zk-SNARK 为什么以及如何工作：明确的解释

马克西姆·佩特库斯
maksym@petkus.info

抽象的

尽管存在多个伟大的资源zk-SNARK建筑，来自原始论文[1]给解释者[2]，由于移动部件的数量庞大，该主题对许多人来说仍然是一个黑匣子。虽然给出了一些拼图，但如果没有缺失的部分，则无法看到完整的画面。

因此，这项工作的重点是通过基于示例的简单明了的方法阐明该主题，并在此过程中回答许多原因，以便更多人能够欣赏最先进的技术、其创新者以及最终的数学之美.

论文的贡献是一个简单的阐述，具有足够的并逐渐增加的复杂性，有必要理解zk-SNARK没有任何学科、密码学或高等数学的先决知识。主要目标不仅是解释它是如何工作的，而且是解释它为什么起作用以及它是如何形成的。

关键词：零知识证明、SNARK、隐私、可验证计算。

[1]位+11；标准杆+13。
[2]丽16；但是16；但是17；加布17。

内容

0 前言

虽然最初计划很短，但该作品现在跨越了几十页，但它需要的先决知识很少，并且可以自由地跳过熟悉的部分。

如果您不熟悉一些使用过的数学符号，请不要担心，会有一些，它们会逐渐介绍，一次一个。

1 简介

零知识简洁的非交互式知识论证（zk-SNARK）是在不透露任何其他信息的情况下证明某事是真实的真正巧妙的方法，但是，为什么它首先是有用的?

零知识证明在无数应用中具有优势，包括：

- 关于私有数据的证明声明：
  - 人一个有超过X在他的银行账户里
  - 去年，银行未与实体进行交易是
  - 在不显示完整 DNA 的情况下匹配 DNA
  - 一个人的信用评分高于Z
- 匿名授权：
  - 证明请求者R有权在不透露其身份（例如登录名、密码）的情况下访问网站的受限区域
  - 证明一个来自允许的国家/州列表，而无需透露具体来自哪一个
  - 证明一个人拥有地铁/地铁的月票而不透露卡的ID
- 匿名付款：
  - 完全脱离任何身份的支付[3]
  - 在不透露收入的情况下纳税
- 外包计算：
  - 外包昂贵的计算并验证结果是否正确，无需重新执行；它开辟了一种去信任计算的范畴
  - 将区块链模型从每个人的计算都更改为单方计算，每个人都进行验证

[3]本+14。

尽管表面上听起来很棒，但底层方法是数学和密码学的“奇迹”，自 1985 年在主要著作“交互式证明系统的知识复杂性”[GMR85] 中引入以来，已经进行了第四个十年的研究随后引入了非交互式证明 [BFM88]，这在区块链的背景下尤为重要。

在任何零知识证明系统，有一个证明者谁想说服一个验证者那一些陈述在不透露任何其他信息的情况下为真，例如，验证者得知证明者有超过X在他的银行帐户中，但没有其他内容（即未披露实际金额）。一个协议应该满足三个属性：

- 完整性——如果陈述是真的，那么证明者可以说服一个验证者
- 健全——作弊证明者不能说服一个验证者虚假的陈述
- 零知识——交互只揭示一个陈述是真的，没有别的

这zk-SNARK术语本身是在 [Bit+11] 中引入的，基于 [Gro10] 并遵循 Pinocchio 协议 [Gen+12; Par+13] 使其适用于一般计算。

2 证明的媒介

让我们从简单开始，尝试证明一些东西，而不用担心零知识、非交互性、形式和适用性。

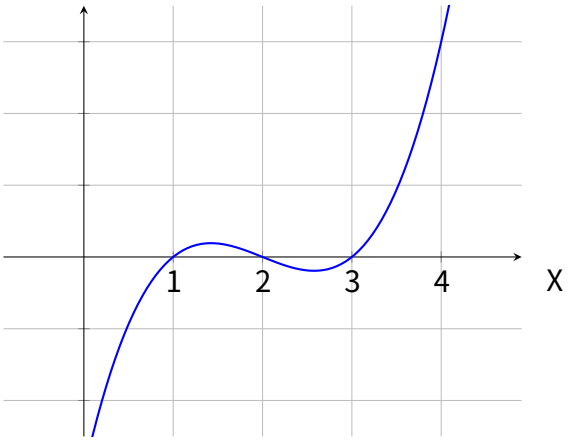
想象一下，我们有一个长度为 10 的位数组，我们想向验证者（例如，程序）证明所有这些位都设置为 1，即，我们知道一个数组，使得每个元素都等于 1。

b = [?, ?, ?, ?, ?, ?, ?, ?, ?, ?]

Hello Verifier 一次只能检查（即读取）一个元素。为了验证该语句，可以通过以任意顺序读取元素并检查它是否真的相等来继续
为 1，如果是，则在第一次检查后对该语句的置信度为1 10=10%，或声明
如果该位等于 0，则完全无效。验证者必须进行下一轮，直到他达到足够的置信度。在某些情况下，人们可能信任证明者并且只需要 50% 的置信度，这意味着必须执行 5 次检查，而在其他需要 95% 置信度的情况下，必须检查所有单元格。很明显，这种证明协议的缺点是必须根据元素数量进行检查，如果我们考虑数百万个元素的数组，这是不切实际的。

让我们考虑多项式，可以将其可视化为图形上的曲线，由数学方程形成：

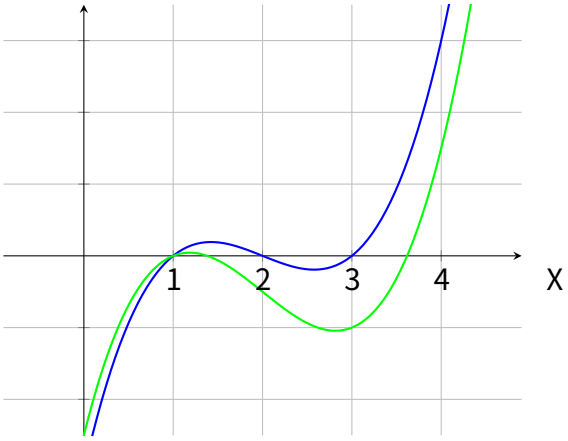
是的



上面的曲线对应于多项式：$f(x) = x_3\text{-}6X_2+11x\ \text{-}6$. 多项式的次数由它的最大指数决定X，在这种情况下是 3。

多项式有一个有利的性质，即如果我们最多有两个不相等的次数多项式d,他们可以相交不超过d点。例如，让我们稍微修改一下原来的多项式$X_3\text{-}6X_2+10x\ \text{-}5$并将其可视化为绿色：

是的



如此微小的变化会产生截然不同的结果。事实上，不可能找到两个不相等的多项式，它们共享一个连续的曲线块[4].

该属性源于寻找共享点的方法。如果我们想找到两个多项式的交集，我们需要将它们等同起来。例如，要找到多项式与X-轴（即，$f(x) =0$），我们等价于$X_3\text{-}6X_2+11x\ \text{-}6 = 0$，这样一个方程的解将是那些共享点：$x =1$、$x =2$和$x =3$，你也可以清楚地看到上图是这样，蓝色曲线与X-轴线。

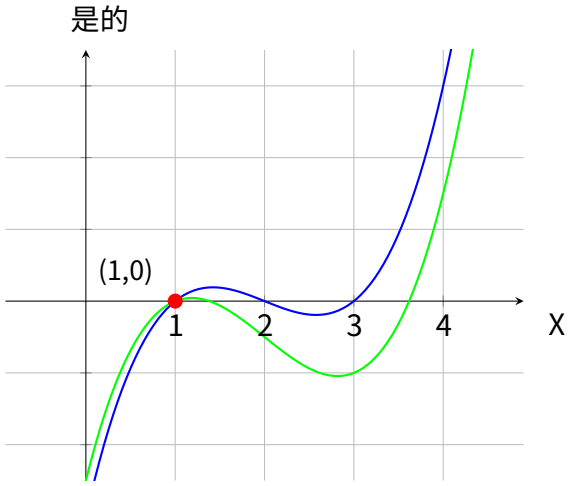
同样，我们可以将多项式的原始版本和修改版本等同起来，以找到它们的交集。

$$X_3\text{-}6X_2+11x\ \text{-}6 =X_3\text{-}6X_2+10x\ \text{-}5$$

$$x\ \text{-}1 = 0$$

[4]排除单点块情况

得到的多项式是 1 次的，有一个明显的解x =1. 因此只有一个交叉点：



任意度数的任何此类方程的结果dpolynomials 最多总是另一个多项式d,因为没有乘法来产生更高的度数。示例：$5X_3+7X_2-\times +2 = 3X_3-X_2+2x -5$，简化为 $2X_3+8X_2-3\times +7 = 0$。代数基本定理告诉我们，度数d多项式最多可以有d 解决方案[5]，因此至多d共享点。

因此我们可以得出结论，评估[6]任意点上的任何多项式的表示都类似于其唯一身份的表示。让我们评估我们的示例多项式x =10.

$$X_3-6X_2+11x -6 = 504\ X_3-6X_2+10x -5 = 495$$

事实上出于所有的选择X要评估，在这些多项式中最多只有 3 个选择具有相同的评估，而所有其他的都会不同。

这就是为什么如果证明者声称知道验证者也知道一些多项式（无论其次数有多大），他们可以遵循一个简单的协议来验证该语句：

- 验证者选择一个随机值X并在本地评估他的多项式
- 验证者给出X给证明者并要求评估有问题的多项式
- 证明者评估他的多项式X并将结果提供给验证者
- 验证者检查本地结果是否等于证明者的结果，如果是，则以高置信度证明该语句

例如，如果我们考虑一个整数范围X从 1 到 10_{77}，其中的点数评价不同是$10_{77}-d$。以后的概率X意外“击中”任何的d共享点等于 $\frac{d}{10_{77}}$，这被认为可以忽略不计。

[5]更多信息请参见第 3.2 节
[6]更多关于多项式评估：[Pik13]

注意：新协议只需要一轮，并给予压倒性的信心（几乎100% 假设d与低效的位检查协议相比，语句中的范围的上限足够小。

这就是为什么多项式是zk-SNARK，尽管也可能存在其他证明媒介。

3 多项式的非交互式零知识

3.1 证明多项式的知识

我们从证明多项式知识的问题开始，然后采用通用方法。在此过程中，我们将发现多项式的许多其他性质。

到目前为止的讨论集中在一个弱的证明概念上，即各方必须相互信任，因为还没有措施来执行协议的规则。例如，证明者不需要知道多项式，他可以使用任何其他可用的方法来得出正确的结果。此外，如果验证者的多项式评估的幅度不大，比如说 10，验证者可以猜测一个数字，并且它被接受的概率是不可忽略的。我们必须解决协议的这种弱点，但首先知道多项式意味着什么？多项式可以表示为以下形式（其中n是多项式的次数）：

$$C_nX_n+ ... +C_1X_1+C_0X_0$$

如果有人说他或她知道一个 1 次多项式（即，$C_1X_1+c_0$），这意味着真正的知道是系数C_0， C_1. 此外，系数可以具有任何值，包括 0。

假设证明者声称知道一个 3 次多项式，这样x =1 和x =2 是所有可能的解决方案中的两个。这样的有效多项式之一是X_3-3X_2+2x =0. 对于x =1：1 - 3 + 2 = 0。对于x =2：8 - 12 + 4 = 0。

让我们首先更仔细地看一下解决方案的解剖结构。

3.2 因式分解

代数基本定理指出，任何多项式都可以分解为线性多项式（即表示一条线的 1 次多项式），只要它是可解的。因此，我们可以将任何有效多项式表示为其因子的乘积：

$$(x - 一个_0)(x - 一个_1)...(x - a_n) = 0$$

此外，如果这些因素中的任何一个为零，则整个方程为零，因此所有一个-s 是唯一的解决方案。

实际上，我们的示例可以分解为以下多项式：

$$X_3-3X_2+2x = (x -0)(x -1)(x -2)$$

解决方案是（的值X）：0, 1, 2，您可以在多项式的任何一种形式上轻松检查这一点，但因式分解的形式具有表面上的所有解（也称为根）。

回到证明者声称他知道根为 1 和 2 的 3 次多项式，这意味着他的多项式具有以下形式：

$$(x - 1)(x - 2) \cdot \ldots$$

换句话说 （x -1) 和 (x -2) 是所讨论多项式的辅因子。因此，如果证明者想要证明他的多项式确实有这些根，而不公开多项式本身，他需要证明他的多项式p(x)是这些辅因子的乘积t(x) = (x -1)(x -2），称为目标多项式，和一些任意多项式h(x) （等于x -0 在我们的例子中），即：

$$p(x) = t(x) \cdot h(x)$$

换句话说，存在一些多项式h(x)这使得t(x)等于p(x),所以 p(x)包含t(x)，所以p(x)有所有的根源t(x)，要证明的事情。

一种自然的寻找方式h(x)是通过除法$h(x) = \frac{p(x)}{t(x)}$.如果证明者找不到这样的h(x)这意味着p(x)没有必要的辅助因子t(x)，在这种情况下，多项式除法将有余数。

在我们的例子中，如果我们划分$p(x) = x^3 - 3x^2 + 2x$由$t(x) = (x-1)(x-2) = x^2 - 3x + 2$：

$$\begin{array}{r} \\ x^2 - 3x + 2 \;\big)\; x^3 - 3x^2 + 2x \\ -\,(x^3 - 3x^2 + 2x) \\ \hline 0 \end{array} \quad \begin{array}{l} x \\ \\ \\ \\ \end{array}$$

注：分母在左，结果在右上，余数在下[7].

我们得到了结果h(x) = x无余。

注意：为简单起见，我们将使用多项式的字母变量来表示其评估，例如，p = p(r)

使用我们的多项式身份检查协议，我们可以比较多项式p(x)和t(x) · h(x)：

- 验证器对随机值进行采样r,计算t = t(r) (即，评估）并给出r给证明者
- 证明者计算$h(x) = \frac{p(x)}{t(x)}$并评估p(r)和h(r)；结果值p, h是提供给验证者
- 验证者然后检查p = t · H，如果是这样，这些多项式是相等的，这意味着p(x) 有t(x)作为辅因子。

[7][Pik14] 上提供了带有示例的多项式除法解释

为了将其付诸实践，让我们为我们的示例执行此协议：

$$p(x) = x_3\text{-}3X_2+2xt(x) =$$
$$(x\text{ -}1)(x\text{ -}2)$$

- 验证者采样一个随机值 23，计算t = t(23) = (23 − 1)(23 − 2) = 462 并给证明者 23
- 证明者计算h(x) =$_{p(x)}$ ~~t(x)~~=X，评估p = p(23) = 10626 和h = h(23) = 23 和提供p, h给验证者
- 验证者然后检查p = t・H：10626 = 462・23, 这是真的, 因此这个陈述被证明了

相反，如果证明者使用不同的p'（X）没有必要的辅因子，例如p'(x) =2X$_3$-3X$_2$+2X，然后：

$$h(x) = \quad 2\times +3$$
$$X_2\text{-}3\times +2 \;)\overline{\;2X_3\text{-}3X_2+2X\;}$$
$$\underline{-\;\;2X_3+6X_2\text{-}4X}$$
$$3X_2\text{-}2X$$
$$\underline{-\;\;3X_2+9x\text{ -}6}$$
$$7x\text{ -}6$$

我们将获得 2x+3，其余 7x−6，即：p(x) = t(x)×(2x+3)+7x−6. 这意味着证明者必须将余数除以t(r)为了评估h(x) =2x+3+$\frac{7x-6}{t(x).}$
因此，由于随机选择X由验证者，有一个低[8]余数的评估概率为 7x -6 将被评估整除t(x)，今后，如果验证者将另外检查p和H必须是整数，这样的证明将被拒绝。然而，检查要求多项式系数也是整数，这对协议造成了很大的限制。

这就是引入密码原语的原因，即使原始评估碰巧是可分的，也无法进行这种划分。

注意：虽然作者的主要目标是简单性，包括使用的数学符号集，但省略普遍存在的符号素数对于进一步的部分是有害的：'.它的基本目的是表示原始变量或函数的某种转换或推导，例如，如果我们想乘v经过2并将其分配给一个单独的变量，我们可以使用素数：v'=2・五。

备注 3.1现在我们可以在不学习多项式本身的情况下检查多项式的特定属性，因此这已经为我们提供了某种形式的零知识和简洁性。尽管如此，这种结构存在多个问题：

[8]但依然不容小觑

- 证明者可能不知道声称的多项式p(x)一点也不。他可以计算评价 t = t(r),选择一个随机数H并设置p = t・H，这将被验证者接受为有效，因为等式成立。

- 因为证明者知道随机点x = r,他可以构造任何具有一个共享点的多项式r和t(r)・h(r)。

- 在原始声明中，证明者声称知道特定次数的多项式，在当前协议中没有执行次数。因此，证明者可以通过使用也满足辅因子检查的更高次多项式来作弊。

我们将在以下部分解决所有问题。

3.3 模糊评价

注释 3.1 的前两个问题是可能的，因为值是以原始形式呈现的，证明者知道r 和t(r)。如果将这些值作为黑匣子给出，那将是理想的，这样人们就不能对协议进行调整，但仍然能够计算对这些模糊值的操作。类似于散列函数的东西，这样在计算时很难回到原始输入。

3.3.1 同态加密

这正是同态加密的设计目的。也就是说，它允许加密一个值并能够对这种加密应用算术运算。实现加密的同态属性有多种方式，我们简单介绍一种。

总体思路是我们选择一个基地[9]自然数G （说5）并加密我们取幂的值G到那个值的力量。例如，如果我们要加密数字 3：

$$5^3=125$$

其中 125 是 3 的加密。如果我们想将这个加密数字乘以 2，我们提出
它是 2 的指数：

$$125^2=15625 = 5^{(3\,2)}=5^{2\times 3}=5^6$$

我们能够将一个未知值乘以 2 并对其进行加密。我们还可以通过乘法将两个加密值相加，例如 3 + 2：

$$5^3 \cdot 5^2=5^{3+2}=5^5=3125$$

类似地，我们可以通过除法减去加密数字，例如 5 − 3：

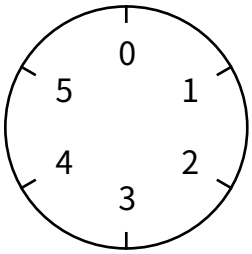
$$\frac{5^5}{} = 5^5 \cdot 5^{-3}=5^{5-3}=5^2=25\ 5^3$$

[9]基数需要具有某些属性

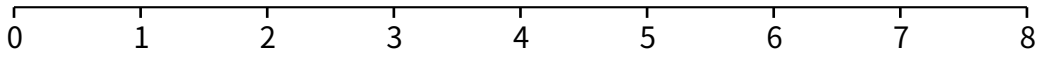
但是，由于基数 5 是公开的，所以很容易回到秘密数字，将加密除以 5 直到结果为 1。步数就是秘密数字。

3.3.2 模运算

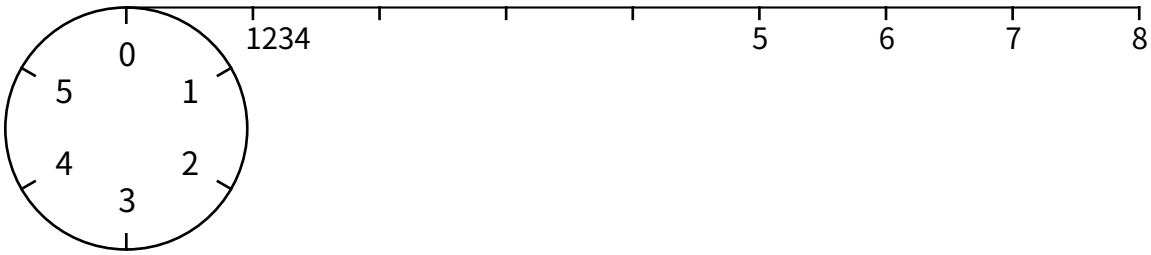
这就是模运算发挥作用的地方。模算术的思想如下：我们声明我们只选择第一个而不是无限的数字集n 自然数，即 0, 1, . . . ,n -1，要使用，如果任何给定的整数超出此范围，我们将“环绕”它。例如，让我们选择六个第一个数字。为了说明这一点，考虑一个有六个等单位刻度的圆；这是我们的范围[10].



