zk-snark原理分析

引言

[问题]:什么是零知识证明?

> 《一千零一夜》里的零知识证明：阿里巴巴与四十大盗的故事其中一个片段。
> 阿里巴巴会芝麻开门的咒语，强盗向他拷问打开山洞石门的咒语，他不想让人听到咒语，便对强盗说：「你们离我一箭之地，用弓箭指着我，你们举起右手，我念咒语打开石门，举起左手，我念咒语关上石门，如果我做不到或逃跑，你们就用弓箭射死我。」
> 这个方案对阿里巴巴没损失，也能帮助他们搞清楚阿里巴巴到底是否知道咒语，于是强盗们同意。强盗举起了右手，只见阿里巴巴的嘴动了几下，石门打开了；强盗举起了左手，阿里巴巴的嘴动了几下，石门又关上了。强盗有点不信，没准这是巧合，多试几次过后，他们相信了阿里巴巴。

通过另一个维度来证明了本维度的信息,但没有泄露信息

零+知识+证明=无+信息+可验证过程（P?=NP）

[问题]:什么是zk-snark?

zk-snark:是zero-knowledge succint non-interactive arguments of knowledge的缩写.是多种零知识证明设计方案中的一种.它是一种基于多项式构造的零知识证明

- zero knowledge：零知识证明。
- succinct：简明的，证据信息较短，方便验证。
- non-interactivity：非交互的，证明者只要提供一个字符串，可放在链上公开验证。
- arguments：证明过程是计算完好（computationally soundness）的，证明者无法在合理的时间内造出伪证（破解）
- of knowledge：对于一个证明者来说，在不知晓特定证明 (witness) 的前提下，构建一个有效的零知识证据是不可能的

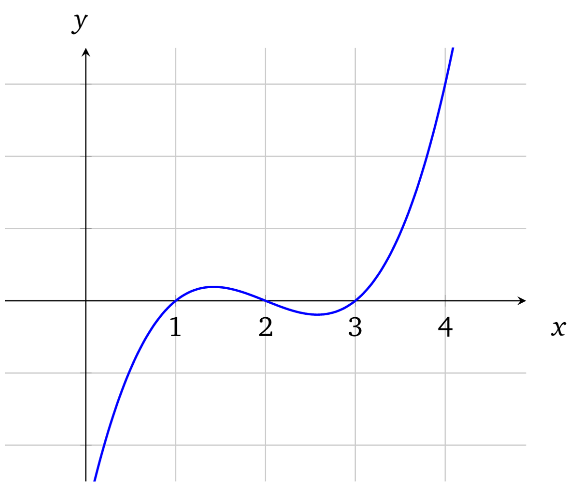
原理分析

[问题]:如何证明某个秘密?

基础：如何证明某个秘密多项式

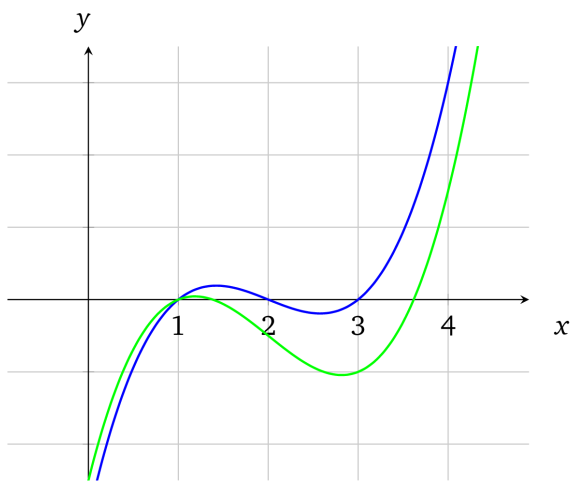
[知识]:多项式 : $f(x) = c_0x^0 + c_1x^1 + c_2x^2 + ...c_dx^d$, x的最大指数为多项式的阶

例如：*f(x) = x³ – 6x² +11x– 6* , 该多项式的阶等于3



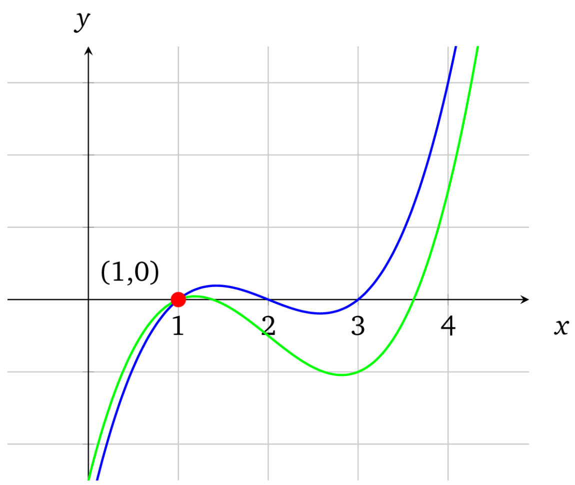
[知识]:两条不同的多项式曲线，不可能完全重合(他们只会相交于一些点)

