零知识证明

星辰实验室

lynndell2010@gmail.com

目　　录

第一章　zk-SNARK 从入门到精通

1.1　零知识证明基础

1.1.1　预备知识

交互式零知识证明：证明方 $\mathcal{P}$ 发送数据给验证方 $\mathcal{V}$，也需要验证方 $\mathcal{V}$ 发送数据给证明方 $\mathcal{P}$[1, 2]。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。该过程没有其他额外的数据交互。与用户将 ECDSA 签名发送出去而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[3]。

多项式时间算法：能够快速计算出结果的算法[4]。

1. 哈希算法：已知私钥 sk 和椭圆曲线基点 G，能够快速计算出公钥 PK[5]

$$PK := sk \cdot G$$

2. 倍点运算：已知原象 x 和哈希函数 $SHA256$，能够快速计算出函数值 y[6]

$$y := SHA256(x)$$

非多项式时间算法：包括指数时间算法、亚指数时间算法[7]。以下计算需要指数时间：

1. 哈希求逆：已知公钥 PK 和椭圆曲线基点 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出私钥 sk，使得以下等式成立

$$PK = sk \cdot G$$

2. 离散对数：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出原象 x，使得以下等式成立

$$y = SHA256(x)$$

根据维基百科，目前，学术届还没有区分 P 问题是否等于 NP 问题。为方便理解 zkSnark，从应用角度出发，我们认为 P 问题不等于 NP 问题。

P 问题：在多项式时间内可解的问题[8, 9]。以下问题多项式时间内可求解：

1. 求公钥？已知私钥 sk 和椭圆曲线基点 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：（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。（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[8]。以下是 3 个典型的 NP 问题：

1. 求私钥？已知公钥 PK 和基点 G，求私钥 sk。要求私钥与公钥满足以下**离散对数关系**

$$PK = sk \cdot G$$

不能在多项式时间内求解出私钥 sk，需要指数时间。但是，一旦给出私钥 sk，则能够在多项式时间内验证该私钥 sk 与公钥 PK 是否满足离散对数关系。

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

$$y = SHA256(x)$$

不能在多项式时间内求解出原象 x，需要指数时间。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。

因此，**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[10]

已知阶为 n 的一个多项式 $z(x)$ 和阶为 $n-1$ 的三组多项式 $u_0(x), u_1(x), ..., u_m(x); v_0(x), v_1(x), ..., v_m(x); w_0(x), w_1(x), ..., w_m(x)$，求向量 $\vec{s} = (1, s_1, ..., s_m)$，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 $\vec{s}$ 分别对三组多项式 $u_0(x), u_1(x), ..., u_m(x); v_0(x), v_1(x), ..., v_m(x); w_0(x), w_1(x), ..., w_m(x)$ 进行线性组合，分别得到三个线性组合多项式

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

然后，将前两个线性组合多项式相乘并减去第三个多项式。这两个过程称为组合运算。

向量 $\vec{s}$ 的维度为 m。如果每个元素 s_i 的取值空间为 a，则将 $(u_0(x) + \sum_{i=1}^{m} s_i \cdot u_i(x)) \cdot (v_0(x) + \sum_{i=1}^{m} s_i \cdot v_i(x)) - (w_0(x) + \sum_{i=1}^{m} s_i \cdot w_i(x))$ 称为**二次算法多项式**，简称 **QAP 多项式**。QAP 多项式的构造为**指数空间** a^m。如果元素 s_i 的取值空间为 0 或 1，则将 $(u_0(x) + \sum_{i=1}^{m} s_i \cdot u_i(x)) \cdot (v_0(x) + \sum_{i=1}^{m} s_i \cdot v_i(x)) - (w_0(x) + \sum_{i=1}^{m} s_i \cdot w_i(x))$ 称为**二次扩张多项式，或二次布尔多项式**，简称 **QSP 多项式**。QSP 多项式的构造为**指数空间** 2^m。因此，这两个组

合运算的计算时间均是**指数时间**。

多项式 $z(x)$ 等于零有 n 个解，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，且商多项式本质上就是 QAP/QSP 多项式等于零的 $n-2$ 个其他解构成的多项式。

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。如果满足，则接受，否则拒绝。因此，需要指数时间才能够暴力搜索出向量 $\vec{s}$。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 $z(x)$ 与构造出的 QAP/QSP 多项式是否满足**整除关系**。**因此，多项式 z(x) 与 QAP/QSP 多项式的整除关系，满足单项性，构成 NP 问题**。该构造至关重要，本文后续的举例需使用多项式整除关系构造 NP 问题。

密码学核心思想：

1. 秘密数据满足加性运算关系，或乘性运算关系；
2. 将数据放到椭圆曲线离散对数点上，形成离散对数困难；
3. 椭圆曲线离散对数点是群元素，群元素可进行二元运算；
4. 群元素相乘，则指数相加，实现加性同态，可重构加性关系；
5. 双线性映射，则指数相乘，实现乘性同态，可重构乘性关系。

