密码学zk系列

第 8 课：zkSnark-Groth16 证明系统

lynndell博士

新火科技 密码学专家 lynndell2010@gmail.com

目录

1 零知识证明基础

1.1 预备知识

交互式零知识证明：证明方$\mathcal{P}$发送承诺给验证方$\mathcal{V}$；验证方$\mathcal{V}$发送挑战给证明方$\mathcal{P}$；证明方$\mathcal{P}$发送响应。与TCP/IP协议的**交互**式一样。

非交互式证明：证明方$\mathcal{P}$生成证明，发送给验证方$\mathcal{V}$；验证方$\mathcal{V}$验证一致性即可。其中，**挑战是由证明方基于当前数据计算哈希获得，操控挑战的难度与POW相同，因此非交互式证明是安全。**该过程没有其他额外的数据交互。例如：用户将ECDSA签名发送出去，共识节点进行一致性验证过程一样，整个过程只有一次数据发送。

多项式时间算法：能够快速计算出结果的算法。例如：

1. **倍点运算：**已知私钥sk和椭圆曲线生成元G，能够快速计算公钥PK

$$PK := sk \cdot G$$

2. **哈希运算：**已知原象x和哈希函数$SHA256$，能够快速计算函数值y

$$y := SHA256(x)$$

非多项式时间算法：包括**指数时间**、**亚指数时间**算法。以下计算需要指数时间：

1. **离散对数：**已知公钥PK和椭圆曲线生成元G，不能够在多项式时间内计算出私钥sk，而需要指数时间才能计算私钥sk，使得以下等式成立

$$PK = sk \cdot G$$

2. **哈希求逆：**已知函数值y和哈希函数$SHA256$，不能够在多项式时间内计算出原象x，而需要指数时间才能计算原象x，使得以下等式成立

$$y = SHA256(x)$$

目前，在理论计算领域中，还没有区分P问题是否等于NP问题。
为方便理解zkSnark，从应用角度出发，认为P问题**不等于**NP问题。
以下从应用角度说明，而非严格学术定义。
P问题：在多项式时间内可解的问题。以下问题多项式时间内可求解：

1. **求公钥？**已知私钥sk和椭圆曲线生成元G，求满足以下离散对数关系的公钥PK

$$PK = sk \cdot G$$

2. **求哈希值？**已知原象x和哈希函数$SHA256$，求满足以下计算关系的函数值y

$$y = SHA256(x)$$

NP问题：

条件1：多项式时间内不可计算的问题，需要指数时间或亚指数时间。

条件2：但是，一旦已知解，则能够在多项式时间内验证解是否正确。

以下是3个典型问题是NP问题：

1. **求私钥？**已知公钥PK和椭圆曲线生成元G，求私钥sk。要求私钥与公钥满足以下**离散对数关系**

$$PK = sk \cdot G$$

（1）不能在多项式时间内求解出私钥sk，需要指数时间，**满足条件1**。（2）但是，一旦给出私钥sk，则能够在多项式时间内快速验证该私钥sk与公钥PK是否满足离散对数关系，**满足条件2**。

2. **求哈希原象？**已知函数值y和哈希函数$SHA256$，求原象x。要求原象x和函数值y满足以下$SHA256$计算关系

$$y = SHA256(x)$$

（1）不能在多项式时间内求解出原象x，需要指数时间，**满足条件1**。（2）但是，一旦给出原象x，则能够在多项式时间内验证该原象x与函数值y是否满足$SHA256$计算关系，**满足条件2**。

NP问题的本质是单向性，不可快速逆向求解，但是能够快速正向验证。

第3个重要的NP问题：多项式整除问题

已知3组的阶小于n多项式$u_0(x), u_1(x), \ldots, u_m(x)$；$v_0(x), v_1(x), \ldots, v_m(x)$；$w_0(x), w_1(x), \ldots, w_m(x)$；一个n阶目标多项式z(x)，z(x)=0的解为$r_1,\ldots,r_n$。且对于任意解$r_j$，三组多项式的函数值的**线性组合等于零**，表达如下：

$$0 = \left(\sum_{i=0}^{m} s_i \cdot u_i(r_j)\right) \cdot \left(\sum_{i=0}^{m} s_i \cdot v_i(r_j)\right) - \left(\sum_{i=0}^{m} s_i \cdot w_i(r_j)\right)$$

求向量$\vec{s} = (1, s_1, \ldots, s_m)$？满足以下整除关系

$$z(x) \left| \left[\left(u_0(x) + \sum_{i=1}^{m} s_i \cdot u_i(x)\right) \cdot \left(v_0(x) + \sum_{i=1}^{m} s_i \cdot v_i(x)\right) - \left(w_0(x) + \sum_{i=1}^{m} s_i \cdot w_i(x)\right) \right] \right.$$

原理分析：向量$\vec{s}$对三组多项式$u_0(x), u_1(x), \ldots, u_m(x); v_0(x), v_1(x), \ldots, v_m(x); w_0(x), w_1(x), \ldots, w_m(x)$进行**线性组合**，分别得到一个**线性组合多项式**

$$u_0(x) + \sum_{i=1}^{m} s_i \cdot u_i(x), v_0(x) + \sum_{i=1}^{m} s_i \cdot v_i(x), w_0(x) + \sum_{i=1}^{m} s_i \cdot w_i(x)$$

向量$\vec{s}$的维度为m。如果每个元素s_i的取值空间为$a > 2$，则将线性组合多项式$(u_0(x) + \sum_{i=1}^{m} s_i \cdot u_i(x)) \cdot (v_0(x) + \sum_{i=1}^{m} s_i \cdot v_i(x)) - (w_0(x) + \sum_{i=1}^{m} s_i \cdot w_i(x))$称为**二次算法多项式**，简称**QAP多项式**。因此，QAP多项式的构造为**指数空间a^m**。

如果每个元素s_i的取值空间为0或1，则将线性组合多项式$(u_0(x) + \sum_{i=1}^{m} s_i \cdot u_i(x)) \cdot (v_0(x) + \sum_{i=1}^{m} s_i \cdot v_i(x)) - (w_0(x) + \sum_{i=1}^{m} s_i \cdot w_i(x))$称为**二次扩张多项式，或二次布尔多项式**，简称**QSP多项式**。因此，QSP多项式的构造为**指数空间2^m**。

目标多项式$z(x)$等于零有n个解$r_1,\ldots,r_n$，而QAP/QSP多项式等于零的解数量为$2n-2$。所以，除$z(x)=0$的n个解以外，QAP/QSP多项式还有$n-2$个其他解。因此，可以计算出**商多项式**$h(x)$，且商多项式$h(x)$就是QAP/QSP多项式等于零的剩余的$n-2$个解构成的多项式。

如果不知道向量$\vec{s}$，则只能**随机选择**一个向量$\vec{s}$，计算QAP/QSP多项式；然后检测$z(x)$与之是否满足整除关系。因此，需要**指数时间**才能够暴力搜索出向量$\vec{s}$。

满足NP问题的条件1：不可快速逆向求解。

但是，一旦给定向量$\vec{s}$，则能够快速基于向量$\vec{s}$构造出QAP/QSP多项式，并快速验证目标多项式$z(x)$与之是否满足**整除关系**。

满足NP问题的条件2：能够快速校验正确性。

因此，目标多项式z(x)与QAP/QSP多项式的整除关系，满足单项性，不可快速逆向求解，但是能够快速验证向量$\vec{s}$的正确性，构成NP问题。
该构造至关重要，后续的举例需使用**多项式整除关系构造NP问题。**

zkSnark核心思想： **1. 秘密数据满足加法/乘法等任意运算关系；** **2. 将秘密数据放到离散对数点上，对外暴漏离散对数点；** **等价于已知多个私钥，对外暴露多个公钥，而不泄露私钥。** **3. 离散对数点是群元素，能够进行二元运算，可重构加法/乘法等任意运算关系；**
用户隐私保护、数据保密性、区块链扩容（数据压缩和计算压缩）等应用需求 **1. 证明方：基于向量$\vec{s}$和电路，生成QAP/QSP多项式、目标多项式$z(x)$和商多项式$h(x)$，构造NP问题；然后基于这3个多项式的系数，计算离散对数点（专业术语称为：多项式承诺），对外暴漏离散对数点；** **2. 验证方：对离散对数点进行双线性映射，重构整除关系，则能够快速验证向量$\vec{s}$的正确性，却不知道向量$\vec{s}$。**

