Multiparty Threshold ECDSA

1. 简介

本文讨论多个参与者如何共同完成 ECDSA 门限签名，主要讨论 2018 年 Rosario Gennaro 和 Steven Goldfeder 在论文 Fast Multiparty Threshold ECDSA with Fast Trustless Setup 中提出的方案，简称 GG18 方案。

2. Generic DSA signature

GG18 作者提出了一个通用的签名方案（G-DSA），它统一了 DSA 签名和 ECDSA 签名的表示方法。

G-DSA 如图 1 所示。其中 x 是私钥， k 随机产生， m 是待签名消息的哈希；而签名数据由 r, s 两部分组成。

Key-Gen On input the security parameter, outputs a private key x chosen uniformly at random in Z_q, and a public key $y = g^x$ computed in $\mathcal{G}$.

Sig On input an arbitrary message M,

- compute $m = H(M) \in Z_q$
- choose $k \in_R Z_q$
- compute $R = g^{k^{-1}}$ in $\mathcal{G}$ and $r = H'(R) \in Z_q$
- compute $s = k(m + xr) \bmod q$
- output $\sigma = (r, s)$

Ver On input M, σ and y,

- check that $r, s \in Z_q$
- compute $R' = g^{ms^{-1} \bmod q} y^{rs^{-1} \bmod q}$ in $\mathcal{G}$
- Accept (output 1) iff $H'(R') = r$.

The traditional DSA algorithm is obtained by choosing large primes p, q such that $q|(p-1)$ and setting $\mathcal{G}$ to be the order q subgroup of Z_p^*. In this case the multiplication operation in $\mathcal{G}$ is multiplication modulo p. The function H' is defined as $H'(R) = R \bmod q$.

The ECDSA scheme is obtained by choosing $\mathcal{G}$ as a group of points on an elliptic curve of cardinality q. In this case the multiplication operation in $\mathcal{G}$ is the group operation over the curve. The function H' is defined as $H'(R) = R_x \bmod q$ where R_x is the x-coordinate of the point R.

Figure 1: Generic DSA

2.1. ECDSA 门限签名的难点

从前面介绍的标准 ECDSA 签名算法可知，签名中涉及了椭圆曲线上的乘法、加法、求逆等运算，**如何把这些运算改造为分布式方式是 ECDSA 门限签名的难点。**

3. 主要思路

下面以 $(3, n)$ 门限为例，介绍一下 3 个参与者共同完成 ECDSA 签名的思路。首先把 ECDSA 签名算法中的随机数 k 拆成由 3 份相加组成 $k = k_1 + k_2 + k_3$ ，把私钥 x 拆成由 3 份相加组成 $x = w_1 + w_2 + w_3$ （详情参考节 4.2）。每个参与者 P_i 都只知道自己的那部分数据：

```
k_1, w_1          # 参与者 P_1 掌握，不泄露给别人
k_2, w_2          # 参与者 P_2 掌握，不泄露给别人
k_3, w_3          # 参与者 P_3 掌握，不泄露给别人
```

再想方设法去构造签名数据 r, s 。

3.1. 求签名中的 r

只要计算出 R，就容易求出 r。下面介绍一下求 $R = g^{k^{-1}}$ 的思路：

$$
\begin{aligned}
R &= g^{k^{-1}} \\
&= g^{\gamma k^{-1} \gamma^{-1}} \\
&= g^{(\sum \gamma_i) k^{-1} \gamma^{-1}} \\
&= \left(g^{(\sum \gamma_i)}\right)^{(k\gamma)^{-1}} \\
&= \left(\prod g^{\gamma_i}\right)^{(k\gamma)^{-1}}
\end{aligned}
$$

求 R 的思路是：

1. 引入一个变量 γ，同样由 3 份（每份由参与者 P_i 自己随机生成）相加组成 $\gamma = \gamma_1 + \gamma_2 + \gamma_3$，每个参与者单独计算 g^{γ_i}，这个是可以公开的，因为由 g^{γ_i} 反推不出 γ_i，这样，大家就可以算出 $\prod g^{\gamma_i}$ 了。
2. 那如何求解 $k\gamma = (\sum k_i)(\sum \gamma_i)$ 呢？每个参与者 P_i 手里只有 k_i, γ_i，且需要保证这两个值的机密，如何一起配合求得 $(\sum k_i)(\sum \gamma_i)$ 呢？使用一种技巧可以实现，把 $k\gamma$ 拆分为 3 份之和，即 $k\gamma = \delta_1 + \delta_2 + \delta_3$，每个参与者 P_i 计算 δ_i，并公开 δ_i 给其它参与者，这不会泄密 k_i, γ_i，这样每个参与者都可以计算 $k\gamma$ 了，其详情可参考节 4.2.2.1。