零知识证明与 zk-SNARK 核心思想：

由于用户隐私保护、数据保密性、区块链扩容等应用需求，

1. 证明方：将上述 NP 问题中 QAP/QSP 多项式、$z(x)$ 多项式和商多项式这三个多项式的系数放到椭圆曲线离散对数点上（专业术语称为：多项式承诺），形成离散对数困难。
2. 验证方：对生成的椭圆曲线离散对数点进行双线性映射，重构整除关系，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，可使用经典的 Σ 协议证明证明方 $\mathcal{P}$ 知道私钥 sk 而不泄露私钥 sk。第 2 个问题无法使用 Σ 协议进行证明，但是可以将第 2 个 NP 问题转换为第 3 个 NP 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，使得验证方能够重构整除关系。最后，验证方验证了向量 $\vec{s}$ 的正确性，但是证明方没泄露向量 $\vec{s}$。

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。如果足够大，则对应的计算复杂度较高，能够抵抗暴力破解。为方便展示计算过程，本文在第1.2.3节的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，如果方程的阶很高，对应的向量 $\vec{s}$ 维度很大，则不会被暴力破解。

1.1.2　零知识证明

零知识证明引言

对于第 1 个 NP 问题：证明方 $\mathcal{P}$ 向验证方 $\mathcal{V}$ 证明其拥有私钥 sk，但不泄露 sk。

1. 证明方 $\mathcal{P}$ 用私钥 sk 和随机数 k，计算 ECDSA 签名 $\sigma = (r, s)$。
2. 验证方 $\mathcal{V}$ 基于 (r, s) 构造椭圆曲线点，结合公钥 PK 重构线性关系，则验证成功。

则验证方 $\mathcal{V}$ 认可证明方 $\mathcal{P}$ 有私钥 sk 而不知道该私钥 sk。

该举例使用数字签名实现零知识证明，可见零知识证明由数字签名发展而来。数字签名中的私钥天然满足对公钥的离散对数关系，但是零知识证明中的秘密 ω 满足任意计算规则 $y = F(\omega)$。因此，零知识证明对离散对数关系进行了巨大的扩展，使得零知识证明有了广泛的应用范畴。

定义 1：**零知识证明**：令 R 为一个高效可计算的二元关系。对于一对二元组 $(x, \omega) \in R$，x 为声明，ω 为秘密见证。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。

一个具体的**知识证明协议** $ZK\{\omega|(x, \omega) \in R\}$ 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**：安全多方计算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作为挑战 e 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

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

$ZK\{\omega|(x,\omega) \in R\}$ 协议具有**完整性**，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：

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

$ZK\{\omega|(x,\omega) \in R\}$ 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**完成性、知识提取的鲁棒性**和**诚实验证方零知识**，则协议 $ZK\{\omega|(x,\omega) \in R\}$ 是一个 Σ 协议[11]。

1.1.3 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：群 $\mathbb{G}$ 的阶为 q，生成元为 G；公开输入为 $Q \in \mathbb{G}$。在该系统下，证明方 $\mathcal{P}$ 证明：其知道 ω 满足离散对数关系 $Q = \omega \cdot G$。

交互式零知识证明

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

2. 挑战： 验证方 $\mathcal{V}$ 选择随机数 e，发送 e；

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

4. 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。

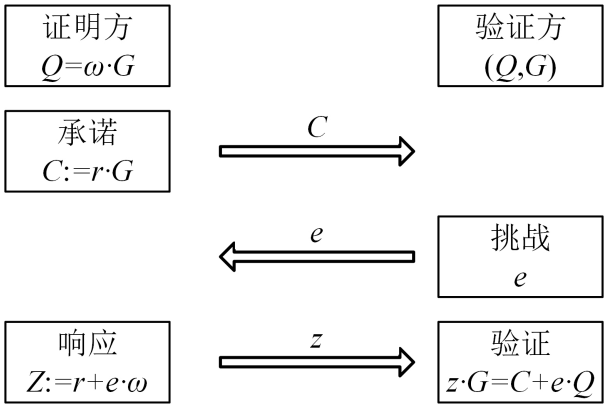


图 1　Σ 协议

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 $z := r + e \cdot \omega$，生成随机数 z；
2. 验证方 $\mathcal{V}$ 基于随机数 z, e 构造椭圆曲线离散对数点 $z \cdot G, e \cdot Q$，并与点 C，在椭圆曲线离散对数上重构线性关系，实现一致性验证。

非交互式零知识证明

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

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

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

4. 验证： 验证方 $\mathcal{V}$ 计算 $e := H(C)$，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，验证方用 Alice 的公钥 PK 对消息签名对 (m, σ) 进行一致性验证

$$\sigma \leftarrow Sign(sk, m)$$

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

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 ω 看作私钥 sk，Q 看作公钥 PK。

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

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

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

4. 验证： 验证方 $\mathcal{V}$ 计算 $e := Hash(C, m)$，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 步的数据 (C, e, z) 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，该协议满足数字签名的定义。

