Reading Notes　Anonymous Voter Model: SEAL

August 24, 2023

这篇文章是 2019-1332 SEAL: Sealed-Bid Auction without Auctioneers 的读书笔记, 作者在这篇文章中提出了一种全新的去中心化密封出价拍卖协议, 具有线性的计算和通信复杂度 $O(c)$, 其中 c 是出价价格的位长度. 同时, 该协议通过一个巧妙的子协议构造实现了完全的去中心化计算. 不过, 在阅读本文时笔者发现该子协议有可以优化计算复杂度的方案, 同时对于原协议在参与方从多个退化至两个时会导致不可用的情况做出了修正, 这些想法会在文章的第三部分涉及.

多方安全匿名投票协议是密码学中的一项关键技术，通过允许多个参与方共同进行投票，同时保护每个投票的隐私. 在很多情况下，例如选举、调查和共同决策中, 这种协议可以确保投票的公平性和透明度, 同时防止任何潜在的欺诈行为. 在此之前的工作有这样几个思路: 利用公钥密码学和盲签名的方式对选票进行盲化处理最终汇集; 利用零知识证明来确保投票的有效性和匿名性, 而不泄露任何有关投票内容的信息; 通过混合网络 (可信计票中心持有) 对投票进行重新排序和加密, 以确保投票的匿名性; 利用区块链的透明性和不可更改性来确保投票记录的公开和永久保存, 等等. 基于多方安全计算的协议研究此前也有不少, 但始终难以绕过需要可信执行第三方以及无法让所有参与方获得结果的问题. 但在这篇文章中，作者介绍了一种新的多方安全匿名投票协议, 该协议不依赖于任何中央权威或可信第三方, 并且利用其机制设计出了一个参与方可以一起计算结果的匿名竞价协议. 在开始协议讲解之前, 我们同上一篇读书笔记一样先对匿名投票协议模型进行描述:

定义 (多方匿名投票协议) 多方匿名投票协议 MAV 由一个元组 (Setup, DistVote, Tally) 给出:

- $\text{Setup}(1^\kappa, n, t) \to (\llbracket \mathbf{pk} \rrbracket_{[n]}, \llbracket \mathbf{sk} \rrbracket_{[n]}, pp)$: Setup 是一个随机算法, 接受安全参数作为输入, 并输出 n 个公钥和私钥对以及公共参数 pp. 第 i 个公钥和私钥对分配给第 i 个参与方.
- $\text{DistVote}(\llbracket \mathbf{pk} \rrbracket_{[n]}, [j : v, S], pp) \to [j : b/\perp]$: DistVote 是一个分布式协议, 通过它, 参与方 j 在集合 S 中的参与方的帮助下投票 v。在协议结束时, j 输出投票的凭证 b(或 $\perp$ 表示失败). 所有其他参与方没有输出.
- $\text{Tally}(\llbracket \mathbf{pk} \rrbracket_{[n]}, \llbracket \mathbf{sk} \rrbracket_{[n]}, [b_1, \ldots, b_n], pp) \to [r/\perp]$: Tally 是一个分布式协议, 通过它, 所有参与方共同计算投票结果. 在协议结束时, 输出投票结果 r(或 $\perp$ 表示失败).

安全匿名投票协议通常需要满足以下几个关键属性:

完整性: 所有合法的投票都被准确计算，并且没有被篡改. 数学上，可以表示为:

$$\forall v \in V, \text{Tally}(\llbracket \mathbf{pk} \rrbracket_{[n]}, \llbracket \mathbf{sk} \rrbracket_{[n]}, [b_1, \ldots, b_n], pp) = \text{true\_result}(v),$$

其中 V 是所有合法投票的集合, true_result 是真实投票结果的函数.

隐私性: 投票者的身份和投票内容之间的关联不被泄露. 数学上，可以表示为:

$$\forall i, j \in [n], i \neq j \Rightarrow \neg \exists A : A(\llbracket \mathbf{pk} \rrbracket_{[n]}, b_i, b_j) = (i, v_i) \text{ 或} (j, v_j),$$

其中 A 是任何潜在的攻击算法.