有了上面这两个信息，R 就可以计算出来了，从而 r 也可以计算出来了。

3.2. 求签名中的 s

签名中的另一部分 $s = k(m + xr)$，其中 m 是消息的哈希，x 是私钥，r 在上一节已经求得了。下面介绍一下求 s 的思路：

$$
\begin{aligned}
s &= k(m + xr) \\
&= (k_1 + k_2 + k_3)m + kxr \\
&= (k_1 + k_2 + k_3)m + (\sigma_1 + \sigma_2 + \sigma_3)r \\
&= (mk_1 + r\sigma_1) + (mk_2 + r\sigma_2) + (mk_3 + r\sigma_3)
\end{aligned}
$$

求 s 的主要思路是使用一种技巧把 kx 拆分为 3 份之和（至于怎么拆分可参考节 4.2.2.2，其实这和上一节求解 $k\gamma$ 时，把 $k\gamma$ 拆分为 3 份之和的原理是一样的），即 $kx = \sigma_1 + \sigma_2 + \sigma_3$，每个参与者 P_i 单独计算自己的那个 $s_i = mk_i + r\sigma_i$，并公开给其它参与者，这样每个参与者就都可以得到 $s = s_1 + s_2 + s_3$ 了。

4. 协议描述

下面仅描述 GG18 协议的“主要逻辑部分”。

4.1. Distributed Key Generation

假设系统中有 4 个参与者（Alice/Bob/Carol/Dave），其中任意 3 个人可以完成签名，而少于 3 个人无法签名。

下面介绍一个 Distributed Key Generation 流程：
1、每个参与者 P_i 随机选择一个 u_i，这就是每个参与者自己的秘密值，需要各参与者自己保存好。假设 Alice/Bob/Carol/Dave 分别选择了：

```
u_1 = 120
u_2 = 10
u_3 = 30
u_4 = 80
```

整体的秘密（私钥）就是 $x = \sum u_i = u_1 + u_2 + u_3 + u_4 = 120 + 10 + 30 + 80 = 240$，每个参与者都不知道这个秘密值，下面将在不引入 Trusted Dealer 的情况下，把秘密值 $x = 240$ 以 $(3, 4)$ 门限的方式分散给 Alice/Bob/Carol/Dave。

2、每个参与者 P_i 把自己的秘密值 u_i 通过 (t, n) Feldman-VSS 分享给其它所有参与者。每个参与者把自己收到的所有分享全部加起来得到 x_i，容易知道 x_i 是 x 的 (t, n) 分享。比如，Alice 选择一个 2 次多项式 $f_a(x) = 3x^2 - 10x + u_1 = 3x^2 - 10x + 120$ 把 120 分享给所有参与者，计算：

```
f_a(1) = 113       # Alice 自己留着
f_a(2) = 112       # 发送给 Bob
f_a(3) = 117       # 发送给 Carol
f_a(4) = 128       # 发送给 Dave
```

类似地，Bob 选择一个 2 次多项式 $f_b(x) = 6x^2 + x + u_2 = 6x^2 + x + 10$ 把 10 分享给所有参与者，计算：

```
f_b(1) = 17        # 发送给Alice
f_b(2) = 36        # Bob 自己留着
f_b(3) = 67        # 发送给 Carol
f_b(4) = 110       # 发送给 Dave
```

Carol 选择一个 2 次多项式 $f_c(x) = 7x^2 - x + u_3 = 7x^2 - x + 30$ 把 30 分享给所有参与者，计算：

```
f_c(1) = 36        # 发送给 Alice
f_c(2) = 56        # 发送给 Bob
f_c(3) = 90        # Carol 自己留着
f_c(4) = 138       # 发送给 Dave
```

Dave 选择一个 2 次多项式 $f_d(x) = x^2 + 2x + u_4 = x^2 + 2x + 80$ 把 80 分享给所有参与者，计算：

```
f_d(1) = 83        # 发送给 Alice
f_d(2) = 88        # 发送给 Bob
f_d(3) = 95        # 发送给 Carol
f_d(4) = 104       # Dave 自己留着
```

然后，Alice/Bob/Carol/Dave 分别计算他们的 x_i：

```
x_1 = f_a(1) + f_b(1) + f_c(1) + f_d(1) = 113+17+36+83 = 249      # Alice 计算并保存
x_2 = f_a(2) + f_b(2) + f_c(2) + f_d(2) = 112+36+56+88 = 292      # Bob 计算并保存
x_3 = f_a(3) + f_b(3) + f_c(3) + f_d(3) = 117+67+90+95 = 369      # Carol 计算并保存
x_4 = f_a(4) + f_b(4) + f_c(4) + f_d(4) = 128+110+138+104 = 480   # Dave 计算并保存
```

至此， **私钥 240 已经通过 $(3, 4)$ 门限的方式分散到了 Alice/Bob/Carol/Dave 4 个参与者中，也就是说 $(1, 249), (2, 292), (3, 369), (4, 480)$ 这 4 个点中任意找 3 个点通过拉格朗日插值得到的多项式，它在 0 处的值就是秘密值 240，** 如图 2 所示。

