Secret Sharing：秘密分享

August 13, 2023

摘要

在本文中，我们将对秘密分享 (SS) 进行探讨，行文逻辑分为两大块：Shamir Secret Sharing 及其衍生和 Duplicated Secret Sharing 及其衍生.

秘密分享是一种密码学技术，允许一个秘密信息被拆分成多个份额，并分发给不同的参与方。这些份额单独看起来是随机的，没有任何意义，只有当收集到足够数量的份额时，才能恢复原始的秘密信息. 最典型的催生这一技术的问题是：如果有一个非常重要的秘密不想让任何一个人单独知道，但又需要让一组人共同保管, 需要如何解决? 秘密分享就是解决这个问题的方法. 它确保没有人能单独访问秘密，只有当足够多的人同意时，秘密才能被访问. 从广义上来说, 传统的加解密过程也是一种秘密分享过程，明文被分解成为密钥和密文两部分，只有两者均具备才能恢复出明文.

在进行更进一步的介绍之前，需要先对这一领域的典型流程和其中的概念进行解释.

1. 选择阈值：假设你有一个秘密，你想将其分享给 5 个人，但只要其中的 3 个人同意，就可以恢复秘密, 这里的 3 就是所谓的“阈值”. 我们一般简记一个秘密分享方案实现了 (t,n) 分享，这里 t 指代恢复秘密的门限或者阈值人数, n 则指代总的参与方个数.

2. 拆分秘密：使用特定的数学方法（例如 Shamir Secret Sharing 算法），将秘密拆分成各个份额.

3. 分发份额：将不同份额分别发给参与方.

4. 恢复秘密：当门限 t 个或更多的人将他们的份额组合在一起时，秘密可以被恢复。少于 t 个人的份额则无法恢复秘密.

综上, 秘密共享方案数学语言描述如下：将共享的秘密 m 分割成 n 个子秘密，并将其分发至 n 个参与者，使得授权子集 Γ 中的参与者可共同恢出复秘密 m，而非授权子集中的参与者无法得到关于秘密 m 的任何信息.

值得注意的是，在标准的秘密分享方案中，通常有一个特定的角色 (我们一般称之为”Dealer” 或” 拆分者”) 负责将秘密拆分成多个份额，并将这些份额分发给参与者。因为拆分秘密需要知道整个秘密，因此这是一个需要谨慎进行的敏感操作。在一些协议中接收分享份额的参与方同时也有可能就是拆分者，这看起来似乎会泄露额外的信息，最典型的一个例子是 BGW 协议 (我们会在第一节中详细讲解) 中双方对于自己的秘密值进行拆分后分享给对方，因此对于自己的秘密输入各方其实是具有完全掌握权的，但在这种模型中，只要对于任意一方存在其不知道的秘密片段，那么他就无从得知整个秘密，这也是这类协议的迷人之处.

1 Shamir Secret Sharing

Shamir(t,n) 门限秘密共享体制基于多项式插值算法设计，它的秘密分配算法如下: 首先假设 $\mathbb{F}_q$ 为 q 元有限域，q 是素数且 q>n. P $= \{P_1, \ldots, P_n\}$ 是参与者集合，P 共享主秘密 s, s$\in \mathbb{F}_q$，秘密管理中心 P_0 按如下所述的步骤对主秘密 s 进行分配，为了可读性起见，以下公式均略去了模 q 操作:

参与者 P_0 秘密地在有限域 $\mathbb{F}_q$ 中随机选取 t−1 个元素, 记为 $a_1, \ldots, a_{t-1}$, 并取以 x 为变元的多项式 $f(x)$

$$f(x) = a_{t-1}x^{t-1} + \cdots + a_1 x^1 + s = s + \sum_{i=1}^{t-1} a_i x^i$$

对于 $1 \le i \le n$, P_0 秘密计算 $y_i = f(i)$;

对于 $1 \le i \le n$, P_0 通过安全信道秘密地将 (i, y_i) 分配给 P_i.

Shamir(t,n) 门限共享体制的秘密重构可以使用通俗的解方程法，即 t 个方程可以确定 t 个未知数，而这 t 个未知数即为包括主秘密 s 在内的多项式 $f(x)$ 的各项系数。如参与者 $P_1, \ldots, P_t$ 掌握了子秘密 $f(1), \ldots, f(t)$，解方程:

$$\begin{aligned}
a_{t-1}1^{t-1} + \cdots + a_1 1^1 + s &= f(1)\\
a_{t-1}2^{t-1} + \cdots + a_1 2^1 + s &= f(2)\\
&\vdots\\
a_{t-1}n^{t-1} + \cdots + a_1 n^1 + s &= f(n)
\end{aligned}$$

即可求解出系数 $a_{t-1}, \ldots, a_1, s$.

利用拉格朗日插值法同样可以恢复秘密. 假设参与解密的 t 个子秘钥分别为 (x_i, y_i), 其中 $y_i = f(x_i)$, $i = 1, \ldots, t$ 且 $i \ne j$ 时 $x_i \ne x_j$. 参与者共同计算

$$\begin{aligned}
h(x) =& \sum_{s=1}^{t} y_{i_s} \prod_{j=1, j\ne s}^{t} \frac{x - x_j}{x_{i_s} - x_{i_j}},\\
=& y_1 \frac{(x-x_2)(x-x_3)\ldots(x-x_t)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_t)}\\
&+ y_2 \frac{(x-x_1)(x-x_3)\ldots(x-x_t)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_t)}\\
&+ \cdots\\
&+ y_t \frac{(x-x_1)(x-x_3)\ldots(x-x_{t-1})}{(x_t-x_1)(x_t-x_2)\ldots(x_t-x_{t-1})}\\
s =& a_0 = f(0) = \sum_{s=1}^{t} y_{i_s} \prod_{j=1, j\ne s}^{t} \frac{-x_j}{x_{i_s} - x_{i_j}}\\
=& \sum_{s=1}^{t} b_s y_{i_s}
\end{aligned}$$

其中, $b_s = \prod_{j=1, j\ne s}^{t} \frac{-x_j}{x_{i_s} - x_{i_j}}$(Lagrange 插值系数), 运算都是 $GF(q)$ 上的运算.

显然，$h(x)$ 是一个 t−1 次的多项式，且因为 $i \ne j$ 时 $x_i \ne x_j$, 每个加式的分母均不为零，因此对于 $i = 1, \ldots, t, y_i = h(x_i) = f(x_i)$. 又根据多项式的性质，如果存在两个最高次均为 t−1

次的多项式，这两个多项式在 t 个互不相同的点所取的值均相同，那么这两个多项式相同. 即 $h(x_i) = f(x_i)$, 进而参与者计算 $h(0) = f(0) = \mathrm{s}$，即可恢复主秘密 s.