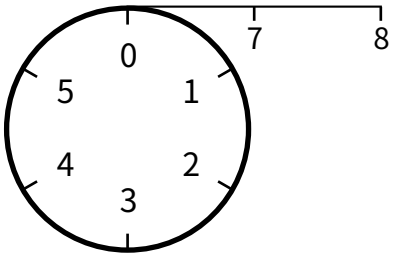
现在让我们看看数字八将落在哪里。打个比方，我们可以把它想象成一根绳子，它的长度是八个单位：



如果我们将绳子连接到圆圈的开头

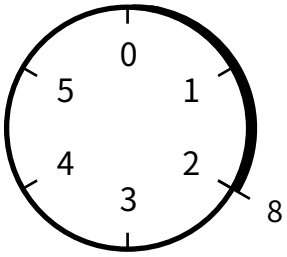


并开始将绳子缠绕在它周围，旋转一圈后，我们还剩下一部分绳子：



因此，如果我们继续这个过程，绳子将在第 2 点结束。

[10]通常称为有限域



它是模运算的结果。不管绳子有多长，它总是会停在圆圈的刻度之一处。因此，模运算将使其保持在一定范围内（在本例中为 0 到 5）。15 个单位的绳索将在 3 处停止，即 6 + 6 + 3（两个完整的圆圈，剩余 3 个单位）。负数的工作方式相同，唯一的区别是我们以相反的方向包装它，对于 -8，结果将是 4。

而且，我们可以进行算术运算，结果总是在 n数字。我们将使用符号“modn”现在来表示数字的范围。例如：

$$3 \times 5 = 3 \quad \text{(模式 6)}$$
$$5 + 2 = 1 \quad \text{(模式 6)}$$

此外，最重要的属性是操作的顺序无关紧要，例如，我们可以先执行所有操作，然后再应用模数或在每个操作后应用模数。例如（$2 \times 4 - 1) \times 3 = 3 \pmod 6$ 等价于：

$$2 \times 4 = 2 \quad \text{(模式 6)}$$
$$2 - 1 = 1 \quad \text{(模式 6)}$$
$$1 \times 3 = 3 \quad \text{(模式 6)}$$

那么到底为什么这有帮助呢？事实证明，如果我们使用模算术，得到一个运算结果，返回原始数字并非易事，因为许多不同的组合将具有相同的结果：

$$5 \times 4 = 2 \quad \text{(模式 6)}$$
$$4 \times 2 = 2 \quad \text{(模式 6)}$$
$$2 \times 1 = 2 \quad \text{(模式 6)}$$
$$\ldots$$

如果没有模算术，结果的大小为它的解决方案提供了线索。否则，这条信息会被隐藏，而常见的算术属性会被保留。

3.3.3 强同态加密

如果我们回到同态加密并使用模运算，例如模 7，我们将得到：

$$5^1=5 \quad (\text{模式 } 7)$$
$$5^2=4 \quad (\text{模式 } 7)$$
$$5^3=6 \quad (\text{模式 } 7)$$
$$\ldots$$

不同的指数会有相同的结果：

$$5^5=3 \quad (\text{模式 } 7)$$
$$5^{11}=3 \quad (\text{模式 } 7)$$
$$5^{17}=3 \quad (\text{模式 } 7)$$
$$\ldots$$

这是它得到的地方难的找到指数。事实上，如果模数足够大，这样做就变得不可行，而现代密码学的很大一部分都是基于这个问题的“难度”。

该方案的所有同态属性都保留在模块化领域中：

$$\text{加密：} \quad 5^3=6 \quad (\text{模式 } 7)$$
$$\text{乘法：} \quad 6^2=(5^3)^2=5^6=1 \quad (\text{模式 } 7)$$
$$\text{添加：} \quad 5^3 \cdot 5^2=5^5=3 \quad (\text{模式 } 7)$$

注意：模块化划分有点复杂，超出了范围。

让我们明确说明加密函数：$E(v) = g^v$(模组n),在哪里v是我们要加密的值。

备注 3.2这种同态加密方案存在局限性，虽然我们可以将加密值乘以未加密值，但不能将两个加密值相乘（和除），也不能对加密值求幂。虽然第一印象很不幸，但这些属性将成为 zk-SNARK 的基石。限制在第 3.6.1 节中讨论

3.3.4 加密多项式

有了这样的工具，我们现在可以评估一个具有加密随机值的多项式 X并修改零知识相应的协议。

让我们看看如何评估多项式$p(x) = x^3-3X^2+2X$。正如我们之前建立的，要知道多项式就是要知道它的系数，在这种情况下，它们是：1、-3、2。因为同态加密不允许对加密值取幂，所以我们必须得到加密的幂值的X从 1 到 3：E(x), $E(x^2)$， 前任^3)，这样我们

可以如下评估加密多项式：

$$(G^{X_3})^1 \cdot (G^{X_2})^{-3} \cdot G^X)^2 = G^{X_3} \cdot G^{X_2} \cdot G^X = G^{1X_3} \cdot G^{-3X_2} \cdot G^{2X} = G^{X_3-3X_2+2X}$$

作为这些操作的结果，我们对我们的多项式进行了加密评估，我们不知道X。这是一个非常强大的机制，并且由于同态特性，相同多项式的加密计算在加密空间中总是相同的。

我们现在可以更新以前版本的协议，获得多项式d:

- 验证者
  - 采样一个随机值年代，即秘密
  - 计算加密s为所有权力一世在 0, 1, ...,d,IE：$E(s_{一世}) = G_{s_{一世}}$
  - 评估未加密目标多项式和s: t(s)
  - 加密的权力s提供给证明者：$E(s_0), E(s_1), ..., E(s_d)$
- 证明者
  - 计算多项式$h(x) = \frac{p(x)}{t(x)}$
  - 使用加密权力G_{s_0}， $G_{s_1}, \ldots, G_{s_d}$和系数C_0， $C_1, \ldots,$ C_n评估
    $E(p(s)) = g_{p(s)} = G_{s_d}^{(\cdot)^C} \cdot \ldots \cdot G_{s_1}^{(\cdot)^{C_1}} \cdot G_{s^0}^{(\cdot)^{C_0}}$ 同样地$E(h(s)) = g_{h(s)}$
  - 所结果的G_p和G_H提供给验证者
- 验证者
  - 验证者的最后一步是检查$p = t(s) \cdot H$在加密空间中：
    $G_p = G_H^{(\cdot)^{t(s)}} \Rightarrow G_p = G_{t(s) \cdot H}$

注意：因为证明者对此一无所知年代，这使得很难提出不合法但仍然匹配的评估。

虽然在这样的协议中，证明者的敏捷性是有限的，但他仍然可以使用任何其他方式来伪造证明，而无需实际使用所提供的权力加密。年代，例如，如果证明者声称仅使用 2 次幂就有一个令人满意的多项式s_3和s_1，这在当前协议中无法验证。

3.4 限制多项式

多项式的知识就是其系数的知识C_0， $C_1, \ldots,$ $C_{一世}$和方式
我们在协议中“分配”这些系数是通过相应的取幂
秘密值的加密权力小号 (IE，埃斯$_{我知道了_{一世}}=G_{C_{一世}}$（）$\cdot s_{一世}$）。我们已经在选择加密权力时限制了证明者年代，但没有强制执行这种限制，例如，可以使用任何可能的方法来找到一些任意值z_p和z_H满足方程$z_p = (z_H)^{t(s)}$
并将它们提供给验证者而不是G_p和G_H. 例如，对于一些随机r$z_H = G_r$
和$z_p = G_{t(s)}$（）^{r}， 在哪里$G_{t(s)}$可以从提供的加密幂计算s。这就是为什么验证者需要仅提供权力加密的证明s被用来计算 G_p和G_H没有别的了。

让我们考虑一个具有一个变量和一个系数的 1 次多项式的基本示例f(x) = c・X和相应的加密s提供E(s) = g_s. 我们正在寻找的是确保只有加密年代，IE，G_s, 同态地“乘以”某个任意系数C没有别的了。因此结果必须始终为 $(G_s)^c$对于一些任意的C。

一种方法是要求对另一个执行相同的操作转移加密值与原始值一起，充当“校验和”的算术模拟，确保结果是原始值的取幂。

这是通过在 [Dam91] 中引入的指数知识假设（或 KEA）实现的，更准确地说：

- 爱丽丝有一个价值一个，她希望 Bob 对任何幂求幂[11]，唯一的要求是只有这个一个可以取幂，仅此而已，以确保她：
  - 选择一个随机 α
  - 计算一个'=一个$^{\alpha}$(模组n)
  - 提供元组 (一个，一个') 发送给 Bob 并要求对每个值执行相同的任意取幂并以结果元组 (乙，乙') 其中指数“α-shift”保持不变，即$b^{\alpha} = b'$(模组n)

- 因为 Bob 无法从元组中提取 α (一个，一个') 除了通过蛮力[12]
  这是不可行的，据推测 Bob 可以产生有效响应的唯一方法是通过以下过程：
  - 选择了一些值C
  - 计算b = $(a)^c$(模组n) 和 b' = $(一个')^c$(模组n)
  - 回复（乙，乙'）

[11]在哪里一个是使用的有限域群的生成器
[12]证明在原始论文中提供

- 得到响应和 α，Alice 检查相等性：

  (二)α=b'

  (一个c)α= (一个')c

  一个c · α= (一个α)c

- 结论：
  - Bob 应用了相同的指数（即，C）到元组的两个值
  - Bob 只能使用原始 Alice 的元组来维持 α 关系
  - 鲍勃知道应用指数C，因为产生有效（乙，乙') 是使用相同的指数
  - 爱丽丝还没学会C出于同样的原因，Bob 无法学习 α[13]

最终，这样的协议向 Alice 提供了 Bob 确实求幂的证明一个通过他已知的某个值，他不能进行任何其他操作，例如乘法、加法，因为这会消除 α 位移关系。

在同态加密上下文中，取幂是加密值的乘法。我们可以在简单的一系数多项式的情况下应用相同的结构 f(x) = c · X：

- 验证者随机选择s, α并提供评估x = 小号对于幂 1 及其“移位”： （Gs， Gα · s)
- 证明者应用系数丙： （（克s)c, (Gα · s)c) = (Gc · s， Gα · c · s)
- 验证者检查： （Gc · s)α=Gα · c · s

这样的建设限制证明者只使用加密的s提供，因此证明者可以指定系数C仅适用于验证者提供的多项式。我们现在可以缩放这样的一项多项式[14]多项式的方法，因为每个项的系数分配是单独计算的，然后同态地“相加”在一起（这种方法由 Jens Groth 在 [Gro10] 中引入）。所以如果给证明者加密的幂s和他们一起转移他可以评估原始多项式和移位多项式的值，其中必须进行相同的检查。特别是对于学位d多项式：

- 验证者提供加密权力Gs0， Gs1, . . . ， Gsd和他们的转变Gαs0， Gαs1, . . . ,Gαsd
- 证明者：
  - 评估（s en）cryp ted p)olynomia(l w)th 提供的权力年代：
    Gp(s)=Gs0 C0 • Gs1 C1 · . . . · Gsd Cd=GC0S0+C1S1+...+CdSd

[13]虽然C是加密它的可能值范围可能不足以保留零知识 属性将在第 3.5 节中讨论。

[14]单项式

– 评估加密的“移位”多项式与相应的幂的 α 位移年代：

$$G_{\alpha p(s)}=\left(G_{\alpha s^0}\right)^{c_0}\left(\right)\cdot G_{\alpha s^1}{}^{c_1}\cdot\ldots\cdot\left(\right)^{c}G_{\alpha s^d}=G_{c_0\alpha s^0+c_1\alpha s^1+\ldots+c_d\alpha s^d}=G_{\alpha(c_0s^0+c_1s^1+\ldots+c_ds^d)}$$

– 提供结果为G_p， $G_{p'}$给验证者

• 验证者检查：$(G_p)_\alpha=G_{p'}$

对于我们之前的示例多项式$p(x) = x^3-3X^2+2X$这将是：

• 验证者提供$E(s^3), E(s^2), E(s)$和他们的转变$E(\alpha s^3), E(\alpha s^2), E(\alpha s)$

• 证明者评估(tes:)$_1$

$$G_p=G_{p(s)}=\left(\right)\cdot G_{s^3}{}^{-3}\left(\right)\cdot G_{s^2}{}^{2}\cdot G_s=G_{s^3}\cdot G_{-3s^2}\cdot G_{2s}=G_{s^3-3s^2+2s}$$

$$G_{p'}=G_{\alpha p(s)}=\left(G_{\alpha s^3}\right)^{1}\left(\right)\cdot G_{\alpha s^2}{}^{-3}\left(\right)\cdot G_{\alpha s}{}^{2}=G_{\alpha s^3}\cdot G_{-3\alpha s^2}\cdot G_{2\alpha s}=G_{\alpha(s^3-3s^2+2s)}$$

• 验证者检查$(G_p)_\alpha=G_{p'}$：

$$\left(G_{s^3-3s^2+2s}\right)^{\alpha}=G_{\alpha(s^3-3s^2+2s)}$$

$$G_{\alpha(s^3-3s^2+2s)}=G_{\alpha(s^3-3s^2+2s)}$$

现在我们可以确定证明者没有使用除了验证者多项式提供的任何其他东西，因为没有其他方法可以保留 α-shift。此外，如果验证者希望确保排除某些权力s在证明者的多项式中，例如，j,他不会提供加密G_{s^j}及其转移$G_{\alpha s^j}$.

与我们一开始的相比，我们现在有了一个健壮的协议。但是，它仍然存在一个显着的缺点零知识属性，不管加密：而理论上多项式系数$C_{-世}$可以有很大范围的值，实际上，它可能非常有限（在前面的示例中为 6），这意味着验证者可以暴力破解有限范围的系数组合，直到结果等于证明者的答案。例如，如果我们考虑每个系数的 100 个值的范围，则 2 次多项式将总共有 100 万个不同的组合，考虑到蛮力将需要少于 100 万次迭代。此外，即使在只有一个系数且其值为 1 的情况下，安全协议也应该是安全的。

3.5 零知识

因为验证者可以提取关于未知多项式的知识$p(x)$仅从证明者发送的数据中，让我们考虑那些提供的值（证明）：G_p， $G_{p'}$,G_H. 他们参与以下检查：

$$G_p=G_H^{\left(\right)t(s)}\qquad\text{（多项式}p(x)\text{有根}t(x)\text{）}$$

$$(G_p)_\alpha=G_{p'}\qquad\text{（使用正确形式的多项式）}$$

问题是我们如何改变证明以使检查仍然有效，但无法提取任何知识？一个答案可以从上一节得出：我们可以将这些值“移动”一些随机数 δ (delta)，例如 $(G_p)_\delta$. 现在，为了提取知识，一个

首先需要找到被认为不可行的δ。此外，这种随机化在统计上与随机无法区分。

为了维持关系，让我们检查验证者的检查。证明者的一个值位于等式的每一侧。因此，如果我们用相同的 δ“移动”它们中的每一个，则方程必须保持平衡。

具体来说，证明者对随机 δ 进行采样，并用它对他的证明值求幂 $(G_{p(s)})_{\delta}$, $(G_{h(s)})_{\delta}$, $G_{\alpha p(s)\delta}$并向验证者提供验证：

$$(G_p)_{\delta} = G_H^{((\quad)\,t(s))}$$

$$(\quad)_{(G_p)_{\delta}}{}^{\alpha} = (\quad)_{G_{p'}}{}^{\delta}$$

合并后，我们可以观察到检查仍然成立：

$$G_{\delta \cdot p} = G_{\delta \cdot t(s)h}$$

$$G_{\delta \cdot \alpha p} = G_{\delta \cdot p'}$$

注意：零知识是多么容易被编织到构造中，这通常被称为“免费”零知识。

3.6 非交互性

至此，我们有一个交互式零知识方案。为什么会这样？因为证明只对原始验证者有效，所以没有其他人（其他验证者）可以信任相同的证明，因为：

- 验证者可以与证明者串通并泄露这些秘密参数s, α允许伪造证明，如备注 3.1 中所述
- 出于同样的原因，验证者可以自己生成假证明
- 验证者必须存储 α 和t(s)直到所有相关证明都得到验证，这允许额外的攻击面，可能会泄露秘密参数

因此，为了证明一个陈述（在这种情况下是多项式知识），需要与每个验证者进行单独的交互。

虽然交互式证明系统有其用例，例如当一个证明者只想说服一个专门的验证者（称为指定验证者[15]）使得该证明不能被重复用于向其他人证明相同的陈述，当一个人需要同时（例如，在区块链等分布式系统中）或永久地说服多方时，这是非常低效的。证明者将需要始终保持在线并为每个验证者执行相同的计算。

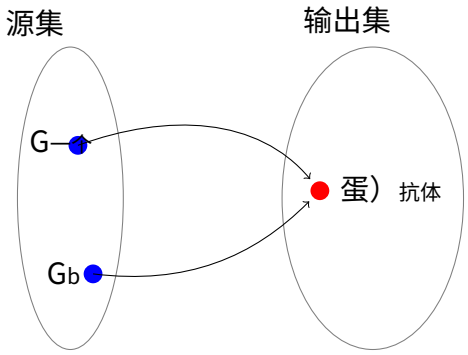
因此，我们需要秘密参数是可重用的、公开的、值得信赖的和不可滥用的。

[15] [JSI96] 中有关指定验证者的更多信息

让我们首先考虑如何保护秘密（t(s), α)在他们生产出来之后。我们可以像验证者加密s在发送给证明者之前。然而正如备注 3.2 中提到的，我们使用的同态加密不支持两个加密值的乘法，这对于两个验证检查来乘以加密t(s)和H也p和α。这就是密码配对适合的地方。

3.6.1 加密值的乘法

密码配对（双线性映射）是一种数学结构，表示为函数 例如*， G*），它给出了两个加密的输入（例如，G一个， Gb) 从一组数字允许将它们确定性地映射到它们在不同输出数字集中的乘法表示，即例如一个， Gb) =蛋）抗体：



.

因为源和输出数集[16]是不同的配对结果不能用作另一个配对操作的输入。我们可以将输出集（也称为“目标集”）视为来自“不同的宇宙”。因此，我们不能将结果乘以另一个加密值，并且名称本身表明我们一次只能将两个加密值相乘。

在某种意义上，它类似于一个散列函数，它将所有可能的输入值映射到一组可能的输出值中的一个元素，并且它不是平凡可逆的。

注意：乍一看，这种限制只能阻碍依赖功能，具有讽刺意味的是，在 zk-SNARK 案例中，它是方案安全性的最重要属性，请参见备注 3.3。

配对函数的基本（技术上不正确）数学类比例如*， G*) 将声明有一种方法可以“交换”每个输入的基数和指数，这样基数 G在转化为指数的过程中被修改，例如，G一个→一个G. 然后将两个“交换的”输入相乘，这样原始一个和b值在相同的指数下相乘，例如：

例如一个， Gb) =一个G • bG= (a)G

[16]通常称为组。

因此，因为在“交换”过程中使用结果（a)G在另一个配对中（例如，e ((ab)G,GC)) 不会产生所需的加密乘法美国广播公司。

配对的核心属性可以用以下等式表示：

例如一个， Gb) =例如b， G一个) =例如抗体， G1) =例如1， G抗体) =例如1， G一个)b=例如1， G1)抗体= 。. .

从技术上讲，配对的结果是不同生成器下原始值的加密产品G的目标集，即例如一个， Gb) =G抗体. 因此它具有同态加密的性质，例如，我们可以将多个配对的加密产品相加：

例如一个， Gb)・例如c， Gd) =G抗体・G光盘=Gab+cd=蛋） ab+cd

注意：密码配对是利用椭圆曲线来实现这些属性，因此从现在开始符号Gn将表示添加到自身的曲线上的生成点ntimes 而不是我们在前几节中使用的乘法组生成器。

调查 [DBS04] 为探索密码配对提供了一个起点。

3.6.2 可信方设置

有了加密配对，我们现在可以设置安全的公共和可重用参数。让我们假设我们信任一个诚实的一方来生成秘密s和α。立刻 α以及所有必要的权力s具有相应的 α 位移被加密（Gα， Gs一世,Gαs一世为了一世在 0, 1, . . . ,d),必须删除原始值。

这些参数通常称为公共参考字符串或 CRS。CRS 生成后，任何证明者和任何验证者都可以使用它来进行非交互式零知识证明协议。虽然不重要，但 CRS 的优化版本将包括对目标多项式Gt(s).