对于第1个NP问题【**求私钥？**】，可使用经典的Σ零知识证明，证明方$\mathcal{P}$知道私钥sk而不泄露私钥sk。

第2个NP问题【**求哈希原象？**】无法使用Σ协议进行证明，但是可以将第2个NP问题**等价转换**为第3个NP问题【**多项式整除问题**】，然后将多项式系数放到椭圆曲线离散对数点上（即多项式承诺），形成离散对数困难，实现零知识；验证方能够重构整除关系。验证方验证了向量$\vec{s}$的正确性，但是不知道向量$\vec{s}$。

注意：如果私钥sk的位宽、哈希函数原象x的位宽、向量$\vec{s}$的维度比较小，则对应的指数计算复杂度较低，则容易被暴力破解。如果位宽足够大，则能够抵抗暴力搜索。**后续，为方便展示计算过程，**在举例使用方程对应的向量$\vec{s}$的维度比较小，容易暴力破解。但是，如果举例中方程的阶很高，对应的向量$\vec{s}$维度很大，构造的NP问题能够抵抗暴力攻击。

1.2 零知识证明

对于第1个NP问题：证明方$\mathcal{P}$证明知道私钥sk，但不泄露sk。

1. 证明方$\mathcal{P}$用私钥sk，计算ECDSA签名$\sigma = (r, s)$。
2. 验证方$\mathcal{V}$基于(r, s)计算椭圆曲线点，结合公钥PK重构**线性关系**，则ECDSA验证成功。验证方$\mathcal{V}$认可：证明方$\mathcal{P}$有私钥sk，而不泄露该私钥sk。

因此，使用**数字签名**实现零知识证明的功能，证明了拥有私钥sk。事实上，零知识证明由数字签名**发展**而来。数字签名中的私钥满足对公钥的**离散对数关系**，但是**零知识证明**中的秘密ω满足**任意计算关系**$y = F(\omega)$。因此，零知识证明将数字签名的**离散对数关系扩展为任意计算关系**，使得零知识证明有了广泛的应用空间。

定义1：零知识证明：令R为一个高效可计算的**二元运算关系**。对于二元组$(x, \omega) \in R$，x为声明，ω为秘密。对于二元运算关系R，证明系统包括系统参数生成$SysGen$，证明P和验证V。

知识证明协议$ZK\{\boldsymbol{\omega}|(\boldsymbol{x},\boldsymbol{\omega})\in\boldsymbol{R}\}$包括5个多项式时间算法，系统参数生成$SysGen$，承诺$Commitment$，挑战$Challenge$，响应$Response$和验证$Verify$。
对于一个固定的安全参数λ，这5个算法如下运行：

- **系统参数：**生成系数所需公共参数CRS

$$CRS \leftarrow SysGen(1^{\lambda})$$

- **承诺：**证明方$\mathcal{P}$选择一个随机数r，计算并发送承诺

$$C \leftarrow Com(x,\omega;r)$$

- **挑战：**验证方$\mathcal{V}$选择从某个域中一个随机数e作为挑战并发送给证明方$\mathcal{P}$。
- **响应：**证明方$\mathcal{P}$对验证方$\mathcal{V}$输出响应

$$z \leftarrow Response(x,\omega;e,r)$$

- **验证：**验证方$\mathcal{V}$基于承诺、挑战和响应计算并输出判断结果

$$Valid/Invalid \leftarrow Verify(x,C,e,z)$$

$ZK\{\omega|(x,\omega)\in R\}$协议具有**完备性**，如果满足以下性质：

$$\Pr\begin{bmatrix} r\leftarrow\mathbb{Z}_p, C\leftarrow Com(x,\omega,r) \\ e\leftarrow\mathbb{Z}_p \\ z\leftarrow Response(x,\omega;e,r) \\ Valid\leftarrow Verify(x,C,z,e) \end{bmatrix}=1$$

$ZK\{\omega|(x,\omega)\in R\}$协议具有**知识提取鲁棒性**，如果满足以下性质：

$$\Pr\begin{bmatrix} Valid\leftarrow Verify(x,C,z,e) \\ Valid\leftarrow Verify(x,C,z',e') \\ \omega\leftarrow Extrator(x,C,z,e,z',e') \\ (x,\omega)\in R \end{bmatrix}=1$$

对秘密证明2次，则验证方可以提取秘密。说明了证明方确实知道秘密。承诺的随机数使用2次，挑战不同，响应不同，则可以列出2个响应方程，解方程组。

$ZK\{\omega|(x,\omega)\in R\}$协议具有**诚实验证方零知识**，如果满足以下性质：

$$\Pr\begin{bmatrix} P(x,\omega)\rightleftarrows V(x,C,z,e) \\ S(x)\rightleftarrows V(x,C,z',e') \\ \{Verify(x,C,z,e)\}\approx\{Verify(x,C,z',e')\} \end{bmatrix}=1$$

定义：如果协议$ZK\{\omega|(x,\omega)\in R\}$具有**完备性、知识提取的鲁棒性**和**诚实验证方零知识**，则该协议$ZK\{\omega|(x,\omega)\in R\}$是一个Σ协议。

1.3 Σ协议

Σ协议包括：（1）生成系统参数CRS，（2）承诺，（3）挑战，（4）响应，（5）验证。其中，系统参数CRS由多方安全计算生成。

系统参数CRS：椭圆曲线群$\mathbb{G}$的阶为q，生成元为G；
公开输入为$Q\in\mathbb{G}$。证明方**$\mathcal{P}$证明**知道秘密ω且ω满足离散对数关系$Q=\omega\cdot G$。

交互式零知识证明

承诺：证明方$\mathcal{P}$选择随机数r，计算并发送：$C:=r\cdot G$；

挑战：验证方$\mathcal{V}$选择随机数e，发送：e；

响应：证明方$\mathcal{P}$计算并发送：$z:=r+e\cdot\omega$；

验证：验证方$\mathcal{V}$验证

$$z\cdot G=C+e\cdot Q$$

如果等式成立，则接受，否则拒绝。

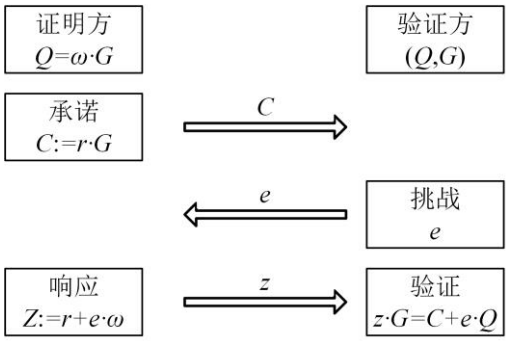


图 1: Σ协议

协议思想：

1. 如图1所示，证明方$\mathcal{P}$将ω与随机数e进行线性组合$z:=r+e\cdot\omega$，生成响应z；
2. 验证方$\mathcal{V}$基于响应z和挑战e构造椭圆曲线离散对数点$z\cdot G, e\cdot Q$，并与承诺C，在椭圆曲线离散对数上**重构线性关系**，实现正确性验证，但是不知道秘密ω。

非交互式零知识证明

承诺：证明方$\mathcal{P}$选择随机数r，计算：$C:=r\cdot G$；

挑战：证明方$\mathcal{P}$计算随机数：$e:=Hash(\mathrm{Q},C)$；（此处有变化）

响应：证明方$\mathcal{P}$计算$z:=r+e\cdot\omega$，证明方发送：C,z

验证：验证方$\mathcal{V}$计算$e:=Hash(\mathrm{Q},C)$，然后验证

$$z\cdot G=C+e\cdot Q$$

如果等式成立，则接受，否则拒绝。

数字签名定义：

Alice用私钥sk对消息m签名$\sigma\leftarrow Sign(sk,m)$

验证方*Verifier*用Alice的公钥PK对消息签名对(m,σ)进行一致性验证

$$Valild/Invalid\leftarrow Verify(pk,m,\sigma)$$

非交互式零知识证明扩展为数字签名

对于离散对数关系$Q=\omega\cdot G$，把秘密ω看作私钥sk，公开输入Q看作公钥PK。

承诺：证明方$\mathcal{P}$选择随机数r，计算：$C:=r\cdot G$；

挑战：证明方$\mathcal{P}$计算随机数：$e:=Hash(\mathrm{Q},C,m)$；（此处有变化）

响应：证明方$\mathcal{P}$计算$z:=r+e\cdot\omega$，证明方发送：C,z,m；

验证：验证方$\mathcal{V}$计算$e:=Hash(\mathrm{Q},C,m)$，然后验证

$$z \cdot G = C + e \cdot Q$$

如果等式成立，则接受，否则拒绝。

核心思想：
第3步的(C, z, m)等价于Alice的对消息m的签名σ；
第4步等价于验证方$\mathcal{V}$使用Alice的公钥Q对消息与签名(m, C, e, z)的一致性验证。
因此，该非交互式零知识证明满足数字签名的定义。