对于有限域 $\mathbb{F}_q$ 上 n-1 次的多项式，设为 $f(x)$, 存在有限域 $\mathbb{F}_q$ 上的 n 个元，记为 $\lambda_1, \ldots, \lambda_n$, 使得:

$$f(0) = \sum_{i=1}^{n} \lambda_i f(i)$$

称 $(\lambda_1, \ldots, \lambda_n)$ 为重组向量 (recombination vector), 其必然存在, 利用线性代数中范德蒙行列式和秩的性质可以给出证明: 设 $f(x) = \sum_{j=0}^{n-1} a_i x^i$, 则 $f(0) = a_0$, 且 a_0 可被表示为:

$$a_0 = (1, 0, 0, \ldots, 0)\,(a_0, a_1, \ldots, a_{n-1})^T$$

考虑一个 n 阶矩阵

$$M = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2^1 & \cdots & 2^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & n^1 & \cdots & n^{n-1} \end{pmatrix}$$

有等式

$$\begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(n) \end{pmatrix} = M \times \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2^1 & \cdots & 2^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & n^1 & \cdots & n^{n-1} \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

由于矩阵 M 满秩，因此存在 $\lambda_1, \ldots, \lambda_n \in \mathbb{F}_q$, 使得

$$(\lambda_1, \ldots, \lambda_n)\, M = (1, 0, \ldots, 0)$$

因此有:

$$\begin{aligned} f(0) = & a_0 = (\lambda_1, \ldots, \lambda_n)\, M\, (a_0, a_1, \ldots, a_{n-1})^T \\ & = (\lambda_1, \ldots, \lambda_n)\, (f(1), f(2), \ldots, f(n))^T \\ & = \sum_{i=1}^{n} \lambda_i f(i) \end{aligned}$$

我们可以通过下式计算重组向量 $(\lambda_1, \ldots, \lambda_n)$:

$$(\lambda_1, \ldots, \lambda_n) = (1, 0, \ldots, 0)\, M^{-1}$$

在获得重组向量后，可构建基于 Shamir 门限体制的安全多方计算协议, 这里以 BGW 协议为例. 首先假设 $P = \{P_1, \ldots, P_n\}$ 是参与者集合, P_i 掌握输入 $x_i\,(1 \le i \le n)$, 需要共同计算的函数为 $f(x_1, \ldots, x_n)$. 在有限域 $\mathbb{F}_q$ 上的 Shamir$(t+1, n)$ 门限体制主要流程为:

输入阶段, 每个参与者 P_i 将自己的输入 x_i, 利用 Shamir$(t+1, n)$ 门限秘密共享体制，秘密选取最高 t 次的随机多项式 $f_i(x)$, 满足 $f_i(0) = x_i$. 然后 P_i 将 $f_i(j)$ 发送给参与者 P_j.

计算阶段, 假设输入的 a 和 b 已经通过至多 t 次的随机多项式 $f_a(x)$ 和 $f_b(x)$ 通过 Shamir 门限体制共享给了各个参与者, $f_a(x) = m_t x^t + \ldots + m_1 x^1 + a$, $f_b(x) = n_t x^t + \ldots + n_1 x^1 + b$, 其中 $m_t, \ldots, m_1, n_t, \ldots, n_1$ 是随机的多项式系数. 参与者 P_i 掌握输入 a 的子秘密 $a_i = f_a(i)$ 和输入 b 的子秘密 $b_i = f_b(i)$. 至多 t 次的原因是多项式的系数是随机产生的，因此 t 次的系数可能为 0.

加法: 令 $c = a + b$, 这一步希望在计算完成后各参与方 P_i 拥有 c 的秘密分享值 c_i. 为了达成这一目的, 每个参与者 P_i 独立计算 $c_i = a_i + b_i, 1 \le i \le n$. $c_1, \ldots, c_n$ 即为 $c = a + b$ 通过随机多项式共享后的结果, 通过多项式插值法或者解方程即可恢复秘密 c. 子秘密可以直接相加, 因为对于 $c_i = a_i + b_i = f_a(i) + f_b(i) = (m_t + n_t)i^t + \ldots + (m_1 + n_1)i + (a + b)$, 多项式的次数并没有发生变化，新的多项式 $f_c(x) = f_a(x) + f_b(x)$ 的最高次数依旧为 t, 并且满足 $f_c(0) = c$ 且 P_i 拥有 $f_c(i)$, 符合前文提到的 Shamir 门限分享. t+1 个参与者共享他们所掌握的 c_i，即可根据 t+1 个方程解 t+1 个未知数，解出 $c = a + b$, 也可直接使用拉格朗日插值法求解.

数乘运算: 作为加法运算的广义拓展, 对于常数 c 与秘密值 a 相乘的分享值计算可以简单套用多次加法, 令 $\mathrm{H(x) = c * f_a(x)}$ 则 $t+1$ 个参与者利用 $< x_i, c * y_i >$ 即可求得自己的份额 $H(x_i)$, 然后利用插值法计算 $h(0)$ 即可得到 $c * a$ 的结果.

乘法: 这一步希望在计算完成后各参与方 P_i 拥有 $a \times b$ 的秘密分享值. 按照上一小节的思路, 一个很自然的想法就是令 $h(x) = f_a(x) \times f_b(x)$, 这样求出来的 $h(x)$ 满足 $h(0) = ab$, 但与此同时产生的问题是 $h(x)$ 变为了最高 2t 次的函数, 导致恢复秘密值的阈限产生变化, 如果 $2t > n$ 会导致无法解密, 这是不可接受的. 同时, 按照这种方法产生的函数 $h(x)$ 一定是可约多项式, 泄露了额外的信息量, 因此需要做出限制.

依然令 $H(x) = f_a(x) \times f_b(x)$, 这里必须限制 $2t \le n$, 容易得到

$$
\begin{aligned}
A \cdot X &= \begin{bmatrix} 1 & 1 & \ldots & 1^{2t} \\ 1 & 2 & \ldots & 2^{2t} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2t+1 & \ldots & (2t+1)^{2t} \end{bmatrix} \begin{bmatrix} ab \\ r_1 \\ \cdots \\ r_{2t} \end{bmatrix} \\
&= \begin{bmatrix} H(1) \\ H(2) \\ \cdots \\ H(2t+1) \end{bmatrix}
\end{aligned}
$$

这里 X 为 $H(x)$ 的系数矩阵, 可知 A 是一个范德蒙矩阵，其可逆，设逆为

$$
A^{-1} = \begin{bmatrix} \lambda_1 & \lambda_2 & \ldots & \lambda_{2t+1} \\ \cdots & \cdots & \ldots & \cdots \\ \cdots & \ldots & \ldots & \cdots \\ \ldots & \ldots & \ldots & \cdots \end{bmatrix}
$$

则有

$$
\mathrm{ab} = \lambda_1 * \mathrm{H}(1) + \lambda_2 * \mathrm{H}(2) + \ldots \lambda_{2\mathrm{t}+1} * \mathrm{H}(2\mathrm{t}+1)
$$

其中 $\lambda_1, \lambda_2, \ldots, \lambda_{2t+1}$ 是公开参数, $H(1), H(2), \ldots, H(2t+1)$ 由各自的参与者 $P_i, 1 \le i \le 2t \le n$ 通过计算 $a_i b_i$ 拥有, 且不可泄露，然后上述问题可以看成每个参与者拥有秘密 $H(x_i)$ 的 BGW 加法和乘法问题, 可以选择新的 t 阶多项式将秘密 $H(x_i)$ 分发, 从而达到降阶效果.

更加严谨的数学语言描述为: 每个参与者 P_i 独立计算 $d_i = a_i b_i, 1 \leq i \leq n$, 接着每个 P_i 独自选取次数为 t 次的随机多项式 $h_i(x)$, 且满足 $h_i(0) = d_i, 1 \leq i \leq n, t < \frac{n}{2}$.

