所有相对偏移**。

## 8.7 多项式的线性组合

多点打开的优化算法的输入为：

- 验证者采样一个随机点 x，计算多项式 a(X),b(X),c(X),d(X)在该点的值；
- 证明者计算多项式的值：a(x),b(x),c(x),d(x),c(ωx),d(ωx)。

这些数据是消退证明的输出。

**多点打开优化算法的步骤如下：**
选择一个随机点 x1，让**电路多项式** a(X),b(X),c(X),d(X)线性无关；
对于每一个点集，**将对应的多项式集合**线性组合组合成一个多项式，得到新的多项式集合 $q_{polys}$。

$$q_1(X) = a(X) + x_1 \cdot b(X)$$
$$q_2(X) = c(X) + x_1 \cdot d(X)$$

同理对于每一个点集，也计算出新的多项式点值集合 q_eval_sets。
根据 q_eval_sets，构造插值多项式集合 r_polys，即对每个多项式的集合，构造一个**低次多项式** $r_1(X), r_2(X)$，使得该低次多项式在该集合内所有点处和 $q_1(X), q_2(X)$ 取值相同

$$r_1(x) = a(x) + x_1 \cdot b(x)$$
$$r_2(x) = c(x) + x_1 \cdot d(x)$$
$$r_2(\omega x) = c(\omega x) + x_1 \cdot d(\omega x)$$

对每个多项式的集合，构造**检查多项式** $f_1(X), f_2(X)$

对应 Dan Boneh 承诺中的低次多项式。
承诺的商多项式，商多项式存在，则原多项式的求值正确。

$$f_1(X) = \frac{q_1(X) - r_1(X)}{X - x}$$
$$f_2(X) = \frac{q_2(X) - r_2(X)}{(X - x)(X - \omega x)}$$

- 如果 $q_1(x)=r_1(x)$，则 $f_1(X)$是一个多项式；
- 如果 $q_2(x)=r_2(x)$且 $q_2(\omega x)=r_2(\omega x)$，则 $f_2(X)$是一个多项式。

使用随机点 x2，将**检查多项式**进行**线性组合**

$$f(X) = f_1(X) + x_2 \cdot f_2(X)$$

选择随机数 x3，并计算 f(X)在该点的取值：$f(x_3)$
选择随机数 x4，用来让 f(X)和 $q_{polys}$ 线性无关，**得到最终需要证明的多项式：**

$$f_{final}(X) = f(X) + x_4 \cdot q_1(X) + x_4^2 \cdot q_2(X)$$

# 9.Halo2 证明系统 2

## 9.1 初始化

系统参数 $Setup = \left(\mathbb{G}, \mathbb{F}, \vec{G} \in \mathbb{G}^n, U, W \in \mathbb{G}\right)$；

需要证明的**运算关系**

$$\mathcal{R} = \left\{ \begin{pmatrix} g(X, c_0, c_1, ..., c_{n_a-1}) \\ a_0(X), a_1(X, c_0), ..., a_{n_a-1}(X, c_0, c_1, ..., c_{n_a-1}) \\ g(\omega^i, c_0, c_1, ..., c_{n_a-1}) = 0, \forall i \in [0, 2^k) \end{pmatrix} : \right\}$$

- $c_0, c_1, ..., c_{n_a-1}$为一些常量随机数，由验证方发送，或由证明方基于当前数据计算哈希获得。
- $a_0(X), a_1(X, c_0), ..., a_{n_a-1}(X, c_0, c_1, ..., c_{n_a-1})$表示证明方的**保密数据多项式**。
- $g(\omega^i, c_0, c_1, ..., c_{n_a-1}) = 0, \forall i \in [0, 2^k)$表示**运算关系**。
- $a_i(\omega^j X, ...)$为$g(X, ...)$中的**成员**。

运算关系 $g(\omega^i, c_0, c_1, ..., c_{n_a-1}) = 0, \forall i \in [0, 2^k)$ 表示门约束、线约束、查找表和定制门的运算汇总多项式。
每个定制门都需要一个打开点，最终需要打开多个点，不是打开 2 个点。

- 对于每个需要承诺的**数据多项式**$a_i(X)$，令 $p_i$ 为该多项式**打开点集合**，例如在 $X, \omega X$ 上打开，则 $p_i$ 集合为$\{0,1\}$。
- 令 q 表示**不同的打开点集合**，例如$q = \{\{0\}, \{0,1\}, \{0,1,-1\}\}$，并且设定$q_0 = \{0\}$。
- $\sigma(i)$ 表示第 i 个多项式$a_i(X)$的打开点集**属于 q 集合中的第**$\sigma(i)$**个集合。**