保密随机数r的作用等价于私钥sk。
1.ECDSA/EdDSA签名算法中随机数r泄露，则私钥sk泄露；$z = r + e \cdot \omega$
2.Tornado cash就是zk证明知道Merkle树中的保密随机数r，就可以提币。

数字签名证明：（1）Alice拥有私钥$sk = \omega$，且（2）ω与消息m具有绑定关系$e = Hash(\mathrm{Q}, C, m)$，且（3）私钥与公钥满足离散对数关系。

Σ协议：证明方$\mathcal{P}$证明其知道ω满足离散对数关系$Q = \omega \cdot G$。由于$Q = \omega \cdot G$即是**天然的NP问题**，又是天然的**离散对数关系。**所以，可以直接基于这个NP问题在椭圆曲线离散对数点上构造**线性关系**，形成离散对数困难。
验证方$\mathcal{V}$验证了NP问题的解的正确性，但是证明方$\mathcal{P}$没泄露秘密ω。

zk-SNARK协议：证明秘密ω满足**任意多项式时间计算关系**$y = F(\omega)$。
任意运算关系可以包括NP问题和P问题的求解。
1. **NP问题：**已知函数值y和哈希函数SHA256，求原象x；已知Merkle根，求Merkle树的叶子和路径；
2. **P问题：**已知两个布尔值a, c，求布尔值b，使得等式$a \oplus b = c$成立；已知方程$x^4 + x^3 + x^2 + x = 120$，求解$x$；

这4个算法中
保密数据：哈希函数的原象x、Merkle树的叶子和路径、布尔值b、方程解$x = 3$以及计算过程的**中间状态值**，记为秘密ω。
公开数据：哈希函数值y、Merkle根、布尔运算结果c和方程的值120，记为statement。

为不泄露秘密ω，需使用R1CS约束（电路约束）**等价描述**算法的运算规则。
将ω满足公开的计算关系$y = F(\omega)$**等价转化**为ω满足公开的R1CS约束，**再等价转化**为向量$\vec{s}$与**多维向量**的内积，**再等价转化**为向量$\vec{s}$与**矩阵**的内积，**再等价转化**为向量$\vec{s}$对**三组多项式**的线性组合运算，**再等价转化**为目标多项式$z(x)$整除QAP/QSP多项式**（构成NP问题）**，**再等价转化**为基于这三个多项式的**系数**计算椭圆曲线离散对数点（**构成零知识**），对外暴漏离散对数点；
验证方$\mathcal{V}$基于离散对数点**重构整除关系**，验证向量$\vec{s}$的正确性，而不知道向量$\vec{s}$。

zk-SNARK协议的**7个等价转化关系**，如图2所示

图 2: zk-SNARK的7个**等价转化关系**

接下来对zk-SNARK协议中的7个等价转换关系进行**举例说明**。

2 多项式时间算法等价转换为**阶为1的等式**

2.1 哈希函数等价于阶为1的等式

哈希函数$SHA256$输入512bits的数据，通过**与、或、非、异或、循环移位**等基础运算的耦合，进行线性变换与非线性变换(扩张与有损压缩)，输出256bits的随机数。以下是$SHA256$的运算原理，循环64次后输出函数值。其中，x,y,z位宽均为32bits，S^i循环右移$i$bits，R^i循环右移$i$bits。

$$\begin{aligned} maj(x,y,z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\ ch(x,y,z) &= (x \wedge y) \oplus (\neg x \wedge z) \\ \Sigma_0(x) &= S^2(x) \oplus S^{13}(x) \oplus S^{22}(x) \\ \Sigma_1(x) &= S^6(x) \oplus S^{11}(x) \oplus S^{25}(x) \\ \sigma_0(x) &= S^7(x) \oplus S^{18}(x) \oplus R^3(x) \\ \sigma_1(x) &= S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x) \end{aligned} \tag{1}$$

以下对**布尔值**和**异或运算**进行等价转化：

举例1：布尔值a和b**等价转换**为阶为1的等式

$$\begin{aligned} (1-a) \times a &= 0 \\ (1-b) \times b &= 0 \end{aligned}$$

举例2：位异或运算$a \oplus b = c$**等价转换**为4个阶为1的等式

$$\begin{aligned} (1-a) \times a &= 0 \\ (1-b) \times b &= 0 \\ (1-c) \times c &= 0 \\ (2a) \times (b) &= (a+b-c) \end{aligned} \tag{3}$$

推论1：布尔值a和b等价为2个阶为1的等式，电路约束；

推论2：位异或运算$a \oplus b = c$等价为4个阶为1的等式，电路约束；

因此，基于推论1和2，可以得出以下推论3和4：

推论3：哈希函数SHA256等价为大约2.5万个阶为1的等式，电路约束；

推论4：Merkle树等价为约2.5万*k个阶为1的等式，电路约束；

上述4个阶为1的等式，即公式(3)，就是**电路约束**。这4个阶为1的等式是乘法等式，所以是**乘法约束**。前3个乘法约束限定输入和输出为布尔值，第4个乘法约束实现异或运算的等价功能。因此，**算法的运算规则等价为电路约束。**

a和b是输入值，c是输出值。证明方把a,b,c发送给验证方，则验证方根据a,b,c和阶为1的等式（电路约束）成功验证，则验证方认可证明方是对布尔值a和b诚实进

行异或运算。类似地，证明方把哈希函数的原象x和函数值y发送给验证方，验证方根据x, y和阶为1的等式成功验证，则能够确定证明方诚实计算哈希函数，且原象x和函数值y是正确的。但是，上述证明与验证过程存在2个缺点：

1. **秘密泄露：**布尔值a, b或哈希原象x泄露，缺乏零知识，后续通过离散对数解决。
2. **验证方的计算复杂度没降低：**阶为1的等式的计算复杂度**等于**原算法的计算复杂度。后续通过构造多项式整除关系解决。

最终实现：计算压缩、数据压缩、零知识。

2.2 数字签名等价于阶为1的等式

（一）预备知识

1. **椭圆曲线离散对数困难问题：**已知椭圆曲线生成元$G = (x_0, y_0)$和公钥$Q = (x_q, y_q)$，计算私钥是困难的

$$Q = sk \cdot G$$

因此，密码学提供计算安全，是**相对安全**，而不是绝对安全。

2. 已知私钥sk和椭圆曲线生成元$G = (x_0, y_0)$，能够在多项式时间内计算公钥

$$Q = (x_q, y_q)$$

（二）数字签名

ECDSA签名：用户的私钥为d，对于消息M，选择随机数$k \in [1, \dots, n-1]$，如下计算：

$$\begin{aligned} (x_1, y_1) &:= k \cdot G \\ r &:= x_1 \bmod n \\ s &:= k^{-1} \cdot (SHA256(M) + dr) \bmod n \end{aligned} \tag{4}$$

则签名为$\sigma = (r, s)$。广播(M, σ, Q)，其中Q是公钥。

签名原理：

1. 随机数k对私钥d进行随机化，防止攻击者计算私钥d。
2. s是随机数k^{-1}、私钥d、消息的摘要值$SHA(M)$和横坐标r的**线性组合**

$$s = k^{-1} \cdot (SHA256(M) + dr) \bmod n$$

攻击者无法根据(r, s)计算出随机数k与私钥d。

ECDSA验证：验证方基于(M, σ, Q)如下计算：

$$\begin{aligned} u_1 &:= SHA256(M) \cdot s^{-1} \bmod n, \\ u_2 &:= r \cdot s^{-1} \bmod n, \\ (x_1, y_1) &:= u_1 \cdot G + u_2 \cdot Q \end{aligned} \tag{5}$$