向各个参与者分配 d_i, 且 $c_{ij} = h_i(j), 1 \leq i, j \leq n$. 所有参与者分配结束后, P_i 掌握了信息 $c_{1i}, c_{2i}, \ldots, c_{ni}$. 同时 $(\lambda_1, \lambda_2, \ldots, \lambda_n)$ 是公开的重组向量, 即 $(\lambda_1, \ldots, \lambda_n)$ 满足 $ab = h(0) = \Sigma_{i=1}^{n} \lambda_i d_i$, 因此 P_i 可以计算 $c_i = \Sigma_{j=1}^{n} \lambda_j c_{ji}$, , 再利用多项式插值法, 即可获得 ab. 运算如下所示:

$$
\begin{array}{ccccccccc}
 & & & & P_1 & P_2 & & P_n \\
d_1 & \rightarrow & \mathrm{h}_1(x) & \rightarrow & \lambda_1 c_{11} & \lambda_1 c_{12} & \ldots & \lambda_1 c_{1n} \\
 & & + & & + & + & & + \\
d_2 & \rightarrow & \mathrm{h}_2(x) & \rightarrow & \lambda_2 c_{21} & \lambda_2 c_{22} & \ldots & \lambda_2 c_{2n} \\
 & & + & & + & + & & + \\
\vdots & & \vdots & & \vdots & \vdots & & \vdots \\
d_n & \rightarrow & \mathrm{h}_n(x) & \rightarrow & \lambda_n c_{n1} & \lambda_n c_{n2} & \ldots & \lambda_n c_{nn} \\
 & & \| & & \| & \| & & \| \\
ab & & H(x) & & c_1 & c_2 & \ldots & c_n
\end{array}
$$

2 Blakley Secret Sharing

Blakley Secret Sharing 基于高斯消元法实现. 根据基本的几何知识我们知道: 两条不平行的线在同一平面上交于一个点, 三个不平行的平面在空间中交于一个点. 更普遍地说, 任何 n 个不平行的 $(n-1)$ 维超平面都交于一个特定的点. 基于这一点可以设计出利用超平面相交特性的秘密分享方案. 大致思路为: 秘密 s 可以编码为交点的任何一个坐标, 通常我们选定为坐标向量的第一位, 而只选取一个维度坐标的原因是: 如果使用所有的坐标进行编码, 那么拥有超平面的计算参与方就可以获得有关秘密的额外信息. 这是因为他们知道秘密必须位于他们的平面上, 秘密值不再完全随机, 而是满足特定的线性方程 (平面方程), 因此可以借此获得更多关于秘密的信息, 即便分发算法再安全也无法实现信息论安全.

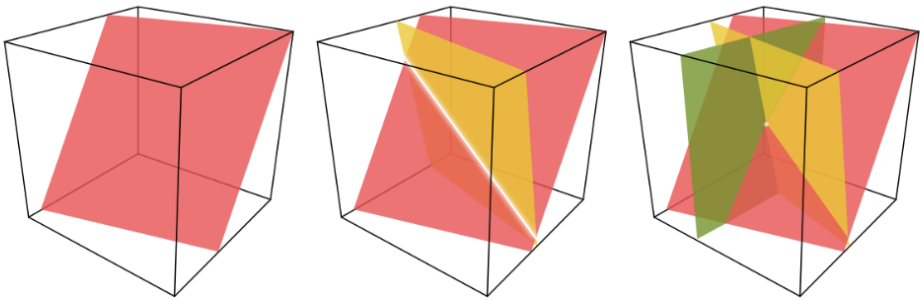

Figure 1: Blakley 方案三维情况: 每个份额都是一个平面, 秘密就是三个份额相交的点. 在第二张图中两个份额不足以确定秘密, 尽管它们确实提供了足够的信息来将其缩小到两个平面相交的线. 只有三个平面才能完全确定交点.

而当恢复时我们只需要平面维数个的参与方参与即可解密出秘密交点, 从而恢复交点向量特定维的秘密值. 接下来将给出基于此原理设计的 Blakley 协议算法流程.

2.1 协议流程

预备

令需要被分享的秘密为 S, 参与方个数为 N, 秘密恢复阈值为 k, 计算范围为有限域 $\mathbb{F}_p$.

分享

1. 分发者生成 k − 1 个随机数 $\{x_1, x_2, \ldots x_{k-2}, y\}$ mod P, 再令 $x_0 = S$, 构造 k 维平面交点 $(x_0, x_1, x_2 \ldots x_{k-2}, y)$.

2. 为 N 个参与方的每一方 $P_i, 1 \leq i \leq N$ 随机生成 k-1 个随机数 $\{a_{i,0}, a_{i,1}, \ldots, a_{i,k-2}\}$, 并根据平面公式 $y = a_{i,0} * x_0 + a_{i,1} * x_1 + a_{i,2} * x_2 \ldots a_{i,k-2} * x_{k-2} + c$, 计算 $c = y - (a_0 * x_0 + a_1 * x_1 + \ldots)$, 将 $\{a_{i,0}, a_{i,1}, \ldots, a_{i,k-2}, c\}$ 发送给参与方.

恢复

利用如下的矩阵方程求解, 即可恢复出交点坐标 $(x_0, x_1, x_2 \ldots x_{k-2}, y).(x_0, x_1, x_2 \ldots x_{k-2}, y)$, 从而恢复出秘密值 S.

$$\begin{bmatrix} a_{1,0} & a_{1,1} & \ldots & a_{1,k-2}, -1 \\ a_{2,0} & a_{2,1} & \ldots & a_{2,k-2}, -1 \\ \ldots & & & \\ a_{n,0} & a_{n,1} & \ldots & a_{n,k-2}, -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \ldots \\ x_{k-2} \\ y \end{bmatrix} = \begin{bmatrix} -c_1 \\ -c_2 \\ \ldots \\ -c_n \end{bmatrix} \bmod P$$

3 CRT Based Secret Sharing

引理: 中国剩余定理

这里仅作结论的展示, 引理证明过程可参考此网址: https://zhuanlan.zhihu.com/p/44591114

假设整数 $m_1, m_2, \ldots, m_n$ 两两互素，则对于任意的整数 $a_1, a_2, \ldots, a_n$ ，方程组

$$\begin{cases} x \equiv & a_1 \mod m_1 \\ x \equiv & a_2 \mod m_2 \\ & \ldots \\ x \equiv & a_n \mod m_n \end{cases}$$

都存在整数解，且该解在 mod N 意义下唯一，其中 $N = \prod_{i=1}^{n} m_i$. 解析解形式为

$$x \equiv \sum_{i=1}^{n} a_i \times \frac{N}{m_i} \times \left[\left(\frac{N}{m_i} \right)^{-1} \right]_{m_i} \mod N$$

中国剩余定理表明，当模数互素时，结构形式如上的含有 n 个一元线性同余方程的同余方程组在模去方程组模数之积的意义下有唯一解。基于这一性质，基于 CRT 的秘密分享思路如下:

协议流程

1. 假设这里需要实现的是门限为 t 的 n 方秘密分享方案,选择 n 个互素的正整数 $m_1, m_2, \ldots, m_n$, 不妨设 $m_1 < m_2 < \ldots < m_n$，让秘密值 S 小于其中最小的 t 个数的乘积，同时又大于最大的 t-1 个数的乘积。这样选择的目的是：使得 S 对于任意大于等于 t 个参与方能够保证在模去各方模数乘积下的唯一解即是 S 本身，同时小于 t 个参与方无法获得解的有效信息.