## 9.2 协议原理

### Halo2 按功能划分为 4 部分

[1] **数据多项式承诺：**按打开点集的不同，将多项式划分为不同集合，然后承诺；

[2] **消退证明：**将 vanishing argument 中 h(X)多项式，根据最大阶 n 进行划分，并逐个承诺；然后，计算多项式的值；

[3] **多点打开证明：**使用多点打开方案，构造最终多项式 $f_{final}(X)$；

[4] **内积证明：**承诺/验证最终的多项式 $f_{final}(X)$。

证明方与验证方**一共进行**$n_a$**轮交互**；交互从 0 轮开始，以下描述第 j 轮的交互。

此时，验证方已经发送挑战值$c_0, ..., c_{j-1}$给证明方。

### 9.2.1 数据多项式承诺（对应 8.3/7.1 节）

**按打开点集的不同，将多项式划分为不同的集合并进行承诺**

a.证明者构造**数据多项式** $a_j'(X)=a_j(X,c_0,...,c_{j-1})$；

b.证明者**数据多项式承诺** $A_j:=\langle\vec{a}',\vec{G}\rangle+[\cdot]W$；

c.验证者发送挑战 $c_j$；

d.证明者构造**运算关系** $g'(X)=g(X,c_0,c_1,...,c_{n_a-1})$ 对应运算关系 $\mathcal{R}$。【该关系代表门约束、线约束、查找表和定制门的汇总多项式】

e.证明者构造随机多项式 r(X)并发送**随机多项式承诺** $R=\langle\vec{r},\vec{G}\rangle+[\cdot]W$；

f.证明者计算**消退证明多项式** $h(X):=\frac{g'(X)}{t(X)}$，阶为 $n_g\cdot(n-1)-n$。其中，$t(X)$ 是根据打开点计算出来的公开的**目标多项式**。

### 9.2.2 消退证明（对应 8.4/7.5 节）

将商多项式 $h(X)$，按协议设定的最大阶 n 进行划分和多项式承诺

a) 将 $h(X)$ 划分为 $h(X)=\sum_{i=0}^{n_g-2}X^{ni}h_i(X)$，每个 $h_i(X)$ 的**阶**不超过 n-1；

b) 对于每个划分的 $h_i(X)$，证明者分别进行**商多项式承诺** $H_i:=\langle\vec{h}_i,\vec{G}\rangle+[\cdot]W$；

c) 验证者发送随机挑战 $x$，并计算**承诺**的线性组合 $H':=\sum_{i=0}^{n_g-2}[x^{ni}]H_i$；

d) 证明者按照验证者发送的 $x$，将 $h_i(X)$ 进行线性组合 $h'(X):=\sum_{i=0}^{n_g-2}x^{ni}\cdot h_i(X)$；

### 9.2.3 多项式求值（对应 8.5/7.6 节）

证明者**发送**这些多项式**在 x 点的多项式求值**：

［1］随机多项式 $r(x)$ 的值为 $r$；

[2] 对于 $i\in[0,n_a)$，数据多项式的值 $a_i'(\omega^{(p_i)_j}x)=(\vec{a}_i)_j$， $j\in[0,n_e)$。

### 9.2.4 多点打开证明（对应 8.6/7.7 节）

**使用多点打开方案，构造最终的随打开点多项式** $f_{final}(X)$。

**反之，如果随机打开点正确，则 $f_{final}(X)$ 正确，则 $h(X)$ 正确，则 $g'(X)$ 正确，则数据满足运算关系 $\mathcal{R}$ 。**

a.证明者和验证者构造 $s_i(X)$，使得 $s_i(X)$ 和 $a_i(X)$ **在 q 集合中第 i 个多项式打开点集合内所有打开点。**例如：在 $X, \omega X$ 等处，**两者的值**相等 $s_i'(\omega^{(p_i)_j} x) = (\vec{a}_i)_j$ 。

b.验证者发送随机挑战 $x_1, x_2$，为 q 中的每一个集合，计算属于 q 集合的**数据多项式承诺的线性组合**

$$Q_{\sigma(i)} := [x_1]Q_{\sigma(i)} + A_i, i = 0, ..., n_a - 1$$

最后，$Q_0$ 需要包含 H 和 R 的承诺：$Q_0 := [x_1^2]Q_i + [x_1]H' + R$

e.同样，证明者初始化 $q_0(X), ..., q_{n_q-1}(X) = 0$，构造对应上述承诺的 q 集合的**数据多项式的线性组合**

$$i = 0, ..., n_a - 1$$
$$q_{\sigma(i)}(X) := x_1 \cdot q_{\sigma(i)}(X) + a'(X),$$