如果等式$r = x_1 \bmod n$成立，则接受，否则拒绝。

验证原理：

1. 基于(r, s)和消息M计算出u_1, u_2；
2. 基于u_1, u_2、椭圆曲线生成元G和公钥Q倍点运算，计算2个点$u_1 \cdot G, u_2 \cdot Q$；
3. 对这两个椭圆曲线离散对数点$u_1 \cdot G, u_2 \cdot Q$进行点加运算，则结果为(x_1, y_1)；
4. 新椭圆曲线点的横坐标x_1与随机数r应该相等的，则验证成功。

ECDSA核心思想：
用户从**私钥角度**计算椭圆曲线离散对数点(x_1, y_1)，并产生(r, s)。
验证方从**公钥Q和签名**(r, s)角度计算椭圆曲线离散对数点(x_1, y_1)，不泄露私钥d。

ECDSA包括3个基本运算：点加运算$u_1 \cdot G + u_2 \cdot Q$、倍点运算$k \cdot G$、模$n$运算。
将点加运算进行如下表达：

$$(x_3, y_3) := (x_1, y_1) + (x_2, y_2)$$

点加运算：
已知两个椭圆曲线离散对数点$(x_1, y_1), (x_2, y_2)$，求第三个点(x_3, y_3)，计算过程如下：

$$\begin{aligned} x_3 &= \frac{x_1 \cdot y_2 + y_1 \cdot x_2}{1 + d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2} \\ y_3 &= \frac{y_1 \cdot y_2 - a \cdot x_1 \cdot x_2}{1 - d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2} \end{aligned} \tag{6}$$

上面是求值。**求值电路：CPU电路是求值电路，计算方知道计算的结果。**
Garbled电路：计算方计算出密文结果，不知道明文结果；计算方把密文结果发送给接收方，接收方能够解密，获得明文结果。
其中，$a = -1, d = -168696/168700$是常量。
当$x_1 = x_2, y_1 = y_2$，就是倍点运算$k \cdot G$，因此以下仅讨论点加运算。
将上述点加运算公式6等价转换为7个阶为1的等式

$$\begin{aligned} (x_1) \times (y_2) &= (A_1) \\ (x_2) \times (y_1) &= (A_2) \\ (A_1) \times (A_2) &= (A_3) \\ (x_3) \times (1 + d \cdot A_3) &= (A_4) \\ (y_1) \times (y_2) &= (A_5) \\ (x_1) \times (x_2) &= (A_6) \\ (y_3) \times (1 - d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2) &= (y_1 \cdot y_2 - a \cdot x_1 \cdot x_2) \end{aligned} \tag{7}$$

zk验证电路：把输入信息和输出结果都放到电路里面。

推论5：点加运算公式6与电路公式7等价。
推论6：ECDSA能够等价转换为阶为1的等式。或ECDSA与某些阶为1的等式等价。

公式7中阶为1的等式就是电路约束。阶为1的乘法等式就是乘法约束；阶为1的加法等式就是加法约束。这7个约束实现点加运算的等价功能。

推论7：任意多项式时间算法均能够拆为阶为1的等式。或任意多项式时间算法与某些阶为1的等式等价。

2.3 多项式时间算法等价于阶为1的等式

任意多项式时间算法（包括上述SHA256和ECDSA）均有对应的阶为1的等式（R1CS约束）。此处，将多项式时间算法限定为一个方程，将方程拆为阶为1的等式。该举例有2个优点：（1）解该方程是P问题，体现了基于P问题构造NP问题；（2）能够较好

的展现出对R1CS约束【电路约束】的优化。

方程$x^4 + x^3 + x^2 + x = 120$如下拆为阶为1的等式

$$\begin{aligned} s_1 &= x * x \\ s_2 &= s_1 * x \\ s_3 &= s_2 * x \\ s_4 &= s_1 + x \\ s_5 &= s_4 + s_2 \\ 120 &= s_5 + s_3 \end{aligned} \tag{8}$$

关键推论：x=3满足方程的解，等价于x=3满足电路约束。
关键推论：x满足任意多项式时间运算y=f(x)，等价于x满足对应的电路约束。
这6个阶为1的等式限定了方程的运算规则，能够用电路约束表达同样的运算规则。
阶为1的等式=电路约束【电路门】
阶为1的乘法等式=乘法约束【乘法门】
阶为1的加法等式=加法约束【加法门】

关键术语：阶为1的等式（Rank 1 Constraint System, R1CS）。
以下术语是等价描述：阶为1的等式、R1CS约束、电路约束。

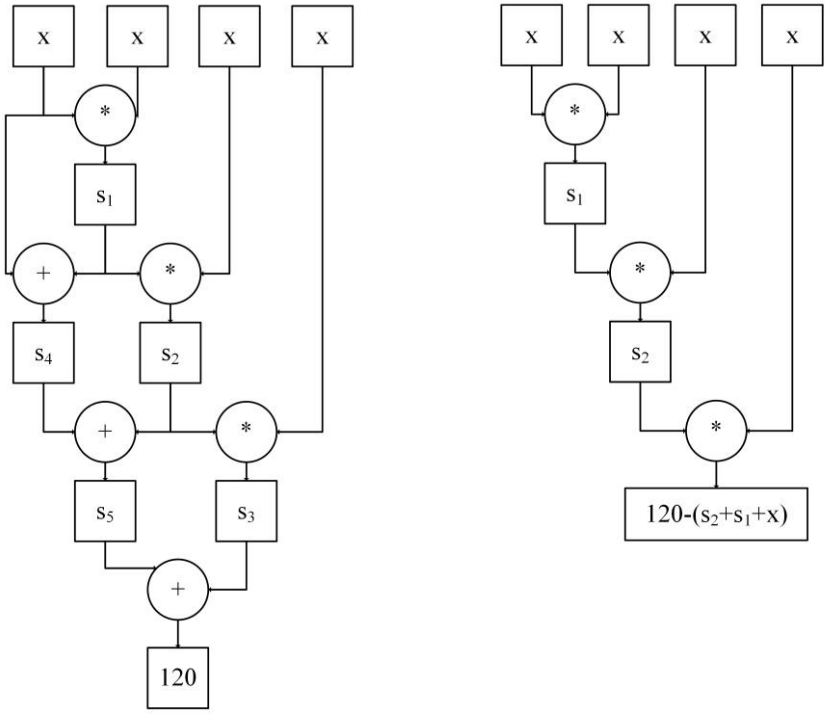
上述6个阶为1的等式包含3个乘法门和3个加法门，可使用3个乘法门和3个加法门实现对应的运算原理，如图1(a)所示。此外，可以将加法约束优化到乘法约束中，则上述6个阶为1的等式8化简为以下3个阶为1的乘法等式9。仅使用3个乘法约束即可实现等价的运算规则，如图1(b)所示。

$$\begin{aligned} s_1 &= x * x \\ s_2 &= s_1 * x \\ 120 - (s_2 + s_1 + x) &= s_2 * x \end{aligned} \tag{9}$$

但是，并非所有的加法门都可以耦合到乘法门中。如果加法门不能够耦合到乘法门中，则将加法门变为乘法门**a*b=c**

$$\begin{aligned} s_4 &= (s_1 + x) * 1 \\ s_5 &= (s_4 + s_2) * 1 \\ 120 &= (s_5 + s_3) * 1 \end{aligned}$$

最终R1CS以**乘法门**为**标准门**。



电路原理1　　电路原理2

图 3: 方程的电路原理

推论8：方程(多项式时间算法)等价于R1CS约束，形式化表达如下

$$x^4 + x^3 + x^2 + x = 120 --(a)$$

$$\Updownarrow$$

$$\left\{\begin{array}{c} s_1 = x * x \\ s_2 = s_1 * x \\ s_3 = s_2 * x \\ s_4 = s_1 + x \\ s_5 = s_4 + s_2 \\ 120 = s_5 + s_3 \end{array}\right\} --(b) \tag{10}$$

$$\Updownarrow$$

$$\left\{\begin{array}{c} s_1 = x * x \\ s_2 = s_1 * x \\ 120 - s_1 - x = s_2 * x \end{array}\right\} --(c)$$

因此，可以得出以下推论：

推论9：任意多项式时间算法（哈希函数SHA256、Merkle树、ECDSA验证、方程等）均可等价为阶为1的等式（也称为：R1CS约束、电路约束、电路门）。

3　ω满足R1CS约束的等价转化

预备知识

1. 向量$\vec{s}$与向量$\vec{s}$的**内积**运算得到一个值y

$$\vec{s} \cdot \vec{s} = (s_1, s_2, s_3) \cdot (s_1, s_2, s_3) = \sum_{i=1}^{3} s_i^2 = y$$