2. 秘密共享的值通过该式生成：$s_i \equiv S(\mathrm{mod} m_i), \forall 1 \leq i \leq n$. 接下来将 s_i 分享给参与方 p_i, 分享完成.

3. 重构过程为：任意满足门限数量的参与方可以利用 CRT 解析解公式重构出秘密值 S.

Asmuth-Bloom 方案

该方案基于上文 CRT-based Secret Sharing 思路完成，在实现细节上有所改动. 在该方案中：

1. 选取适合的参与方 n+1 和门限 k，令 $2 \leq k \leq n+1$.

2. 选择 n 个两两互素的正整数：$m_0 < m_1 < \ldots < m_n$，满足 $m_0 \cdot m_{n-k+2} \ldots m_n < m_1 \ldots m_k$.

3. 秘密值 S 在模 m_0 的剩余类集合 $\mathbb{Z}/m_0\mathbb{Z}$ 中选取.

4. 选取一个随机数 α，满足 $S + \alpha \cdot m_0 < m_1 \cdots m_k$.

5. 秘密共享的值：$I_i = (s_i, m_i), s_i \equiv S + \alpha \cdot m_0 (\mathrm{mod} m_i), \forall 1 \leq m_i \leq n$. 分发给各参与方，分享完成.

4 Brickell Secret Sharing

刚才提到的秘密分享方案中每个参与方的解密能力都是平均的，这么说的原因是任意随机选取的阈限个参与方即可恢复秘密，在另外一些情况下我们可能需要限定只有特定组合的参与方才能够进行解密，这种方案在参与方有利益相关的情况下尤为重要. 这类分享方案中最为典型的利用到了线性代数中线性相关组的概念，在开始具体协议之前先做简单回顾.

4.1 预备知识: 线性相关向量组

线性相关组是线性代数中的一个基本概念，表示这些向量之间存在某种线性关系. 如果一组向量中的某个向量可以通过其他向量的线性组合表示出来，那么这组向量就被称为线性相关的. 线性相关组的操作主要涉及到以下几个方面：

1. 检测线性相关性：通过矩阵的秩（rank）来判断一组向量是否线性相关。如果矩阵的秩小于向量的数量，那么这组向量就是线性相关的.

2. 线性组合：线性组合是指通过一组向量的加权和来构造新的向量。这在秘密共享中可以用来构造特定的解密组合.

3. 线性方程组：线性相关组可以用来解线性方程组，从而找到特定的解密方案.

在本节将要涉及的秘密分享方案中，将会利用到不同向量组和 $(1, 0, 0, \ldots, 0)$ 之间的线性相关性来进行秘密的分享和复现, 在此基础上能够赋予不同的参与方不同的解密能力.

4.2 协议流程

原协议为了数学描述的严谨性作了诸多定义和规则导致直接阅读非常晦涩难懂, 这里先给出一个原理性的简单流程, 并佐以一个简单例子, 希望对理解有所帮助.

秘密拆分

设秘密为 S, 确定 n 和向量维度 d, 选择模数 p. 然后确定解密组合 $\{\{v_1, v_2, v_3\}, \{v_1, v_4\}\}$, 其中 v_i 表示第 i 个 d 维向量，该解密规则表示第一、二、三个人和第一、四个人分别可以合作恢复出秘密。确定所有人共享的 n 个向量 $\{v_1, \ldots, v_n\}$ ，要求解密规则里的任何一组向量都可以线性构成 $(1, 0, 0)$, 而不在解密规则里的组合无法构成. 生成 $d-1$ 个小于 p 的随机数 $a_2, \ldots, a_d$, 同时将 S 的值赋值给 a_1. 分别计算 $S_i = \alpha \cdot v_i$, 其中 $\alpha = (a_1, \ldots, a_n)$.

秘密分发

将 (S_i, i) 作为钥匙分发给秘密持有者.

秘密恢复

利用满足解密规则的秘密持有者的向量构造 (1,0,0) . 例如: 通过 $c_1v_1+c_2v_2+c_3v_3=(1,0,0)$ 解出 c_1,c_2,c_3 ，然后将参数带入 $(c_1S_1+c_2S_2+c_3S_3)\,\mathrm{modp}=S$, 得出秘密 S 的值.

5 Duplicated Secret Sharing

复制秘密共享是基于另一种思路的秘密共享技术。这次选取的典型案例是一个半诚实的三方复制秘密共享协议，用于在三方环境下的安全多方计算和秘密共享，可以容忍最多一个腐化用户，其相比于 Shamir(2, 3) 来说有非常小的通信量和计算量。并且在布尔电路情况下和算术电路情况下均可以有等价的实现.

5.1 预备知识: 共享零和随机数的实现

我们需要实现 Alice、Bob、Candy 三方在本地生成随机数 a_1,a_2,a_3 并且满足 $a_1\oplus a_2\oplus a_3=0$. 实现方式为：Alice、Bob、Candy 分别生成随机数 ρ_1,ρ_2,ρ_3, Alice 将 ρ_1 发送给 Bob, Bob 将 ρ_2 发送给 Candy, Candy 将 ρ_3 发送给 Alice. 接下来 Alice 计算 $a_1=\rho_1\oplus\rho_3$，Bob 计算 $a_2=\rho_2\oplus\rho_1$，Candy 计算 $a_3=\rho_3\oplus\rho_2$.

容易验证：$a_1\oplus a_2\oplus a_3=\rho_1\oplus\rho_3\oplus\rho_2\oplus\rho_1\oplus\rho_3\oplus\rho_2=0$。

5.2 布尔电路实现

首先介绍在布尔电路下的情景，假设参与者分别为 Alice、Bob 和 Candy，三者的序号分别记为 1、2、3。在复制秘密共享中，一个单比特的秘密 X 会被生成为三个子秘密 x_1,x_2,x_3，且 $x=x_1\oplus x_2\oplus x_3$。具体方式为：秘密分享者首先生成三个随机数 a_1,a_2,a_3，并且满足 $a_1\oplus a_2\oplus a_3=0$。让子秘密 $x_1=a_3\oplus x$，$x_2=a_1\oplus x$，子秘密 $x_3=a_2\oplus x$。则 $x_1\oplus x_2\oplus x_3=a_1\oplus a_2\oplus a_3\oplus x\oplus x\oplus x=x$。让 Alice、Bob 和 Candy 分别持有 $(a_1,x_1),(a_2,x_2),(a_3,x_3)$，将这种秘密分享方式简记为 $[x]$。接下来实现布尔电路上的完备操作集合异或 (加法) 和与 (乘法). 对于秘密值 x 和 y，我们设定两者已经均使用上文的方案进行了分享，此时 Alice 已经持有了 (a_1,x_1), (b_1,y_1)，Bob 持有了 (a_2,x_2), (b_2,y_2)，Candy 持有了 (a_3,x_3), (b_3,y_3).