不可否认性: 投票者不能否认自己的投票. 数学上, 可以表示为:

$$\forall i \in [n], \neg \exists A : A(\llbracket \mathbf{pk} \rrbracket_{[n]}, b_i) \neq v_i,$$

其中 A 是任何潜在的攻击算法.

公平性: 没有人可以在投票结束前得知投票结果. 数学上, 可以表示为:

$$\forall i \in [n], \neg \exists A : A(\llbracket \mathbf{pk} \rrbracket_{[n]}, [b_1, \ldots, b_{i-1}]) = \text{true\_result}(v),$$

其中 A 是任何潜在的攻击算法.

鲁棒性: 协议对恶意参与者的攻击具有抵抗力. 数学上, 可以表示为:

$$\forall i \in [n], \forall A : \Pr[A(\llbracket \mathbf{pk} \rrbracket_{[n]}, b_i) \text{ 成功攻击}] \leq \text{negl}(\kappa),$$

其中 A 是任何潜在的攻击算法, negl 是一个可以忽略的函数.

部分协议提出了抗胁迫性这一概念, 这是考虑到部分选民在参加投票时可能遭受到胁迫等情况, 例如著名的 Helios 互联网投票系统. 选民可以在受胁迫时选择提交一个特殊的 “胁迫证据”, 该证据看起来像是对特定候选人的投票, 但实际上不会计入最终结果. 不过很多协议对此性质并没有做特殊要求或者设计.

1 预备知识

1.1 Discrete Logarithm 离散对数困难问题

P 问题是指可以在多项式时间内解决的问题, 而 NP 问题是指可以在多项式时间内验证解的问题. 也就是说, 对于 NP 问题并没有确定步骤的求解方案, 只能通过猜测解然后通过多项式时间算法来对猜测的答案正确性进行验证, 现代密码学很多问题都是基于 NP 难问题求解困难性.

离散对数问题是一个典型的 NP 问题: 给定一个素数 p 和一个整数 g, 求解 x 使得 $g^x \equiv h \mod p$ 成立是非常困难的. 这个问题的困难性是许多密码学算法的基础. 在本文中, 我们需要利用到的性质除了上条还有如下一个设定:

给定一个群 G 和一个生成元 g, 以及两个整数 a 和 b, 我们可以容易地计算 g^a 和 g^b. 然而, 计算 g^{ab} (即求解离散对数问题的变体) 通常被认为是困难的. 这个问题在许多密码学应用中是基础问题, 例如在 Diffie-Hellman 密钥交换协议中. 尽管我们可以容易地计算出 g^a 和 g^b, 但是从这些值推导出 g^{ab} 在目前的技术下被认为是计算上不可行的, 除非我们知道 a 和 b 的具体值. 这个问题的困难性是许多现代密码学协议安全性的基础.

1.2 Vickrey Auction 第二高出价拍卖机制

Vickrey 投票, 也称为第二价格拍卖, 是一种独特的拍卖机制, 由经济学家威廉·维克里(William Vickrey) 于 1961 年提出. 这种拍卖方式的核心思想是鼓励参与者真实地报价, 即按照他们真实的估值进行出价. 原文的附加部分对初始协议作了适应 Vickrey 投票的模型修正, 所以在此处简单涉及.

拍卖流程

Vickrey 投票的方案相对简单, 具体步骤如下:

出价阶段: 参与者私下提交自己的出价. 这些出价秘密发送, 其他参与者不知道彼此出价.

确定赢家: 拍卖者收集所有的出价, 赢家是出价最高的参与者.

支付阶段: 赢家支付的价格不是其自己的出价, 而是第二高的出价.