例如：*f1(x) = x³ – 6x² +11x– 6* 和*f2(x) = x³ – 6x² +11x– 5*



[知识]:求解两个多项式交点的方法是令两个多项式相等 例如：$x^3 - 6x^2 + 11x - 6 = x^3 - 6x^2 + 10x - 5$

$=> x = 1,$ 因此两个多项式交点为 $(1, 0)$



[知识]:任意一个由阶数为 *d* 的多项式组成的等式，都可以化简为另外一个阶数至多为 *d* 的多项式

$f_1(x) = f_2(x) => f(x) = f1(x) - f2(x)$

例如：$x^3 - 6x^2 + 11x - 6 = x^3 - 6x^2 + 10x - 5 => x - 1 = 0$

[知识]:对于一个阶数为 *d* 的多项式至多有 *d* 个解,因此两个d阶多项式至多有d个交点

[知识]:如果一个阶数为d的多项式曲线上的点的个数为P，则随机取一个点，抽取到交点的概率为d / p，因此用一个多项式伪造另一个多项式，被随机抽查通过的概率为d/p

[问题]: 假设证明者A知道一个秘密多项式，验证人B如何验证A知道该秘密多项式?

方法一：A直接公布秘密多项式的所有系数(**知道某个多项式等价于知道该多项式的所有系数**)

方法二：设计一个验证协议

[问题]: 如何设计一个验证协议?

假设A声称知道一个d阶秘密多项式(B已知该秘密多项式,也可以只知道某些特殊点的多项式计算结果).

验证协议V1.0

验证步骤:

- B随机选择一个特殊点 x， 并记录该 x 对应的多项式计算结果f(x)
- B把x发送给A,并由其计算相应的多项式结果f`(x)
- A将 f`(x)发送给B
- B验证 f`(x)是否等于f(x). 如果等于，则证明A确实知道该秘密多项式，否则不知道

存在一些问题:

- 无法保证A使用秘密多项式计算f`(x)
- B与A可以相互破解秘密多项式(即求解代数方程)

  ......

[知识]:如果一个多项式有解，则可以因式分解成线性多项式 $f(x)=(x-a_0)(x-a_1)\ldots(x-a_n)=0, a_i$代表多项式的解

[知识]:如果一个多项式有某些解，则其可以被其目标多项式整除

$f(x)=h(x)t(x)$,目标多项式：$t(x)=(x-a_0)(x-a_1)\ldots(x-a_n), a_i$代表多项式的某些解

验证协议V2.0

验证步骤:

- B 挑选一个随机值 *r*, 计算 $t=t(r)$ 并记录， 然后将 *r* 发送给 A
- A 计算 $h(x)=f(x)/t(x)$，$f=f(r)$和$h=h(r)$，并将计算结果 *f*, *h* 发送给 B
- B 验证 $f=th$，如果多项式相等，则A知道秘密多项式

存在一些问题:

- 由于r的暴露，A可以任意伪造一个h,使得 *f*= *t*·*h*成立
- A的秘密多项式阶数不可控
- B与A可以相互破解秘密多项式

  ……

以上验证协议协议最大的问题就是暴露了随机值r以及由r计算得到的计算值。

假设存在一个黑盒可以在不泄露r以及由r计算出的一系列的h(r),f(r)等值的情况下,验证*f(r)= t(r)·h(r)*,则可以避免以上存在的大部分问题

[知识]:同态隐藏

如果加密函数E(x)满足以下特性,则认为有同态性

加法同态：$E(a+b)=E(a)+E(b)$

乘法同态：$E(ka)=kE(a)$

双线性映射：假设有两个群$G_1, G_2, G_1 * G_1=G_2, g$是$G_1$的生成元，$e(g,g)$是$G_2$的生成元，则$e(g^a,g^b)=e(g,g)^{ab}=e(g^b,g^a)$

验证协议V3.0

验证步骤:

- B挑选一个随机值 s, 并秘密隐藏起来.
- B计算s对应的各个指数次幂的加密值$\{E(s^i), i\in[0,d]\}$以及目标多项式t(s)的加密值E(t(s)),并将这些加密值发送给A.
- A利用$\{E(s^i), i\in[0,d]\}$,计算f(s)的加密值E(f(s)), $E(f(s))=a_0E(s^0)+a_1E(s^1)+\ldots a_dE(s^d)$以及h(s)的加密值E(h(s)),并将其发送给B
- B验证$E(f(s))=E(h(s))*E(t(s))$,则可以判定A是否知道秘密多项式

存在一些问题:

- 忽略多项式中的几项来伪造加密值
- 暴力破解

  ……

[知识]:KOE假设

α对:假设满足加密函数E(x) 的两个点 a,b , 如果 b=αa ,则称(a,b)为α对

KCA过程

- B随机选择一个α生成α对（a，b），α自己保存，（a，b）发送给A
- 由于同态函数的乘法性质，A无法算出α是什么.只能使用参数γ生成(a',b')=(γ·a,γ·b)，把(a',b')回传给B.
- (a',b')也是一个α对，b'=γ·b=γα·a=α(γ·a)=α·a'
- B校验(a',b')是否是α对,即$e(b',g)=e(a',g^{\alpha})$,如果相等,则可以认为A知道某个数γ，且不会向B泄露γ具体的值

d-KCA过程

- B发送一系列的α对(a1,b1)，(a2,b2)给A
- A使用c数组[c1,c2]，生成新的α对(a',b')=(c1·a1+c2·a2,c1·b1+c2·b2)，把(a',b')回传给B。
- (a',b')也是一个α对，b'=c1·b1+c2·b2=c1·α·a1 + c2·α·a2=α·(c1·a1 + c2·a2)=α·a'
- B校验(a',b')是否是α对,即$e(b',g)=e(a',g^{\alpha})=e(g,g)^{\alpha\cdot(c1\cdot a1+c2\cdot a2)}$，可以断言A知道某个c数组[c1,c2],且不会向B泄露c具体的值

KCA过程和d-KCA过程也称为KOE假设,其作用是保证确实使用多项式进行计算,但无法保证是否使用了正确的多项式,使用的多项式完全可能是伪造的.

例如：多项式$f(x)=1+x^1+x^2$, A宣称知道该多项式，即知道系数$[1,1,1]$, B为了验证发送了$[E(s^0),E(s^1),E(s^2)]$

以及$[E(\alpha s^0),E(\alpha s^1),E(\alpha s^2)]$，但是$A$按照错误的多项式关系$f'(x)=1+x^1$计算$\alpha$对，此时$d-KCA$过程依然成立，

B无法发现A遗漏了x^2项

验证协议V4.0(多项式盲证)

验证步骤:

- B挑选随机值 s,α, 并秘密隐藏起来.
- B计算s对应的各个指数次幂的加密值$\{E(s^i), i\in[0,d]\}$以及相应的α对值$\{E(\alpha s^i), i\in[0,d]\}$ 和目标多项式t(s)的加密值E(t(s)),并将这些加密值发送给A.

- A利用$\{E(s^i), i \in [0,d]\}$,计算E(f(s)), $E(f(s)) = a_0E(s^0) + a_1E(s^1) + \dots a_dE(s^d)$,并将其发送给B
- A利用$\{E(\alpha s^i), i \in [0,d]\}$,计算E(αf(s)), ,并将其发送给B
- A利用$\{E(s^i), i \in [0,d]\}$,计算h(s)的加密值E(h(s)),并发送给B
- B验证E(f(s))和E(αf(s))是否是α对
- B验证$E(f(s)) = E(h(s)) * E(t(s))$,则可以判定A是否知道秘密多项式

存在一些问题:

- 暴力破解(多项式阶数比较小的情况下)
- 某些特定值也能通过验证

  ......

扩展:一般秘密计算问题的证明

[问题]:如何将一般秘密计算证明问题转化成秘密多项式证明问题?