本质上该协议是证明：（1）Alice 拥有私钥 $sk = \omega$，且（2）ω 与消息 m 具有绑定关系 $e = Hash(A, m)$。

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，形成离散对数困难问题。最后，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 ω。

zk-SNARK 协议：需要证明 ω 满足任意计算关系 $y = F(\omega)$[12, 13]。该任意计算包括 NP 问题和 P 问题。

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 树的叶子和节点；
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，哈希函数的原象 x、Merkle 树的叶子和节点、布尔值 b、方程解 $x = 3$ 以及计算过程的中间状态值就是秘密，记为 ω 或 $witness$。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。公开 R1CS 约束（电路约束），然后将 ω 满足任意计算关系 $y = F(\omega)$ 等价转化为 ω 满足 R1CS 约束并**合并 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 $\vec{s}$ 对三组多项式的组合运算，再等价转化为目标多项式 $z(x)$ 整除 QAP 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（即多项式承诺），形成离散对数困难问题；最后，验证方 $\mathcal{V}$ 基于椭圆曲线离散对数点（多项式承诺）重构整除关系，验证了向量 $\vec{s}$ 的正确性，但是证明方 $\mathcal{P}$ 没泄露向量 $\vec{s}$。

zk-SNARK 协议的 7 个核心等价转化关系，如图2所示

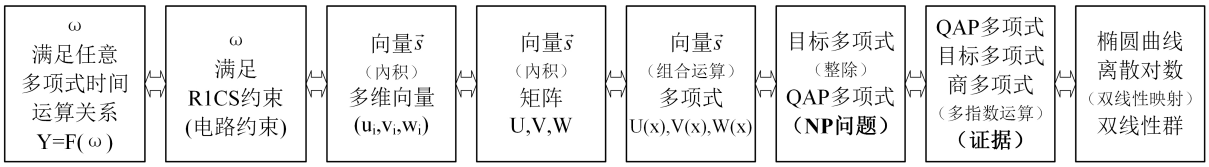


图 2　zk-SNARK 核心等价转化关系

接下来对 zk-SNARK 协议中的核心等价转换关系进行详细叙述。

1.2　多项式时间算法等价转换为阶为 1 的等式

1.2.1　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。

$$\begin{aligned} ma(x,y,z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\ ch(x,y,z) &= (x \wedge y) \oplus (\neg x \wedge z) \\ \Sigma_0(x) &= S^2(x) \oplus S^{13}(x) \oplus S^{22}(x) \\ \Sigma_1(x) &= S^6(x) \oplus S^{11}(x) \oplus S^{25}(x) \\ \sigma_0(x) &= S^7(x) \oplus S^{18}(x) \oplus R^3(x) \\ \sigma_1(x) &= S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x) \end{aligned} \tag{1.1}$$

以下对布尔值和异或运算进行等价转化：

布尔值 a 和 b，如下拆为阶为 1 的等式

$$\begin{aligned} (1-a) \times a &= 0 \\ (1-b) \times b &= 0 \end{aligned} \tag{1.2}$$

位异或运算 $a \oplus b = c$，如下拆为 4 个阶为 1 的等式

$$\begin{aligned} (1-a) \times a &= 0 \\ (1-b) \times b &= 0 \\ (1-c) \times c &= 0 \\ (2a) \times (b) &= (a+b-c) \end{aligned} \tag{1.3}$$

推论 1：*布尔值 a 和 b 等价于阶为 1 的等式1.2*;

推论 2： 位异或运算 $a \oplus b = c$ 等价于 *4* 个阶为 *1* 的等式*1.3*。

因此，基于推论1和2，可以得出以下推论：

推论 3： 哈希函数 *SHA256* 或公式*1.1*等价于某些阶为 *1* 的等式。

推论 4： *Merkle* 树等价于某些阶为 *1* 的等式。

上述 3 个阶为 1 的等式就是电路约束。这 3 个阶为 1 的等式是乘法等式，所以是乘法约束。公式1.3的前 3 个乘法约束限定输入为布尔值，第 4 个乘法约束实现异或运算的等价功能。一方面：可以基于这 4 个等式，设计实际的硬件电路或 FPGA。另一方面：可以用程序表达这 4 个运算规则，则等价于表达出电路约束，等价于实现电路运算原理。因此，算法的运算规则等价于电路约束。

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 和上述阶为 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。

在零知识证明协议中，*witness* 是布尔值 a, b 或原象 x，*statement* 是 c 或哈希值 y，算法的计算规则就是电路约束（R1CS 约束）。因此，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，所以验证方的计算量没降低。后续通过多项式放到离散对数上解决。

1.2.2 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**：已知基点 $G = (x_0, y_0)$ 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 和基点 $G = (x_0, y_0)$，可在多项式时间内计算公钥 $Q = (x_q, y_q)$。

（二）数字签名

ECDSA 签名[14]：用户的私钥为 d，对于消息 M，选择随机数 $k \in [1, ..., n-1]$，如下计算：

$$\begin{aligned} (x_1, y_1) &:= k \cdot G \\ r &:= x_1 \bmod n \\ s &:= k^{-1} \cdot (SHA256(M) + dr) \bmod n \end{aligned} \tag{1.4}$$

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

ECDSA 签名原理解析：

1. 选择一个随机数 k，计算椭圆曲线离散对数 $k \cdot G$。随机数 k 对私钥 d 进行随机化，防止攻击者计算出私钥 d。
2. 计算出两个随机数 (r, s) 就是签名。这两个随机数 (r, 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