博弈论原理

我们以一个简单的例子进行说明为何 Vickrey 投票能够鼓励真实出价:

假设有三个参与者 A、B 和 C, 他们对同一物品的真实估值分别为 100 元、80 元和 70 元. 如果他们都按照真实估值出价，那么 A 将以 80 元的价格赢得物品，这是第二高的出价. 如果 A 试图通过降低出价来节省费用, 例如出价 75 元, 那么 B 将以 75 元的价格赢得物品, 因为 A 的出价低于 B 的 80 元. 如果 A 试图通过提高出价来确保赢得物品, 例如出价 110 元, 那么 A 仍将以 80 元的价格赢得物品, 因为赢家支付的是第二高的出价. 因此, 无论 A 如何改变出价, 其支付的价格都是 80 元，或者失去赢得物品的机会. 这就是为什么真实出价是最优策略的原因. 该方案的博弈论原理可以归纳为以下两点:

无需猜测他人出价: 由于赢家支付的是第二高的价格, 因此参与者无需猜测其他人可能的出价来调整自己的出价. 他们只需考虑自己的真实估值.

真实出价是最优策略: 在 Vickrey 投票中，真实出价是支配策略，即无论其他人如何出价，真实出价都是最优选择. 如果出价低于真实估值, 可能会失去赢得物品的机会; 如果出价高于真实估值, 则可能会以高于真实估值的价格赢得物品.

综上, Vickrey 投票是一种有效的拍卖机制，能够激励参与者按照真实估值进行出价. 它消除了对其他参与者出价的猜测，确保了拍卖的公平性和透明度. 然而，它也有一些潜在的缺点，例如可能的串通行为，因此在实际应用中可能需要额外的机制来确保其有效性和安全性.

1.3 零知识证明

零知识证明 (Zero-Knowledge Proof) 由 S.Goldwasser, S.Micali 及 C.Rackoff 在 20 世纪 80 年代初提出, 证明者能够在不向验证者提供任何有用的信息的情况下, 使验证者相信某个论断是正确的. 零知识证明实质上是一种涉及两方或更多方的协议，即两方或更多方完成一项任务所需采取的一系列步骤. 在这一模型中, 通常存在两方, 需要证明自己论断的证明者与验证论断的验证者, 通常“P”表示“证明者 (Proofer)”, 作为零知识证明的参与方, 他需要在证明命题真实性的同时不会泄漏任何相关信息; “V”表示“验证者 (Verifier)”: 作为零知识证明的另一参与方, 验证证明者提出的命题以及对应的证明是不是正确.

一个零知识证明的过程通常包括如下几个阶段:

承诺阶段 (Commit): 证明者针对命题做出承诺, 并等待验证者提出挑战并进行验证.

挑战阶段 (Challenge): 验证者选择随机数, 对提出的承诺进行挑战.

回应挑战阶段 (Response): 证明者将收到的随机数并结合给出的承诺, 返回挑战的回应.

验证阶段 (Verify): 验证者验证, 挑战的回应是否正确, 错误的话那就证明失败, 如果成功就可进行下一次挑战, 直到可以相信的概率达到验证者接受的条件, 这样就证明成功.

同时, 在零知识证明中, 需要满足三个性质:

正确性: 没有人能够假冒 P 使这个证明成功. 如果不满足这条性质, 也就是 P 不知道“知识”, 再怎么证明, V 也很难相信 P 拥有正确的知识.

完备性: 如果 P 和 V 都是诚实的, 并证明过程的每一步都进行正确的计算, 那么这个证明一定是成功的. 也就是说如果 P 知道“知识”, 那么 V 会有极大的概率相信 P.

零知识性: 证明执行完之后, V 只获得了 “P 拥有这个知识” 这条信息, 而没有获得关于这个知识本身的任何信息.

具体的协议讲解会在 ZKP 专题科普中涉及, 此处暂不给出具体案例, 原文中所有的 ZKP 使用均可以通过非交互式 Schnorr 协议完成证明.

2 模型

2.1 子协议

作者设计的拍卖协议中的基本构建块是一个原始子协议, 该操作可以在不泄露每个单独位的情况下安全地计算二进制输入的逻辑 “或” 运算. 作者选择修改 Hao-Zielinski 的匿名否决网络 (AV-net) 协议来达到这一目的, 选择此协议是因为它在轮数、计算和通信带宽方面具有最佳效率. 然而, 在正式使用之前需要做出一部分调整来适应最终构建方案. 修改后的 AV-net 协议的工作方式如下:

假设有 n 个投票者希望找出是否有一个投票者想否决某项提案. 换句话说, 他们希望安全地计算许多输入位的逻辑 “或” 函数, 每个位来自一个单独的实体. 设 G 是一个包含 p 个元素的群, 其中决策性 Diffie-Hellman 问题 (1.1) 被认定为难解问题. 设 g 是 G 的随机生成元, G 中的所有计算都是关于素数模数 q 的模运算, 但为了简单起见, 这里省略了 mod q 表示法, 每个投票者 V_i 持有一个秘密位 $v_i \in \{0,1\}$, 并在两轮中计算逻辑 “或” $\bigvee_{i=1}^{n} v_i$. 对于任何两个整数 a 和 b, 其中 $a < b$, 我们用 $[a,b]$ 表示集合: $\{a, a+1, \ldots, b\}$.

第一轮: 每个投票者 $V_i : i \in [1,n]$ 选择两个随机元素 $(x_i, r_i) \in Z_p^2$, 并计算 $X_i = g^{x_i}, R_i = g^{r_i}$. V_i 将 (X_i, R_i) 连同非交互式零知识 (NIZK) 证明一起发布到公共信道上, 通过使用 Schnorr 签名方案分别证明 x_i 和 r_i 的合法构造.

第二轮: 每个投票者 $V_i : i \in [1,n]$ 计算 $Y_i = \prod_{j=1}^{i-1} X_j / \prod_{j=i+1}^{n} X_j$. V_i 同时计算一个加密的选票 b_i, 并将其连同 NIZK 证明一起发布到公共信道上, 以证明 b_i 的格式正确.

$$b_i = \begin{cases} Y_i^{x_i} & \text{if } v_i = 0 \\ R_i^{x_i} & \text{if } v_i = 1 \end{cases}$$

第二轮中的 NIZK 证明是为了证明以下陈述: $(b_i = Y_i^{x_i}) \vee (b_i = R_i^{x_i})$. 请注意, 第一项等同于证明 $\{X_i, Y_i, b_i = Y_i^{x_i}\}$ 是一个 DDH 元组, 第二项等同于证明 $\{X_i, R_i, b_i = R_i^{x_i}\}$ 是一个 DDH 元组. 因此, b_i 的格式正确性的 NIZK 证明是两个子陈述的析取证明. 有关零知识证明的更多细节可以在原文附录中找到.

第二轮后, 每个人都可以在从公告板获取选票 $b_i : i \in [1,n]$ 后计算 $B = \prod_{i=1}^{n} b_i$. 很容易看出, 如果每个输入都是 0, $B = 1$; 否则, 如果至少有一个输入是 1, 那么 $B \neq 1$ 的概率非常大 (即 B 是 G 中的随机元素). 因此, 所有输入位的逻辑 “或” 已由所有参与者安全地计算出来, 而没有泄露每个单独位的值. 该协议的关键部分是 $\prod_i Y_i^{x_i} = 1$, 即随机因子 x_i 全部被消除.

2.2 竞价模型

在以上子协议的基础上我们可以实现安全的多方最高竞价协议构造, 原文较为冗长所以我们直接给出简化版本的描述. 首先所有参与方将自己的出价转换为二进制表示, 然后从二进制最高位开始逐个比特使用上文提到的匿名投票协议直到出现第一个投票结果为 1 的位 (表明有人在这