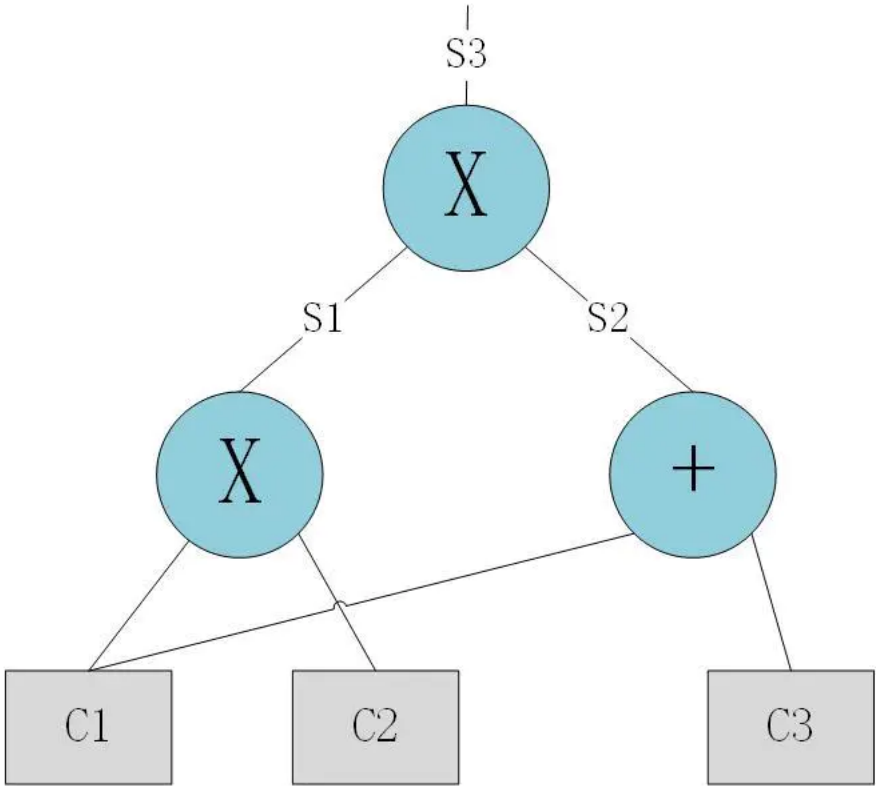
[知识]:R1CS电路描述

关系:一般秘密计算问题中秘密中的变量总会存在一些潜在的等式关系

例如:假定A需要向B证明他知道c1，c2，c3，使(c1·c2)·(c1+c3)=7，c1，c2，c3需要对B保密.

(c1·c2)·(c1+c3)=7就是c1，c2，c3的等式关系.

R1CS电路描述:等式关系本身对应的是一种算数运算过程, 由于和逻辑电路很相似,所以将其成为算数电路,并用特殊的方式来表示算数电路,这种表示方法成为称为R1CS电路描述.



等式关系可能比较复杂,因此还需要使用一种特定的**拍平**操作将其进行化简

拍平操作:即通过增加中间变量,把复杂等式关系化简(**降阶**)成最简形式的**约束**

例如:

$$(c1 \cdot c2) \cdot (c1 + c3) = 7 => \begin{cases} S1 = C1 * C2 \\ S2 = C1 + C3 \\ S3 = S1 * S2 \end{cases}$$

使用向量$V = [V_0 = \text{常数}1, V_1, V_2, \dots, V_d]$,表示约束中的所有变量.再把每一个门电路表示为等价的向量点积形式.对每个门电路，我们定义一组向量（A,B,C），使得$V.A * V.B - V.C = 0$

例如:

$$\begin{cases} S1 = C1 * C2 \\ S2 = C1 + C3 \\ S3 = S1 * S2 \end{cases} => A = \begin{bmatrix} 0,1,0,0,0,0,0 \\ 1,0,0,0,0,0,0 \\ 0,0,0,0,1,0,0 \end{bmatrix} B = \begin{bmatrix} 0,0,1,0,0,0,0 \\ 0,1,0,1,0,0,0 \\ 0,0,0,0,0,1,0 \end{bmatrix} C = \begin{bmatrix} 0,0,0,0,1,0,0 \\ 0,0,0,0,0,1,0 \\ 0,0,0,0,0,0,1 \end{bmatrix}$$

[知识]:QAP二次算数程序

QAP就是将向量的验证转化为多项式的验证的过程,其核心是使用拉格朗日插值法.

拉格朗日插值法:提出是对于某一个未知函数的一组观测或者实验数据，寻找一个多项式函数，使这个多项式函数,即 $Z(x) = [Z_1(x), Z_2(x), \ldots\ldots Z_d(x)]$

通过拉格朗日插值法三个向量组表示成3个多项式数组,即

$A(x) = [A_1(x), A_2(x), \ldots\ldots A_d(x)]$
$B(x) = [B_1(x), B_2(x), \ldots\ldots B_d(x)]$
$C(x) = [C_1(x), C_2(x), \ldots\ldots C_d(x)]$

因此原本向量组的验证关系转化成一个多项式的验证关系:

$f(x) = V.A(x) * V.B(x) - V.C(x) = H(x) * t(x)$
当 $x = 1, 2, \ldots$ 时, $f(x) = 0$

假设 $L(x) = V.A(x) = \sum_{i=0}^{m} V_i A_i(x), R(x) = V.B(x) = \sum_{i=0}^{m} V_i B_i(x), O(x) = V.C(x) = \sum_{i=0}^{m} V_i C_i(x)$
则 $f(x) = L(x) * R(x) - O(x)$. 此多项式最高阶为 $2d$

原多项式盲证验证逻辑也等价于

$e(g^{L(s)}, g^{R(s)}) = e(g,g)^{L(s)R(s)} = e(g^{t(s)}, g^{h(s)})e(g^{O(s)}, g) = e(g,g)^{t(s)h(s)}e(g^{O(s)}, g)$,3次双线性映射

通用验证协议V1.0

初始化设置

- 生成随机数s, α
- 计算生成证明密钥:$\{g^{s^i} | i \in [0,d]\}, \{g^{\alpha s^i} | i \in [0,d]\}, g^{l_i(s)}, g^{r_i(s)}, g^{o_i(s)}, g^{\alpha l_i(s)}, g^{\alpha r_i(s)}, g^{\alpha o_i(s)}\}$
- 计算生成验证密钥:$E(t(s)) = g^{t(s)}, E(\alpha) = g^{\alpha}$

证明人

- 假设变量为$V = \{v_i, i \in [0,d]\}$
- 计算$h(x) = (l(x)r(x) - o(x))/t(x)$,并利用证明密钥计算$g^{H(s) = \prod_0^d g^{h_i s^i}}$
- 计算隐私输入证明值$g^{L(s)} = \prod_{i=0}^{d} (g^{v_i l_i(s)})$,同理$g^{R(s)}, g^{O(s)}$
- 计算隐私输入证明值的α对$g^{L'(s)} = \prod_{i=0}^{d} (g^{v_i \alpha l_i(s)})$,同理$g^{R'(s)}, g^{O'(s)}$
- 生成证明$\pi = \{g^{L(s)}, g^{R(s)}, g^{O(s)}, g^{L'(s)}, g^{R'(s)}, g^{O'(s)}, g^{H(s)}\}$

验证者

- 获取证明$\pi = \{g^{L(s)}, g^{R(s)}, g^{O(s)}, g^{L'(s)}, g^{R'(s)}, g^{O'(s)}, g^{H(s)}\}$
- 检查α对$e(g^{L'(s)}, g) = e(g^{L(s)}, g^{\alpha})$
- 检查α对$e(g^{R'(s)}, g) = e(g^{R(s)}, g^{\alpha})$
- 检查α对$e(g^{O'(s)}, g) = e(g^{O(s)}, g^{\alpha})$
- 检查多项式等式$e(g^{L(s)}, g^{R(s)}) = e(g,g)^{t(s)h(s)}e(g^{O(s)}, g)$

存在一些问题:

- 暴力破解(多项式阶数比较小的情况下)
- 某些特定值也能通过验证

  ......

[知识]:匹诺曹协议

以上的验证协议还存在存以下一些问题:

1)$L(x), R(x), O(x)$需要保证其是由多项式计算得到,否则可能出现拼凑的多项式计算结果

例如E(L(s))=E(t(s)) E(R(s))=E(2) E(O(s))=E(t(s)),E(h(s))=E(1)

$e(g^{L(s)}, g^{R(s)}) = e(g,g)^{L(s)R(s)} = e(g,g)^{2t(s)} = e(g^{t(s)}, g^{h(s)}))e(g^{O(s)}, g^1)) = e(g^{t(s)}, g^1))e(g^{t(s)}, g^1))$

因此利用d-KCA过程的特性,证明人需要提供可供验证的α对 $(E(L(s), E(\alpha L(s))), (E(R(s), E(\alpha R(s))), (E(O(s), E(\alpha O(s)))$

但是$L(x), R(x), O(x)$如果使用相同的α对来验证,则证明者可以通过一定的技巧通过d-KCA 验证过程

例如:原有的多项式等式关系变更为$O(x)R(x) - L(x) = H(x)T(x)$,此时相关多项式计算结果为 $L(x) = \sum_{i=0}^{m} V_i C_i(x), L'(x) = \sum_{i=0}^{m} V_i \alpha C_i(x)$,由于α对相同,d-KCA 验证过程依旧成立

因此需要提供不同的α对来解决,即 $(E(L(s), E(\beta_l L(s))), (E(R(s), E(\beta_r R(s))), (E(O(s), E(\beta_o O(s)))$

2)$L(x), R(x), O(x)$需要保证使用同一个向量V生成而不是使用不同的向量V来绕过约束检查.