最后，构造多项式 $q_0(X) := x_1^2 \cdot q_0(X) + x_1 \cdot h'(X) + r(X)$ 。对应上一步的承诺。

f.证明者和验证者同时初始化低次多项式 $r_0(X), ..., r_{n_q-1}(X) = 0$，构造 $s_i(X)$ **多项式的线性组合**

$$i = 0, ..., n_a - 1$$
$$r_{\sigma(i)}(X) := x_1 \cdot r_{\sigma(i)}(X) + s_i(X),$$

最后，构造多项式 $r_0(X) := x_1^2 \cdot r_0(X) + x_1 \cdot h + r$ 。

| 低次多项式 $r_1(X), r_2(X)$ 对应 Dan Boneh 承诺中的低次多项式。 |
|---|

验证方会使用证明方提供的 r 和 $\vec{a}$ 计算**消退证明多项式** $h := \dfrac{g'(x)}{t(x)}$ 。

g.证明方计算**商多项式承诺** $Q' := \langle \vec{q}', \vec{G} \rangle + [\cdot]W$ ，其中 $q'(X)$ 表达如下

$$\textbf{商多项式} q'(X) = \sum_{i=0}^{n_q-1} x_2^i \left[ \frac{q_i(X) - r_i(X)}{\prod_{j=0}^{n_e-1} \left( X - \omega^{(q_i)_j} x \right)} \right]$$

h.验证者发送随机挑战 $x_3$；

i.证明者发送所有的 $q_i(X)$ 多项式在 $x_3$ 点处的**商多项式求值** $u_i := q_i(x_3), i \in [0, n_q)$，然后验证方发送挑战 $x_4$；

j.证明者和验证者通过承诺的**线性组合**，构造**最终多项式承诺的线性组合** $P = Q' + x_4 \sum_{i=0}^{n_q-1} [x_4^i] Q_i$；并计算出该多项式在 $x_3$ 点处的**最终多项式值的线性组合**

$$v = p(x_3) = \sum_{i=0}^{n_q-1} \left( x_2^i \cdot \frac{\vec{u}_i - r_i(x_3)}{\prod_{j=0}^{n_e-1} (x_3 - \omega^{(q_i)_j} x)} \right) + x_4 \sum_{i=0}^{n_q-1} (x_4 \cdot \vec{u}_i)$$

k.证明者构造出**最终线性组合多项式** $p(X) = q'(X) + [x_4] \sum_{i=0}^{n_q-1} [x_4^i] q_i(X)$

**最终线性组合多项式** $p(X)$ = 随机打开点多项式 + 数据多项式

### 9.2.5 内积证明

**使用内积证明方案，承诺/验证最终的多项式** $p(X)$

a.证明者选择一个**随机多项式 s(X)添加零知识**，阶为 n-1,使得 $s(x_3)=0$，然后发送其承诺

$$S := \langle \vec{s}, \vec{G} \rangle + [\cdot] W$$

b.验证者发送随机挑战 $\xi, z$；

c.双方同时计算线性组合向量 $\vec{P}' := \vec{P} - [v]\vec{G}_0 + [\xi]\vec{S}$；

d.证明者准备多项式线性组合 $p'(X) = p(X) - v + \xi s(X)$；

e.以下采用内积方式，证明**多项式承诺与多项式的值**

$$P = Com(p(X)) \wedge v = p(x_3)$$

**（使用第 6.5 节方案 4：折半打开，并行承诺）**

令秘密向量 $\vec{p}'$ 为多项式 p'(X)的系数，且 $\vec{b} = (x_3^0, x_3^1, ..., x_3^{n-1})$

- 交互 k 轮，每一轮中，证明者计算**折半承诺值**

$$L_j = \langle \vec{p}_{hi}', \vec{G}_{lo}' \rangle + \left[ z \langle \vec{p}_{hi}', \vec{b}_{lo}' \rangle \right] \cdot U + [\cdot] W$$

$$R_j = \langle \vec{p}_{lo}', \vec{G}_{hi}' \rangle + \left[ z \langle \vec{p}_{lo}', \vec{b}_{hi}' \rangle \right] \cdot U + [\cdot] W$$

- 验证者回应**挑战** $u_j$；
- 双方计算**折半向量** $\vec{G}' = \vec{G}_{lo}' + u_j \cdot \vec{G}_{hi}', \vec{b}' = \vec{b}_{lo}' + u_j \cdot \vec{b}_{hi}'$；
- 证明者计算**折半秘密** $p' = p_{lo}' + u_j^{-1} \cdot p_{hi}'$；

最后，证明者发送**最终折半为常量** $c = p'_0$ 和盲因子 r。

**验证方：** 验证 $P' + \sum_{j=0}^{k-1}[u_j^{-1}]L_j + \sum_{j=0}^{k-1}[u_j]R_j = [c]\vec{G}_0' + [c\vec{b}_0 z]G_0' + [r]W$

lynndell 博士 新火科技 密码学专家 lynndell2010@gmail.com