此外，CRS 分为两组（对于一世在 0, 1, . . . ,d):

- 证明密钥[17]: (Gs一世,Gαs一世)
- 验证码：（Gt(s)， Gα)

能够将加密值相乘，验证者可以在协议的最后一步检查多项式：

- 让验证密钥验证器进程收到加密多项式评估Gp， GH， Gp'
  来自证明者：
  - 检查 p = t・H 在加密空间中：
    ( ) ( )
    例如p， G1 =例如吨， GH 这相当于 蛋）p=蛋）吨・H
  - 检查多项式（mial r）限制：
    e (gp， Gα) =例如p',G

[17]也被称为评估键

3.6.3 信任中的一个

虽然可信设置是有效的，但它并不有效，因为 CRS 的多个用户将不得不信任一个删除的 α 和年代，因为目前没有办法证明[18]. 因此，有必要最小化或消除这种信任。否则，不诚实的一方将能够在不被发现的情况下制作假证据。

实现这一目标的一种方法是生成一个合成的多方使用前几节中介绍的数学工具进行 CRS，这样这些方都不知道秘密。这是一种方法，让我们考虑三个参与者 Alice、Bob 和 Carol，对应的索引为 A、B 和 C，对于一世在 1、2、. . . ,d:

- (虽)Alice 样本 he)r 随机s一个和α一个并发布她的 CRS：

  $G_{s_{一个}^{一世}}, G_{\alpha_{一个}^{一世}}, G_{\alpha_{一个}s_{一个}^{一世}}$

- 鲍勃采样他的s乙和α乙并通过同态增强 Alice 的加密 CRS 米重复：

  $\left(G_{s_{一个}^{一世}}, G_{\alpha_{一个}^{一世}}, G_{(\alpha_{一个})\alpha_{乙}^{一世}}\right), \left(G_{\alpha_{一个}s_{一个}^{一世}}, G_{\alpha_{一个}\alpha_{乙}^{一世}}\right) = G_{(小号_{一个}s_{乙})_{一世}}, G_{\alpha_{一个}\alpha_{乙}^{一世}}, G_{\alpha_{一个}\alpha_{乙}(小号_{一个}s_{乙})_{一世}}$

  a(nd) 发布结果的两方 Alice-Bob CRS：

  $G_{s_{AB}^{一世}}, G_{\alpha_{AB}^{一世}}, G_{\alpha_{ABs_{AB}^{一世}}}$

- 卡罗尔和她也是s

  $\left(G_{s_{AB}^{一世}}, G_{\alpha_{AB}^{一世}}\right)^{s_C}, \left(G_{\alpha_{ABs_{AB}^{一世}}}\right)^{\alpha_C} = G_{(小号_{一个}s_{乙}s_C)_{一世}}, G_{\alpha_{一个}\alpha_{乙}\alpha_C}, G_{\alpha_{一个}\alpha_{乙}\alpha_C(小号_{一个}s_{乙}s_C)_{一世}}$

  a(nd 发布 Alice-Bo)b-Carol CRS：

  $G_{s_{美国广播公司}^{一世}}, G_{\alpha_{美国广播公司}}, G_{\alpha_{美国广播公司}s_{美国广播公司}^{一世}}$

作为这种协议的结果，我们有复合$s_{一世}=s_{一世}$ $_{一个}s_{乙}s_{一世}^{一世}$c, 和 $\alpha = \alpha_{一个}\alpha_{乙}\alpha_C$和不参与者学习其他参与者的秘密参数，除非他们串通。其实为了学习s和α，必须与其他所有参与者勾结。因此，即使一个人是诚实的，也无法提供假证明。

注意：可以根据需要为尽可能多的参与者重复此过程。

可能存在的问题是如何验证参与者是否与 CRS 的每个值一致，因为对手可以采样多个不同的$s_1, s_2, \ldots$和$\alpha_1, \alpha_2, \ldots$,并将它们随机用于不同的权力小号 (或提供随机数作为增强的公共参考字符串），使 CRS 无效和不可用。

幸运的是，因为我们可以使用配对来乘以加密值，所以我们能够执行一致性检查，从第一个参数开始，并确保每个下一个参数都是从它派生的。参与者发布的每个 CRS 都可以检查如下：

- 我们取1的幂s作为规范值并检查所有其他幂是否与其一致：

[18]无知证明是一个积极研究的领域 [DK18]

例如$_{s-世,G}$ $=e\left(G_{s1}, G_{s我-1}\right)\Big|_{一世\in\{2,\ldots,d\}}$

例如：

- 电源2：$e\left(G_{s2}, G\right)=e\left(G_{s1}, G_{s1}\right)$ ⇒ 蛋）$_{s2}$=蛋）$_{s1+1}$
- 电源 3：$e\left(G_{s3}, G\right)=e\left(G_{s1}, G_{s2}\right)$ ⇒ 蛋）$_{s3}$=蛋）$_{s1+2}$,等等

- W(e now c)heck i(f the α-)s丨上一步中的值变化是正确的： 例如$_{s-世,G}$ $\alpha=e\left(G_{\alpha s-世,G}\right)\Big|_{一世\in[d]}$

例如：

- 电源 3：$e\left(G_{s3}, G_{\alpha}\right)=e\left(G_{\alpha s3}, G\right.$⇒蛋）$_{s3\cdot\alpha}$=蛋）$_{\alpha s3}$， ETC。

在哪里一世∈ {2, . . . ,d}是“一世位于 2, 3, . . . ,d”和 [d]是 1, 2, 的缩写形式。. . ,d,这是下一节更方便的符号

请注意，虽然我们验证每个参与者都与他们的秘密参数一致，但使用先前发布的 CRS 的要求并未对每个下一个参与者强制执行（在我们的示例中为 Bob 和 Carol）。因此，如果对手是链中的最后一个，他可以忽略先前的 CRS 并从头开始构造有效参数，就好像他是链中的第一个，因此是唯一知道秘密的人s和α。

我们可以通过额外要求除第一个参与者之外的每个参与者加密来解决这个问题
并发布他的秘密参数，例如，Bob 还发布：

$$\left(G_{s乙,}G_{\alpha乙,}G_{\alpha乙s-世\ 乙}\right)\Big|_{一世\in[d]}$$

这允许验证 Bob 的 CRS 是 Alice 参数的正确倍数，对于一世在
1, 2, . . . ,d:

- 例如$_{s-世B,}$ G $=$例如$_{s-个,}$ $G_{s-世乙}^{一世}$
- $e\left(g_{\alpha AB}, g\right)=e\left(g_{\alpha-个}, G_{\alpha乙}\right)$
- 例如$_{\alpha ABSAB,}$ $G^{一世}$ $=$例如$_{\alpha-个s-个一世,}G_{\alpha乙s乙}^{一世}$

同样，Carol 必须证明她的 CRS 是 Alice-Bob 的 CRS 的适当倍数。

这是一个强大的 CRS 设置方案，不完全依赖于任何一方。事实上，只要一方是诚实的并且删除并且从不共享其秘密参数就足够了，即使所有其他方都串通了。所以CRS设置中不相关的参与者越多[19]最微弱的假证明的可能性，如果竞争方参与，概率变得可以忽略不计。该方案允许涉及对设置的易读性有疑问的其他不受信任的方，因为验证步骤确保他们不会破坏（其中还包括使用弱 α 和s)最后的公共参考字符串。

[19]有时称为仪式 [Wil16]

3.7 多项式知识的简洁非交互论证

我们现在准备巩固进化的zk-SNARKOP协议。正式，为简洁起见，
我们将使用大括号来表示一组由旁边的下标填充的元素
例如，它$s_{-世}$ $\{\}_{-世\in[d]}$ 表示一个集合$s_1, s_2, \ldots, s_d$.

经商定目标多项式t(x)和学位d证明者多项式：

- 设置
  - 抽样随机值s, α
  - 计算加密G_α和 $\{\}$ $G_{s-世}$ $_{-世\in[d]}$ ，$\{\}$ $G_{\alpha s-世}$ $_{-世\in\{0,\ldots,d\}}$
  - 证明密钥： $(\{\}\{\})$ $G_{s-世}$ $_{-世\in[d]}$ ，$G_{\alpha s-世}$ $_{-世\in\{0,\ldots,d\}}$
  - 验证密钥： $(G_\alpha, G_{t(s)})$
- 证明
  - 分配系数 $\{C_{-世}\}_{-世\in\{0,\ldots,d\}}$(即知识），$p(x) = c_dX_d + \cdots + C_1X_1 + C_0X_0$
  - 计算多项式$h(x) = \frac{p(x)}{t(x)}$
  - 评估加密多项式$G_{p(s)}$和$G_{h(s)}$使用 $\{\}$ $G_{s-世}$ $_{-世\in[d]}$
  - 评估加密的移位多项式$G_{\alpha p(s)}$使用 $\{\}$ $G_{\alpha s-世}$ $_{-世\in\{0,\ldots,d\}}$
  - 样本随机 δ
  - 设置随机证明 $\pi = (G_{\delta p(s)}, G_{\delta h(s)}, G_{\delta\alpha p(s)})$
- 确认
  - 将证明 π 解析为$(G_p, G_H, G_{p'})$
  - 检查多项式限制 $e(G_{p'}, G) = e(g_p, G_\alpha)$
  - 检查多项式辅因子 $e(g_p, g) = e(G_{t(s)}, G_H)$

备注 3.3如果可以将配对结果重用于另一个乘法，那么这样的协议将是完全不安全的，因为证明者可以分配$G_{p'} = e(g_p, G_\alpha)$然后将通过“多项式限制”检查：

$$e(e(g_p, G_\alpha), g) = e(g_p, G_\alpha)$$

3.7.1 结论

我们来到了知识协议的零知识简洁非交互式参数，用于多项式问题的知识，这是一个利基用例。虽然可以声称证明者可以轻松构建这样的多项式p(x)只需乘以t(x)通过另一个有界多项式使其通过测试，该构造仍然有用。

验证者知道证明者有一个有效的多项式，但不知道是哪个特定的。我们可以添加多项式其他属性的额外证明，例如：除以多个多项式，是多项式的平方。可能存在接受、存储和奖励所有已证明多项式的服务，或者需要对必要形式的未知多项式进行加密评估。然而，拥有通用方案将允许无数的应用。

4个通用零知识证明

我们已经用一个简单但足够的例子铺平了道路，其中涉及了大部分zk-SNARK 机器，现在可以推进该计划以执行零知识程序。

4.1 计算

让我们考虑一个简单的伪代码程序：

```
算法 1操作取决于输入
  功能计算（w，a，b）
      如果w然后
          返回一个×b
      别的
          返回a + b
      万一
  结束函数
```

从高级的角度来看，它与多项式完全无关，我们有协议。因此，我们需要找到一种方法将程序转换为多项式形式。那么第一步就是把程序翻译成数学语言，比较简单，同样的语句可以表示如下（假设w是 0 或 1）：

$$f(w, a, b) = w(a \times b) + (1 - w)(a + b)$$

执行 calc(1, 4, 2) 并评估F（1, 4, 2) 将产生相同的结果： 8. 相反 calc(0, 4, 2) 和F（0, 4, 2) 都将被解析为 6。我们可以用这种方式表示任何类型的有限程序。

那么我们需要证明（在这个例子中），对于表达式的输入 (1, 4, 2) f(w, a, b)输出为 8，换句话说，我们检查相等性：

$$w(a \times b) + (1 - w)(a + b) = 8$$

4.2 单一操作

我们现在有了用数学语言表达的通用计算，但我们仍然需要将其转换为多项式领域。让我们仔细看看什么是计算

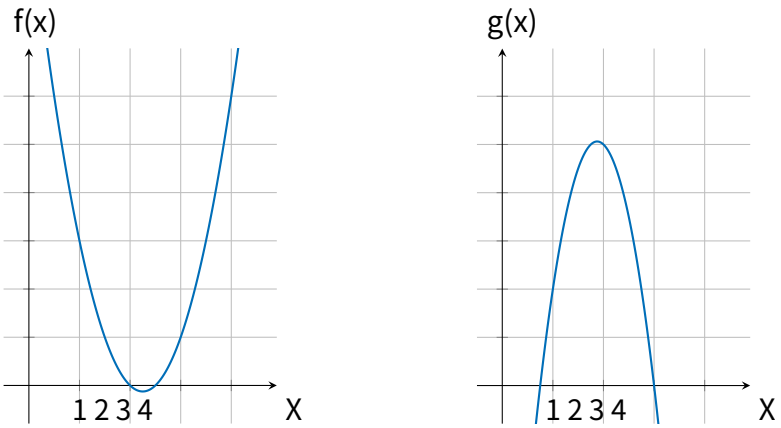
简而言之。其核心的任何计算都由以下形式的基本操作组成：

左操作数操作员右操作数 = 输出

两个操作数（即值）正在由一个运算符（例如，+、-、×, ÷)。 例如，操作数 2 和 3 以及运算 为了符“乘法”这些将解析为 2×3 = 6。因为任何复杂的计算（或程序）都只是一系列操作，首先我们需要找出多项式可以表示的单个操作。

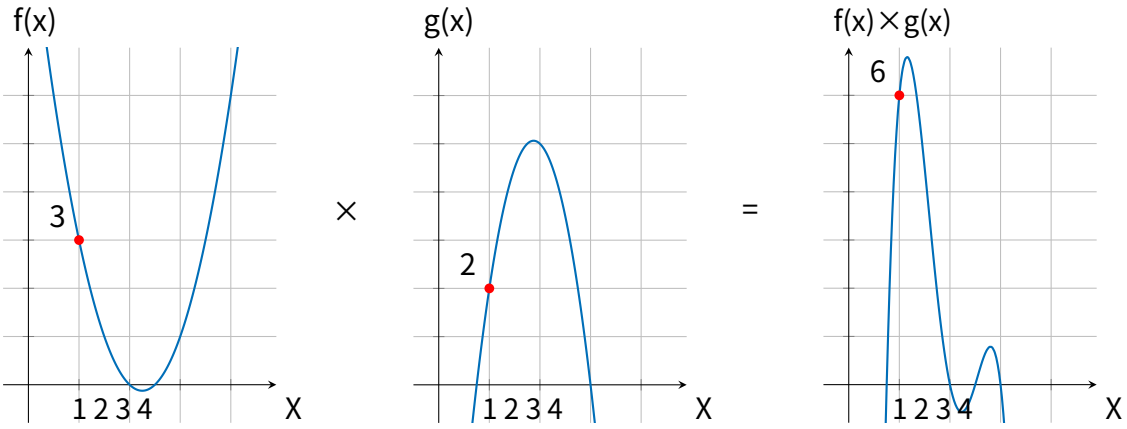
4.2.1 多项式的算术性质

让我们看看多项式是如何与算术运算相关的。如果你取两个多项式 f(x)和g(x)并尝试，例如，将它们相乘h(x) = f(x)×g(x),评估结果h(x)在任何x = r将是评估结果的乘积f(r)和g(r)。让我们考虑以下两个多项式：f(x) =2X2−9x+10 和g(x) = -4X2+15x−9、以图形的形式可视化：



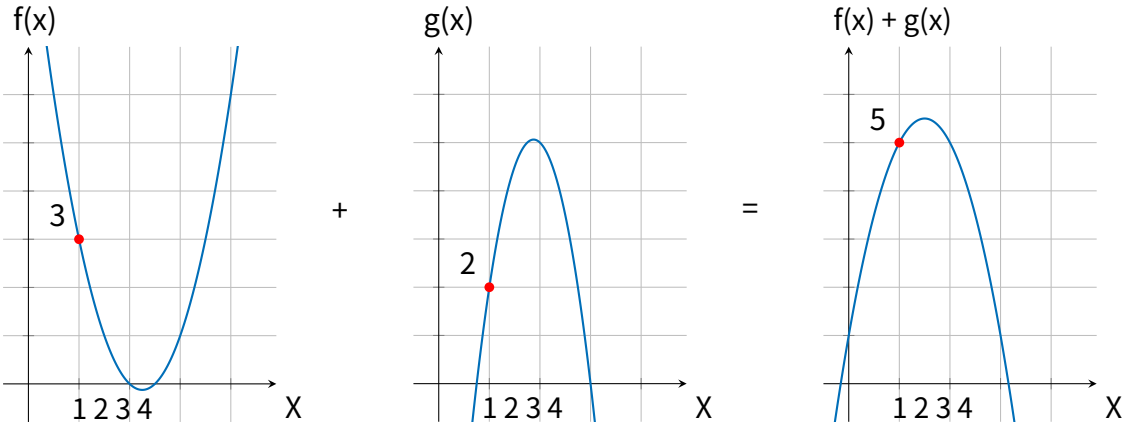
为了x =1 这些将评估为：F（1) = 2 - 9 + 10 = 3,G（1) = -4 + 15 - 9 = 2。

让我们将多项式相乘：h(x) = f(x)×g(x) = -8X4+66X3-193X2+231x -90. 视觉上的乘法可以看作：



如果我们检查评估x =1 在结果多项式上f(x)×g(x)我们将得到：H（1) =
− 8 + 66 − 193 + 231 − 90 = 6，因此值为x =1 个f(x)和g(x)已经成倍增加，并且分别在每隔一个X。

同样，如果我们添加f(x)和g(x)我们将得到 -2X2+6x+1 评估为 5 在x =1.



注：其他评价X-s 也被加在一起，例如，检查x =2、x =3.

如果我们可以将操作数值表示为多项式（我们确实可以按照概述），那么通过算术属性，我们将能够获得操作数所施加的操作的结果。

4.3 执行操作

如果证明者声称有两个数字相乘的结果，那么验证者如何检查呢？为了证明单个操作的正确性，我们必须对提供的操作数强制输出（结果）的正确性。如果我们再看一下操作的形式：

左操作数操作员右操作数 = 输出

同样可以表示为运算多项式：

l(x)操作员r(x) = o(x)

一些选择的地方A：

- l(x) -在一个表示（计算为）左操作数的值
- r(x) -在一个表示右操作数的值
- o(x) -在一个表示操作的结果（输出）

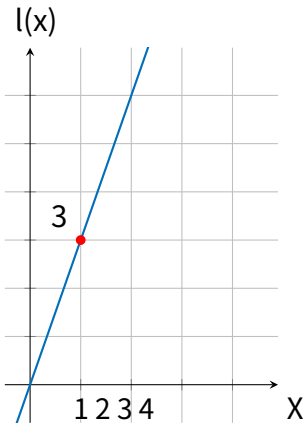
因此，如果这些多项式的运算正确地表示了操作数和输出，则计算升（一）操作员r(a) = o(a)应该持有。并且移动输出多项式牛）在等式的左边升（一）操作员r(a) - o(a) =0 浮出水面的事实是运算多项式l(x)操作员r(x) - o(x) =0 必须在

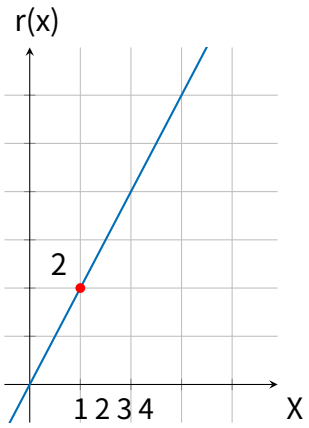
一个，如果由输出多项式牛）是由产生的正确结果 操作员在表示的值上操作数多项式l(x)和r(x)。今后运算多项式必须有根一个如果它是有效的，因此它必须包含辅因子 (x-a) 正如我们之前建立的（参见分解，第 3.2 节），这是目标多项式 我们证明反对，即t(x) = x - a。

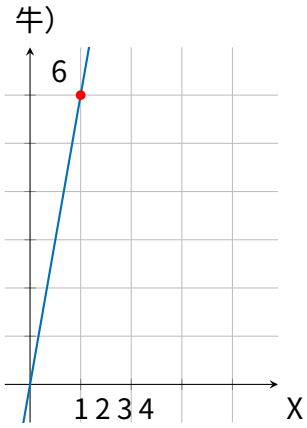
例如，让我们考虑操作：

$$3\times 2 = 6$$

它可以用简单的多项式表示l(x) =3x, r(x) =2x, o(x) =6X，评估为相应的值一个=1，即，l(1) = 3;r(1) = 2;o(1) = 6。







注：价值一个可以是任意的。

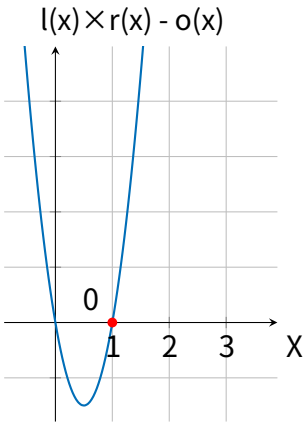
那么运算多项式将是：

$$l(x)\times r(x) = o(x)$$

$$3X\times 2x = 6X$$

$$6X_2\text{-}6x = 0$$

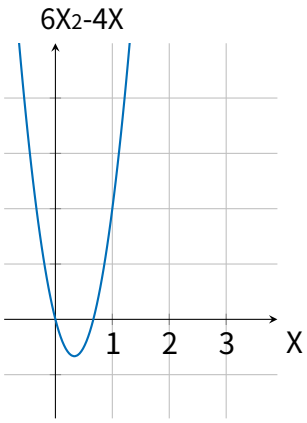
可视化为：



值得注意的是，运算多项式有 (x -1) 作为辅助因素：

$$6X_2\text{-}6x = 6x(x\text{ -}1)$$

因此，如果证明者提供这样的多项式l(x), r(x), o(x)而不是以前p(x)那么验证者将接受它为有效的，因为它可以被整除t(x)。相反，如果证明者试图作弊并将输出值替换为 4，例如，o(x) =4X，然后运算多项式 将是 $6X_2\text{-}4x = 0$：



哪个没有解决方案x =1、今后l(x)×r(x)-o(x)不能被t(x)没有余数：

$$h(x) = \begin{array}{r} 6\times +2 \\ x-1\ \overline{)\ 6X^2-4X} \\ -\ 6X^2+6X \\ \hline 2X \\ -\ 2\times +2 \\ \hline 2 \end{array} \quad \Rightarrow \quad h(x) = 6\times +2 + \frac{2}{x-1}$$

因此这样的不一致的操作不会被验证者接受[20].