加法 (异或)

要计算 $[z]=[x+y]$，Alice、Bob 和 Candy 只需要分别本地计算 $x_i+y_i,i\in\{1,2,3\}$ 即可. 以 Alice 为例，因为 $x_1=a_3\oplus x$, $y_1=b_3\oplus y$, 则 $x_1\oplus y_1=a_3\oplus x\oplus b_3\oplus y=(a_3\oplus b_3)\oplus(x\oplus y)=(a_3\oplus b_3)\oplus z$, 若将 $a_1\oplus b_1$ 记为 c_1, $a_2\oplus b_2$ 记为 c_2, $a_3\oplus b_3$ 记为 c_3, 则 Alice、Bob 和 Candy 在分别完成本地计算 $x_i+y_i,i\in\{1,2,3\}$ 后, Alice 可以得到 $z_1=c_3\oplus z$，Bob 可以得到 $z_2=c_1\oplus z$，Candy 可以得到 $z_3=c_2\oplus z$; 且 $c_1\oplus c_2\oplus c_3=a_1\oplus a_2\oplus a_3\oplus b_1\oplus b_2\oplus b_3=0\oplus 0=0$, 由此满足之前的秘密分享定义，有 $z_1\oplus z_2\oplus z_3=z$, $c_1\oplus c_2\oplus c_3=0$.

乘法 (与)

要实现乘法首先需要另外引入三个随机数，记为 α,β,γ，且满足 $\alpha\oplus\beta\oplus\gamma=0$，Alice 掌握 α，Bob 掌握 β，Candy 掌握 γ. 具体的实现方式在本节预备知识部分已经涉及, 这里假设已经预计算完成.

Alice 计算 $r_1=x_1y_1\oplus a_1b_1\oplus\alpha$，并把 r_1 发送给 Bob；Bob 计算 $r_2=x_2y_2\oplus a_2b_2\oplus\beta$，并把 r_2 发送给 Candy；Candy 计算 $r_3=x_3y_3\oplus a_3b_3\oplus\gamma$，并把 r_3 发送给 Alice. 接着 Alice 本地

计算 $c_1 = r_1 \oplus r_3$, $z_1 = r_1$；Bob 本地计算 $c_2 = r_2 \oplus r_1$，$z_2 = r_2$；Candy 本地计算 $c_3 = r_3 \oplus r_2$，$z_3 = r_3$。正确性证明有

$$\begin{aligned} r_1 &= x_1 y_1 \oplus a_1 b_1 \oplus \alpha, \\ r_2 &= x_2 y_2 \oplus a_2 b_2 \oplus \beta, \\ r_3 &= x_3 y_3 \oplus a_3 b_3 \oplus \gamma, \end{aligned}$$

又因为

$$\begin{aligned} x_1 y_1 &= (a_3 \oplus x)(b_3 \oplus y) = a_3 b_3 \oplus a_3 y \oplus b_3 x \oplus xy, \\ x_2 y_2 &= (a_1 \oplus x)(b_1 \oplus y) = a_1 b_1 \oplus a_1 y \oplus b_1 x \oplus xy, \\ x_3 y_3 &= (a_2 \oplus x)(b_2 \oplus y) = a_2 b_2 \oplus a_2 y \oplus b_2 x \oplus xy, \end{aligned}$$

因此有

$$\begin{aligned} x_1 y_1 \oplus x_2 y_2 \oplus x_3 y_3 &= a_1 b_1 \oplus a_2 b_2 \oplus a_3 b_3 \oplus x(b_1 \oplus b_2 \oplus b_3) \oplus y(a_1 \oplus a_2 \oplus a_3) \oplus xy \\ &= a_1 b_1 \oplus a_2 b_2 \oplus a_3 b_3 \oplus xy \end{aligned}$$

把 $x_1 y_1 \oplus x_2 y_2 \oplus x_3 y_3$ 的结果代入到 $z_1 \oplus z_2 \oplus z_3$ 可得:

$$\begin{aligned} z_1 \oplus z_2 \oplus z_3 &= r_1 \oplus r_2 \oplus r_3 \\ &= x_1 y_1 \oplus a_1 b_1 \oplus \alpha \oplus x_2 y_2 \oplus a_2 b_2 \oplus \beta \oplus x_3 y_3 \oplus a_3 b_3 \oplus \gamma \\ &= x_1 y_1 \oplus x_2 y_2 \oplus x_3 y_3 \oplus a_1 b_1 \oplus a_2 b_2 \oplus a_3 b_3 \oplus \alpha \oplus \beta \oplus \gamma \\ &= a_1 b_1 \oplus a_2 b_2 \oplus a_3 b_3 \oplus xy \oplus a_1 b_1 \oplus a_2 b_2 \oplus a_3 b_3 \\ &= xy \end{aligned}$$

而 $c_1 \oplus c_2 \oplus c_3 = r_1 \oplus r_3 \oplus r_2 \oplus r_1 \oplus r_3 \oplus r_2 = 0$，即:

$$\begin{aligned} z_1 &= c_3 \oplus xy \\ z_2 &= c_1 \oplus xy \\ z_3 &= c_2 \oplus xy \end{aligned}$$

其中 $z_1 \oplus z_2 \oplus z_3 = xy$，$c_1 \oplus c_2 \oplus c_3 = 0$，由此得证乘法的正确性.

5.3 算术电路实现

上一节介绍了在布尔电路下的三方复制秘密共享方案，这一节将讲解如何将其扩展到环 2^n 下. 首先是在环 2^n 下生成三个随机数 a_1, a_2, a_3, 并且满足 $a_1 + a_2 + a_3 = 0$. 在预备知识中已经介绍过满足条件 $a_1 \oplus a_2 \oplus a_3 = 0$ 的随机数生成方式, 满足 $a_1 + a_2 + a_3 = 0$ 的三元组只需对上次的方式进行一些小更改:

Alice, Bob, Candy 分别生成随机数 ρ_1, ρ_2, ρ_3, Alice 将 ρ_1 发送给 Bob, Bob 将 ρ_2 发送给 Candy, Candy 将 ρ_3 发送给 Alice. 接着 Alice 计算 $a_1 = \rho_1 - \rho_3$, Bob 计算 $a_2 = \rho_2 - \rho_1$, Candy 计算 $a_3 = \rho_3 - \rho_2$. 显然, $a_1 + a_2 + a_3 = \rho_1 - \rho_3 + \rho_2 - \rho_1 + \rho_3 - \rho_2 = 0$.

分享方案

假设秘密为 x, y, Alice, Bob, Candy 已经按照前文的方法生成零和随机数组 (a_1, a_2, a_3) 和 (b_1, b_2, b_3). 则令 $x_1 = a_3 - x$, $x_2 = a_1 - x$, $x_3 = a_2 - x$; $y_1 = b_3 - y$, $y_2 = b_1 - y$, $y_3 = b_2 - y$. Alice 持有 (x_1, a_1),(y_1, b_1) , Bob 持有 (x_2, a_2),(y_2, b_2) , Candy 持有 (x_3, a_3),(y_3, b_3). 对于单个秘密 (以 x 为例), 则只需要任意两方参与即可通过减法恢复秘密 x. 对于一个算术电路的完备操作集合为加法和乘法, 接下来将在上文已经建立的 x, y 分配标记的基础上实现各种算术运算.

加法

加法的实现方式与布尔电路上的加法实现方式类似, Alice, Bob 和 Candy 在模 2^n 下直接本地计算 $x_i + y_i$ 即可. Alice 计算 $z_1 = x_1 + y_1 = a_3 - x + b_3 - y = (a_3 + b_3) - (x + y)$, $c_1 = a_1 + b_1$. 同理 Bobi 计算 $z_2 = x_2 + y_2 = a_1 - x + b_1 - y = (a_1 + b_1) - (x + y)$, $c_2 = a_2 + b_2$; Candy 计算 $z_3 = (x_3 + y_3) = a_2 - x + b_2 - y = (a_2 + b_2) - (x + y)$, $c_3 = a_3 + b_3$. 可以验证只需要任意两方参与即可通过减法恢复秘密, 且秘密份额的形式与之前相同.