2. 向量$\vec{s}$与矩阵A的**内积**运算得到向量x^{T}

$$\vec{s}\cdot A=(s_1,s_2,s_3)\cdot\begin{vmatrix}a_{1,1}&a_{1,2}&a_{1,3}\\a_{2,1}&a_{2,2}&a_{2,3}\\a_{3,1}&a_{3,2}&a_{3,3}\end{vmatrix}=\left(\sum_{i=1}^{3}s_i\cdot a_{i,1},\sum_{i=1}^{3}s_i\cdot a_{i,2},\sum_{i=1}^{3}s_i\cdot a_{i,3},\right)$$
$$=x^{\mathrm{T}}$$

3.1 ω等价于向量$\vec{s}$

命题1：方程的解$x=3$与向量$\vec{s}$等价。对约束等式10的(c)，有以下等价关系

$$\boldsymbol{x=3}$$
$$\Updownarrow$$
$$\boldsymbol{\vec{s}=[1,out,x,s_1,s_2]}$$

证明： 令$\vec{s}=[1,out,x,s_1,s_2]$。已知$x=3$，能够根据公式10的(c)逐步计算出$s_1,s_2,out,1$，从而能够构造向量$\vec{s}$；
反之，向量$\vec{s}$包含x，因此已知向量$\vec{s}$，能够计算出$x=3$。
因此，知道解$x=3$与知道向量$\vec{s}$是等价的。**其中，向量$\vec{s}$中的1能够表达任意常量。如果算法中有任意常量，则任意常量均是1的倍数。**该倍数存储到后续的多维向量中，则能够表达任意常量。此处，out是方程的计算结果120。

对约束等式10的(b)，有以下等价关系

$$x=3$$
$$\Updownarrow$$
$$\vec{s}=[1,out,x,s_1,s_2,s_3,s_4,s_5]$$

可见R1CS约束等式(b)中的向量需要包含更多的中间状态变量s_3,s_4,s_5。在下一节，R1CS约束等式(b)对应的**矩阵维度**也更大，进而在后续步骤中会产生阶数更高的多项式和更复杂的拉格朗日插值多项式（或傅里叶变换）。因此，下一节的R1CS约束约束等价转化为矩阵时，使用优化的R1CS约束等式(c)。

业务语言：保密数据（方程的解$x=3$、哈希函数原象x，Merkle树的叶子和节点、ECDSA数据M签名σ和公钥Q）就是零知识证明中的$witness$。

实际应用时，为防止证明方$\mathcal{P}$欺骗验证方$\mathcal{V}$，不能将所有数据保密起来，需要公开局部参数。所以，$1,out$需要公开。
因此，将向量$\vec{s}$拆为**公开数据**与**保密数据**。

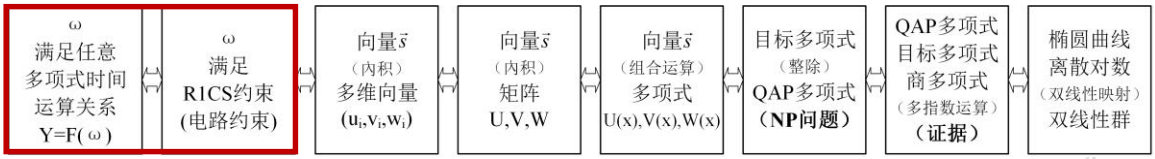
公开数据记为$statement=(1,out)$；out是方程的计算结果120。

保密数据记为$witness=(x,s_1,s_2,s_3)$。

对于Layer2应用场景，out通常是Merkle root。

因此，得出zk-SNARK协议的核心构造： **数据分为公开数据$statement$和保密数据$witness$，则构造向量$\vec{s}=(statement;witness)$**

得出zk-SNARK协议的核心等价关系（一）：
ω满足任意多项式时间算法的计算关系$Y = F(\omega)$等价转换为ω满足R1CS约束，即电路约束。



3.2 ω满足R1CS约束等价转换为向量内积

对于第1个阶为1的等式（**R1CS约束**）$s_1 = x * x$，如下进行向量内积**等价转化**：

$$\begin{cases} \vec{s} \cdot [0,0,0,1,0] = [1, out, x, s_1, s_2] \cdot [0,0,0,1,0] = s_1 \\ \vec{s} \cdot [0,0,1,0,0] = [1, out, x, s_1, s_2] \cdot [0,0,1,0,0] = x \\ \vec{s} \cdot [0,0,1,0,0] = [1, out, x, s_1, s_2] \cdot [0,0,1,0,0] = x \end{cases}$$

令$w_1 = [0,0,0,1,0], u_1 = [0,0,1,0,0], v_1 = [0,0,1,0,0]$。
因此，有以下**等价关系**：

$$\begin{aligned} & s_1 = x * x \\ & \Updownarrow \\ & \vec{s} \cdot w_1 = \vec{s} \cdot u_1 * \vec{s} \cdot v_1 \end{aligned} \tag{11}$$

同理，第2和3个阶为1的等式（R1CS约束）

$$\begin{aligned} & s_2 = s_1 * x \\ & 120 - (s_2 + s_1 + x) = s_2 * x \end{aligned}$$

有如下进行向量内积等价转化：

$$\begin{cases} \vec{s} \cdot [0,0,0,0,1] = [1, out, x, s_1, s_2] \cdot [0,0,0,0,1] = s_2 \\ \vec{s} \cdot [0,0,0,1,0] = [1, out, x, s_1, s_2] \cdot [0,0,0,1,0] = s_1 \\ \vec{s} \cdot [0,0,1,0,0] = [1, out, x, s_1, s_2] \cdot [0,0,1,0,0] = x \end{cases}$$

$$\begin{cases} \vec{s} \cdot [0,1,-1,-1,-1] & = [1, out, x, s_1, s_2] \cdot [0,1,-1,-1,-1] = 120 - (s_2 + s_1 + x) \\ \vec{s} \cdot [0,0,0,0,1] & = [1, out, x, s_1, s_2] \cdot [0,0,0,0,1] = s_2 \\ \vec{s} \cdot [0,0,1,0,0] & = [1, out, x, s_1, s_2] \cdot [0,0,1,0,0] = x \end{cases}$$

令$w_2 = [0,0,0,0,1]$，$u_2 = [0,0,0,1,0]$，$v_2 = [0,0,1,0,0]$；$w_3 = [0,1,-1,-1,-1]$，$u_3 = [0,0,0,0,1]$，$v_3 = [0,0,1,0,0]$。
因此，有以下等价关系：

$$
\begin{aligned}
&s_2 = s_1 * x \\
&120 - (s_2 + s_1 + x) = s_2 + s_1 \\
&\Updownarrow \\
&\vec{s} \cdot w_2 = \vec{s} \cdot u_2 * \vec{s} \cdot v_2 \\
&\vec{s} \cdot w_3 = \vec{s} \cdot u_3 * \vec{s} \cdot v_3
\end{aligned}
\tag{12}
$$

因此，得出zk-SNARK协议的核心等价关系（二）：
ω满足R1CS约束等价转换为向量$\vec{s}$与多维向量$(u_i, v_i, w_i), i = 1,2,3$的内积。

3.3 向量内积等价转化为向量与矩阵的内积

将等式11和等式12中的向量，按照顺序排列，组成三个矩阵W, U, V：

$$
W = \begin{bmatrix} w(1) \\ w(2) \\ w(3) \end{bmatrix} = \begin{bmatrix} 0,0,0,1,0 \\ 0,0,0,0,1 \\ 0,1,-1,-1,-1 \end{bmatrix},
$$