$L(x), R(x), O(x)$作为多项式存在线性相关性,可以产生一个新的线性关系$L(x) + R(x) + O(x)$.

通过对该线性关系的d-KCA过程可以来保证$L(x), R(x), O(x)$在计算时使用了相同的向量V,即添加一个额外的β对

$Z(s) = \sum_0^d g^{z_i(s)} = \sum_0^d g^{\beta v_i (l_i(s) + r_i(s) + O_i(s))}$

验证$e(g^{v_i(l_i(s) + r_i(s) + O_i(s))}, g^{\beta}) = e(g^{Z_i(s)}, g)$

但是由于多个多项式的线性相加会存在合并同类项的情况

例如:

$$L(x) * R(x) - O(x) = \sum_{i=0}^{m} V_i A_i(x) \sum_{i=0}^{m} V_i B_i(x) - \sum_{i=0}^{m} V_i C_i(x)$$

在$L(x) = R(x)$的情况下,可以使用不同的向量V_l, V_r, V_o, 且 $V_1 = 2V_o - V_r$,此时验证等式依然可能成立

因此需要给每个多项式添加单独偏置,即$Z(s)=\beta_l L(s)+\beta_r R(x)+\beta_o O(x)$

验证$e(g^{v_i(\beta_l l_i(s)+\beta_r r_i(s)+\beta_o O_i(s))},g)=e(g^{Z_i(s)},g)$

3)$L(x),R(x),O(x)$存在多项式变形的可能性,

例如:$L(x),R(x),O(x)$同时产生相同偏移,此时由于随机数加密值是暴露的,可以构造 $(E(\beta L(s)+\beta)$,此时$Z(s)=\beta_l L(s)+\beta_r R(x)+\beta_o O(x)+\beta_l+\beta_r++\beta_o$也可以被构造出来,从而通过一致性验证.因此需要将单独偏执的加密值保护起来,比如乘以一个新的随机数γ,

验证$e(g^{L(s)},g^{\beta_l\gamma})e(g^{R(s)},g^{\beta_r\gamma})e(g^{O(s)},g^{\beta_o\gamma})=e(g^{Z(s)},g^{\gamma})$,4次双线性映射

4)优化双线性映射次数

双线性映射属于比较负责的运算，会消耗一定的运算资源。之前的计算都是按照同一个生成元进行同态加密运算的，需要消耗四次双线性映射，如果采用不同生成元运算可以减少双线性运算次数，从而减少消耗。

通用验证协议V2.0

初始化设置

- 生成随机数$\alpha_l,\alpha_r,\alpha_o,\beta,\gamma,\rho_l,\rho_r,以及\ \rho_o=\rho_l*\rho_r$
- 生成三个生成元$g_l=E(\rho_1)=g^{\rho_l},g_r=E(\rho_r)=g^{\rho_r},g_o=E(\rho_o)=g^{\rho_o}$
- 生成一组证明密钥$E(s^i)=g^{s^i},i\in[0,d],\{g_l^{l_i(s)},g_r^{r_i(s)},g_o^{o_i(s)},g_l^{\alpha_l l_i(s)},g_r^{\alpha_r r_i(s)},g_o^{\alpha_o o_i(s)},g_l^{\beta l_i(s)},g_r^{\beta r_i(s)},g_o^{\beta o_i(s)}\}$
- 生成一组验证密钥$E(t(s))=g_o^{t(s)},E(\alpha_1)=g^{\alpha_l},E(\alpha_r)=g^{\alpha_r},E(\alpha_o)=g^{\alpha_o},E(\beta\gamma)=g^{\beta\gamma},E(\gamma)=g^{\gamma}$

证明人

- 假设变量为$V=\{v_i,i\in[0,d]\}$
- 计算$h(x)=(l(x)r(x)-o(x))/t(x)$,并利用证明密钥计算$g^{H(s)=\prod_0^d g^{h_i s^i}}$
- 计算隐私输入证明值$g_l^{L(s)}=\prod_{i=0}^d(g_l^{v_i l_i(s)})$,同理$g_r^{R(s)},g_o^{O(s)}$
- 计算隐私输入证明值的α对$g_l^{L'(s)}=\prod_{i=0}^d(g_l^{v_i\alpha l_i(s)})$,同理$g_r^{R'(s)},g_o^{O'(s)}$
- 计算一致性检查值$g^{Z(s)}=\prod_{i=0}^d(g_l^{\beta l_i(s)}g_r^{\beta r_i(s)}g_o^{\beta l_o(s)})^{v_i}$
- 生成证明$\pi=\{g_l^{L(s)},g_r^{R(s)},g_o^{O(s)},g_l^{L'(s)},g_r^{R'(s)},g_o^{O'(s)},g^{H(s)},g^{Z(s)}\}$