乘法

乘法的实现方式需要 Alice、Bob 和 Candy 利用本节开头的随机数生成方式, 生成满足 $\alpha + \beta + \gamma = 0$ 的条件随机数 α, β, γ. Alice 计算 $r_1 = (\mathrm{x_1y_1 + a_1b_1} + \alpha)/3$, 并且把 r_1 发送给 Bob; Bob 计算 $r_2 = (x_2y_2 + a_2b_2 + \beta)/3$, 并且把 r_2 发送给 Candy; Candy 计算 $r_3 = (x_3y_3 + a_3b_3 + \gamma)/3$, 并且把 r_3 发送给 Alice.

Alice 让 $z_1 = -2r_3 - r_1, \quad c_1 = r_3 - r_1$;

Bob 让 $z_2 = -2r_1 - r_2, \quad c_2 = r_1 - r_2$;

Candy 让 $z_3 = -2r_2 - r_3, \quad c_3 = r_2 - r_3$.

显然: $c_1 + c_2 + c_3 = r_3 - r_1 + r_1 - r_2 + r_2 - r_3 = 0$, 并且可以验证:

$$
\begin{aligned}
(r_1 + r_2 + r_3) &= \frac{x_1y_1 - a_1b_1 + \alpha + x_2y_2 - a_2b_2 + \beta + x_3y_3 - a_3b_3 + \gamma}{3} \\
&= \frac{(x_1y_1 + x_2y_2 + x_3y_3 - a_1b_1 - a_2b_2 - a_3b_3)}{3} \\
&= \frac{(a_1 - x)(b_1 - y) + (a_2 - x)(b_2 - y) + (a_3 - x)(b_3 - y) - a_1b_1 - a_2b_2 - a_3b_3}{3} \\
&= \frac{(-x\,(b_1 + b_2 + b_3) - y\,(a_1 + a_2 + a_3) + 3xy)}{3} \\
&= xy
\end{aligned}
$$

因此有:

$$
\begin{aligned}
z_1 &= -2r_3 - r_1 \\
&= -2r_3 - r_1 - r_2 + r_2 \\
&= r_2 - r_3 - (r_1 + r_2 + r_3) \\
&= r_2 - r_3 - xy \\
&= c_3 - xy
\end{aligned}
$$

同理可以推导出 $z_2 = c_1 - xy$, $z_3 = c_2 - xy$, 可以验证只需要任意两方参与即可通过减法恢复秘密, 且秘密份额的形式与之前相同. 故操作完成后各方拥有了 xy 的份额.

上面两节介绍了在布尔电路下和数字电路下的三方复制秘密共享, 以及其实现加法和乘法的方式. 本节介绍两种三方复制秘密共享下的截断 (Truncate) 操作. 其中一种截断方式只需要三个参与者中的两者参加, 不需要三者共同进行截断, 以此减少截断操作所需要的通信量. 在三方复制秘密共享中，任意两方合谋即可恢复出秘密，即其中任意两方就可以进行截断操作.

6 安全性进阶：可验证秘密分享方案 (VSS)

可验证的秘密共享 VSS 方案是对传统秘密共享方案的修正，在通常的秘密分享方案基础上附加验证操作构成，主要用于解决不诚实的分发中心问题. 在 VSS 方案中，分发者不但分发秘密的碎片，而且广播对秘密碎片的承诺，当个成员收到其碎片时，要验证其碎片是否正确；在秘密重构阶段，每个参与者也采用同样方法验证其他成员秘密碎片的正确性.

VSS 主要抵抗以下两种主动攻击：

- 分发者在秘密分发协议中发送错误碎片给部分或全部参与者.
- 协议参与者在秘密重构协议中提交错误碎片.

常见的可验证的秘密共享方案包括 Feldman 的 VSS 方案以及 Pedersen 方案.

6.1 Feldman Verifiable Secret Sharing

Feldman 流程与 Shamir 基本一致, 保证安全性的手段是使用了承诺与公开验证机制. 承诺这一密码学组件将会在之后的文章中继续涉及.

秘密分发

可信中心选取 t 阶随机多项式:

$$f(x) = a_{t-1}x^{t-1} + \ldots + a_2x^2 + a_1x + a_0 \in GF(q)[x]$$

其中常数 $a_0 = s$ 为要分发的秘密.

可信中心选择 n 个非零的互不相同的元素 $x_1, x_2, \ldots, x_n \in GF(q)$, 计算 $y_i = f(x_i), 1 \leq i \leq n$, 将子密钥 (x_i, y_i) 分配给共享者 P_i, 这里 x_i 是公开的, 而 y_i 为 P_i 秘密持有. 接下来可信中心计算并广播承诺 $B_j = g^{a_j}, 0 \leq j < t$. 参与者 P_i 收到自己的碎片 y_i 后, 判断 $g^{y_i} = \Pi_{j=0}^{t-1} B_j^{x_i^l}$ 是否成立, 该式右侧相当于在指数上对 $f(x_i)$ 进行运算, 如果等式成立, 则接受该碎片为有效; 否则, P_i 请求可信中心重新发送正确的碎片.

数学推导为: 若可信中心向 P_i 传送了正确的碎片 y_i, 则有

$$\begin{aligned} g^{y_i = g^{f(x_i)}} &= g^{a_{t-1}r_i^{t-1} + \ldots + a_2x_i^2 + a_1x_i + a_0} \\ &= g^{a_{t-1}x_i^{t-1}} \ldots g^{a_2x_i^2} g^{a_1x_i} g^{a_0} \\ &= B_{t-1}^{x_i^{t-1}} \ldots B_2^{x_i^2} B_7^{x_i} B_0 \\ &= \Pi_{j=0}^{t-1} B_j^{x_i^j} \end{aligned}$$

秘密重构

假设每个参与者都接受到正确的碎片, 他们中的任意 t 个可执行 Shamir 门限秘密分享方案中的重构算法恢复出原始秘密, 即 P_i 向参与重构的其他人广播自己的碎片, 这样每个参与重构的成员均可验证所接收到的碎片的有效性，然后使用 Lagrange 插值公式计算出秘密 s.

Feldman 的 VSS 方案中, 由于可信中心在广播承诺时将 $B_0 = g^{a_0} = g^s$ 也作为一个承诺发出, 因此方案仅是计算安全的.

6.2 Pedersen Verifiable Secret Sharing

Pedersen 扩展了 Feldman 的方案，将 Shamir 秘密共享方案与承诺方案相结合, 构造出了一个高效、安全的非交互式可验证秘密共享方案，且验证信息不会直接泄露秘密 s, 因此是信息论安全的.

秘密分发

可信中心选取两个 t 阶随机多项式:

$$a(x) = a_{t-1}x^{t-1} + \ldots + a_2x^2 + a_1x + a_0 \in GF(q)[x]$$
$$b(x) = b_{t-1}x^{t-1} + \ldots + b_2x^2 + b_1x + b_0 \in GF(q)[x]$$

其中常数 $a_0 = s$ 为要分发的秘密.