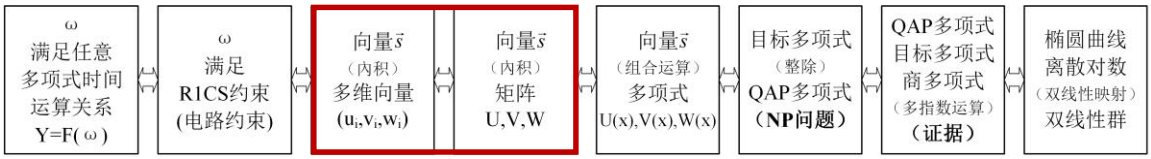
$$
U = \begin{bmatrix} u(1) \\ u(2) \\ u(3) \end{bmatrix} = \begin{bmatrix} 0,0,1,0,0 \\ 0,0,0,1,0 \\ 0,0,0,0,1 \end{bmatrix},
$$

$$
V = \begin{bmatrix} v(1) \\ v(2) \\ v(3) \end{bmatrix} = \begin{bmatrix} 0,0,1,0,0 \\ 0,0,1,0,0 \\ 0,0,1,0,0 \end{bmatrix}
$$

因此，有如下等价运算关系：

$$
\begin{aligned}
&\vec{s} \cdot w(1) = \vec{s} \cdot u(1) * \vec{s} \cdot v(1) \\
&\vec{s} \cdot w(2) = \vec{s} \cdot u(2) * \vec{s} \cdot v(2) \\
&\vec{s} \cdot w(3) = \vec{s} \cdot u(3) * \vec{s} \cdot v(3) \\
&\Updownarrow \\
&\vec{s} \cdot W = \vec{s} \cdot U * \vec{s} \cdot V
\end{aligned}
$$

因此，得出zk-SNARK协议的核心等价关系（三）：
向量$\vec{s}$与多维向量$(u(i), v(i), w(i)), i = 1,2,3$的内积等价转换为向量$\vec{s}$与矩阵$U, V, W$的内积。



4 向量与矩阵内积的等价转化

预备知识

命题2：多项式值表达等价于多项式系数表达。

证明：设$n-1$阶多项式为

$$f(x)=\sum_{i=0}^{m} a_i x^{i-1}$$

（1）已知多项式的值$f_0,\ldots,f_m$和横坐标$x_0,\ldots,x_m$，可以计算出多项式的系数$a_0,\ldots,a_m$；计算方法包括**解方程组、拉格朗日插值法、快速傅里叶变换**等。

（2）已知多项式的系数$a_0,\ldots,a_m$和横坐标$x_0,\ldots,x_m$，可以计算出多项式的值$f_0,\ldots,f_m$。快速计算方法：**快速傅里叶变换**等。

因此，多项式值表达等价于多项式系数表达。

4.1 向量对三组多项式的组合运算

将方程$x^4+x^3+x^2+x=120$转换为R1CS约束后，得到以下三个W,U,V矩阵

$$W=\begin{bmatrix}0,0,0,1,0\\0,0,0,0,1\\0,1,-1,-1,-1\end{bmatrix},U=\begin{bmatrix}0,0,1,0,0\\0,0,0,1,0\\0,0,0,0,1\end{bmatrix},V=\begin{bmatrix}0,0,1,0,0\\0,0,1,0,0\\0,0,1,0,0\end{bmatrix}$$

将矩阵中的元素当作**多项式的值**，例如对于矩阵W

$$\begin{aligned}&w_0(1)=0,w_1(1)=0,w_2(1)=0,\ \ w_3(1)=1,\ \ w_4(1)=0\\&w_0(2)=0,w_1(2)=0,w_2(2)=0,\ \ w_3(2)=0,\ \ w_4(2)=1\\&w_0(3)=0,w_1(3)=1,w_2(3)=-1,w_3(3)=-1,w_4(3)=-1\end{aligned}$$

对于多项式值表达**等价转化为**多项式系数表达，以下介绍的拉格朗日插值法不是最优算法，但在理解上是最直观的。**最优算法是快速傅里叶变换、基4时分的Cooley-Tukey蝶形变换，运算可并行化，并使用GPU/FPGA等硬件加速。**

对于多项式的值，每一列有3个函数值，所以取任意3个横坐标，$x=1,2,3$，作为**横坐标**。可使用**拉格朗日插值多项式**，基于多项式的值计算多项式的系数

$$f(x)=\sum_{k=1}^{t} f_k \prod_{j=1,j\neq k}^{t}\frac{x-x_j}{x_k-x_j}$$

如下计算多项式的系数表达

$$\begin{aligned}w_0(x)&=0\frac{(x-2)(x-3)}{(1-2)(1-3)}+0\frac{(x-1)(x-3)}{(2-1)(2-3)}+0\frac{(x-1)(x-2)}{(3-1)(3-2)}=0\\w_1(x)&=0\frac{(x-2)(x-3)}{(1-2)(1-3)}+0\frac{(x-1)(x-3)}{(2-1)(2-3)}+1\frac{(x-1)(x-2)}{(3-1)(3-2)}=\frac{1}{2}(x^2-3x+2)\\w_2(x)&=0\frac{(x-2)(x-3)}{(1-2)(1-3)}+0\frac{(x-1)(x-3)}{(2-1)(2-3)}-1\frac{(x-1)(x-2)}{(3-1)(3-2)}=-\frac{1}{2}(x^2-3x+2)\\w_3(x)&=1\frac{(x-2)(x-3)}{(1-2)(1-3)}+0\frac{(x-1)(x-3)}{(2-1)(2-3)}-1\frac{(x-1)(x-2)}{(3-1)(3-2)}=-x^2+4x-4\\w_4(x)&=0\frac{(x-2)(x-3)}{(1-2)(1-3)}+1\frac{(x-1)(x-3)}{(2-1)(2-3)}-1\frac{(x-1)(x-2)}{(3-1)(3-2)}=-\frac{1}{2}(3x^2-11x+8)\end{aligned}$$

则**多项式系数**表达$W(x)=[w_0(x),w_1(x),w_2(x),w_3(x),w_4(x)]$；

以此类推，可计算出U矩阵和V矩阵的**多项式系数表达**

$$\begin{aligned}U(x)&=[u_0(x),u_1(x),u_2(x),u_3(x),u_4(x)]\\V(x)&=[v_0(x),v_1(x),v_2(x),v_3(x),v_4(x)]\end{aligned}$$

则有以下等价关系

$$\vec{s}\cdot W=\vec{s}\cdot U*\vec{s}\cdot V$$
$$\Updownarrow$$
$$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)=0,x=1,2,3$$

其中，等式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$是向量$\vec{s}$对三组多项式的组合运算，运算结果称为**二次算法多项式**，简称**QAP多项式**。

因此，得出zk-SNARK协议的核心等价关系（四）：
向量$\vec{s}$与矩阵内积等价转换为向量$\vec{s}$对多项式$w_i(x),u_i(x),v_i(x),i=0,\dots,m$的组合运算。

4.2 目标多项式整除QAP多项式构造NP问题

上述拉格朗日插值多项式引入**横坐标**为$x=1,2,3$，则能够构造多项式

$$z(x)=(x-1)(x-2)(x-3)$$

$z(x)$称为**目标多项式**。也可以取任意**横坐标**变量，如令$x=4,5,6$，则重新使用拉格朗日插值计算函数值，并令目标多项式为$z(x)=(x-4)(x-5)(x-6)$。

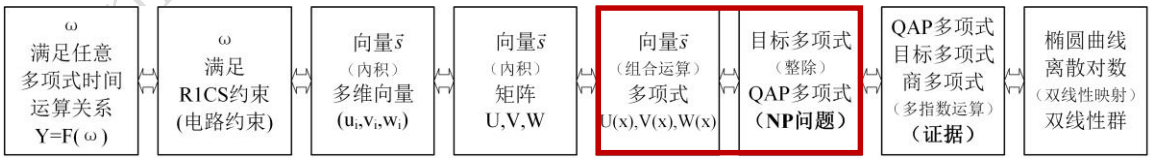
当$x=1,2,3$，目标多项式$z(x)$和QAP多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$均为零

$$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)=0,x=1,2,3$$
$$z(x)=(x-1)(x-2)(x-3)=0,x=1,2,3$$

此外，QAP多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$的阶高于目标多项式$z(x)$，所以QAP多项式等于零还有其他解。因此，目标多项式是QAP多项式的因子。换言之，目标多项式$z(x)$与QAP多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$呈**整除关系**

$$z(x)|(\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x))$$

因此，得出zk-SNARK协议的核心等价关系（五）：
向量$\vec{s}$对三组多项式$w_i(x),u_i(x),v_i(x),i=0,\dots,m$的组合运算等价转换为目标多项式$z(x)$整除QAP多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$。构成NP问题