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

因此，攻击者无法根据 (r, s) 计算出随机数 k 与私钥 sk。

ECDSA 验证：验证方基于 (M, σ, Q) 如下计算：

$$\begin{aligned} u_1 &:= SHA256(M) \cdot s^{-1} \bmod n, \\ u_2 &:= r \cdot s^{-1} \bmod n, \\ (x_1, y_1) &:= u_1 \cdot G + u_2 \cdot Q \end{aligned} \tag{1.5}$$

如果等式 $r = x_1 \bmod n$ 成立，则接受，否则拒绝。

ECDSA 验证原理解析：

1. 基于两个随机数 (r, s) 和消息 M 计算出 u_1, u_2；
2. 基于 u_1, u_2、基点 G 和公钥 Q 进行倍点运算，计算出 2 个椭圆曲线点 $u_1 \cdot G, u_2 \cdot Q$；
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 思想精华：用户从私钥角度计算椭圆曲线离散对数点 (x_1, y_1)，并产生两个随机数 (r, s)，使得验证方能够从公钥的角度重新计算出椭圆曲线离散对数点 (x_1, y_1)。

ECDSA 算法包括 3 个基本的运算，分别为椭圆曲线离散对数点加运算 $u_1 \cdot G + u_2 \cdot Q$、倍点运算 $k \cdot G$、域元素相乘的模 n 运算。

为方便后续分析，将点加运算 $(x_1, y_1) := u_1 \cdot G + u_2 \cdot Q$ 进行如下修改：

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

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：

$$\begin{aligned} x_3 &= \frac{x_1 \cdot y_2 + y_1 \cdot x_2}{1 + d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2} \\ y_3 &= \frac{y_1 \cdot y_2 - a \cdot x_1 \cdot x_2}{1 - d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2} \end{aligned} \tag{1.6}$$

其中，$a = -1, d = -168696/168700$ 是两个常量。

当 $x_1 = x_2, y_1 = y_2$，就是倍点运算，因此以下仅讨论点加运算。

将上述点加运算公式1.6拆为 7 个阶为 1 的等式

$$\begin{aligned} (x_1) \times (y_2) &= (A_1) \\ (x_2) \times (y_1) &= (A_2) \\ (A_1) \times (A_2) &= (A_3) \\ (x_3) \times (1 + d \cdot A_3) &= (A_4) \\ (y_1) \times (y_2) &= (A_5) \\ (x_1) \times (x_2) &= (A_6) \\ (y_3) \times (1 - d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2) &= (y_1 \cdot y_2 - a \cdot x_1 \cdot x_2) \end{aligned} \tag{1.7}$$

推论 5：*点加运算公式1.6与等式 1.7 等价。*

推论 6：*ECDSA 能够拆为阶为 1 的等式。或 ECDSA 与某些阶为 1 的等式等价。*

这些阶为 1 的等式就是电路约束。其中，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。一方面：可以基于这 7 个阶为 1 的等式（R1CS 约束），设计实际的运算电路或 FPGA。另一方面：可以用程序表达这 7 个运算规则，则等价于表达出电路约束，等价于实现电路运算原理。因此，算法的运算规则就是电路约束。

证明方把 $(x_1, y_1), (x_2, y_2), (x_3, y_3), A_1, A_2, A_3, A_4, A_5, A_6$ 发送给验证方，则验证方根据阶为 1 的等式 1.8 成功验证，则验证方确定证明方是诚实进行点加运算的。

类似地，证明方把消息 M、签名值 σ 和公钥 Q 发送给验证方，验证方根据某些阶为 1 的等式进行验证，则能够确定证明方诚实计算了 ECDSSA 签名。

但是，仍存在以下两个缺点：

1. **消息太多**：或秘密泄露。交易消息 M、签名值 σ 和公钥 Q 泄露。解决方案：仅存储交易消息 M 中的公开数据 *pubdata* 和 Merkle 根，剩余的所有消息隐藏起来，如签名和公钥当作 *witness*。
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签名验证计算复杂度。后续通过多项式和离散对数解决该问题。

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。*

1.2.3　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）均有一个对应的阶为 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 个作用：（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

$$\begin{aligned} s_1 &= x * x \\ s_2 &= s_1 * x \\ s_3 &= s_2 * x \\ s_4 &= s_1 + x \\ s_5 &= s_4 + s_2 \\ 120 &= s_5 + s_3 \end{aligned} \tag{1.8}$$

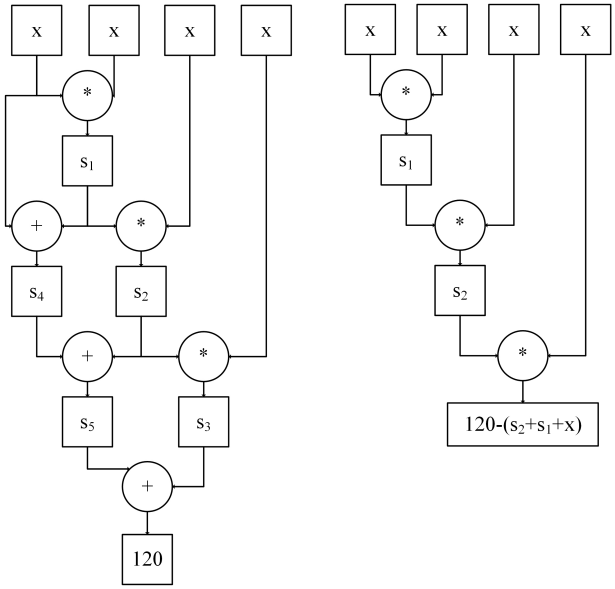
这 6 个阶为 1 的等式限定了方程的运算规则，能够用电路约束表达同样的运算规则。电路约束即能用硬件电路或 FPGA 实现，也可以用软件实现等价的功能。阶为 1 的等式 = 电路约束，阶为 1 的乘法等式 = 乘法约束，阶为 1 的加法等式 = 加法约束。