验证者

- 获取证$\pi=\{g_l^{L(s)},g_r^{R(s)},g_o^{O(s)},g_l^{L'(s)},g_r^{R'(s)},g_o^{O'(s)},g^{H(s)},g^{Z(s)}\}$
- 验证是否采用多项式计算: $e(g_l^{L'(s)},g)=e(g_l^{L(s)},g^{\alpha_l})$,同理R,O
- 验证是否一致性向量V:$e(g_l^{L(s)}g_r^{R(s)}g_o^{O(s)},g^{\beta\gamma})=e(g^{Z(s)},g^{\gamma})$
- 验证多项式等式成立:$e(g_l^{L(s)},g_r^{R(s)})=e(g_o^{t(s)},g^{H(s)})e(g_o^{O(s)},g)$

[知识]:随机偏置

为了防止暴力破解以及加密证明的滥用,还需引入随机偏置

$假设\ L'(x)=L(x)+\delta_1 T(x),R'(x)=R(x)+\delta_2 T(x),O'(x)=O(x)+\delta_3 T(x)$
$则\ f'(x)=L'(x)*R'(x)-O'(x)=(L(x)+\delta_1 T(x))(R(x)+\delta_2 T(x))-(O(x)+\delta_3 T(x))=(L(x)R(x)-O(x))+L(x)\delta_2 T(x)+\delta_1 T(x)R(x)+\delta_1\delta_2 T(x)$
$=T(x)(H(x)+L(x)\delta_2+R(x)\delta_1+\delta_1\delta_2 T(x)-\delta_3)$
$令\ H'(x)=(H(x)+L(x)\delta_2+R(x)\delta_1+\delta_1\delta_2 T(x)-\delta_3)$
$则\ f'(x)=L'(x)*R'(x)-O'(x)=H'(x)T(x)依然可以验证$

此外,以上的推导都是默认向量V中的元素都是隐私输入,但是在实际使用中有些元素是可以作为公开输入,从而实现复用等目的.因此,向量V可以被分成两部分$V_p=\{v_i,i\in[0,m]\}和\ V_v=\{v_i,i\in(m+1,d]\}$,同理之后所用到的$L(x),R(x),O(x)$等多项式也可以分解成两部分.在验证时,由验证方计算相应的向量乘积(SA(x)SB(x)-SC(x)=H(x)T(x))即可.

通用验证协议(终结版)

初始化过程

选择生成元g和一个配对函数e()

将一个函数进行QAP转换

生成随机数$\alpha_l,\alpha_r,\alpha_o,\beta,\gamma,\rho_l,\rho_r,以及\ \rho_o=\rho_l*\rho_r$

生成三个生成元$g_l=E(\rho_1)=g^{\rho_l},g_r=E(\rho_r)=g^{\rho_r},g_o=E(\rho_o)=g^{\rho_o}$

生成一组证明密钥
$E(s^i)=g^{s^i},i\in[0,d],\{g_l^{l_i(s)},g_r^{r_i(s)},g_o^{o_i(s)},g_l^{\alpha_l l_i(s)},g_r^{\alpha_r r_i(s)},g_o^{\alpha_o o_i(s)},g_l^{\beta l_i(s)},g_r^{\beta r_i(s)},g_o^{\beta o_i(s)}|i\in(m+1,d]\},g_l^{t(s)},g_r^{t(s)},g_o^{t(s)},g_l^{\alpha_l t(s)},g_r^{\alpha_r t(s)},g_o^{\alpha_o t(s)},g_l^{\beta t(s)},g_r^{\beta t(s)},g_o^{\beta t(s)}$

生成一组验证密钥$E(t(s))=g_o^{t(s)},E(\alpha_1)=g^{\alpha_l},E(\alpha_r)=g^{\alpha_r},E(\alpha_o)=g^{\alpha_o},E(\beta\gamma)=g^{\beta\gamma},E(\gamma)=g^{\gamma},\{g_l^{l_i(s)},g_r^{r_i(s)},g_o^{o_i(s)}|i\in[0,m]\}$