4.4 操作证明

让我们修改我们最新的协议以支持单次乘法运算证明。回想一下，以前我们有多项式知识的证明p(x),但现在我们处理三个 l(x)、r(x)、o(x)。虽然我们可以定义p(x) = l(x)×r(x)-o(x)有两种反驳。首先，在我们的协议中，加密值的乘法（即，l(s)×r(s))在证明阶段是不可能的，因为配对只能使用一次，并且它是“多项式限制”检查所必需的。其次，这将为证明者留下一个机会，可以随意修改多项式的结构，但仍保持有效的辅因子t(x)，例如p(x) = l(x) 或者p(x) = l(x) - r(x)甚至p(x) = l(x)×r(x) + o(x),只要p(x)有根一个。这种修改实际上意味着证明是关于不同的陈述，这当然是不希望的。

这就是多项式求值的原因l(s), r(s), o(s)必须由证明者单独提供。这意味着多项式知识必须调整。本质上，验证者需要在加密空间中检查的是l(s)×r(s)-o(s) = t(s)h(s)。虽然验证者可以使用加密配对执行乘法，但减法（-牛）是一项昂贵的操作[21]这就是我们搬家的原因牛）等式右边：l(x)r(x) = t(x)h(x)+o(x)。

[20]如第 3.2 节所述

[21]需要找到的倒数$G_{o(s)}$

在加密空间中，验证者的检查转化为：

例如l(s)， Gr(s)) (=例如t(s)， Gh(s)) (e Go(s)， G)

蛋）l(s)r(s)=蛋）t(s)h(s)・蛋）o(s)

蛋）l(s)r(s)=蛋）t(s)h(s)+o(s)

注意：回想一下，密码配对的结果支持通过乘法进行的加密加法，请参见第 3.6.1 节。

虽然设置阶段保持不变，这是更新后的协议：

- 证明
  - 将相应的系数分配给l(x), r(x), o(x)
  - 计算多项式h(x) =l(x)×r(x)-o(x) / t(x)
  - 评估加密多项式Gl(s),Gr(s),Go(s)和Gh(s)使用 { } Gs一世 一世∈[d]
  - 评估加密的移位多项式Gαl(s),Gαr(s),Gαo(s)使用 { } Gαs一世 一世∈{0,...,d}
  - 设置证明 π =Gl(s)， Gr(s)， Go(s)， Gh(s)， Gαl(s)， Gαr(s)， Gαo(s) ( )
- 确认
  - 将证明 π 解析为Gl， Gr， G○， GH， Gl',Gr',G○' ( )
  - 多项式限制检查： 例如l',g) = e(gl， Gα) 例如r',g) = e(gr， Gα) 例如○',g) = e(g○， Gα)
  - 有效操作检查：例如l， Gr=例如t(s)， GH・e (g○， G) ( ) ( )

这样的协议允许证明两个值相乘的结果是正确计算的。

人们可能会注意到，在更新后的协议中，我们不得不放弃零知识零件。这样做的原因是为了使过渡更简单。我们将在后面的部分中回到它。

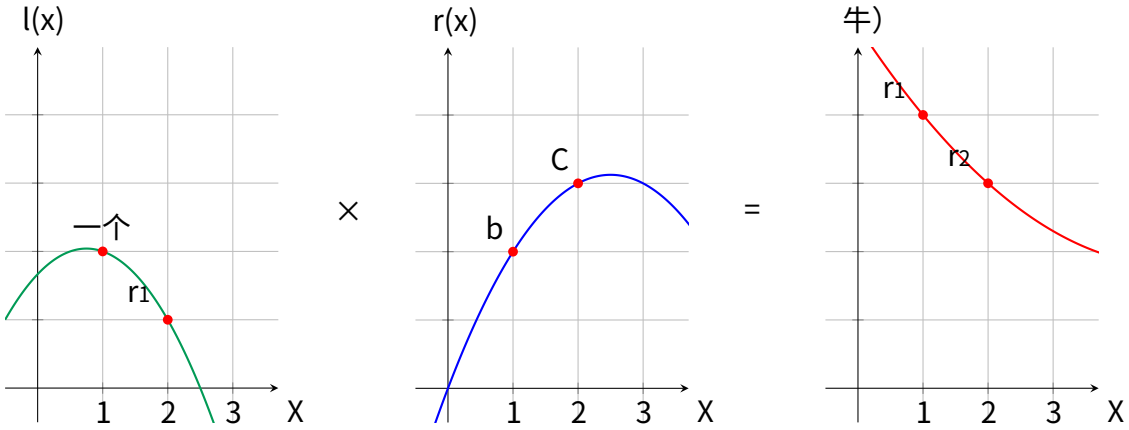
4.5 多重操作

我们可以证明单个操作，但是我们如何扩展以证明多个操作（这是我们的最终目标）？让我们尝试添加另一个操作。考虑计算产品的需要：一个×b×C。在元素操作模型中，这意味着两个操作：

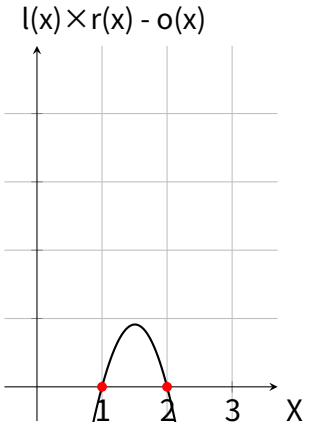
一个× b = r1

r1 × C = r2

如前所述，我们可以通过使操作数多项式在某个任意值处计算为相应值来表示这样的操作X，例如1. 多项式的性质并不限制我们以不同的方式表示其他值X，例如2，例如：



这种独立性使我们能够执行一次进行两个操作，而不会将它们“混合”在一起，即没有干扰。这种多项式算术的结果将是：



可见运算多项式有根x =1 和x =2.因此这两个操作都是执行正确。

让我们看一下 3 乘法 2 的示例×1×3×2、可以按如下方式执行：

$$2 \times 1 = 2$$
$$2 \times 3 = 6$$
$$6 \times 2 = 12$$

我们需要将它们表示为操作数多项式，这样对于由X∈ {1, 2, 3}l(x)相应地通过2, 2和6, 即通过点 (1, 2), (2、 2), (3、 6)，类似地r(x)∋ (1、 1), (2、 3), (3、 2） 和牛）∋ (1、 2), (2、 6), (3、 12）。

但是，我们如何找到通过这些点的多项式呢？对于我们有多个点的任何情况，都必须使用特定的数学方法。

4.5.1 多项式插值

为了构建操作数和输出多项式我们需要一种方法，它给定一组点产生一个弯曲多项式以这样的方式通过所有这些点，它被称为插值有不同的方法可用：

- 具有未知数的方程组
- 牛顿多项式
- 内维尔算法
- 拉格朗日多项式
- 快速傅里叶变换

让我们以前者为例。这种方法的思想是最多存在唯一的次数多项式n还未知系数通过给定的ñ +1 个点使得对于每个点 {(X—世, 是—世)}—世∈[n+1]多项式在X—世应该等于是的—世. 在我们的例子中，对于三点，它将是形式为 2 的多项式：

$$斧头^2+bx + c = y$$

让我们均衡的每个点的评估多项式左操作数多项式 (绿色）并通过用其他系数表达每个系数来求解方程组：

$$\begin{cases} l(1) = 2 \\ l(2) = 2 \\ l(3) = 6 \end{cases} \Rightarrow \begin{cases} 一个（1）^2+b・1+c =2 \\ 一个（2）^2+b・2 +c =2 \\ 一个（3）^2+b・3+c =6 \end{cases} \Rightarrow \begin{cases} a + b + c =2 \\ 4一个+2b + c =2 \\ 9一个+3b + c =6 \end{cases} \Rightarrow \begin{cases} a =2 -b - c \\ 2b =2 - 4(2 -b - c) - c \\ -c =6 - 9(2 -b - c) -3b \end{cases}$$

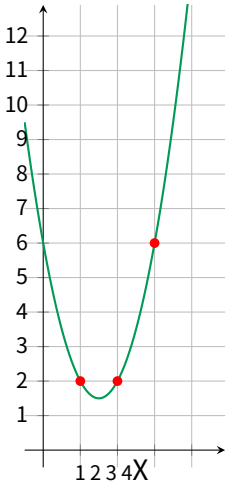
$$\Rightarrow \begin{cases} a =2 -b - c \\ b =\frac{6-3C}{2} \\ c = -12 + 6乙+9C \end{cases} \Rightarrow \begin{cases} a =2 -b - c \\ b =\frac{6-3C}{2} \\ -c = -12 + 6(\frac{6-3C}{2}) + 9C \end{cases} \Rightarrow \begin{cases} a =2 \\ b = -6 \\ -c =6 \end{cases}$$

因此左操作数多项式是：

$$l(x)=2X^2-6\times +6$$

对应于下图：

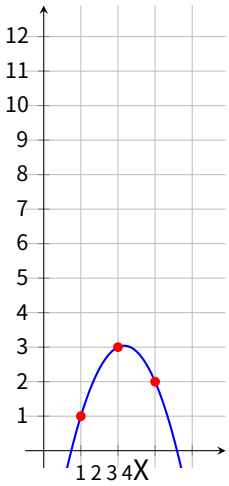
l(x)



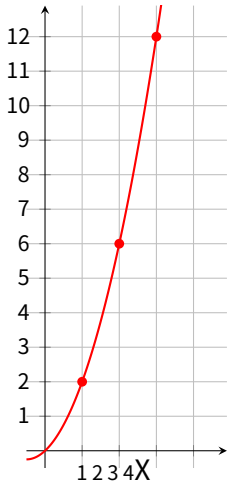
我们可以找r(x)和牛）以同样的方式：

$$r(x)= \frac{-3X_2+13x-8}{2}; \quad 牛) = x_2+X$$

r(x)

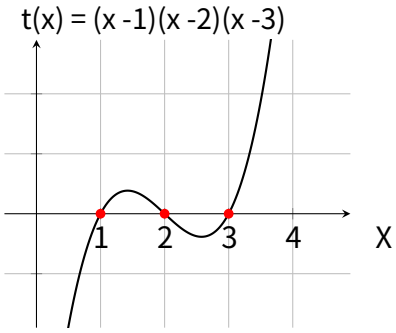


牛)

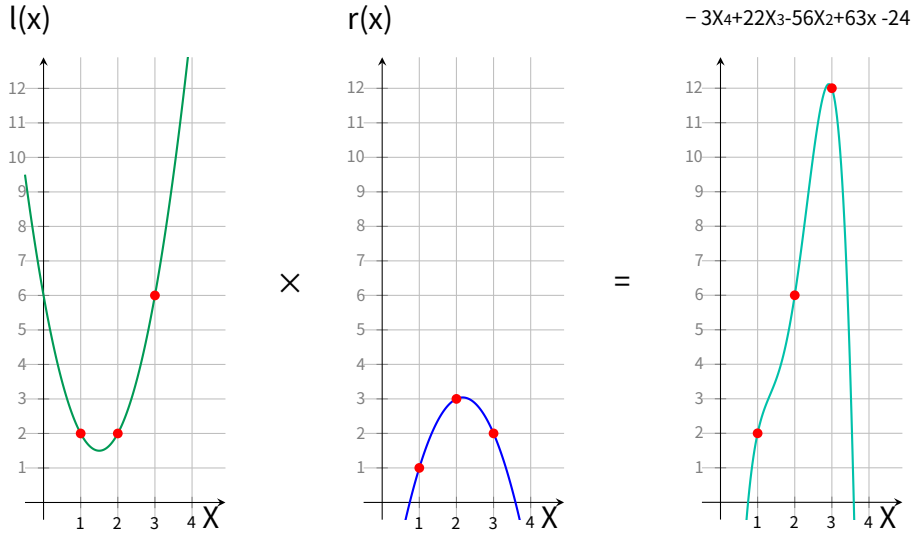


4.5.2 多操作多项式

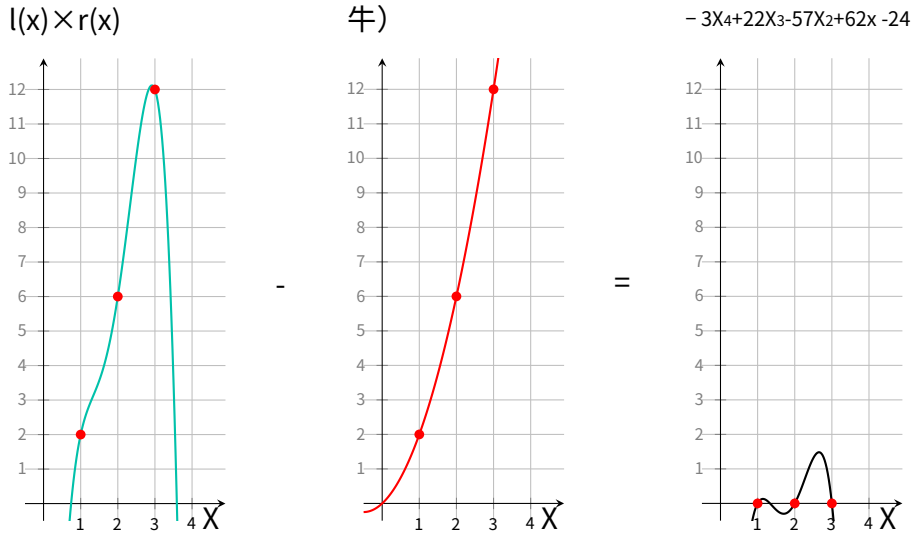
现在我们有了代表三个操作的操作数多项式，让我们一步一步地看看如何验证每个操作的正确性。回想一下，验证者正在寻找平等 l(x)×r(x) - o(x) = t(x)h(x)。在这种情况下，因为操作表示在点 X∈{1, 2, 3} 目标多项式必须在那些X-s，换句话说，t(x)必须是 1、2 和 3，其基本形式为：



首先，l(x)和r(x)相乘，结果为：



其次，牛）从结果中减去l(x)×r（x）：



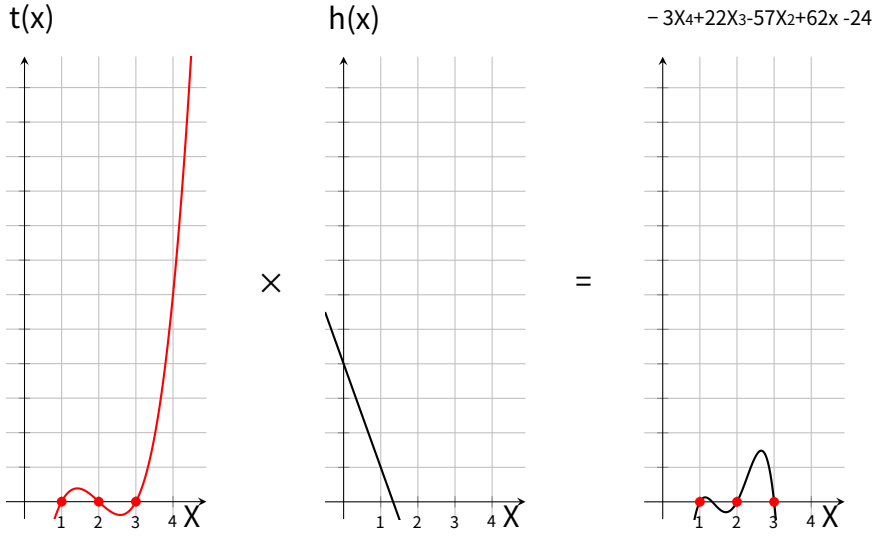
已经可以看出，每个操作数的乘法都对应一个正确的结果。对于最后一步证明者需要提供一个有效的辅因子：

$$h(x) = \frac{l(x) \times r(x) - o(x)}{t(x)} = \frac{-3X^4+22X^3-57X^2+62x -24\ (x -1)(}{x -2)(x -3)}$$

使用长除法我们得到：

$$h(x) = X^3-6X^2+11x-6 \;\Big)\overline{\begin{array}{r} -3\times+4 \\ -3X^4+22X^3-57X^2+62x-24 \\ X^3+33X^2-18X \\ \hline 4X^3-24X^2+44x-24 \\ -4X^3+24X^2-44\times+24 \\ \hline 0 \end{array}}$$

和h(x) = -3× +4个验证者可以计算t(x)h(x)：



现在很明显，l(x)×r(x) - o(x) = t(x)h(x)这是必须证明的。

4.6 可变多项式

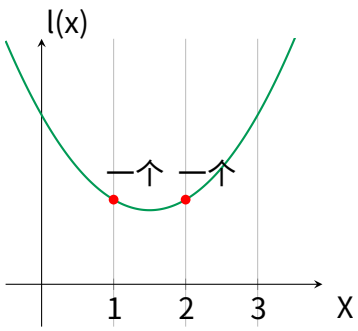
使用这种方法，我们可以一次证明许多操作（例如，数百万甚至更多），但它有一个严重的缺点。

如果正在证明执行的“程序”使用相同的多变的，在不同的操作中作为操作数或输出，例如：

一个×b=r_1

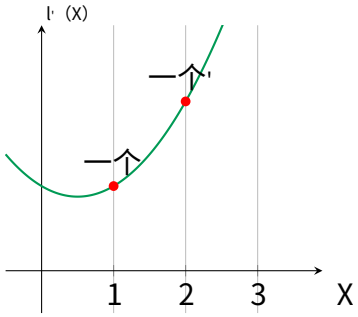
一个×C=r_2

这一个必须在左操作数多项式对于这两种操作：



尽管如此，因为我们的协议允许证明者将任何系数设置为多项式，所以他不受限制设置不同的值一个对于不同的操作（即，由一些代表

X），例如：



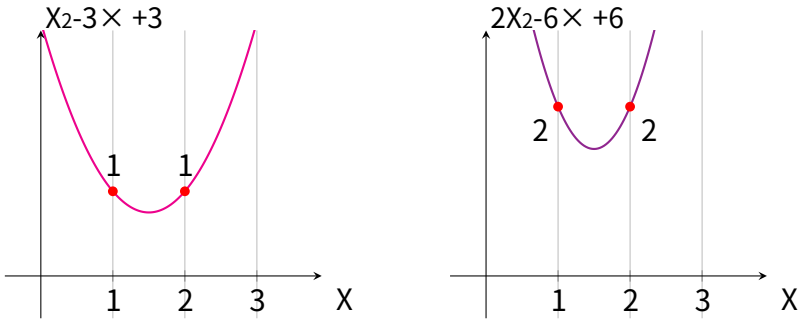
这种自由破坏了一致性，并允许证明者证明执行者不感兴趣的其他程序的执行。因此，我们必须确保任何变量在它所使用的每个操作中都只能有一个值。