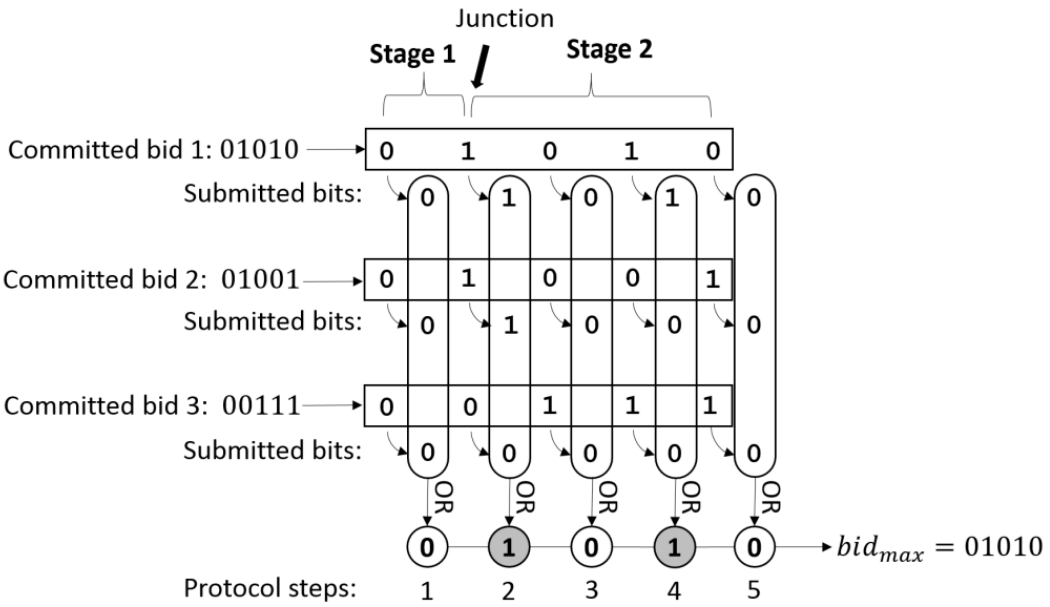

Figure 1: 一个实际拍卖情况的模拟

一位出价为 1), 则在此位出价为 0 的买家立刻可以得知自己已经输掉竞标, 则在接下来轮中其无论原出价的对应位为何都投票为 0, 直到最后得出最高价, 出该价的买家得知自己胜出. 流程如下图所示, 值得一提的是原文大量使用 NIZK 来证明构造合法, 详细证明过程可以参考原文.

2.3 Vickrey 拓展

上面讨论的主要协议是为最高价投标拍卖设计的. 将其扩展以支持第二高价密封投标 (即, 维克里拍卖) 的直接方法如下. 首先运行协议以确定最高出价和赢家, 然后第二次运行协议并排除赢家以计算第二高的出价, 承诺第二高出价的投标人保持匿名, 即可实现第二高价密封投标.

3 优化子协议

3.1 协议流程

我们回到上文中实现单比特或的结构的讨论, 仍然假设有 n 个参与方参与匿名投票, 此处我们暂时假定 $n \geq 3$, 因为在未作处理的情况下 $n = 2$ 会导致退化, 在下一节我们会通过构造一个假想第三方的方式来掩盖真实投票值, 从而对两方情况进行补全. 假设 n 个参与方名为 $p_1, p_2, \ldots, p_n$, 协议交互同样共有两轮流程, 密钥选择和投票公布阶段, 在协议开始运作之前参与方选定群公共生成元 g, 随后参与方 p_i 选择私秘秘钥 k_i 并计算和发布 g^{x_i}, 至此第一轮交互完成.

接下来是投票计算阶段，对于参与方 p_i，分其“同意”与“不同意”进行讨论：

若参与方 p_i 表示同意，其选票 v_i 计算为: $v_i = g^{(x_{i-1}-x_{i+1})\times x_i}$，这里我们规定 $x_{n+1} = x_1$, $x_0 = x_n$; 若参与方 p_i 表示不同意, 其选票 v_i 计算为: $v_i = g^{r_i}$, 其中 r_i 为随机选择的群中元素, 选票计算完成. 接下来各方依次发布自己的选票, 投票阶段完成. 在以上内容完成后进行合票计算,

将各方选票相乘, 如果各方均同意的情况下选票计算结果 R 计算为:

$$\begin{aligned} R &= \prod_{i=1}^{n} v_i \\ &= g^{(x_n - x_2)\times x_1 + (x_1 - x_3)\times x_2 \ldots + \left(x_{n_1} - x_1\right)\times x_n} \\ &= g^{x_1 x_n - x_1 x_2 + x_1 x_2 - x_2 x_3 + \ldots + x_{n-1} x_n - x_1 x_n} \\ &= g^0 = 1 \end{aligned}$$

但当任意一方选票为不同意时 R 以不可忽略的概率不为 1, 此处概率依赖于安全参数 κ, 回顾选票的匿名性，在诚实参与方安全模型下，对于参与方 p_i，其选票可能为 $v_i = g^{(x_{i-1} - x_{i+1})\times x_i}$ 或 $v_i = g^{r_i}$, 他人已知的情报包括 g^{x_i}、$g^{x_{i-1}}$ 和 $g^{x_{i-1}}$(等价于 $g^{x_{i-1} - x_{i+1}}$), 从中推断出选票是否为 $v_i = g^{(x_{i-1} - x_{i+1})\times x_i}$ 等价于从 g^a 和 g^b 推导 $g^a \times b$ 的离散对数难题, 从而其两种可能选票完全不可分辨.