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 个乘法等式和 3 个加法等式，可使用 3 个乘法约束和 3 个加法约束实现运算原理，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1.8化简为以下 3 个阶为 1 的乘法等式1.9。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。

$$\begin{aligned} s_1 &= x * x \\ s_2 &= s_1 * x \\ 120 - (s_2 + s_1 + x) &= s_2 * x \end{aligned} \tag{1.9}$$



(a) 电路原理 1　　(b) 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

$$
\begin{aligned}
& x^4 + x^3 + x^2 + x = 120 - -(a) \\
& \Updownarrow \\
& \left.\begin{array}{l} s_1 = x * x \\ s_2 = s_1 * x \\ s_3 = s_2 * x \\ s_4 = s_1 + x \\ s_5 = s_4 + s_2 \\ 120 = s_5 + s_3 \end{array}\right\} - -(b) \\
& \Updownarrow \\
& \left.\begin{array}{c} s_1 = x * x \\ s_2 = s_1 * x \\ 120 - s_1 - x = s_2 * x \end{array}\right\} - -(c)
\end{aligned}
\tag{1.10}
$$

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）。

1.3　witness 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}

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

1.3.1　*witness* 等价于向量 $\vec{s}$

本节将 *witness* 即方程的解 $x = 3$（也可以是哈希函数原象 x，Merkle 树的叶子和节点、ECDSA 中的消息 M、签名 σ 和公钥 Q）等价转化为向量 $\vec{s}$。

命题 1：*方程的解 $x = 3$ 与向量 $\vec{s}$ 等价。对约束等式1.10的 (c)，有以下等价关系*

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

证明：令 $\vec{s} = [1, out, x, s_1, s_2]$。已知 $x = 3$，能够根据公式1.10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。*out* 是方程的计算结果 120。

对约束等式1.10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：隐私数据（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*。

在实际应用时，为防止证明方 $\mathcal{P}$ 欺骗验证方 $\mathcal{V}$，不能将所有数据隐藏起来，需要公开局部参数。所以，$1, out$ 需要公开，x, s_1, s_2, s_3 隐藏。因此，将向量 $\vec{s}$ 拆为公开数据与隐私数据。公开数据记为 $statement = (1, out)$，隐私数据记为 $witness = (x, s_1, s_2, s_3)$。这里的 *out* 是方程的计算结果 120。对于 Layer2 技术，*out* 则是 Merkle 根。

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

基于推论9与核心构造，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。

1.3.2 *witness* 满足 R1CS 约束等价于向量内积

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：

$$\left\{\begin{array}{l}\vec{s} \cdot [0,0,0,1,0] = [1, out, x, s_1, s_2] \cdot [0,0,0,1,0] = s_1 \\ \vec{s} \cdot [0,0,1,0,0] = [1, out, x, s_1, s_2] \cdot [0,0,1,0,0] = x \\ \vec{s} \cdot [0,0,1,0,0] = [1, out, x, s_1, s_2] \cdot [0,0,1,0,0] = x\end{array}\right\}$$

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：

$$\begin{array}{c} s_1 = x * x \\ \Updownarrow \\ \vec{s} \cdot w_1 = \vec{s} \cdot u_1 * \vec{s} \cdot v_1 \end{array} \tag{1.11}$$

同理，第 2,3,4 个阶为 1 的等式（R1CS 约束）

$$s_2 = s_1 * x$$

$$120 - (s_2 + s_1 + x) = s_2 * x$$

有如下进行向量内积转化：

$$\left\{\begin{array}{l}\vec{s} \cdot [0,0,0,0,1] = [1, out, x, s_1, s_2] \cdot [0,0,0,0,1] = s_2 \\ \vec{s} \cdot [0,0,0,1,0] = [1, out, x, s_1, s_2] \cdot [0,0,0,1,0] = s_1 \\ \vec{s} \cdot [0,0,1,0,0] = [1, out, x, s_1, s_2] \cdot [0,0,1,0,0] = x\end{array}\right\}$$

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

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：

$$\begin{gathered}s_2=s_1*x\\ 120-(s_2+s_1+x)=s_2+s_1\\ \Updownarrow\\ \vec{s}\cdot w_2=\vec{s}\cdot u_2*\vec{s}\cdot v_2\\ \vec{s}\cdot w_3=\vec{s}\cdot u_3*\vec{s}\cdot v_3\end{gathered}\tag{1.12}$$