注意：在这种情况下，变量与常规的计算机科学定义不同，它是不可变的，每次执行只分配一次。

4.6.1 单变量操作数多项式

让我们考虑一个简单的情况（如当前示例），其中我们只有一个变量（例如，一个）在所有左操作数中使用左操作数多项式I(x)。我们必须找出是否有可能确保这个多项式代表相同的值一个对于每个操作。证明者可以设置不同值的原因是他可以控制每个指数的每个系数X。因此，如果这些系数是恒定的，那将解决可变性问题。

让我们仔细看看包含相等值的多项式。例如，相应地检查两个表示两个操作的相等值的多项式（即，在x =1 和 x =2)，其中第一个多项式包含值 1，第二个包含值 2：



请注意，相应的系数在每个多项式中是成比例的，因此第二个中的系数是第一个中的两倍，即：

$$2X_2-6\times +6 = 2\times (X_2-3\times +3)$$

因此，当我们想同时改变多项式中的所有值时，我们需要改变它的比例，这是由于多项式的算术性质，如果我们将多项式乘以一个数字，则在所有可能的情况下进行评估X也将相乘（即缩放）。要验证，请尝试将第一个多项式乘以 3 或任何其他数字。

因此，如果验证者需要强制证明者在所有操作中设置相同的值，那么应该只能修改比例而不是单个系数。

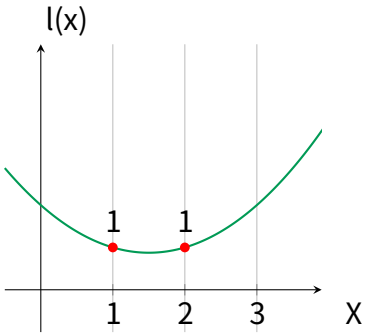
那么如何保持系数比例呢？我们可以从考虑提供什么作为证据开始左操作数多项式。这是一个加密的评估l(x)在某个秘密 年代：克l(s)，即它是一个加密的数字。从 3.4 节我们已经知道如何限制验证者只使用提供的指数s通过 α-shift，同态乘法是唯一可用的操作。

与限制单个指数类似，验证者可以一次限制整个多项式。而不是提供单独的加密Gs1， Gs2, . . . ,Gsd以及它们的 α 位移Gαs1， Gαs2, . . . ,Gαsd
议定书继续：

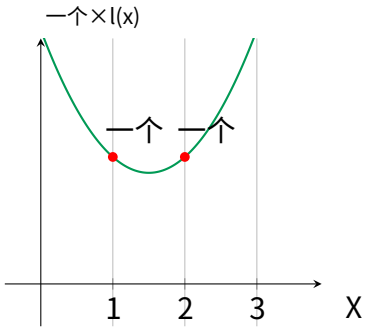
- 设置
  - 构造相应的操作数多项式l(x)具有相应的系数
  - 随机抽样 α 和s
  - 设置加密的证明密钥l(s)它是“移位”对：（Gl(s)， Gαl(s)）
  - 设置验证密钥：（Gα)
- 证明
  - 具有操作数的值v
    - *乘操作数多项式：（Gl(s)）
    - *乘法移位操作数多项式：（Gαl(s)）
  - 提供操作数多项式乘法证明：（Gvl(s)， Gv αl(s)）
- 确认
  - 将证明解析为（ ）Gl， Gl'
  - 验证比例：e（ ）Gl',G （ ）=e Gl， Gα

证明者需要以相同的 α-shift 响应，并且因为他无法从证明密钥中恢复 α，所以保持这种偏移的唯一方法是将两种加密相乘Gl(s)和Gαl(s)
由相同的值。因此证明者不能修改单个系数l(x),例如，如果l(x) = ax2+bx + c他只能一次将整个多项式乘以某个值v: v(轴2+bx + c) = vax2+vbx + vc。不能与另一个多项式相乘，因为 配对，和单个指数的 α 位移s不可用。证明者既不能加也不能减，因为Gα(l(x)+a'X2+c')6=Gαl(x) • G一个'X2 • GC'(这又需要知道未加密的α）。

我们现在有了协议，但是如何操作数多项式l(x)应该建造吗？由于任何整数都可以通过乘以 1 得出，因此对于每个相应的操作，多项式的计算结果应为 1，例如：



这允许证明者分配的价值A：



备注 4.1由于验证密钥包含Gα可以添加（或减去）任意值v'到多项式，即：

$$G^{vl(s)} \cdot G^{v'} = G^{vl(s)+v'}$$

$$G^{\alpha vl(s)} \cdot (G^{\alpha})^{v'} = G^{\alpha(vl(s)+v')}$$

例如$\left(\alpha(vl(s)+v'), G\right)\left(=e\right.$ $\left.G^{vl(s)+v'}, G^{\alpha}\right)$

因此，可以修改多项式超出验证者的预期并证明不同的陈述。我们将在 4.9.3 节中解决这个缺点。

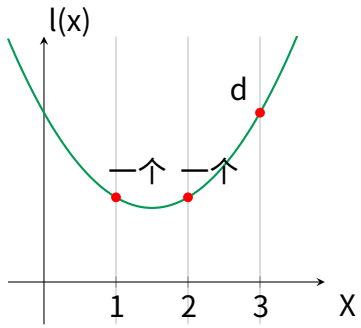
4.6.2 多变量操作数多项式

现在，只有当所有左操作数都使用相同的变量时，我们才能单独设置值。如果我们再添加一个怎么办d:

—个×b=r1

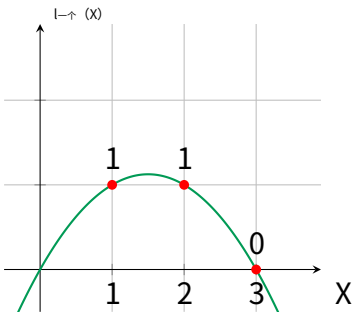
—个×C=r2

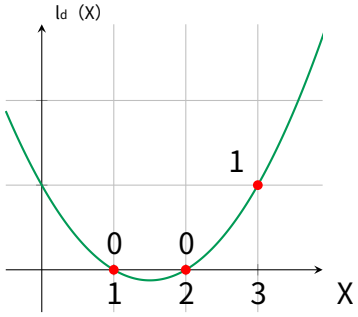
d×C=r3



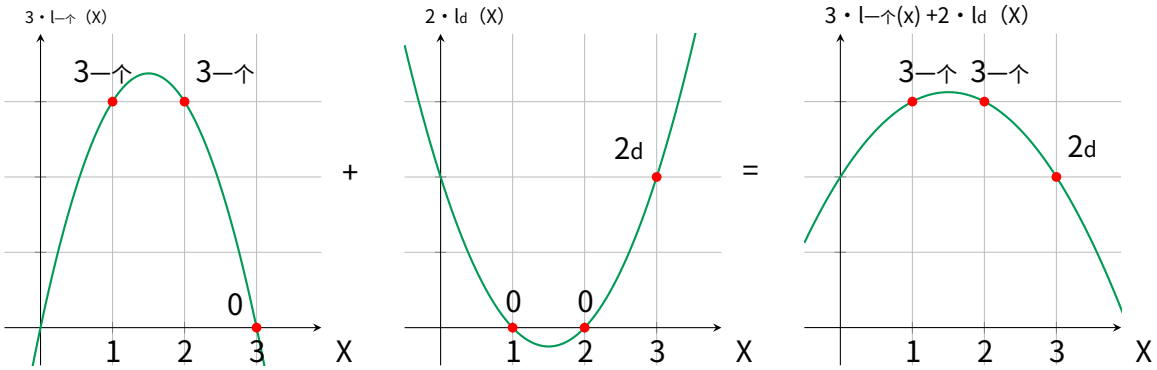
如果我们使用相同的方法，我们将无法为每个变量单独设置值，并且每个不同的变量将一起相乘。因此，这样的受限多项式只能支持一个多变的。如果我们检查多项式的性质，我们将看到将多项式相加会增加对这些多项式的不同评估。因此我们可以将操作数多项式l(x)进入操作数变量多项式l—个（X）和ld（X）（注意下标）这样变量一个和d是分配和上一节类似地分别限制，然后加在一起表示所有左操作数的变量。因为我们添加操作数变量多项式我们需要一起确保只有一个变量 由操作数多项式表示每个操作。

使用算术属性，我们可以构造每个操作数变量多项式这样如果 多变的被用作相应操作中的操作数，则其值为 1，否则为 0。连续乘以 0 乘以任何值将保持为零，并且当加在一起时将被忽略。对于我们的示例l—个（X）必须符合评估l—个(1) = 1,l—个(2) = 1 和 l—个(3) = 0 和ld（X）在 1 和 2 处为零，但在 1 处x =3：





因此，我们可以分别设置每个变量的值，然后将它们加在一起以获得操作数多项式，例如，如果一个=3和d =2：



注意：我们在值旁边使用下标来指示它代表哪个变量，例如，3一个是一个变量一个用值实例化3.

让我们表示这样的复合操作数多项式从现在开始使用大写字母，例如， L(x) = 人一个(x) + dld（X），及其评价值为大号，IE，L = L(s)。这种结构只有在每个操作数变量多项式受验证者限制，与左操作数的交互应相应改变：

- 设置
  - 构造l一个(x), ld（X）这样它通过 1 在“操作X”在所有其他操作中使用它并通过 0
  - 随机抽样s, α
  - 评估和加密未赋值的变量多项式：
    Gl一个(s)， Gld(s)
  - 计算转移这些多项式中：
    Gαl一个(s)， Gαld(s)
  - 设置证明密钥：（
    Gl一个(s)， Gld(s)， Gαl一个(s)， Gαld(s) ）
  - 设置验证密钥：（G
    α)
- 证明
  - 分配价值观一个和d到变量多项式：( )
    Gl一个(s) 一个(Gld(s) d
  - 分配相同的价值观转移多项式：（
    Gαl一个(s)) 一个(Gαld(s)) d
  - 全部添加分配可变多项式形成操作数多项式：
    GL(s)=G人一个(s) • Gdld(s)=G人一个(s)+dld(s)
  - 添加移位赋值变量多项式形成一个移位的操作数多项式：
    GαL(s)=G阿尔法一个(s) • G达尔d(s)=Gα(人一个(s)+dld(s))
  - 提供有效转让的证明左操作数：（
    GL(s)， GαL(s) ）
- 确认
  （ ）
  - 将证明解析为G大号， G大号'
  - 检查提供的多项式是原始提供的倍数的总和无助的
    gned 变量多项式：
    ( )
    例如大号',g = e （ ） 检查
    G大号， Gα

    阿尔法一个(s) + α dld(s) = α× (人一个(s) + dld(s))

笔记：L(s)和αL(s)一次表示所有可变多项式，并且由于 α 仅用于评估可变多项式，因此证明者别无选择，只能使用提供的评估并将相同的系数分配给原始和移位的可变多项式。

结果证明者：

- 无法修改提供可变多项式通过改变它们的系数，除了“分配”值，因为证明者只呈现这些多项式的加密评估，并且因为必要的加密幂s不能单独使用它们的 α 位移

- 无法向提供的多项式添加另一个多项式，因为 α 比率将被破坏

- 不能通过乘以其他多项式来修改操作数多项式u(x),这可能会不成比例地修改值，因为在预配对空间中无法进行加密乘法

注意：如果我们添加（或减去）一个多项式，例如，l—个（X），对另一个，例如，l' d(x) = cd · ld(x) + C'—个 · l—个（X），这并不是对多项式的真正修改ld（X），而是改变
的结果系数l—个（X），因为最后总结出来的是：

$$L(x) = c_{—个} \cdot l_{—个}(x) + l'_d(x) = (c_{—个} + C'_{—个}) \cdot l_{—个}(x) + c_d \cdot l_d(X)$$

虽然证明者限制了多项式的使用，但仍然存在一些不需要抵消的自由：

- 如果证明者决定不添加一些分配的变量多项式是可以接受的 l—世（X）形成操作数多项式L(x)因为它和赋值是一样的
  0：G人—个（X）=G人—个(x)+0ld（X）

- 如果证明者多次添加相同的变量多项式是可以接受的，因为它与分配该值的倍数一次相同，例如，G人—个（X）· G人—个（X）· G人—个（X）=G3人—个（X）

这种方法同样适用于右操作数和输出多项式R（x），O（x）。

4.7 建筑属性

作为这种修改的副作用，还有许多额外的有用特性。

4.7.1 常数系数

在上述构造中，我们一直在使用评估未赋值的变量多项式 1 或 0 作为表示变量是否为用过的在操作中或不是。当然，没有什么可以阻止我们使用其他系数，包括负系数，因为

我们可以插通过任何必要点的多项式[22]. 此类操作的示例如下：

$$2_{一个} \times 1b = 3r$$

$$-3_{一个} \times 1b = -2r$$

因此我们的程序现在可以使用常数系数，例如：

```
算法 2常数系数
  功能计算（w，a，b）
      如果w然后
          返回3a×b
      别的
          返回5a×2b  _
      万一
  结束函数
```

这些系数将在设置阶段“硬连线”，类似于 1 或 0 将是不可变的。我们可以相应地修改运算形式：

$$C_{一个} \cdot 一个 \times C_b \cdot b = C_r \cdot r$$

或者更正式地，对于变量$v_{一世} \in \{v_1, v_2, ..., v_n\}$：

$$C_l \cdot v_l \times C_r \cdot v_r = C_{○} \cdot v_{○}$$

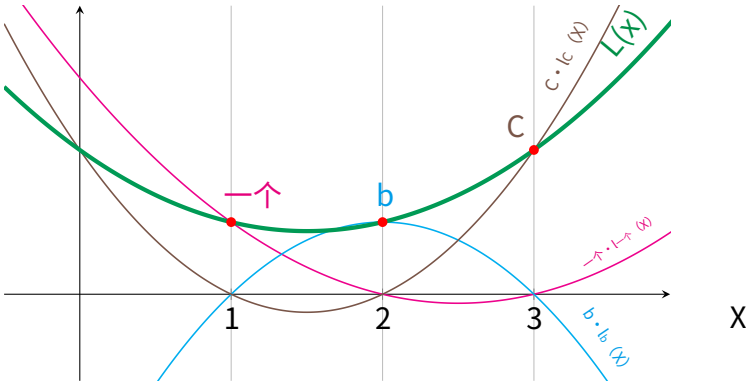
在哪里l, r, o是操作中使用的变量的索引。

注意：同一变量的常数系数在不同的操作和操作数/输出中可能不同。

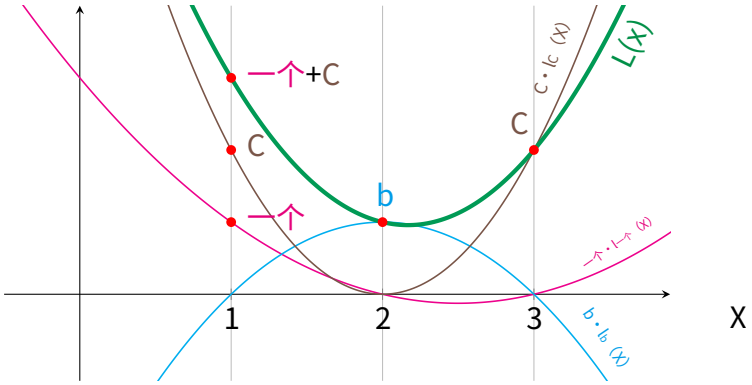
4.7.2 免费添加

考虑到更新的结构，很明显，在多项式表示中，每个 操作数由一些不同的表达X是所有的总和操作数变量多项式这样只有一个用过的变量可以具有非零值，而其他所有值都为零。该图最好地说明了这一点：

[22]前提是没有两个操作占用相同X



我们可以利用这种结构并允许添加任意数量的必要变量 对于每个操作数在操作。例如在第一个操作中，我们可以添加a + c 菲首先，然后才将它乘以其他一些操作数，例如，(一+三）×b=r,这可以表示为：



因此，可以在单个变量中添加任意数量的当前变量操作数，根据各自程序的需要，对它们中的每一个使用任意系数来生成将在相应操作中使用的操作数值。这种属性有效地允许将操作结构更改为：

$$(C_{l,\text{一个}} \cdot a + c_{l,b} \cdot \text{乙} + 。。.) \times (C_{r,\text{一个}} \cdot a + c_{r,b} \cdot \text{乙} + 。。.) = (C_{\text{哦，一个}} \cdot a + c_{\text{哦，b}} \cdot \text{乙} + 。。.)$$

或者更正式地，对于变量v一世∈ {v1, v2, ..., vn}和操作数变量系数Cl,一世∈ {Cl,1， Cl,2， ...， Cr,n},Cr,一世∈ {Cr,1， Cr,2， ...， Cr,n},C哦，一世∈ {Co,1， Co,2， ...， C哦，n}：

$$\sum_{\text{我=1}} C_{l,\text{一世}} \cdot v_{\text{一世}} \times \sum_{\text{我=1}} C_{r,\text{一世}} \cdot v_{\text{一世}} = \sum_{\text{我=1}} C_{\text{哦，一世}} \cdot v_{\text{一世}}$$

注意：每个操作的操作数都有自己的一组系数C。

4.7.3 加减除法

到目前为止，我们一直主要关注乘法运算。然而，为了能够执行一般计算，现实生活中的程序还需要加法、除法和减法。

添加在上一节中，我们已经确定我们可以在单个操作数的上下文中添加变量，然后将其乘以另一个操作数，例如，（3—+二）×d=r,但是如果我们只需要加法而不需要乘法，例如，如果一个程序需要计算 a + b，我们可以将其表示为：

(a + b)×1=r

注意：因为我们的构造需要一个常数系数和一个变量 (C・v) 对于每个操作数，值1表示为C—・v—，而同时C—=1可以“硬连线”成相应的多项式，即v—是一个变量，可以被赋予任何值，因此我们必须强制执行v—通过 4.10 节中解释的约束。

减法减法几乎与加法相同，唯一的区别是一个负系数，例如，对于 a - b：

(一个 + -1・b)×1=r

分配如果我们检查除法运算因素 除数=结果我们会看到结果的
除法是我们需要相乘的数字除数通过生产因素。因此我们可以通过乘法来表达同样的意思：除数×结果=因素。最后，
如果我们要证明除法运算一个 b=r，可以表示为：

b×r =一个

注意：运算的构造也称为“约束”，因为多项式构造表示的运算本身并不计算结果，而是检查证明者是否已经知道变量（包括结果），并且它们是有效的对于操作，即证明者必须提供一致的值，无论它们是什么。

注意：所有这些算术运算都已经存在；因此不需要修改操作的结构。

4.8 示例计算

有了一般运算的构造，我们可以将我们原来的算法 1 转换为一组运算，并进一步转换为多项式形式。让我们考虑算法的数学形式（我们将使用变量v捕获评估结果）：

w× (一个×b) + (1 -w)× (a + b) = v

它有 3 个乘法，由于运算构造只支持 1 个，所以至少会有 3 个运算。但是，我们可以简化方程：

w× (一个×b) + a + b - w× (a + b) = v

w× (一个×b - a - b) = v - a - b

现在它需要两次乘法，同时保持相同的关系。完整的操作是：

1：　　1・一个×1・b　　=　　1・米

2：　　1・w　×　1・米+-1・一个+-1・b　　=　　1・v+-1・一个+-1・b

我们还可以添加一个要求w是二进制的，否则证明者可以使用任何值w渲染计算不正确：

3：　　1・w×1・w　　=1・w

看看为什么w只能是 0 或 1，我们可以将方程表示为w^2-w =0 及以上 (w -0)(w -1) = 0 其中 0 和 1 是唯一的解决方案。

这些总共有 5 个变量，其中 2 个在左操作数中，4 个在右操作数中，5 个在输出中。操作数多项式为：

L(x)=一个・l一个(x) + w・lw（X）

R(x)=米・r米(x) + 一个・r一个(x) + b・rb(x) + w・rw（X）

牛）=米・○米(x) + v・○v(x) + 一个・○一个(x) + b・○b(x) + w・○w（X）

其中每个可变多项式必须为 3 个操作中的每一个计算相应的系数，如果变量不存在于操作的操作数或输出中，则计算为 0：

l一个(1)= 1;l一个(2)= 0;l一个(3)= 0;r米(1)= 0;r米(2)=　　1个；r米(3)= 0;○米(1)= 1;○米(2)=　　0;○米(3)= 0;

lw(1)= 0;lw(2)= 1;lw(3)= 1;r一个(1)= 0;　　r一个(2)= -1;　　r一个(3)= 0;　　○v(1)= 0;　　○v(2)=　　1个;○v(3)= 0;

rb(1)= 1;　　rb(2)= -1;　　rb(3)= 0;　　○一个(1)= 0;　　○一个(2)= -1;　　○一个(3)= 0;

rw(1)= 0;rw(2)=　　0;rw(3)= 1;　　○b(1)= 0;○b(2)= -1;　○b(3)= 0;