3.2 效率与安全性

首先我们来看原协议在效率上与优化版本的计算效率差距在哪里. 由于选票相乘等效于指数项上相加, 因此我们可以设置一个指数矩阵来表示指数上的 $x_i x_j$ 加减的结果, 如图 2 及图 3 所示:

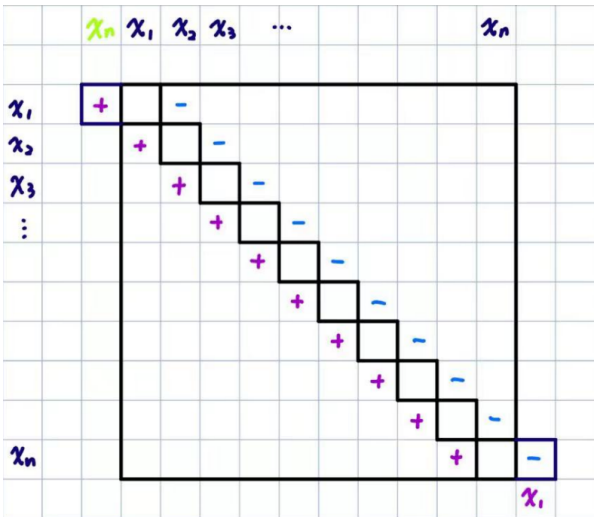

Figure 2: 优化协议

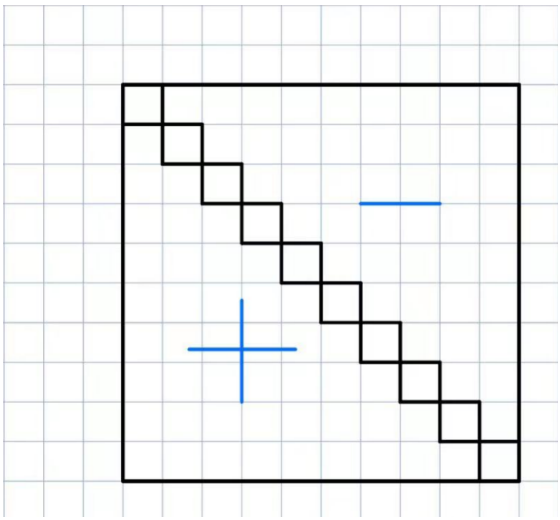

Figure 3: 原子协议

可以看到优化后的子协议的运算量只在主对角线左右两侧的副对角线上进行, 而原协议的计算需要对矩阵的上三角和下三角部分全部计算, 我们以 $g^{x_i x_j}$ 为一次单位运算, 则在 n 方参与的情况下优化后的协议需要 2n 次单位运算, 而原协议需要 n(n-1) 次单位运算, 计算效率得到大幅提高.

接下来我们来看安全性. 上面说的模型安全性相较于原协议较弱, 只能容忍至多一方恶意方参与, 在存在两方恶意方并且其在协议中的位置 i_1 和 i_2 满足 $i_1 \pm 2 \equiv i_2\,(modn)$（两方中间夹了一方诚实参与者）, 那么这两方可以通过密钥选择阶段构造 $x_{i_1} = x_{i_2} + k$ 来构造攻击对, 而夹在两方中间的诚实参与方则完全无法通过 $g^{x_{i_1}}$ 和 $g^{x_{i_2}}$ 来推导这样的 k 是否存在, 或者这种这样的勾结行为是否存在.

而在投票结果发布后, 我们设中间的诚实参与方为 p_o, 已知 k 和 g^{x_o} 的情况下验证 p_o 投票是否为 g^{kx_o} 是一个常数时间复杂度的算法, p_o 的投票遭到泄露.