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。

1.3.3　向量内积等价转化为向量与矩阵的内积

将等式1.11和等式1.12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：

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

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。

1.4 向量与矩阵内积的等价转化

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

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

已知多项式的值 $f_0,...,f_m$ 和横坐标 $x_0,...,x_m$，可以计算出多项式的系数 $a_0,...,a_m$；计算方法包括解方程组、拉格朗日插值法、离散傅里叶变换等。反之，已知多项式的系数 $a_0,...,a_m$ 和横坐标 $x_0,...,x_m$，可以计算出多项式的值 $f_0,...,f_m$。因此，多项式值表达等价于多项式系数表达。

1.4.1 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，如对矩阵 W

$$w_0(1)=0,w_1(1)=0,\ w_2(1)=0,\ \ w_3(1)=1,\ \ w_4(1)=0$$
$$w_0(2)=0,w_1(2)=0,\ w_2(2)=0,\ \ w_3(2)=0,\ \ w_4(2)=1$$
$$w_0(3)=0,w_1(3)=1,w_2(3)=-1,w_3(3)=-1,w_4(3)=-1$$

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 个变量，如 $1,2,3$，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

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

则多项式 $W(x) = [w_0(x), w_1(x), w_2(x), w_3(x), w_4(x)]$，以此类推，可以计算出多项式

$$U(x) = [u_0(x), u_1(x), u_2(x), u_3(x), u_4(x)]$$
$$V(x) = [v_0(x), v_1(x), v_2(x), v_3(x), v_4(x)]$$

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 $w_i(x), u_i(x), v_i(x), i = 0, ..., m$ 组合运算。

1.4.2　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。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定理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

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

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**

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

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 $w_i(x),u_i(x),v_i(x),i=0,...,m$ 的组合运算等价转换为目标多项式 $z(x)$ 整除 QAP 多项式 $\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$。

上述等式中的目标多项式 $z(x)$ 和多项式 $w_i(x),u_i(x),v_i(x),i=0,...,m$ 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 $z(x)$、阶小于或等于 n 的三组多项式 $w_i(x),u_i(x),v_i(x),i=0,...,m$，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

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

该问题构成 NP 问题。

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。验证方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 协议的核心等价关系：
（六）目标多项式 $z(x)$ 整除 QAP 多项式 $\vec{s}\cdot W(x)-\vec{s}\cdot U(x)*\vec{s}\cdot V(x)$ 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（多项式承诺），形成离散对数困难问题。
（七）证明方 $\mathcal{P}$ 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（多项式承诺）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 $\mathcal{V}$ 能够基于椭圆曲线离散对数点（多项式承诺）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 *witness* 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，叶子与节点满足 Merkle 树），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。因此，用户交易会成功。

1.5　zk-SNARK 协议框架

zk-SNARK 协议包括以下 10 步骤：

步骤 0：（初始化 a）通过安全多方计算生成与电路无关的系统参数 *CRS*。

步骤 1：证明方 $\mathcal{P}$ 证明拥有 *witness* 且 *witness* 满足任意计算关系 R。

步骤 2：证明方 $\mathcal{P}$ 证明拥有 *witness* 且 *witness* 满足任意多项式时间算法。

步骤 3：证明方 $\mathcal{P}$ 证明拥有 *witness* 且 *witness* 满足 R1CS 约束。

步骤 4：向量 $\vec{s}$ 与矩阵 U, V, W 的内积 $\vec{s} \cdot W = \vec{s} \cdot U * \vec{s} \cdot V$。

步骤 5：向量 $\vec{s}$ 对多项式进行组合运算 $\vec{s} \cdot W(x) - \vec{s} \cdot U(x) * \vec{s} \cdot V(x) = 0$。

步骤 6：目标多项式 $z(x)$ **整除** QAP 多项式 $z(x) \Big| [\vec{s} \cdot W(x) - \vec{s} \cdot U(x) * \vec{s} \cdot V(x)]$

构成 NP 问题

步骤 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 *CRS*。

步骤 8：将 QAP 多项式、目标多项式、商多项式放到椭圆曲线离散对数点上。

步骤 9：验证方 $\mathcal{V}$ 在椭圆曲线离散对数点上重构整除关系，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椭圆曲线的离散对数上，验证方在椭圆曲线点上重构整除关系，实现正确性验证，却不知道多项式的系数（或不知道解向量 $\vec{s}$），即零知识。
2. 计算复杂度很高的任意多项式时间算法，等价转化为 R1CS 约束、矩阵、多项式。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椭圆曲线离散对数点上（多项式承诺），所以证明方 $\mathcal{P}$ 的计算复杂度很高；而验证方仅需要使用椭圆曲线点重构整除关系，则实现向量 $\vec{s}$ 的正确性验证，所以计算复杂度较低。

算法评价

1：Groth16 的 CRS 包含 R1CS 约束等价转化来的多项式 $W(x), U(x), V(x)$，非常具体。

优点： 证明方 $\mathcal{P}$ 直接使用 CRS 中的多项式 $W(x), U(x), V(x)$ 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，只能表达唯一电路，不能表达其他电路，所以表达能力极差。如果对 layer2 的电路进行修改，则步骤 7 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 R1CS 约束多项式，而只包含大量的椭圆曲线离散对数点。

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。线约束计算复杂度很高，导致证明生成缓慢。

1.6 Groth16 协议详解

1.6.1 协议原理

初始化： 选择随机数 $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{Z}_p^*$，计算系统参数 CRS