○w(1)= 0;○w(2)= 0;○w(3)= 1;

因此，辅因子多项式是t(x) = (x -1)(x -2)(x -3），这将确保正确计算所有三个操作。

接下来我们利用多项式插值来找到每个可变多项式：

l一个（X）= $\frac{1}{2}X^2 - \frac{5}{2}\times +3$;　　r米（X）= $-x^2+4x -3$;　　○米（X）= $\frac{1}{2}X^2 - \frac{5}{2}\times +3$;

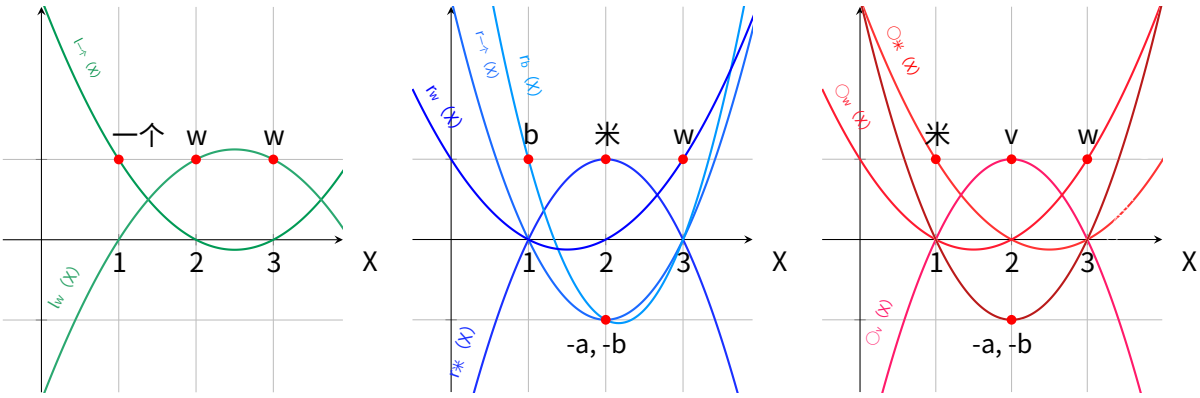
lw（X）= $-\frac{1}{2}X^2+x\frac{5}{2}$;　　r一个（X）= $X^2-4\times +3$;　　○v（X）= $-x^2+4x -3$;

rb（X）= $\frac{3}{2}X^2-\frac{13}{2}\times +6$;　　○一个（X）= $X^2-4\times +3$;

rw（X）= $\frac{1}{2}X^2-\frac{3}{2}\times +$1个;　　○b（X）= $X^2-4\times +3$;

○w（X）= $\frac{1}{2}X^2-\frac{3}{2}\times +$1个;

绘制为：



我们已经准备好通过多项式来证明计算。首先，让我们为函数选择输入值，例如w =1、一个=3、b =2. 其次，从操作中计算中间变量的值：

$$米=一个\times b=6$$

$$v = w(m - a - b) + a + b = 6$$

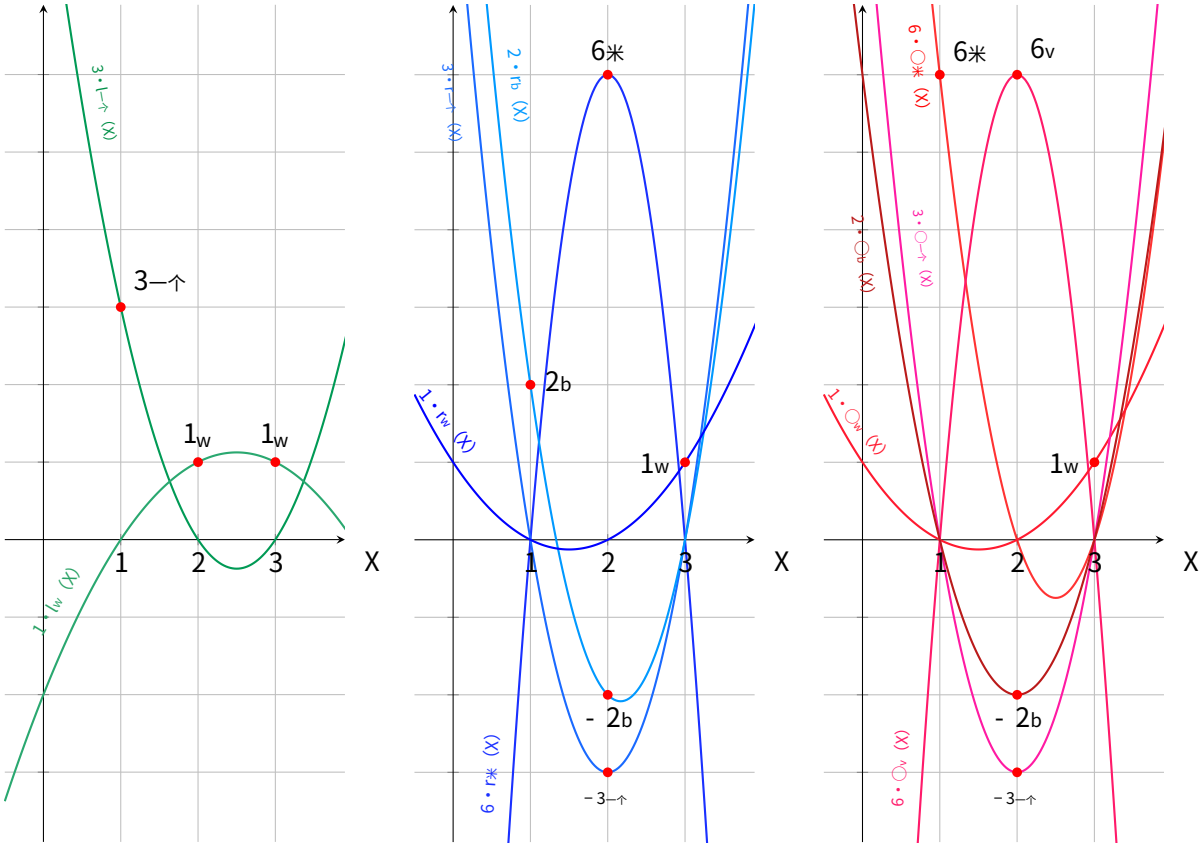
之后，我们将计算结果所涉及的所有值分配给相应的可变多项式并将它们相加以形成操作数和输出多项式：

$$L(x)=3\cdot l_{一个}(x)+1\cdot l_{w}(X)=x_2-5\times+7$$

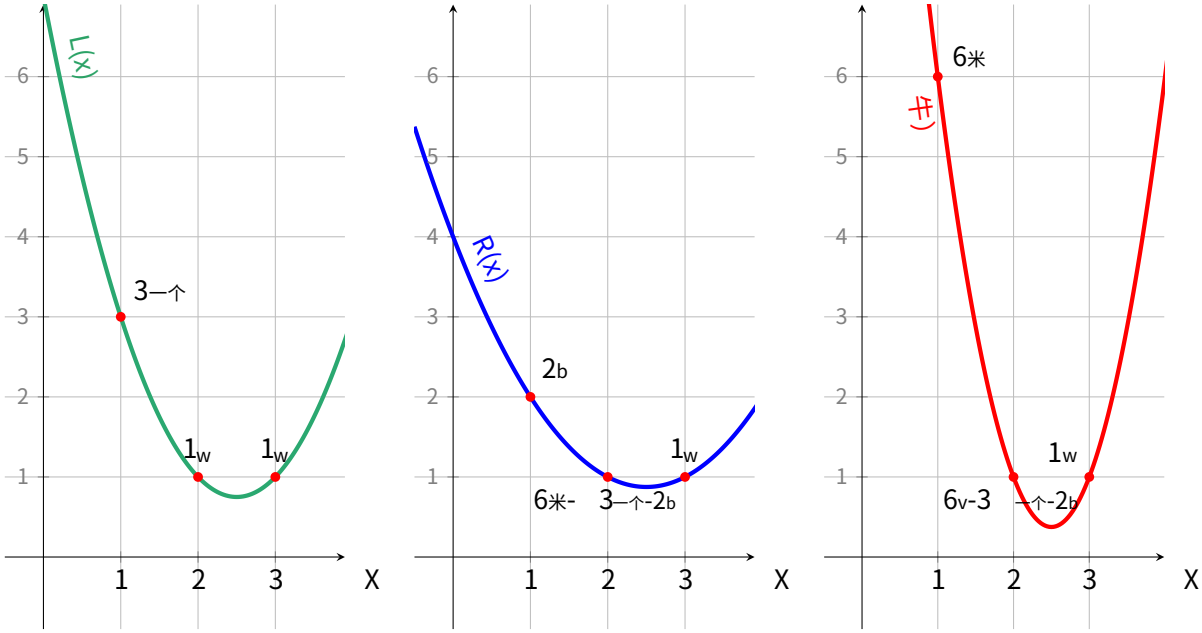
$$R(x)=6\cdot r_{米}(x)+3\cdot r_{一个}(x)+2\cdot r_{b}(x)+1\cdot r_{w}(X)=\frac{1}{2}X_2-2\ \frac{1}{2}\times+4$$

$$牛)=6\cdot \bigcirc_{米}(x)+6\cdot \bigcirc_{v}(x)+3\cdot \bigcirc_{一个}(x)+2\cdot \bigcirc_{b}(x)+1\cdot \bigcirc_{w}(X)=2X_{\frac{1}{2}}-12\qquad\frac{1}{2}\times+16$$

在图表形式中，这些是：



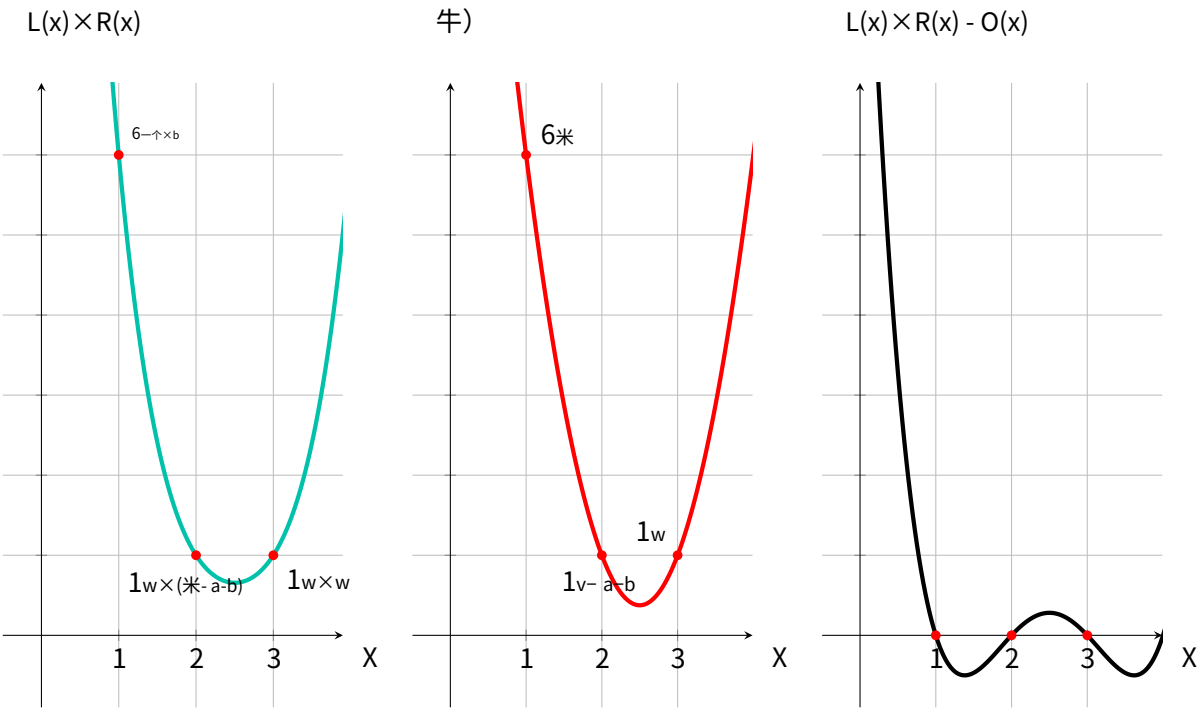
总结起来表示相应操作中的操作数和输出值：



我们需要证明L(x)×R(x) - O(x) = t(x) h(x),因此我们发现h(x)：

$$h(x) = \frac{L(x)\times R(x) - O(x)}{t(x)} = \frac{\pm X_4 - 5X_3 + 35 \quad \overline{2}X_2 - 25\times + 12}{(x-1)(x-2)(x-3)} = \frac{1}{2}x - 2$$

在图形形式中，它表示为：



多项式可见的地方L(x)×R(x) - O(x)有解决方案x =1、x =2和x =3，因此t(x)是它的辅因子，如果我们使用不一致的变量值，情况就不会如此。

这就是在多项式级别上证明正确计算执行的变量值知识的方式。然后，证明者正在处理协议的加密部分。

4.9 可验证计算协议

我们对多项式协议（第 3.7 节）的知识进行了许多重要的修改以使其具有通用性，所以现在让我们看看它是如何定义的。假设商定的功能F（*)计算的结果是证明的主题，以及操作的数量d,变量数n并对应于它们的系数

{Cl,我,j， Cr,我,j， C哦，我,j}一世∈{1,...,n},j∈{1,...,d}：

- 设置
  - 构造可变多项式对于左操作数 {l—世（X）}—世∈{1,...,}这样对于所有操作j∈ {1, . . . ,d}他们评估相应的系数，即l—世(j) = cl,我,j, 同样对于右操作数和输出
  - 随机抽样s, α
  - 计算t(x) = (x -1)(x -2) 。. . (x - d)及其评价Gt(s)
  - 计算证明密钥：
  - 计算验证密钥：

$$\left(\{G_{sk}\}_{k\in[d]}, \{G_{l-世(s)}, G_{r-世(s)}, G_{○-世(s)}, G_{αl-世(s)}, G_{αr-世(s)}, G_{αo-世(s)}\}_{一世\in\{1,...,\}}\right)$$

$$\left(G_{t(s)}, G_{α}\right)$$

- 证明
  - 计算函数F（*)因此相应的变量值 {v一世}一世∈{1,...,}
  - 计算h(x) =L(x)×R(x)-O(x),在哪里L(x) = $\sum_{我=1}^{n} v_{一世} \cdot l_{一世}(X)$ ，同样地R(x), O(x)
  - 分配变量值和总结获得操作数多项式：
    $G_{L(s)}=G_{l_1(s)}^{(\quad)^{v_1}} \cdot \cdot G_{l_n(s)}^{(\quad)^{v_n}}$ ， $G_{R(s)}=\prod_{我=1}(G_{r_{一世}(s)})^{v_{一世}}$ ， $G_{O(s)}=\prod_{我=1}(G_{○_{一世}(s)})^{v_{一世}}$
  - 将变量值赋给移位多项式：
    $G_{\alpha L(s)}=\prod_{我=1}(G_{\alpha l_{一世}(s)})^{v_{一世}}$ ， $G_{\alpha R(s)}=\prod_{我=1}(G_{\alpha r_{一世}(s)})^{v_{一世}}$ ， $G_{\alpha O(s)}=\prod_{我=1}(G_{\alpha o_{一世}(s)})^{v_{一世}}$
  - 计算加密评估$G_{h(s)}$使用提供的权力年代：$\{G_{s_k}\}_{k\in[d]}$
  - 设置证明：$(G_{L(s)}, G_{R(s)}, G_{O(s)}, G_{\alpha L(s)}, G_{\alpha R(s)}, G_{\alpha O(s)}, G_{h(s)})$
- 确认
  - 将证明解析为$(G_{大号}, G_{R}, G_{○}, G_{大号'}, G_{R'}, G_{○'}, G_{H})$
  - 可变多项式限制检查：
    例如$(g_{大号}, G_{\alpha})$ =例如$(g_{大号'}, g)$, $e(g_{R}, G_{\alpha})$ =例如$(g_{R'}, g)$, $e(g_{○}, G_{\alpha})$ =例如$(g_{○'}, G)$
  - 有效操作检查：
    例如$(G_{大号}, G_{R})$ =例如$(G_{吨}, G_{H})$ · 例如$(G_{○}, G)$

注意：使用符号$\prod$可以简洁地表达多个元素的乘积，即 $\prod_{我=1}^{n} v_{一世} = v_1 \cdot v_2 \cdot \ldots \cdot v_n$.