证明人

获取向量$V=\{v_i,i\in[0,d]\}$

使用向量V计算生成$L(x),R(x),O(x)$

生成本地随机数$\delta_1, \delta_2, \delta_3$

计算$H(x) = (\frac{L(x)R(x)-O(x)}{t(x)} + L(x)\delta_2 + R(x)\delta_1 + \delta_1\delta_2T(x) - \delta_3)$,以及H(s)

计算隐私输入证明值$g_l^{L_p(s)} = (g_l^{\delta_1 t(s)})\prod_{i=m+1}^{d}(g_l^{v_i l_i(s)})$,同理$g_r^{R_p(s)}, g_o^{O_p(s)}$

计算隐私输入证明值的α对$g_l^{L'_p(s)} = (g_l^{\delta_1 \alpha_l t(s)})\prod_{i=m+1}^{d}(g_l^{v_i \alpha_l l_i(s)})$,同理$g_r^{R'_p(s)}, g_o^{O'_p(s)}$

计算线性相加值$g^{Z(s)} = (g_l^{\delta 1 \beta t(s)})(g_r^{\delta 2 \beta t(s)})(g_o^{\delta 3 \beta t(s)})\prod_{i=m+1}^{d}(g_l^{\beta v_i l_i(s)})(g_r^{\beta v_i r_i(s)})(g_o^{\beta v_i o_i(s)})$

生成证明$\pi = \{g_l^{L_p(s)}, g_r^{R_p(s)}, g_o^{O_p(s)}, H(s), g_l^{L'_p(s)}, g_r^{R'_p(s)}, g_o^{O'_p(s)}, g^{Z(s)}\}$

验证者

获取证明$\pi = \{g_l^{L_p(s)}, g_r^{R_p(s)}, g_o^{O_p(s)}, H(s), g_l^{L'_p(s)}, g_r^{R'_p(s)}, g_o^{O'_p(s)}, g^{Z(s)}\}$

计算公共输入证明值$g_l^{L_v(s)} = \prod_{i=0}^{m}(g_l^{v_i l_i(s)})$,同理$g_r^{R_v(s)}, g_o^{O_v(s)}$

验证多项式计算α对 $e(g_l^{L_p(s)}, g^{\alpha_l}) = e(g_l^{L'(s)}, g)$,同理验证$g_r^{R_p(s)}, g_o^{O_p(s)}$

验证多项式一致性$e(g_l^{L(s)} g_r^{R(s)} g_o^{O(s)}, g^{\beta\gamma}) = e(g^{Z(s)}, g^{\gamma})$

验证多项式等式:$e(g_l^{L_v(s)} g_l^{L_p(s)}, g_r^{R_p(s)} g_r^{R_v(s)}) = e(g_o^{t(s)}, g^{h(s)})e(g_o^{O_p(s)} g_o^{O_v(s)}, g)$

[知识]:可信初始过程CRS公共字符串

以上协议中初始化过程中生成的所有随机值都必须在被加密后公开, 加密处理后的密文被称为CRS公共字符串,而原始的随机值被称为有毒废料,必需被舍弃,否则一旦被泄露,则有证明造假的风险.

为了保证这些有毒废料被真正移除,需要借助MPC多方协同计算来保证安全.

性能对比

	Size (approximately)	Runtime (minutes, estimate)	
	Proof	**Prover**	**Verifier**
Fractal	250 kB		
Halo	20 kB		
SuperSonic	8 kB		75ms
Plonk	1 kB		
Marlin	1 kB		8-20ms
Sonic	2 kB		
Groth16	0.2 kB	1-2 minutes	1-10ms
zk-STARK	>100 kB		

应用场景

- zcash:混币
- zk-rollup:2层
- zk-swap:匿名交易
- zk-vm:智能合约

优化路线

- 计算速度:矩阵计算(快速傅立变换)\并行计算
- 初始化可信设置:通用可信化设置\无可信化设置

附加

常用R1CS电路

范围证明

证明某个私密输入w的值在0到15之间，约束构建如下:

$$
\begin{gathered}
2^0 \cdot w_0 + 2^1 \cdot w_1 + 2^2 \cdot w_2 + 2^3 \cdot w_3 - w = 0 \\
w_0 \cdot (w_0 - 1) = 0 \\
w_1 \cdot (w_1 - 1) = 0 \\
w_2 \cdot (w_2 - 1) = 0 \\
w_3 \cdot (w_3 - 1) = 0
\end{gathered}
$$

用途：金额证明

参考文献

【1】Why and How zk-SNARK Works: Definitive Explanation