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

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

注意：x 是具体值，所以多项式 $u(x), v(x), w(x)$ 是具体的多项式值；σ_1, σ_2 中与多项式 $u(x), v(x), w(x)$ 无关的项在步骤 0 完成，与电路多项式 $u(x), v(x), w(x)$ 相关的项在步骤 7 完成。注意：此处的多项式 $u(x), v(x), w(x)$ 就是上一节讨论的多项式 $W(x), U(x), V(x)$。

证明： 证明方 $\mathcal{P}$ 选择随机数 $r, s \leftarrow \mathbb{Z}_p$，基于向量 $\vec{s} = (a_1, ..., a_m)$ 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。因此，证明为 $([A]_1, [B]_2, [C]_1)$

$$[A]_1 = \left(\alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta \right) \cdot G_1$$

$$[B]_2 = \left(\beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta \right) \cdot G_2$$

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

验证：验证方 $\mathcal{V}$ 对椭圆曲线离散对数点重构整除关系

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

如果等式成立，则验证成功，否则拒绝。

1.6.2　协议分析

(一)、一致性

$$Left: A=\alpha+\sum_{i=0}^{m}a_iu_i(x)+r\delta$$

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

$$A\cdot B=\left(\alpha+\sum_{i=0}^{m}a_iu_i(x)+r\delta\right)\cdot\left(\beta+\sum_{i=0}^{m}a_iv_i(x)+s\delta\right)$$

$$=\alpha\beta+\alpha s\delta+r\delta\beta+rs\delta^2+\alpha\cdot\sum_{i=0}^{m}a_iv_i(x)+\beta\sum_{i=0}^{m}a_iu_i(x)+\sum_{i=0}^{m}a_iu_i(x)\sum_{i=0}^{m}a_iv_i(x)+s\delta\sum_{i=0}^{m}a_iu_i(x)+r\delta\sum_{i=0}^{m}a_iv_i(x)$$

$$Right:$$

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

$$=\alpha\beta+(As)\,\delta+(rB)\,\delta-rs\delta^2+\alpha\cdot\sum_{i=0}^{l}a_iv_i(x)+\beta\sum_{i=0}^{l}a_iu_i(x)+\sum_{i=0}^{l}a_i\omega_i(x)+\alpha\cdot\sum_{i=l+1}^{m}a_iv_i(x)+\beta\sum_{i=l+1}^{m}a_iu_i(x)+\sum_{i=l+1}^{m}a_i\omega_i(x)+h(x)z(x)$$

$$=\alpha\beta+(As)\,\delta+(rB)\,\delta-rs\delta^2+\alpha\cdot\sum_{i=0}^{l}a_iv_i(x)+\beta\sum_{i=0}^{l}a_iu_i(x)+\sum_{i=0}^{l}a_i\omega_i(x)+\alpha\cdot\sum_{i=l+1}^{m}a_iv_i(x)+\beta\sum_{i=l+1}^{m}a_iu_i(x)+\sum_{i=l+1}^{m}a_i\omega_i(x)+h(x)z(x)$$

$$=\alpha\beta+(As)\,\delta+(rB)\,\delta-rs\delta^2+\alpha\cdot\sum_{i=0}^{m}a_iv_i(x)+\beta\sum_{i=0}^{m}a_iu_i(x)+\sum_{i=0}^{m}a_i\omega_i(x)+h(x)z(x)$$

$$=\alpha\beta+\left(s\delta\alpha+s\delta\sum_{i=0}^{m}a_iu_i(x)+rs\delta^2\right)+\left(r\delta\beta+r\delta\sum_{i=0}^{m}a_iv_i(x)+rs\delta^2\right)-rs\delta^2+\alpha\cdot\sum_{i=0}^{m}a_iv_i(x)+\beta\sum_{i=0}^{m}a_iu_i(x)+\sum_{i=0}^{m}a_i\omega_i(x)+h(x)z(x)$$

(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 $\sum_{i=0}^{m}a_iu_i(x)\sum_{i=0}^{m}a_iv_i(x)=\sum_{i=0}^{m}a_i\omega_i(x)+h(x)z(x)$ 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 $\sum_{i=0}^{m}a_iu_i(x)\sum_{i=0}^{m}a_iv_i(x)-\sum_{i=0}^{m}a_i\omega_i(x)$，即证明方知道该 NP 问题的解。

(三)、证明方无法作弊：

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

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

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

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 α，在计算 B 的时候需要诚实包含 β，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

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

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

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

证明方只要修改任意一个参数，都无法使得验证等式成立

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

原理分析

1. α 和 β 迫使证明 A, B, C 必须采用同一套向量 $statement = a_0, ..., a_m$ 参数，而不能是其他参数。反之：如果 A 采用第 1 套参数 $a_0, ..., a_m$ 参数。B 采用第 2 套参数 $a_0', ..., a_m'$，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 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，让对方计算不出 $witness$。

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 γ, δ。添加 γ, δ 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 $witness = a_{l+1}, ..., a_m$。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。