所有变量多项式的集合{l一世(x), r一世(x), o一世（X）}一世∈{1,...,}和目标多项式 t(x)被称为二次算术程序(质量保证计划[23]）。

虽然该协议足够健壮以允许进行一般计算验证，但有两个安全考虑必须解决。

4.9.1 操作数和输出的不可互换性

因为我们对所有的操作数使用相同的 α可变多项式限制没有什么可以阻止证明者：

- 使用来自其他操作数的可变多项式，例如，大号'(s) = o1(s) + r1(s) + r5(s) + 。. .
- 交换操作数多项式完全，例如，O(s)和L(s)将导致操作 O(s)×R(s)=L(s)
- 重复使用相同的操作数多项式，例如，L(s)×L(s)=O(s)

[23] Gen+12。

这种可互换性意味着证明者可以改变执行并有效地证明一些其他计算。防止这种行为的明显方法是对不同的操作数使用不同的α-s，具体我们修改：

- 设置
  - ...
  - 样本随机 $\alpha_l, \alpha_r, \alpha_\bigcirc$而不是α
  - 计算相应的“班次” $\{G_{\alpha_l^{-世(s)}}, G_{\alpha_r^{-世(s)}}, G_{\alpha_\bigcirc^{-世(s)}}\}_{世\in\{1...\}}$
  - 证明密钥： $(\{G_s^k\}_{k\in[d]}, \{G_{l^{-世(s)}}, G_{r^{-世(s)}}, G_{\bigcirc^{-世(s)}}, G_{\alpha_l^{-世(s)}}, G_{\alpha_r^{-世(s)}}, G_{\alpha_\bigcirc^{-世(s)}}\}_{世\in\{1...\}})$
  - 验证密钥： $(G_{t(s)}, G_{\alpha_l}, G_{\alpha_r}, G_{\alpha_\bigcirc})$
- 证明
  - ...
  - 将变量分配给“移位”多项式 $\prod G_{\alpha_l L(s)} = \prod_{我=1}^{n} (G_{\alpha_l^{-世(s)}})^{v^{-世}}$; $G_{\alpha_r R(s)} = \prod_{我=1} (G_{\alpha_r^{-世(s)}})^{v^{-世}}$; $G_{\alpha_\bigcirc O(s)} = \prod_{我=1} (G_{\alpha_\bigcirc^{-世(s)}})^{v^{-世}}$
  - 设置证明： $(G_{L(s)}, G_{R(s)}, G_{O(s)}, G_{\alpha_l L(s)}, G_{\alpha_r R(s)}, G_{\alpha_\bigcirc O(s)}, G_{h(s)})$
- 确认
  - ...
  - 可变多项式限制检查： $e(G_{例如_{大号}}, G_{\alpha_l})$ $=$ $e(G_{例如_{大号'}}, G)$, $e(G_R, G_{\alpha_r})$ $=$ $e(G_{R'}, G)$, $e(例如_\bigcirc, G_{\alpha_\bigcirc})$ $=$ $e(例如_{\bigcirc'}, G)$

现在不可能使用来自其他操作数的可变多项式，因为 $\alpha_l, \alpha_r, \alpha_\bigcirc$证明者不知道。

4.9.2 操作数之间的变量一致性

对于任何变量$v^{-世}$我们必须分配它的价值可变多项式对于每个对应ding 操作数，即$G_{l^{-世(s)}}$ $(v^{-世}, G_{r^{-世(s)}})$ $(v^{-世}, G_{\bigcirc^{-世(s)}})$ $v^{-世}$.因为每一个的有效性操作数多项式单独检查，没有强制要求在相应的变量中使用相同的变量值可变多项式。这意味着变量的值v_1在左操作数可以不同于变量v_1在右操作数或输出中。

我们可以通过已经熟悉的限制多项式的方法来强制跨操作数的变量值相等（就像我们对变量多项式所做的那样）。如果我们可以在所有操作数上创建一个“移位校验和”变量多项式，这将限制证明者只能分配相同的值。验证者可以将每个变量的多项式组合成一个，例如， $G_{l^{-世(s)}+r^{-世(s)}+o^{-世(s)}}$，并将其移动一些其他随机值 β，即$G_{\beta(l^{-世(s)}+r^{-世(s)}+o^{-世(s)})}$. 这改变了

多项式被提供给证明者以将变量的值与变量多项式一起分配：

$$(\ )_{G_{l,\text{—世}(s)}} v_{l,\text{—世}}(\ ),(\ )_{v_{r,\text{—世}}, G_{r\text{—世}(s)}}, (\ )_{G_{\bigcirc\text{—世}(s)}} v_{\text{哦，—世}}, (\ )_{G_{\beta(l\text{—世}(s)+r\text{—世}(s)+o\text{—世}(s))}} v_{\beta,\text{—世}}$$

并且β被加密并添加到验证密钥中G_β. 现在，如果所有的值$v_{\text{—世}}$是相同的（即，$v_{l,\text{—世}}=v_{r,\text{—世}}=v_{\text{哦，—世}}=v$

$$\text{例如}(\ )_{v_{l,\text{—世}}\cdot l\text{—世}(s)\cdot G_{v_{r,\text{—世}}\cdot r\text{—世}(s)\cdot G_{v_{\text{哦，—世}}\cdot\bigcirc\text{—世}(s)}, G_\beta} (\ )_{G_{v_{\beta,\text{我}}\cdot\beta(l\text{—世}(s)+r\text{—世}(s)+o\text{—世}(s))}, G} = e$$

为了一世$\in\{1,\ldots,n\}$),该等式应成立：

虽然这是一个有用的一致性检查，但由于不可忽略的概率，至少有两个 l(s), r(s), o(s)可能具有相同的评估值或一个多项式可被另一个整除等，这将允许证明者考虑值$v_{l,\text{—世}}, v_{r,\text{—世}}, v_{\text{哦，—世}}, v_{\beta,\text{我}}$使得其中至少有两个不相等但等式成立，从而使检查无效：

$$(v_{l,\text{—世}}\cdot l\text{—世}(s)+v_{r,\text{—世}}\cdot r\text{—世}(s)+v_{\text{哦，—世}}\cdot\bigcirc\text{—世}(s))\cdot\beta=v_{\beta,\text{我}}\cdot\beta\cdot(l\text{—世}(s)+r\text{—世}(s)+o\text{—世}(s))$$

例如，让我们考虑一个单一的操作，在这种情况下l(x) = r(x)。我们将对这两者的评估表示为w = l(s) = r(s)和y = o(x)。然后等式将如下所示：

$$\beta(v_l w + v_r w + v_\bigcirc y) = v_\beta\cdot\beta(w + w + y)$$

这种形式允许，对于一些任意的v_r和$v_\bigcirc$， 设置$v_\beta=v_\bigcirc$,$v_l=2v_\bigcirc-v_r$，这将转化为：

$$\beta(2v_\bigcirc w - v_r w + v_r w + v_\bigcirc y) = v_\bigcirc\cdot\beta(2w + y)$$

因此，这种一致性策略是无效的。缓解这种情况的一种方法是对每个操作数使用不同的 β，确保操作数的可变多项式将具有不可预测的值。以下是协议修改：

- 设置
  - 。. . 样本随机 $\beta_l, \beta_r, \beta_\bigcirc$
  - 计算、加密并添加到证明密钥变量一致性多项式：$\{G_{\beta_l l\text{—世}(s)+\beta_r r\text{—世}(s)+\beta_\bigcirc\bigcirc\text{—世}(s)}\}_{\text{—世}\in\{1,\ldots,\}}$
  - 加密 β-s 并添加到验证密钥：$G_{\beta_l}, G_{\beta_r}, G_{\beta_\bigcirc}$ ( )
- 证明
  - 。. . 给变量赋值变量一致性多项式：$G_{z\text{—世}(s)}=G_{\beta_l l\text{—世}(s)+\beta_r r\text{—世}(s)+\beta_\bigcirc\bigcirc\text{—世}(s)}\ v_{\text{—世}}$ 为了一世$\in\{1,\ldots,\}$
  - 在加密空间中添加分配的多项式：$\prod_{\text{我}=1}^{n} G_{z\text{—世}(s)}=G_{\beta_l L(s)+\beta_r R(s)+\beta_\bigcirc O(s)}$
  - 添加到证明：$G_{Z(s)}$
- 确认

– 。. . 检查提供的一致性操作数多项式和“校验和”

多项式：

( ) ( ) ( ) ( )

例如大号， Gβl ・例如R， Gβr ・例如○， Gβ○ =e Gz， G

这相当于：

蛋）βlL+βrR+β○○=蛋）z

相同的变量值回火技术在这种构造中将失败，因为不同的 β-s 使相同的多项式不适合操作。然而，有一个类似于备注 4.1 中的缺陷，具体是因为术语Gβl， Gβr,Gβ○是公开可用的，对手可以修改任何变量多项式的零指数系数，因为它不依赖于年代，IE，Gβls0=Gβl.

4.9.3 变数和变数一致性多项式的不可延展性

可变多项式的延展性

让我们用以下两个操作来举例说明 4.1：

一个×1=b

3一个×1=C

预期的结果是b = 一个和c =3一个，有明确的关系c =3湾。这意味着左操作数的变量多项式有评估l一个(1) = 1 和l一个(2) = 3. 不管是什么形式l一个（X），证明者可以不成比例地分配一个，通过提供修改

多项式l' 一个(x) = 人一个(x) +1.因此评估将是l' 一个(1) =一个+1 和l' 一个(2) = 3一个+1、因此结果b = a +1 和c =3一个+1 在哪里C6=3乙，有效地意味着 一个对于不同的操作是不同的。

因为证明者可以访问Gαl和Gβl他可以同时满足正确的操作数多项式 和变量值一致性检查：

・。. . 证明：

– 通过不成比例地分配变量形成左操作数多项式a: L(x) = a・l一个(x) +1

– 像往常一样形成右操作数和输出多项式： R(x) = r1(x),
O(x) = b・○b(x) + c・○c（X）

– 计算余数h(x) =L(x)・R(x)-O(x) t(x)

– 计算加密：GL(s)=Gl一个(s) 一个・G1( )和往常一样GR(s)， GO(s)

– 计算 α 位移：GαL(s)=Gαl一个(s) 一个・Gα( )和往常一样GαR(s)， GαO(s)

– 计算变量一致性多项式： ∏ (
Gz(s)= Gβl一世(s)+βr一世(s)+β○○一世(s)) 一世•Gβl=Gβl(L(s)+1)+βrR(s)+β○O(s)

一世∈{1、a,b,c}

下标在哪里一世表示对应变量的符号，而指数一世代表

变量的值；而且未定义可变多项式等于零。

_ 设置证明：G(L(s)， GR(s)， GO(s)， GαlL(s)， GαrR(s)， Gα○O(s)， GZ(s)Gh(s))

・确认：

– 可变多项式限制检查：( )

例如大号',g = e (G大号，Gα) ⇒ 例如(αa・l一个(s)+α，G) (=例如人一个(s)+1，Gα)

和通常一样GR',G○'

– 变量值一致性检查 ( ) ( ) ( ) ( )

例如大号，Gβl ・例如R，Gβr ・例如○，Gβ○ =e Gz，G ⇒

蛋） （一个・l一个+1)βl+RβrO+β○=蛋） βl(L+1)+βrR+β○○

– 有效操作检查例如大号，GR) =例如吨，GH)・例如○，G）

可变一致性多项式的延展性

此外，可用性Gβl,Gβr,Gβ○允许在不同的操作数中使用相同变量的不同值。例如，如果我们有一个操作：

一个×一个=b

可以用变量多项式表示：

l一个(x) = x,　r一个(x) = x,　○一个(x) =0

lb(x) =0,　rb(x) =0,　○b(x) = x

虽然预期的输出是b = 一个2，我们可以设置不同的值一个，例如一个=2,一个=5 如下：

・证明：

– 。..形成左操作数多项式一个=2：L(x) =2l一个(x) +10lb（X）

– 形成右操作数多项式一个=5：R(x) =2r一个(x) +3 + 10rb（X）

– 形成输出多项式b =10：O(x) =2○一个(x) +10○b（X）

– 。..计算加密：(

GL(s)= (Gl一个(s)) 2 • (Glb(s)) 10 =G2l一个(s)+10lb(s)

GR(s)= (Gr一个(s)) 2 ・(G) 3・ (Grb(s)) 10 =G2r一个(s)+3+10rb(s)

GO(s)= (G○一个(s)) 2 • (G○b(s)) 10 =G2○一个(s)+10○b(s)

– 计算变量一致性多项式：(

GZ(s)=Gβll一个(s)+βrr一个(s)+β○○一个(s)) 2 ( ) • Gβr 3 • (Gβllb(s)+βrrb(s)+β○○b(s)) 10 =

Gβl（2l一个(s)+10lb(s)) + βr（2r一个(s)+3+10rb(s)) + β○（2○一个(s)+10○b(s))

・确认

– 。 . . 变量值一致性检查，应持有：

例如大号， Gβl ・例如R， Gβr ・例如○， Gβ○ =e ( ) Gz，G

注意：多项式○—个(x), lb(x), rb（X）实际上可以忽略，因为它们对任何值都评估为 0X，但是，为了完整性，我们会保留它们。

这种能力破坏了健全性的证明。很明显，加密的 β-s 不应该提供给证明者。

非延展性

解决延展性的一种方法是使Gβl,Gβr,Gβ○从验证密钥不兼容Gz(s)通过在设置阶段将它们在加密空间中乘以随机秘密 γ (gamma)：Gβlγ， Gβrγ， Gβ○γ. 连续这样的掩码加密不允许修改的可行性Gz(s)以一种有意义的方式，因为Z(s)不是 γ 的倍数，例如，Gz(s) ・Gv' ・βlγ=

Gβl(L(s)+ v'γ)+βrR(s)+β○O(s). 因为证明者不知道 γ，所以更改将是随机的。修改需要我们平衡协议乘法中的变量值一致性检查方程Z(s)由γ：

・设置

– 。 . .样本随机 βl, βr, β○, γ

– 。 . .设置验证密钥：。 . . ,Gβlγ， Gβrγ， Gβ○γ， Gγ ( )

・证明。 . .

・确认

– 。 . . 变量值一致性检查应保持：

例如大号， Gβlγ ・例如R， Gβrγ ・例如○， Gβ○γ =e ( ) Gz，Gγ

需要注意的是，我们排除了变量多项式为 0 度的情况（例如，
l1(x) =1X0），否则将允许以可变一致性公开 β 的加密
证明密钥的多项式Gβil—世(s)+βrr—世(s)+β○○—世(s) { } —世∈{1,...,} 如果任何两个操作数 /
输出为零，例如，对于l1(x) =1、r1(s) =0,○1(s) =0 这将导致Gβil1(s)+βrr1(s)+β○○1(s)= Gβl.

我们也可以类似地面具α-s 来解决可变多项式。然而，它是没有必要的，因为任何修改可变多项式需要体现在变量一致性多项式这是不可能修改的。

4.9.4 变量值一致性检查的优化

这变量值一致性check 现在生效了，但是它为验证密钥增加了 4 个昂贵的配对操作和 4 个新术语。Pinocchio 协议 [Par+13] 巧妙地选择了生成器G对于每个操作数根深蒂固“转变”：

・设置

- 。..样本随机 β, γ, ρl, ρr并设置 ρ○= ρl • ρr
- 设置生成器Gl=Gρl,Gr=Gρr,G○=Gρ○
- s(et 证明密钥：

{ } { } }) Gsk , Gl , Gr , G○ , Gαl , Gαr , Gα○ , Gβl • Gβr • Gβo
k∈[d] t l r l r ○

- 设置验证密钥： ( ) Gt(s) ○, Gαl,Gαr,Gα○,Gβγ， Gγ

• 证明

- 。..分配变量值 ∏

GZ(s)= n( ) Gβl • Gβr (s) • Gβo v
我=1 l r ○

• 确认

- 。..可变多项式限制检查：（) ( ) 例如 [， G=e Gl,Gα l , 同样对于 GR r,G○○
- 变量值一致性检查：( ) (( ) 例如 [GR • G○,Gβγ =e Gz， Gγ)
- 有效操作检查：( ) ( ) ( ) 例如 [GR =e G吨GH e G○,G⇒

蛋）ρlρrLR=蛋）ρlρrth+ρlρr○

生成器的这种随机化进一步增加了安全性可变多项式 延展性，在备注 4.1 中描述，无效，因为对于预期的变化，它必须是 ρ 的倍数l, ρr或 ρ○, 原始或加密版本不可用（假设，如前所述，我们不处理可能暴露加密版本的 0 度变量多项式）。

优化使验证密钥两个元素更小，并消除了验证步骤中的两个配对操作。

注意：Jens Groth 的 2016 年论文 [Gro16] 中有进一步的协议改进。

4.10 约束

我们的分析主要集中在操作的概念上。但是，该协议实际上并不是“计算”，而是检查输出值是否是操作数值的操作的正确结果。这就是为什么它被称为约束，即验证者约束证明者为预定义的“程序”提供有效值，无论它们是什么。多个约束称为约束系统 (在我们的例子中，它是一个 rank 1 约束系统或 R1CS）。

注意：这意味着找到所有正确解决方案的一种方法是对所有可能的值组合执行蛮力并仅选择“有效”的组合，或者使用更复杂的约束满足技术 [con18]。

因此我们也可以使用约束来确保其他关系。例如，如果我们要确保变量的值一个只能是 0 或 1（即二进制），我们可以用简单的约束来做到这一点：

$$一个 \times 一个 = 一个$$

我们也可以约束一个只有2：

$$(一 - 2) \times 1 = 0$$

一个更复杂的例子是确保数字一个是一个 4 位数字[24]，换句话说，可以表示一个有 4 位。我们也可以称其为“确保数字范围”，因为 4 位数字可以表示 2_4组合，因此有 16 个数字，范围从 0 到 15。在十进制数字系统中，任何数字都可以表示为以 10 为底的幂的总和（作为我们手上手指的数量）和相应的系数，例如 $123 = 1 \cdot 10_2 + 2 \cdot 10_1 + 3 \cdot 10_0$. 类似地，二进制数可以表示为以 2 为底的幂与相应系数的和，例如，1011（二进制）$= 1 \cdot 2_3 + 0 \cdot 2_2 + 1 \cdot 2_1 + 1 \cdot 2_0 = 11$（十进制）。

因此，如果一个是一个 4 位数字，那么$a = b_3 \cdot 2_3 + b_2 \cdot 2_2 + b_1 \cdot 2_1 + b_0 \cdot 2_0$对于一些布尔值 b_0, b_1, b_2, b_3. 约束可以如下：

1：
$$一个 \times 1 = 8 \cdot b_3 + 4 \cdot b_2 + 2 \cdot b_1 + 1 \cdot b_0$$

并确保b_0, b_1, b_2, b_3只能是二进制我们需要添加：

2：
$$b_0 \times b_0 = b_0$$

3：
$$b_1 \times b_1 = b_1$$

4：
$$b_2 \times b_2 = b_2$$

5：
$$b_3 \times b_3 = b_3$$

可以通过这种方式应用相当复杂的约束，确保使用的值符合规则。需要注意的是，上述约束 1 在

当前操作的建设：

$$\sum_{我=1} C_{i,一世} \cdot v_{一世} \times \sum_{我=1} C_{r,一世} \cdot v_{一世} = \sum_{我=1} C_{哦，一世} \cdot v_{一世}$$

因为价值1（和2从前面的约束）必须通过C・v—， 在哪里C可以根植于证明密钥中，但v—可能有任何价值，因为证明者提供了它。虽然我们可以强制执行C・v通过设置为 0c =0，很难找到强制执行的约束v—在我们受限的建筑中成为 1。因此，验证者应该有一种方法来设置v—.

[24]也叫啃

4.11 公共输入和一

如果无法根据验证者的输入检查证明，则证明的可用性将受到限制，例如，知道证明者在不知道结果和/或值是什么的情况下将两个值相乘。虽然可以在证明密钥中“硬连线”要检查的值（例如，乘法的结果必须始终为 12），但这需要为每个所需的“验证者输入”生成单独的密钥对。

因此，如果验证者可以为计算指定一些值（输入或/和输出），包括v–，而不是证明者。

首先，让我们考虑证明值$G_{L(s)}$,$G_{R(s)}$,$G_{O(s)}$. 因为我们使用的是同态加密，所以可以增加这些值，例如，我们可以添加另一个加密多项式评估$G_{L(s)} \cdot G_{l_v(s)}=G_{L(s)+l_v(s)}$，这意味着验证者可以将其他变量多项式添加到已经提供的多项式中。因此，如果我们可以从证明者可用的变量多项式中排除必要的变量多项式，验证者将能够在这些变量上设置他的值，而计算检查应该仍然匹配。

这很容易实现，因为验证者已经在限制证明者选择多项式时，他可以使用 α-shift。因此，这些可变多项式可以从证明密钥移动到验证密钥，同时消除其 α-s 和 β 校验和对应项。

必要的协议更新：

- 设置
  - 。..分开所有n可变多项式分为两组：
    - *验证者的米+1：
      大号v(x) = l0(x) + l1(x) + 。..+ l米, 和类似的Rv（X）和◯v（X），
      其中索引 0 保留用于v–=1
    - *证明者的n - 米：
      大号p(x) = l米+1(x) + 。..+ ln（X），和类似的Rp（X）和◯p（X）
  - 设置证明密钥：
    ({ } { G_{s^k} $_{k\in[d]}$, $G_{l_{i\neq 世}(s)}$, $G_{r_{i\neq 世}s}$ $G_{\bigcirc_{一世}(s)}$, $G_{\alpha l_{一世}(s)}$, $G_{\alpha r_{一世}(s)}$, $G_{\alpha\bigcirc\bigcirc_{一世}(s)}$, $G_{\beta l_{一世}(s)}$ $G_{\beta r_{一世}(s)}$ $G_{\beta o_{一世}(s)}$ }) $_{一世\in\{米+1,\ldots,\}}$
  - a（添加到验证密钥：{ }) $_{一世\in\{0,\ldots,米\}}$
    ..., $G_{l_{i\neq 世}(s)}$, $G_{r_{i\neq 世}(s)}$, $G_{\bigcirc_{一世}(s)}$
- 证明
  - 。..计算h(x)考虑验证者的多项式：h(x) = 在哪里L(x) = Lv(x) + Lp (X)，同样对于R(x), O(x) $\frac{L(x)\cdot R(x) - O(x)}{t(x)}$,
  - p(提供证明：
) $G_{大号_p(s)}$, $G_{R_p(s)}$, $G_{\bigcirc_p(s)}$, $G_{\alpha l大号_p(s)}$, $G_{\alpha r R_p(s)}$, $G_{\alpha\bigcirc\bigcirc_p(s)}$, $G_{Z(s)}$, $G_{h(s)}$

• 确认

– 分配验证者的变量多项式值并加到 1：

G大号v(s)=Gl0(s) • 尺寸 我=1 (Gl−世(s)) v−世

同样对于GRv(s) r 和G○v(s)

– 可变（可变）lynom(ials re)严格检查：
例如大号Gαl =e G大号l p，G同样对于GRp r 和G○p ○

– 变量（变量值 con) 持续性检查：
例如大号GRp r G○p,Gβγ =e (Gz，Gγ)

– va（盖子操作检查：）
例如大号v(s) G大号l,GR s r v( ) GRp =例如吨 (○，H)(G例如○v(s)G○p,G)

注意：根据协议属性（第 4.6.1 节）1由多项式表示 $l_0(x)$, $r_0(x)$, o_0（X）在相应的操作中已经有适当的值，因此不需要分配。

注意：验证者必须在验证步骤上做额外的工作，这与他分配的变量数量成正比。

实际上，这会将证明者的一些变量交给验证者，同时仍然保持等式的平衡。因此有效操作check 应该仍然成立，但前提是证明者使用了与验证者用于其输入的相同的值。

1 的值是必不可少的，它允许导出任何数字[25]通过乘以一个常数项，例如，乘以一个通过 123：

$$1 \cdot 一个 \times 123 \cdot v_- = 1 \cdot r$$

4.12 计算的零知识证明

自从引入通用计算协议（第 4.4 节操作证明）以来，我们不得不放弃零知识属性，使过渡更简单。至此，我们已经构建了一个可验证的计算协议。

以前做多项式的证明零知识我们使用了随机 δ-shift，这使得证明与随机无法区分（第 3.5 节）：

$$\delta p(s) = t(s) \cdot \delta h(s)$$

通过计算，我们证明：

$$L(s) \cdot R(s) - O(s) = t(s)h(s)$$

[25]在选定的有限域中

虽然我们可以将这种方法应用于使用相同 δ 的多个多项式，即提供随机值 δL(s), δR(s), δ2O(s), δ2 h(s),这将通过配对满足有效操作检查：

$$蛋）_{\delta 2}L(s)R(s)=蛋）_{\delta 2}(t(s)h(s)+O(s))$$

问题是具有相同的 δ 会妨碍安全性，因为我们在证明中分别提供了这些值：

- 如果两个不同的多项式评估具有相同的值（例如， $G_{\delta L(s)}=G_{\delta R(s)}$等），即学习一些知识

- 之间的价值差异可能不显着L(s)和R(s)可以允许保理
  通过暴力破解这些差异，例如，如果L(s) =5R(s),迭代检查
  $G_{L(s)}=G_{R(s)}$ 我， 为了一世∈ {1...否}将揭示 5×仅 5 个步骤的差异。可以对加密的加法操作执行相同的蛮力，例如，$G_{L(s)}=G_{R(s)+5}$
- 可以发现证明元素之间的其他相关性，例如，如果例如$_{\delta L(s)}$， $G_{\delta R(s)}$) = 例如$_{\delta 2}$O(s)， G）然后L(x) ・R(x) = O(x)，等等

注意：优化 4.9.4 使此类数据挖掘更加困难，但仍然允许发现关系，除了验证者可以选择 ρl, ρr以一种有助于揭示知识的特殊方式[26].