上述等式中的目标多项式$z(x)$和多项式$w_i(x),u_i(x),v_i(x),i=0,\dots,m$均是公开的，而向量$\vec{s}$是保密的。

已知阶为n的目标多项式$z(x)$、阶小于n的三组多项式$w_i(x),u_i(x),v_i(x),i=0,\dots,m$，这些多项式的解相同。求向量$\vec{s}$，构造QAP多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot$

$V(x)$，使得目标多项式$z(x)$整除QAP多项式

$$z(x)|[\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)]$$

该问题构成NP问题。

接下来，构造零知识。对外暴漏离散对数点。

证明方$\mathcal{P}$不公开该NP问题的向量$\vec{s}$，而是将向量$\vec{s}$构造出的QAP多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$除以目标多项式$z(x)$，得到商多项式$h(x)$

$$h(x):=(\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x))/z(x)$$

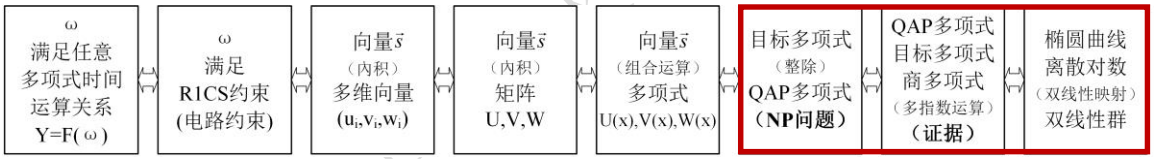
然后，将QAP多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$、目标多项式$z(x)$和商多项式$h(x)$的系数放到椭圆曲线离散对数上（密码学专业术语为：多项式承诺），生成证明（离散对数点）。

验证方获得离散对数点，基于双线性映射重构整除关系，验证了向量$\vec{s}$的正确性，却不知道解向量$\vec{s}$。多项式承诺与双线性映射等计算原理将在下一节详细叙述。

因此，得出zk-SNARK协议的等价关系：

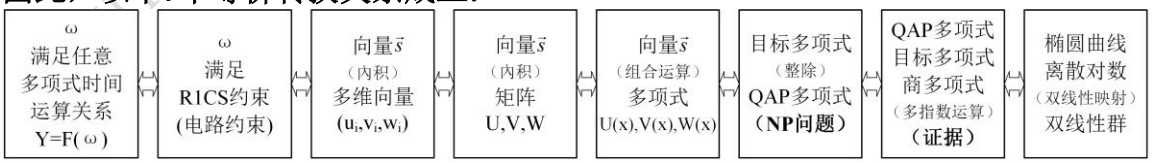
（六）目标多项式$z(x)$整除QAP多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$（构成NP问题）等价转化为基于三个多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x),h(x),z(x)$的系数计算椭圆曲线离散对数点（NP问题放离散对数点上），对外暴漏离散对数点。

（七）证明方$\mathcal{P}$基于多项式$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x),h(x),z(x)$系数计算离散对数点等价转化为验证方$\mathcal{V}$重构整除关系，验证向量$\vec{s}$的正确性，却不知道向量$\vec{s}$。



反之，如果验证方$\mathcal{V}$能够基于离散对数点重构整除关系，则向量$\vec{s}$对三组多项式的组合运算是正确的，则向量与矩阵的内积运算正确，则向量与多维向量的内积正确，则ω满足R1CS约束，则$x=3$是方程的解（或x是哈希函数的原象，a,b是布尔值，叶子与节点在Merkle树上），则证明方$\mathcal{P}$对数据的运算是正确，则用户提交了正确交易数据，则旷工认可用户提交的交易单，则交易成功。

因此，以下7个等价转换关系成立：



5 zk-SNARK协议框架

zk-SNARK协议包括以下10个步骤：

0.（初始化a）生成与电路无关的局部系统参数$CRS1$。

1.证明方$\mathcal{P}$证明拥有ω且ω满足任意运算关系R。

2.证明方$\mathcal{P}$证明拥有ω且ω满足R1CS约束。
3.向量$\vec{s}$与多维向量的内积$\vec{s}\cdot w_i=\vec{s}\cdot u_i*\vec{s}\cdot v_i$
4.向量$\vec{s}$与矩阵U,V,W的内积$\vec{s}\cdot W=\vec{s}\cdot U*\vec{s}\cdot V$。
5.向量$\vec{s}$对多项式进行组合运算$\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)=0$。
6.目标多项式$z(x)$整除QAP多项式$z(x)|[\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)]$构成NP问题
7.（初始化b）基于电路多项式$W(x),U(x),V(x)$生成局部系统参数$CRS2$。
8. 证明方$\mathcal{P}$将QAP多项式、目标多项式、商多项式放到离散对数点上，对外暴漏离散对数点A B C，实现零知识。
9.验证方$\mathcal{V}$重构整除关系，验证向量$\vec{s}$的正确性，却不知道向量$\vec{s}$。

分析
1. 将QAP多项式、目标多项式和商多项式的系数放到的离散对数上，验证方在离散对数点上重构整除关系，完成向量$\vec{s}$正确性验证，却不知道解向量$\vec{s}$，即**零知识**。
2. **复杂度很高的任意多项式时间算法，**等价转化为R1CS约束、多维向量、矩阵、多项式系数表达。最后与向量对三组项式组合运算生成QAP多项式，并与目标多项式、商多项式的系数均放到离散对数点上，所以**证明方$\mathcal{P}$的计算复杂度很高**；而验证方仅需要使用离散对数点进行**双线性映射**，**重构整除关系**，则实现向量$\vec{s}$的正确性验证，**所以验证复杂度较低，实现计算压缩。**

算法评价
1：Groth16的CRS包含R1CS约束**等价转化**的**电路多项式$W(x),U(x),V(x)$，非常具体。**
优点：证明方$\mathcal{P}$使用CRS中的**电路多项式$W(x),U(x),V(x)$**生成**证明，速度很快**。
缺点：这个CRS包含的**电路多项式$W(x),U(x),V(x)$**是由R1CS约束转换而来，**已经固化，只能表达唯一运算电路，不能表达其他运算电路**，所以表达**能力极差**。如果对layer2的电路进行修改，则步骤6的**初始化b**局部CRS也需要修改，则Layer1的合约参数CRS也需要对应修改才能够进行一致性验证。

2：PLONK的**CRS**不包含**电路多项式$W(x),U(x),V(x)$，只包含大量的离散对数点。**
优点：CRS的表达能力很强，能够用于任意多项式时间电路生成证明。
缺点：CRS表达能力太强，R1CS约束力不够，不足以防止证明方$\mathcal{P}$作弊，所以引入额外的**线约束**。**线约束**计算复杂度相对较高，导致证明生成缓慢。

6 Groth16协议详解
6.1 协议原理
初始化CRS：选择【**有毒废料**】随机数$\alpha,\beta,\gamma,\delta,x\leftarrow\mathbb{Z}_p^*$，计算系统参数CRS

$$[\sigma_1]_1 = \left(\begin{array}{c} \alpha, \beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}, \left\{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\gamma}\right\}_{i=0}^{l} \\ \left\{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}\right\}_{i=l+1}^{n-1}, \left\{\frac{x^i z(x)}{\delta}\right\}_{i=0}^{n-2} \end{array} \right) \cdot G_1$$

$$[\sigma_2]_2 = \left(\beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}\right) \cdot G_2$$

有毒废料两种计算方法：

（1）硬件计算：输出CRS后砸坏硬件；**需要经常更新电路，则该方法不可取。**

（2）安全多方计算：只要有一个参与方删除**有毒废料**，则CRS安全。

缺点：算法更新，则电路更新，则CRS更新，但是CRS计算复杂度比较高。

Plonk的**CRS是通用的，**算法更新，则电路更新，仅更新VK，**而不需要更新基于有毒废料计算出的那部分CRS。**

Groth16CRS分为2个集合，证明需要使用的部分记为PK，验证需要使用的部分记为VK。

证明： 证明方$\mathcal{P}$选择随机数$r, s \leftarrow \mathbb{Z}_p$，$a_0 = 1$，基于向量$\vec{s} = (a_1, \ldots, a_m)$和CRS，构造线性组合关系，并将组合多项式的系数放到椭圆曲线离散对数点上，形成离散对数困难问题。**对外暴漏3个离散对数点$([A]_1, [B]_2, [C]_1)$。**

$$[A]_1 = (\alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta) \cdot G_1$$
$$[B]_2 = (\beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta) \cdot G_2$$
$$[C]_1 = \left(\frac{\sum_{i=l+1}^{m} a_i(\beta u_i(x)+\alpha v_i(x)+w_i(x))+h(x)z(x)}{\delta} + As + Br - rs\delta\right) \cdot G_1$$
$$[B]_1 = (\beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta) \cdot G_1$$