$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta = \alpha + \sum_{i=0}^{l} a_i u_i(x) + \sum_{i=l+1}^{m} a_i u_i(x) + r\delta$$

$$B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta = \beta + \sum_{i=0}^{l} a_i v_i(x) + \sum_{i=l+1}^{m} a_i v_i(x) + s\delta$$

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

5. **线性关系**：证明方构造出的证明 A, B, C 与 CRS 呈线性关系，所以线性证明。因为 CRS 是系统参数，所以证明方不需要与验证方交互，因此是非交互式线性证明系统。

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 $z(x)$。

$Simulator$:

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

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

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

$Verifier$:

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

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

$$= \alpha \cdot \beta + \sum_{i=0}^{l} \alpha_i \left(\beta u_i(x) + \alpha v_i(x) + \omega_i(x)\right) + AB - \alpha\beta - \sum_{i=0}^{l} a_i \left(\beta u_i(x) + \alpha v_i(x) + \omega_i(x)\right)$$

$$= AB$$

验证通过。但是，该验证过程不要求 $\sum_{i=0}^{m} a_i u_i(x) \sum_{i=0}^{m} a_i v_i(x) = \sum_{i=0}^{m} a_i \omega_i(x) + h(x)z(x)$ 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。通常理解为拥有陷门，则能够构造证明，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 $\mathcal{P}$。

1. 模拟器 $\mathcal{S}$ 没有知识，验证方使用模拟器 $\mathcal{S}$ 提供的证明计算不出任何知识。
2. 从验证方角度，模拟器 $\mathcal{S}$ 与证明方 $\mathcal{P}$ 一模一样（分布不可区分）。
3. 因此，验证方基于证明方 $\mathcal{P}$ 生成的证明，也计算不出任何知识，实现零知识。

参 考 文 献

[1] Goldwasser S., Micali S., Rackoff C. The knowledge complexity of interactive proof systems. SIAM Journal on computing, 1989, 18(1) : 186 – 208.

[2] Lund C., Fortnow L., Karloff H., et al. Algebraic methods for interactive proof systems. Journal of the ACM (JACM), 1992, 39(4) : 859 – 868.

[3] Groth J. Short pairing-based non-interactive zero-knowledge arguments. International Conference on the Theory and Application of Cryptology and Information Security. 2010 : 321 – 340.

[4] Matuszek D. https://www.seas.upenn.edu/ cit596/notes/dave/p-and-np2.html. . 1996.

[5] Hankerson D., Menezes A J., Vanstone S. Guide to elliptic curve cryptography. Springer Science & Business Media, 2006.

[6] 180-4 F. SHA256. https://www.movable-type.co.uk/scripts/sha256.html. .

[7] Matuszek D. https://www.seas.upenn.edu/ cit596/notes/dave/p-and-np3.html. . 1996.

[8] Cook S A. P-NP problems. https://en.wikipedia.org/wiki/P_versus_NP_problem. .

[9] Cook S A. The complexity of theorem-proving procedures. Proceedings of the third annual ACM symposium on Theory of computing. 1971 : 151 – 158.

[10] Gennaro R., Gentry C., Parno B., et al. Quadratic span programs and succinct NIZKs without PCPs. Annual International Conference on the Theory and Applications of Cryptographic Techniques. 2013 : 626 – 645.

[11] Ludlum R. The Sigma Protocol. Macmillan, 2002.

[12] Parno B., Howell J., Gentry C., et al. Pinocchio: Nearly practical verifiable computation. 2013 IEEE Symposium on Security and Privacy. 2013 : 238 – 252.

[13] Groth J. On the size of pairing-based non-interactive arguments. Annual international conference on the theory and applications of cryptographic techniques. 2016 : 305 – 326.

[14] Johnson D., Menezes A., Vanstone S. The elliptic curve digital signature algorithm (ECDSA). International journal of information security, 2001, 1(1) : 36 – 63.

[15] Jack Schwartz R Z. Schwartz-Zippel lemma. https://en.wikipedia.org/wiki/Schwartz-Zippel_lemma. .

[16] kilic. Multiset-checks. https://hackmd.io/@arielg/ByFgSDA7D. .

[17] kilic. Plookup for SHA256. https://hackmd.io/xfgP5_uMTZyaEJJG4EJoRQ?view. .

[18] Sin7Y. Custom Gates. https://zhuanlan.zhihu.com/p/375786894. .

[19] Reitwiesner G W. Non Adjacent Form(NAF). https://en.wikipedia.org/wiki/Non-adjacent form. .

[20] Straka M. Recursivesnarks. https://www.michaelstraka.com/posts/recursivesnarks/. .

[21] Chiesa A., Tromer E. Proof-Carrying Data and Hearsay Arguments from Signature Cards. ICS : Vol 10. 2010 : 310 – 331.

[22] Ben-Sasson E., Chiesa A., Tromer E., et al. Scalable zero knowledge via cycles of elliptic curves. Algorithmica, 2017, 79(4) : 1102 – 1160.

[23] Bertoni G., Daemen J., Peeters M., et al. Keccak. Annual international conference on the theory and applications of cryptographic techniques. 2013 : 313 – 314.

[24] Gabizon A., Williamson Z J. plookup: A simplified polynomial protocol for lookup tables. IACR Cryptol. ePrint Arch, 2020, 2020 : 315.