可信中心选择 n 个非零的互不相同的元素 $x_1, x_2, \ldots, x_n \in GF(q)$, 计算 $y_i = (s_i, s_{i2}) = (a(x_i), b(x_i)), 1 \leq i \leq n$. 计算完成后将子密钥 (x_i, y_i) 分配给共享者 P_i, 分享完成后 x_i 公开, y_i 为 P_i 的秘密份额. 接下来可信中心计算并广播承诺

$$C_j = g^{a_j}h^{b_j}, 0 \leq j < t$$

参与者 P_i 收到自己的碎片 y_i 后, 判断 $g^{s_i}h^{s_i} = \prod_{j=0}^{t-1} C_j^{x_i^l}$ 是否成立. 如果成立, 则接受该碎片为有效; 否则, P_i 请求可信中心重新发送正确的碎片.

数学推导为: 若可信中心向 P_i 传送了正确的碎片 y_i, 则有

$$\begin{aligned} g^{s_i}h^{s_i2} =& g^{a(x_i)}h^{b(x_i)} = \left(g^{a_{t-1}x_i^{t-1}+\ldots+a_2x_i^2+a_1x_i+a_0}\right)\left(h^{b_{t-1}x_i^{t-1}+\ldots+b_2x_i^2+b_1x_i+b_0}\right) \\ &= \left(g^at-1h^{b_{t-1}}\right)^{x_i^{t-1}} \ldots \left(g^{a_2}h^{b_2}\right)^{x_i^2}\left(g^{a^a}h^{b_1}\right)^{x_i} g^{a_0}h^{b_0} \\ &= C_{t-1}^{x_i^{t-1}} \ldots C_2^{x_i^2}C_1^{x_i}C_0 \\ &= \prod_{j=0}^{t-1} C_j^{x_i^j} \end{aligned}$$

验证完毕, 并且此验证过程无法获得有关秘密 s 的有效信息.

秘密重构

假设每个参与者都接受到正确的碎片, 他们中的任意 t 个可执行 Shamir 门限秘密分享方案中的重构算法恢复出院时秘密, 即 P_i 向参与重构的其他人广播自己的碎片, 这样每个参与重构的成员均可验证所接收到的碎片的有效性, 然后使用 Lagrange 插值公式计算出秘密 s.

Pedersen 的 VSS 方案中, 可信中心在广播承诺时与秘密 s 相关的信息仅为 $C_0 = g^sh^{b_0}$, 没有泄露关于 s 的任何信息, 因此方案是信息论安全的.

7 去中心化的无分发者随机秘密共享

在该秘密体制中不存在秘密分发者, 仅有参与者 $P_1, P_2, ..., P_n$, 他们以交互的方式协商出一个随机秘密 s, 并各自得到该秘密的一个碎片 s_i, 但每个参与者都不知道这个随机秘密的具体值, 除非他们公布自己的碎片并重构该秘密.

基于 Shamir 的秘密分享体制的一个无秘密分发者的 (t, n) 秘密分享体制, 称为 Joint-Shamir-RSS 方案 (Joint Shamir Random Secret Sharing). 方案流程为:

1. 每个参与者 P_i 选择一个 t-1 次随机多项式 $f_i(x) = a_{i,t-1}x^{t-1} + \ldots + a_{i2}x^2 + a_{i1}x + a_{i0} \in GF(q)[x]$, 以 $a_{i0} = f_i(0)$ 作为自己要让 $P_1, P_2, \ldots, P_n$ 分享的秘密.

2. P_i 计算 $s_{ij} = f_i(x_j)$, 发送给 $P_j, j = 1, 2, \ldots, n$, 如下所示:

	P_1	P_2	P_j	P_n
P_1	$f_1(x_1)$	$f_1(x_2)$	$f_1(x_j)$	$f_1(x_n)$
P_2	$f_2(x_1)$	$f_2(x_2)$	$f_2(x_j)$	$f_2(x_n)$
P_i	$f_i(x_1)$	$f_i(x_2)$	$f_i(x_j)$	$f_i(x_n)$
P_n	$f_n(x_1)$	$f_n(x_2)$	$f_n(x_j)$	$f_n(x_n)$

3. P_j 收到其他参与者的值 $s_{ij} = f_i(x_j), i = 1, 2, \ldots, n$, 计算 $s_j = \sum_{i=1}^{n} f_i(x_j)$ 作为自己最终分享秘密的碎片. 从协议中可以看出, 若令 $f(x) = \sum_{i=1}^{n} f_i(x)$, 则 $f(x_j) = \sum_{i=1}^{n} f_i(x_j)$.

4. 秘密重构阶段, 只需任意 t 个参与者使用自己拥有的秘密碎片 s_i 执行 Shamir 秘密分享体制的秘密重构即可, 恢复出的秘密为各方初始随机选择的 a_{i0} 的和.

8 Proactive Secret Sharing

Proactive secret sharing(主动秘密共享) 常用于保护分布式系统中的秘密信息. 它是一种定期更新秘密共享方案中分布式密钥 (份额) 的方法, 从而使攻击者有更少的时间来破坏份额, 只要攻击者访问的数量少于阈值或法定人数, 系统就保持安全, 这与非主动方案形成对比: 如果在秘密的生命周期内阈值数量的份额被破坏, 秘密就会被泄露. 将时间约束考虑在内的模型将鲁棒性扩展到时间域, 这一模型由 Rafail Ostrovsky 和 Moti Yung 于 1991 年提出, 并且已被用于安全多方计算的密码协议领域和阈值密码系统中.

起初对于主动秘密共享方案的需求并不是很强烈, 但随着云存储等技术的广泛普及, 对于数据安全长期存储的需求逐渐凸显, 保持长期不变的存储份额给了攻击者足够的时间来对系统进行攻击. 主动秘密共享概念的提出就是基于这一假设, 其目的是保护敏感信息免受长期攻击的侵害.

PSS 的所使用到的核心概念是密钥旋转 (重新分配共享) 和份额追回, 用于解决这两个问题: 假设 Alice 的计算机受到威胁, (1)Bob/Candy/Dave 能否改组其份额以使原来的份额过时? (2) 如果 Alice 后来更换了一台安全的设备, 她又如何才能重新加入该协议参与者团体? 在这一模型下我们希望没有所谓的可信执行方, 因为如果有了受信任的执行方, 显然 Bob/Candy/Dave 可以将自己拥有的密钥发送给可信执行方, 由其重构秘密后生成新的多项式, 并将新的份额分配给 Bob/Candy/Dave, 并且以后还可以在 Alice 拥有新的安全设备时向 Alice 发送新共享. 所以我们着重考虑的是一个去中心化的更新协议, 即不需要可信执行者即可实现安全更新的方案, 接下来讲解的方案由 Herzberg 于 1995 年提出.

份额转换 (Share Reshuffling)

我们用一个简单的例子来说明这一步是怎么操作的. 假设最初的密钥分配方案为 Shamir 门限方案, 参与方为 Alice、Bob、Candy 和 Dave, 不失一般性, 假设最初的秘密分享多项式为二次多项式 $f(x) = x^2 - 4x + 5$ 其中要被分享的秘密为 $s = 5$, 四人获得的秘密份额分别为 $f(1), f(2), f(3), f(4)$. 解密门限为 $t = 2$. 现在 Alice 主机受到攻击, 我们需要将其剔除并且更新各方拥有的份额, 并且剩余的各方仍然能够通过某种方法恢复出原始秘密.