为了提高安全性的做法也很简单, 表示同意的选票 $v_i = g^{(x_{i-1}-x_{i+1})\times x_i}$ 在计算时括号内对称地前后加减项即可实现, 例如: $v_i = g^{(x_{i-2}+x_{i-1}-x_{i+1}-x_{i+2})\times x_i}$, 在三方恶意参与方的情况下达到绝对安全, 不过该方案计算复杂度略高, 要保证大部分诚实参与方安全的情况下前后加减项各为 $n/4$ 个即可.

4 退化情况修正

我们先来看原协议退化到两方时会发生什么情况: 假设两个参与方 p_1,p_2 参与投票, 我们以 p_1 为例, 在上文模型的情况下如果 p_1 表示同意, 其计算选票的方式为:

$$v_1 = g^{(x_0-x_2)\times x_1} = g^{(x_2-x_2)\times x_1} = g^0 = 1$$

这明显泄露了 p_1 的选票意图, 因此我们必须做出修正. 接下来是笔者进行修正时的尝试思路:

一个自然的想法是引入一个模拟的第三方来保证选择的隐私性, 这个假想第三方的选择 (恒表示同意) 由实际的两个参与方共同决定, 最后共同计算. 由于最终计票的方式是相乘, 等效于在指数上相加, 于是考虑在 p_1 和 p_2 方引入 p_3 选择碎片的概念, p_1 和 p_2 独立选择 $x_3^{'}$ 和 $x_3^{''}$, 彼此不会知道对方的选择, 但是处于上帝视角的我们可以在此标记 $x_3 = x_3^{'} + x_3^{''}$, 这个 x_3 就可以作为假想 p_3 选择的密钥.

初步尝试的想法是双方共同构造 p_3 的同意投票

$$v_3 = g^{(x_2-x_1)\times x_3} = g^{(x_2-x_1)\times (x_3^{'}+x_3^{''})} = g^{(x_2-x_1)\times x_3^{'}} \times g^{(x_2-x_1)\times x_3^{''}}$$

其中 $g^{(x_2-x_1)}$ 由于之前公布的 g^{x_1} 和 g^{x_2} 完全可以计算, 从而分别持有 $x_3^{'}$ 和 $x_3^{''}$ 的 p_1 和 p_2 分别可以计算 $g^{(x_2-x_1)\times x_3^{'}}$ 和 $g^{(x_2-x_1)\times x_3^{''}}$ 并作为一个乘法因子附加在自己的选票中发布. 但在此问题解决后出现了另外一个问题, 我们回归此时 p_1 和 p_2 选票的计算, 按照原协议的流程两者均同意时双方选票各应该为:

$$v_1 = g^{(x_3-x_2)\times x_1} = g^{(x_3^{'}+x_3^{''}-x_2)\times x_1} = g^{(x_3^{'}-x_2)\times x_1} \times g^{x_1\times x_3^{''}}$$
$$v_2 = g^{(x_1-x_3)\times x_2} = g^{(x_1-x_3^{'}-x_3^{''})\times x_2} = g^{(x_1-x_3^{''})\times x_2} \times g^{-x_2\times x_3^{'}}$$

注意到上式结果右侧存在各参与方无法自己计算的项, 但是 $g^{x_1\times x_3^{''}}$ 可被 p_2 计算, $g^{-x_2\times x_3^{'}}$ 可被 p_1 计算, 于是我们可以考虑交换最右侧计算项, 即两者表示同意的选票可以表示为:

$$v_1 = g^{(x_3^{'}-x_2)\times x_1} \times g^{-x_2\times x_3^{'}} \times g^{(x_2-x_1)\times x_3^{'}}$$
$$v_2 = g^{(x_1-x_3^{''})\times x_2} \times g^{x_1\times x_3^{''}} \times g^{(x_2-x_1)\times x_3^{''}}$$

通过这种打乱方式可以实现对于真实投票值的混淆与掩盖. 注意, 从投票结果倒推出的信息不属于协议泄露, 该协议并不会泄露选票内容.