最大优点：proof size小。验证复杂度很低。

验证： 验证方$\mathcal{V}$对椭圆曲线离散对数点**重构整除关系**

$$e([A]_1, [B]_2) = e(\alpha G_1, \beta G_2) \cdot e\left(\sum_{i=0}^{l} \frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\gamma} G_1, \gamma G_2\right) \cdot e([C]_1, \delta G_2)$$

如果等式成立，则验证成功，否则拒绝。

6.2 协议分析

（一）、一致性

$$\begin{aligned} Left: A &= \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta \\ B &= \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta \\ A \cdot B &= (\alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta) \cdot (\beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta) \\ &= \alpha\beta + \alpha s\delta + r\delta\beta + rs\delta^2 + \alpha \cdot \sum_{i=0}^{m} a_i v_i(x) + \beta \sum_{i=0}^{m} a_i u_i(x) + \sum_{i=0}^{m} a_i u_i(x) \sum_{i=0}^{m} a_i v_i(x) + s\delta \sum_{i=0}^{m} a_i u_i(x) + r\delta \sum_{i=0}^{m} a_i v_i(x) \end{aligned}$$

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

（二）、完备性分析：

如果验证方验证等式$A \cdot B = \alpha\beta + \frac{\sum_{i=0}^{l} a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))+h(x)z(x)}{\gamma}\gamma + C\delta$成立，则完备性中$\sum_{i=0}^{m} a_i u_i(x) \sum_{i=0}^{m} a_i v_i(x) = \sum_{i=0}^{m} a_i \omega_i(x) + h(x)z(x)$一定是相等的。因此，证明方证明了其知道$z(x)$整除QAP多项式$\sum_{i=0}^{m} a_i u_i(x) \sum_{i=0}^{m} a_i v_i(x) - \sum_{i=0}^{m} a_i \omega_i(x)$，即证明方知道该NP问题的解。

（三）、证明方无法作弊：

$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta$$

$$B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta$$

$$C = \frac{\sum_{i=l+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + \omega_i(x)) + h(x)z(x)}{\delta} + As + rB - rs\delta$$

验证等式$A \cdot B = \alpha \cdot \beta + \frac{\sum_{i=0}^{l} a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))+h(x)z(x)}{\gamma} \cdot \gamma + C \cdot \delta$的右边包含$\alpha \cdot \beta$，所以**证明方**在计算$A$的时候需要诚实包含$\alpha$，在计算$B$的时候需要诚实包含$\beta$，$A$和$B$相乘，才会出现$\alpha \cdot \beta$。如果证明方作弊，令$A$中包含$\alpha \cdot \beta$，而$B$中包含或不包含$\alpha \cdot \beta$，则证明方无法构造出$\alpha \cdot \sum_{i=0}^{m} a_i v_i(x) + \beta \sum_{i=0}^{m} a_i u_i(x) + \sum_{i=0}^{m} a_i u_i(x) \sum_{i=0}^{m} a_i v_i(x)$这些多项式。类似地，证明方需要诚实进行以下构造

$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta$$

$$B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta$$

$$C = \frac{\sum_{i=l+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + \omega_i(x)) + h(x)z(x)}{\delta} + As + rB - rs\delta$$

此外，证明方只要修改任意一个参数，都无法使得验证等式成立

$$A \cdot B = \alpha \cdot \beta + \frac{\sum_{i=0}^{l} a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))+h(x)z(x)}{\gamma} \cdot \gamma + C \cdot \delta$$

原理分析

1. α和β迫使证明A, B, C必须采用同一套向量$statement = a_0, \ldots, a_m$参数，而不能是其他参数。**反之：**如果计算$A$采用第1套参数 $a_0, \ldots, a_m$参数。计算B采用第2套参数$a_0', \ldots, a_m'$，那么C应该采用第1套还是第2套参数呢？答：不管C采用第1套还是第2套参数，等式都不会成立。如果计算C采用第1套参数，计算出来的表达式与A有对应关系，却与B没有对应关系，等式肯定不成立。

2. 对于A, B, C的构造，需要证明方选择随机数r, s。如果不添加随机数r, s，那么证明A, B, C就失去了随机性，验证方可以根据等式求解出$witness$。因此，算法中添加随机数的算法功能叫作：盲化或随机化，让验证方计算不出$witness$。

3. **γ,δ的作用：** $A\cdot B=\alpha\cdot\beta+\frac{\sum_{i=0}^{l}a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))+h(x)z(x)}{\gamma}\cdot\gamma+C\cdot\delta$，验证等式的最后两项是关于$\gamma,\delta$。添加$\gamma,\delta$后，验证等式的右边的值独立于$\alpha\cdot\beta$。因此，也需要在$A$和$B$的构造中嵌入$\gamma\delta$，使得$A\cdot B$的结果独立于$\alpha\cdot\beta$。

4. **证明方必须要知道$witness$：** 在构造过程中，以下等式涉及$witness=a_{l+1},\dots,a_m$。如果证明方不知道$witness$，则无法构造出正确的证明$A,B,C$，使得验证等式成立。

$$A=\alpha+\sum_{i=0}^{m}a_iu_i(x)+r\delta=\alpha+\sum_{i=0}^{l}a_iu_i(x)+\sum_{i=l+1}^{m}a_iu_i(x)+r\delta$$
$$B=\beta+\sum_{i=0}^{m}a_iv_i(x)+s\delta=\beta+\sum_{i=0}^{l}a_iv_i(x)+\sum_{i=l+1}^{m}a_iv_i(x)+s\delta$$
$$C=\frac{\sum_{i=l+1}^{m}a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))+h(x)z(x)}{\delta}+As+rB-rs\delta$$

5. **线性关系：** 证明方构造出的证明A,B,C与CRS呈线性关系，所以验证方能够重构线性关系。

6. 证明方生成证明A,B,C的过程中，C的生成使用了目标多项式$z(x)$；但是，**模拟器$\mathcal{S}$**生成证明过程中没有使用目标多项式$z(x)$。

$Simulator$:
$A=\alpha+\sum_{i=0}^{m}a_iu_i(x)+r\delta;$
$B=\beta+\sum_{i=0}^{m}a_iv_i(x)+s\delta$
$C=\frac{AB-\alpha\beta-\sum_{i=0}^{l}a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))}{\delta}$

$Verifier$:
$$\alpha\cdot\beta+\frac{\sum_{i=0}^{l}a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))}{\gamma}\cdot\gamma+C\cdot\delta$$
$$=\alpha\cdot\beta+\frac{\sum_{i=0}^{l}a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))}{\gamma}\cdot\gamma+\left(\frac{AB-\alpha\beta-\sum_{i=0}^{l}a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))}{\delta}\right)\cdot\delta$$
$$=\alpha\cdot\beta+\sum_{i=0}^{l}a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))+AB-\alpha\beta-\sum_{i=0}^{l}a_i(\beta u_i(x)+\alpha v_i(x)+\omega_i(x))$$
$$=AB$$

验证通过。但是，该验证过程不要求$\sum_{i=0}^{m}a_iu_i(x)\sum_{i=0}^{m}a_iv_i(x)=\sum_{i=0}^{m}a_i\omega_i(x)+h(x)z(x)$相等。即模拟器$\mathcal{S}$不需要知道目标多项式$z(x)$整除线性组合多项式，一样能够使得验证方验证通过。**理解为模拟器$\mathcal{S}$拥有陷门，能够构造证明，使得验证通过。**虽然等式验证通过，但是目标多项式$z(x)$不整除线性组合多项式，模拟器$\mathcal{S}$完美模拟证明方$\mathcal{P}$。

安全性分析：

1. 从验证方角度，模拟器$\mathcal{S}$生成证明与证明方$\mathcal{P}$生成证明**不可区分**。

2. 模拟器$\mathcal{S}$没有秘密，验证方使用模拟器$\mathcal{S}$提供的证明计算不出任何秘密。
3. 验证方基于证明方$\mathcal{P}$生成的证明，也计算不出任何秘密，实现零知识。

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