因此，我们需要对每个多项式求值有不同的随机性（δ-s），例如：

$$\delta_l L(s) \cdot \delta_r R(s) - \delta_○ O(s) = t(s) \cdot (\Delta_? h(s))$$

为了解决右边的不等式，我们只能修改证明的值h(s),在不改变协议的情况下更可取。Delta (Δ) 这里代表我们需要应用的差异h(s)为了抵消等式另一边的随机性和?表示乘法或加法运算（依次适应除法和减法）。如果我们选择通过乘法应用 Δ (?=×)这意味着不可能以压倒性的概率找到 Δ，因为随机化○

○

$$\Delta = \frac{\delta_l L(s) \cdot \delta_r R(s) - \delta_○ O(s)}{t(s)h(s)}$$

我们可以设置 $\delta_○ = \delta_l \cdot \delta_r$，它转换为：

$$\Delta = \frac{\delta_l \delta_r (L(s) \cdot R(s) - O(s))}{t(s)h(s)} = \delta_l \delta_r$$

然而，如前所述，这阻碍了零知识属性，更重要的是，这种构造不会适应验证者的输入多项式，因为它们必须是相应 δ-s 的倍数，这需要交互。

[26]只要不是多元化设置

我们可以尝试在评估中添加随机性：

$$(L(s) + \delta_l) \cdot (R(s) + \delta_r) - (O(s) + \delta_\bigcirc) = t(s) \cdot (\Delta \times h(s))$$

$$\Delta = \frac{\overbrace{L(s)R(s) - O(s) + \delta_r L(s) + \delta_l R(s) + \delta_l \delta_r - \delta_\bigcirc}^{t(s)h(s)}}{t(s)h(s)} = 1 + \frac{\delta_r L(s) + \delta R(s) + \delta \delta_{LR} - \delta_\bigcirc}{t(s)h(s)}$$

但是由于随机性，它是不可分割的。即使我们通过将每个相乘来解决这个问题 δ和t(s)h(s),因为我们通过乘以h(s),和 Δ 将由加密的评估组成（即，$G_{L(s)}$等）将无法计算$G_{\Delta h(s)}$不使用配对（其结果在另一个数字空间中）。同样{ise c}计算不可能通过 Δ 的加密评估h(x)使用加密权力$G_{s-世}$

，因为程度

h(x)和 Δ 是d,因此 Δ 的程度h(x)最多 2d。此外，不可能计算这种随机操作数多项式评估$G_{L(s)+\delta l}$ $_{t(s)h(s)}$出于同样的原因。

因此我们应该尝试通过加法来应用 Δ (?= +)，因为它可用于同态加密的值。

$$(L(s) + \delta_l) \cdot (R(s) + \delta_r) - (O(s) + \delta_\bigcirc) = t(s) \cdot (\Delta + h(s))$$

$$\Delta = \frac{L(s)R(s) - O(s) + \delta_r L(s) + \delta_l R(s) + \delta \quad _l\delta_r - \delta_\bigcirc - t(s)h(s)}{t(s)} \Rightarrow$$

$$\Delta = \frac{\delta_r L(s) + \delta R(s) + \delta \delta_{LR} \delta_\bigcirc}{t(s)}$$

分子中的每一项都是 δ 的倍数，因此我们可以通过将每个 δ 与吨（秒）：

$$(L(s) + \delta_l t(s)) \cdot (R(s) + \delta_r t(s)) - (O(s) + \delta_\bigcirc t(s)) = t(s) \cdot (\Delta + h(s))$$

$$L(s)R(s) - O(s) + t(s)(\delta_r L(s) + \delta_l R(s) + \delta_l \delta_r t(s) - \delta_\bigcirc) = t(s)\Delta + \quad t(s)h(s)$$

$$\Delta = \delta_r L(s) + \delta_l R(s) + \delta_l \delta_r t(s) - \delta_\bigcirc$$

我们可以在加密空间中有效地计算：

$$G_{L(s)+\delta l t(s)} = G_{L(s)} \cdot \left(G_{t(s)}\right)^{\delta_l}, \text{等等}$$

$$G_\Delta = \left(G_{L(s)}\right)^{\delta_r} \cdot \left(G_{R(s)}\right)^{\delta_l} \cdot G_{t(s)}^{\delta_l \delta_r} \cdot G^{-\delta_\bigcirc}$$

这导致通过有效操作在隐藏加密值的同时进行检查。

$$\text{大号} \cdot R - O + t(\delta_r L + \delta_l R + \delta_l \delta_r t - \delta_\bigcirc) = t(s)h + t(s)(\delta_r L + \delta_l R + \delta_l \delta_r t - \delta_\bigcirc)$$

建设是统计零知识由于添加了 δ 的均匀随机倍数l, δr, δ○(见 [Gen+12] 的定理 13）。

注意：这种方法也与验证者的操作数一致，例如，$G_{\text{大号}p+\delta l\text{吨}} \cdot G_{\text{大号}v}$ $_l$ $_{l=G_{\text{大号}p+Lv+\delta l\text{吨}}}$，因此，有效操作检查仍然成立，但前提是证明者已使用验证者的值来构建证明（即，Δ = δr（大号p +大号v) + δl（Rp+Rv) + δlδrt - δ○)，有关详细信息，请参阅下一节。

为了使“变量多项式限制”和“变量值一致性”检查与零知识修改，需要添加以下参数到

证明密钥：

$$G_{l}^{t(s)},\ G_{r}^{t(s)},\ G_{\bigcirc}^{t(s)},\ G_{l}^{\alpha t(s)},\ G_{r}^{\alpha t(s)},\ G_{\bigcirc}^{\alpha\bigcirc t(s)},\ G_{l}^{\beta t(s)},\ G_{r}^{\beta t(s)},\ G_{\bigcirc}^{\beta t(s)}$$

奇怪的是，最初的匹诺曹协议 [Par+13] 主要关注的是可验证的计算，而较少关注零知识属性，这是一个小的修改，几乎免费。

4.13 zk-SNARK 协议

考虑到所有逐步改进，知识协议的最终零知识简洁非交互式参数是（零知识组件是可选的，并以不同的颜色）：

- 设置
  - 选择发电机G和密码配对e
  - 对于一个函数f(u) = y和n其中总变量米是输入/输出变量，
    转换成多项式形式[27]{l一世(x), r一世(x), o一世（X）}一世∈{0,...,}, t(x)学位d (等于操作数）和大小ñ +1
  - 随机抽样s, ρl, ρr, αl, αr, α○, β, γ
  - 设置 ρ○= ρl • ρr和操作数生成器Gl=Gρl,Gr=Gρr,G○=Gρ○
  - 设置证明密钥：（{ }

$$G^{s_{k}}_{k\in[d]},\ G_{l}^{t_{世}(s)},\ G_{r}^{t_{世}(s)},\ G_{\bigcirc}^{t_{世}(s)}{}_{一世\in\{0,\ldots,\}},$$

$$\{G_{l}^{\alpha l-世(s)},\ G_{r}^{\alpha r-世(s)},\ G_{\bigcirc}^{\alpha\bigcirc-世(s)},\ G_{r}^{\beta l-世(s)}G_{\bigcirc}^{\beta r-世(s)\,l}\ G^{\beta o-世(s)}\}_{一世\in\{米+1,\ldots,\}},$$

$$G_{l}^{t(s)},\ G_{r}^{t(s)}G_{\bigcirc}^{t(s)},\ G_{l}^{\alpha t(s)},\ G_{r}^{\alpha t(s)},\ G_{\bigcirc}^{\alpha\bigcirc t(s)\,r},\ G_{l}^{\beta t(s)},\ G_{r}^{\beta t(s)},\ G_{\bigcirc}^{\beta t(s)\,r}$$

  - 设置验证密钥：（

$$\{G_{1},\ G_{\bigcirc}^{t(s)},\ G_{l-世(s)},\ G_{r}^{r-世(s)},\ G_{\bigcirc}^{\bigcirc-世(s)\,l}\}_{一世\in\{0,\ldots,米\}},\ G_{\alpha l},G_{\alpha r},G_{\alpha\bigcirc},G_{\gamma},\ G_{\beta\gamma}$$

- 证明
  - 对于输入你，执行计算f(u)获取值 {v一世}一世∈{米+1,...,}对于所有中间变量
  - 将所有值分配给未加密的变量多项式L(x) = l0(x) + $\sum_{我=1}^{n} v_{一世} \cdot l_{一世}(X)$
    同样地R(x), O(x)
  - 样本随机 δl, δr和δ○
  - 菲ndh(x) = $\dfrac{L(x)R(x) - O(x)}{t(x)}$ + δlt(x) + δlR(x) + δlδrt(x) - δ○

[27]二次算术程序

- 将证明者的变量值分配给加密的变量多项式并申请
  零知识δ-转移G大号p(s)
- 分配它的α位移对G
- 分配变量值一致性多项式
  Gz(s)=Gβt(s)
- 计算证明G大号p(s)，
- 确认
- 将提供的证明解析为
- 将输入/输出值分配给验证者的加密多项式并加到 1：
- 可变（可变）lynom(ials re)严格检查：
  例如大号
- 变量（变量值 con) 持续性检查：
  例如大号
- va（盖子操作检查：）例
  如大号p

5 结论

我们最终得到了一个有效的协议，它允许证明计算：

- 简洁——独立于计算量，证明是恒定的、小尺寸的
- 非交互式——只要计算出证明，它就可以用来说服任意数量的验证者，而无需与证明者直接交互
- 有争论的知识——该陈述以不可忽略的概率是正确的，
  即，假证明是不可行的；此外证明者知道对应的值[28]对于真实的陈述，例如，如果陈述是“乙是 sha256(一个）”那么证明者知道一些一个这样乙 =沙256(一个）这很有用，因为乙只能根据以下知识计算一个以及计算不可行一个从乙只要[29]

[28]目击者
[29]假设一个有足够的熵

• 在零知识——从证明中提取任何知识是不可行的，即它与随机的无法区分

由于多项式、模算术、同态加密、椭圆曲线密码学、密码配对和发明人的独创性的独特性质，有可能实现初级。

该协议证明了唯一的有限执行机器计算的正确性，它在一次操作中可以将几乎任意数量的变量相加，但只能执行一次乘法。因此，有机会优化程序以有效利用这种特殊性以及使用最小化操作数量的结构。

验证者不必知道任何秘密数据即可验证证明，因此任何人都可以以非交互方式发布和使用正确构造的验证密钥，这一点至关重要。这与证明只能说服一方的“指定验证者”方案相反，因此它是不可转让的。在zk-SNARK在上下文中，如果不可信或单方生成密钥对，我们可以实现此属性。

零知识证明构造领域不断发展，引入了优化（[Ben+13；Gro16；GM17]）、可更新的证明和验证密钥（[Gro+18]）等改进，以及新的构造（Bulletproofs [Bün+ 17]、ZK-STARK [Ben+18]、Sonic [Mal+19]）。

致谢

我们感谢 Mary Maller 和 Andrew Miller 对这项工作的宝贵意见。

6 参考文献

[位+11] Nir Bitansky、Ran Canetti、Alessandro Chiesa 和 Eran Tromer。从可提取的碰撞阻力到简洁的非交互式知识论证，然后再次返回。密码学 ePrint 档案，报告 2011/443。https://eprint.iacr.org/2011/443。2011 年。

[标准杆+13] 布莱恩·帕诺、克雷格·金特里、乔恩·豪威尔和玛丽安娜·雷科娃。Pinocchio：几乎实用的可验证计算。密码学 ePrint 档案，报告 2013/279。 https://eprint.iacr.org/2013/279。2013 年。

[Rei16] 克里斯蒂安·赖特维斯纳。简而言之，zkSNARKs。2016 年。网址：https://blog.ethereum。org/2016/12/05/zksnarks-in-a-nutshell/（于 2018 年 5 月 1 日访问）。

[但16] 维塔利克·布特林。二次算术程序：从零到英雄。https : // medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558 2016.（访问时间：2018-05-01）。

[但是17] 维塔利克·布特林。zk-SNARKs：引擎盖下。2017 年。网址：https://medium.com/@VitalikButerin/zk-snarks-under-the-hood-b33151a013f6（于 2018-05-01 访问）。

[加布17] 阿里尔·加比松。解释 SNARK。https://z.cash/blog/snark-explain/。2017.（访问时间：2018-05-01）。

[本+14] Eli Ben-Sasson、Alessandro Chiesa、Christina Garman、Matthew Green、Ian Miers、Eran Tromer 和 Madars Virza。Zerocash：来自比特币的去中心化匿名支付。密码学 ePrint 档案，报告 2014/349。https://eprint.iacr。组织/2014/349。2014 年。

[GMR85] S Goldwasser、S Micali 和 C Rackoff。“交互式证明系统的知识复杂性”。在：第十七届 ACM 计算理论年度研讨会论文集。斯托克'85。美国罗德岛普罗维登斯：ACM，1985 年，第 291-304 页。伊斯本：0-89791-151-2。土井：10.1145/22145.22178。网址：http://doi。acm.org/10.1145/22145.22178。

[BFM88] 曼努埃尔·布鲁姆、保罗·费尔德曼和西尔维奥·米卡利。《非交互零知识及其应用》。在：第二十届 ACM 计算理论年度研讨会论文集。斯托克'88。美国伊利诺伊州芝加哥：ACM，1988 年，第 103-112 页。 伊斯本：0-89791-264-0。土井：10.1145/62212.62222。网址：http://doi.acm.org/10。1145/62212.62222。

[格罗10] 延斯·格罗斯。“基于短配对的非交互式零知识论证”。在： 密码学和信息安全理论与应用国际会议。施普林格。2010，第 321-340 页。

[Gen+12] 罗萨里奥·根纳罗、克雷格·金特里、布莱恩·帕诺和玛丽安娜·雷科娃。没有 PCP 的二次跨度程序和简洁的 NIZK。密码学 ePrint 档案，报告 2012/215。https://eprint.iacr.org/2012/215。2012 年。

[Pik13] 斯科特派克。评估多项式函数。2013 年。网址：http://www.mesacc.edu/~scotz47781/mat120/notes/polynomials/evaluating/evaluating.html (于 2018 年 5 月 1 日访问）。

[Pik14] 斯科特派克。除以多项式。http://www.mesacc.edu/~scotz47781/mat120/notes/divide_poly/long_division/long_division.html。2014.（访问时间：2018-05-01）。

[大坝91] 伊万·达姆加德。“迈向实用的公钥系统，防止选择的密文攻击”。在：年度国际密码学会议。施普林格。1991 年，第 445-456 页。

[JSI96] Markus Jakobsson、Kazue Sako 和 Russell Impagliazzo。“指定验证者证明及其应用”。在：密码技术理论与应用国际会议。施普林格。1996 年，第 143-154 页。

[DBS04] Ratna Dutta、Rana Barua 和 Palash Sarkar。基于配对的加密协议：调查。密码学 ePrint 档案，报告 2004/064。https://eprint。iacr.org/2004/064。2004 年。

[DK18] Apoorvaa Deshpande 和 Yael Kalai。无知证明和 2 消息证人隐藏的应用。密码学 ePrint 档案，报告 2018/896。https://eprint.iacr.org/2018/896。2018 年。

[威尔16] 佐科·威尔科克斯。仪式的设计。2016 年。网址：https://z.cash/blog/the-design-of-the-ceremony/ (于 2018 年 5 月 1 日访问）。

[格罗16] 延斯·格罗斯。关于基于配对的非交互式参数的大小。密码学 ePrint 档案，报告 2016/260。https://eprint.iacr.org/2016/260。2016. 维基百科贡献者。约束满足。

[con18] 维基百科，免费的百科全书。2018.（访问时间：2018-08-05）。

[本+13] Eli Ben-Sasson、Alessandro Chiesa、Eran Tromer 和 Madars Virza。冯诺依曼架构的简洁非交互式零知识。密码学 ePrint 档案，报告 2013/879。https://eprint.iacr.org/2013/879。2013.延斯格罗斯和玛丽马勒。刻薄的签名：

[GM17] 来自模拟可提取 SNARK 的最小知识签名。密码学 ePrint 档案，报告 2017/540。https://eprint.iacr.org/2017/540。2017. Jens Groth、Markulf Kohlweiss、Mary

[格罗+18] Maller、Sarah Meiklejohn 和 Ian Miers。可更新和通用的通用参考字符串，适用于 zk-SNARKs。 密码学 ePrint 档案，报告 2018/280。https://eprint.iacr.org/2018/280。2018 年。

[本+17] Benedikt Bünz、Jonathan Bootle、Dan Boneh、Andrew Poelstra、Pieter Wuille 和 Greg Maxwell。Bulletproofs：机密交易的简短证明等。 密码学 ePrint 档案，报告 2017/1066。https://eprint.iacr.org/2017/1066。2017 年。

[本+18] Eli Ben-Sasson、Iddo Bentov、Yinon Horesh 和 Michael Riabzev。可扩展、透明和后量子安全的计算完整性。密码学 ePrint 档案，报告 2018/046。https://eprint.iacr.org/2018/046。2018 年。

[马尔+19] Mary Maller、Sean Bowe、Markulf Kohlweiss 和 Sarah Meiklejohn。Sonic：来自线性大小通用和可更新结构化参考字符串的零知识 SNARK。密码学 ePrint 档案，报告 2019/099。https://eprint.iacr。org/2019/099。2019 年。