为了达成这一目的, Bob 随机选择一个 2 次多项式 g(x), 满足 g(0)=0. 例如, Bob 可以选取 $g(x) = x^2 - 2x$. 现在我们令 $f'(x) = f(x) + g(x)$, 由于 $g(0) = 0$, 因此 $f'(0) = f(0)$, 这意味着新的随机多项式的常数项与原来秘密分配多项式的常数项相同.

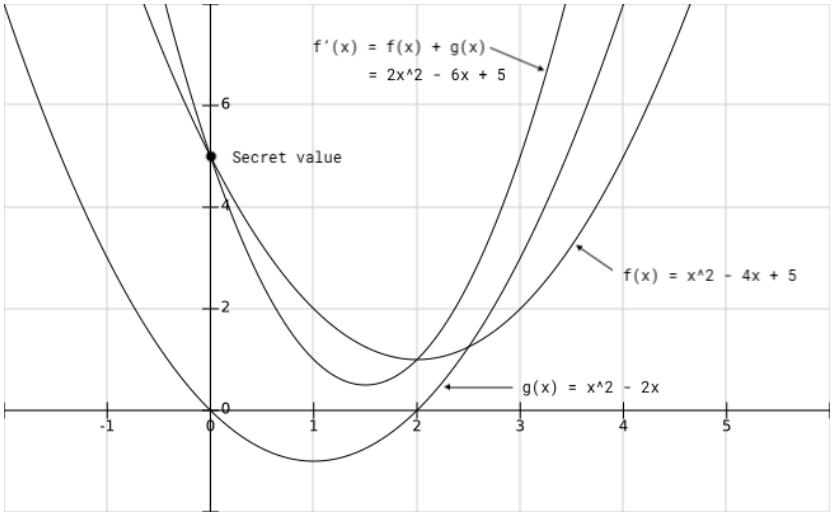

Figure 2: 份额转换: $f'(x)$ 和 $f(x)$ 拥有同样的常数项

并注意到

$$\begin{aligned} f'(1) =& f(1) + g(1) \\ f'(2) =& f(2) + g(2) \\ &\vdots \end{aligned}$$

由于我们在第一节提到的 Shamir 分配方案的加法特性, 上式意味着, 如果 Bob 将 $g(3)$ 发给 Candy, 并且将 $g(4)$ 发给 Dave. 则 Candy 和 Dave 分别根据 $f(3)$ 和 $g(3)$、$f(4)$ 和 $g(4)$ 就能够分别计算出 $f'(3)$ 和 $f'(4)$, Bob 自身也能计算出 $f'(2)$. 通过使用 $f'(2)$, $f'(3)$ 和 $f'(4)$, Bob、Candy 和 Dave 可以根据需要重建秘密, 因为 $f'(0) = f(0)$. 因此，他们可以舍弃 f(x) 的值, 由于 $f(2), f(3), f(4)$ 不再存在, 保留在 Alice 受损计算机中的值 $f(1)$ 将变得无用. 同时, 没有人 (包括 Bob) 在改组过程前后能够知道 $f(0)$ 或 $f'(0)$ 的值.

但是, 尽管 Bob 不知道 $f(0)$ 或 $f'(0)$, 但显然他比 Candy 和 Dave 处于更有利的地位. 因此，实际上该算法不仅需要 Bob 生成随机多项式 g(x) 并将一些值分配给 Candy 和 Dave, 同样，Candy 应该生成随机多项式 $h(x)$ 并将一些值分配给其他对等参与方, 并且 Dave 应该生成另一个随机多项式 $k(x)$ 并将一些值也分配给对等体. 在这种情况下，每个参与方都是平等的, 并且算法对所有参与方公平.

更明确地说, Bob, Candy, Dave 应该共同生成一个新的多项式 $f'(x)$:

$$f'(x) = f(x) + g(x) + h(x) + k(x)$$

其中 $g(x)$ 由 Bob 生成, Candy 是 $h(x)$, Dave 是 $k(x)$. 并且 $g(0) = h(0) = k(0) = 0$.

份额追回 (Share Recovery)

现在 Alice 重新获得了安全的新设备, 在最坏假设下, Alice 丢失了最初自己持有的 $f(1)$, 为了让 Alice 重新加入该参与者集合, 我们希望 Alice 能够计算自己的份额, 但又不会泄露秘密, 这里可以重复利用上一小节的技术.

在上一部分中，Bob 必须生成一个满足 $g(0) = 0$ 的随机多项式 $g(x)$, 因为我们希望新生成的秘密分享多项式 $f'(x)$ 和原先秘密分享多项式 $f(x)$ 有相同的常数项. 推广到恢复 Alice 的份额

$f(1)$ 时, 我们希望新生成的多项式在 $x = 1$ 处的值保持不变 (因为在本示例中, Alice 的 x=1), 即 Bob 应该生成 $g(1) = 0$ 的随机多项式. 具体实现上, 如果 Bob、Carol、Dave 共同生成一个新的函数 $l(x)$ 的分享, 其满足:

$$l(x) = f(x) + b(x) + c(x) + d(x)$$

其中 $b(x), c(x), d(x)$ 分别由 Bob、Carol、Dave 生成且满足 $b(1) = c(1) = d(1) = 0$, 三人分别对 $b(x), c(x), d(x)$ 作 Shamir 分享并且各自在本地计算得到 $l(2), l(3), l(4)$, 计算结果可以通过安全信道发送给 Alice, Alice 通过 Lagrange 插值法可以恢复出 $l(x)$, 从而恢复出 $f(1) = l(1)$. 如果想要 Alice 也得到到经过更新的份额, 按照上一节的方案对 Alice 进行更新即可, 当然这一切要建立在 Alice 的新设备已经排除泄露风险的情况下进行.

以上就是对原文协议原理的简单概述, 在原协议中有部分实现细节不同但大体思路类似, 希望这个简化过程对理解原协议有所帮助.

9 结语

秘密共享不仅是一个令人着迷的理论问题, 还是现代密码学和信息安全的核心组成部分. 通过本文的探讨, 我们不难看出, 从 Shamir 的多项式构造到 Blakley 的几何视角, 再到 Brickell 的特定应用, 秘密共享的概念和应用正在不断扩展和深化.

我们还深入研究了中国剩余定理在秘密共享中的作用, 通过数学方法高效地解决了共享和重建秘密的问题. 而 Duplicated Secret Sharing 则向我们展示了如何在保证安全的同时增加系统的健壮性.

更进一步, 我们探讨了如何确保秘密共享过程的透明和可信. Verifiable Secret Sharing 和 Proactive Secret Sharing 为我们提供了这一解决方案的灵感, 使我们能够在更广泛、更复杂的环境中部署秘密共享方案.

总的来说, 这些概念共同描绘了一幅秘密共享的多维画卷, 涵盖了理论、实践、安全、可验证性和灵活性等方面. 它们突显了秘密共享作为现代安全架构中不可或缺的一部分的重要性. 随着未来对这些方法的进一步研究和改进, 我们有望实现更加全面、高效和安全的信息共享模型. 我们期待秘密共享继续引领安全计算的新方向, 推动技术向前迈进.

感谢阅读至此，希望本文对你有所帮助.