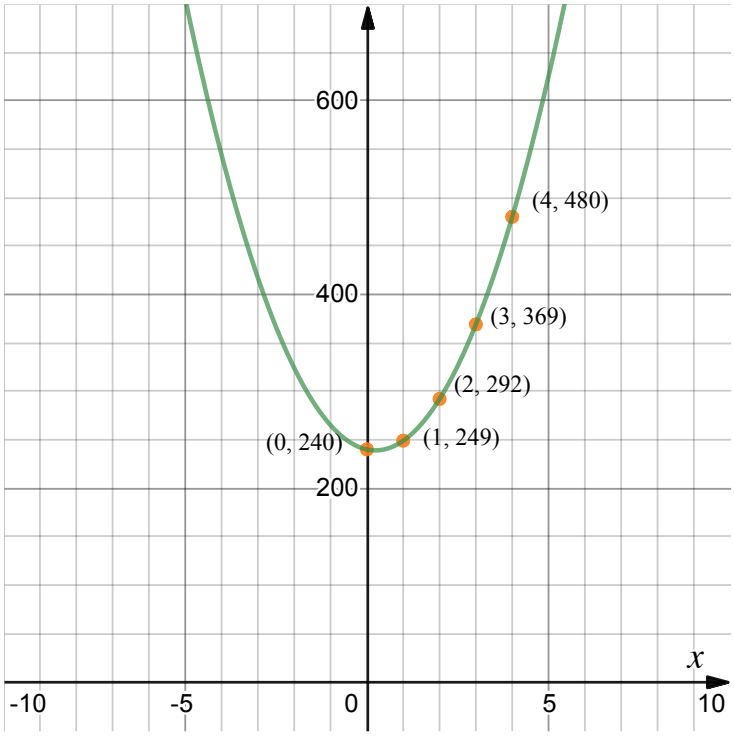

Figure 2: Key Generation，这些点在一个 2 次多项式上（多项式次数为最小阈值减 1），这个曲线（2 次多项式）是由 4 个参与者各自的曲线（2 次多项式）相加而成

可以把上面过程认为是一种“不需要 Trusted Dealer 的 Shamir秘密分享”。

上面介绍了如何把私钥（即 240）分散保存到 4 个参与者中。但如何在不知道 240 这个私钥的情况下，各参与者得到公钥呢？其实很简单，每个参者与公开自己的 $u_i * G$ （这并不会导致 u_i 泄露），公钥就是：

```
y = u_1 * G + u_2 * G + u_3 * G + u_4 * G
  = (u_1 + u_2 + u_3 + u_4) * G
```

上面描述的 DKG 过程是 1991 年 Pedersen 在论文 A Threshold Cryptosystem without a Trusted Party 中提出的，可称为 Pedersen's DKG。

不过，1999 年，Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin 在论文 Secure Distributed Key Generation for Discrete-Log Based Cryptosystems 中指出 Pedersen's DKG 不是安全的，具体来说就是：如果 Pedersen's DKG 过程中攻击者控制了 2 个参与者（共 n 个参与者），那攻击者可以使私钥的分布不再是均匀分布。后来在 2003 年，Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin 在论文 Secure Applications of Pedersen’s Distributed Key Generation Protocol 中指出，Pedersen's DKG 在某些场景下也足够安全。

GG18 中的 DKG 算法是在 Pedersen's DKG 的基础上增加其他手段来确保安全，这里不详细介绍。

4.2. Signature Generation

现在，Alice/Bob/Carol 这 3 个人想要进行签名，他们手头的“部分秘密”分别为 249，292，369

它们是整体秘密 $x = 240$ 的 $(3, 4)$ Feldman-VSS 分享方案，我们先求得整体秘密（即 $x = 240$ ）的 3 人 Additive Sharing 分享方案。即求 w_1, w_2, w_3 ，且满足 $w_1 + w_2 + w_3 = 240$ ，其细节可参考节 8.2，这里直接给出答案：

```
w_1 = 747
w_2 = -876
w_3 = 369
```

4.2.1. Phase 1

Alice/Bob/Carol 选择 k_i, γ_i ，例如：

```
k_1 = 30          # Alice 随机选择
k_2 = 41          # Bob 随机选择
k_3 = 25          # Carol 随机选择
γ_1 = 19          # Alice 随机选择
γ_2 = 8           # Bob 随机选择
γ_3 = 16          # Carol 随机选择
```

定义 $k \triangleq \sum k_i = 30 + 41 + 25 = 96$ ，定义 $\gamma \triangleq \sum \gamma_i = 19 + 8 + 16 = 43$

4.2.2. Phase 2

每两个人都进行两次 MtA 协议（参考节 8.1），具体介绍如下。

4.2.2.1. 第一次 MtA 协议（为求 kγ 做准备）

第一次 MtA 协议： P_i, P_j 对 k_i, γ_j 使用 MtA，把 P_i 得到的结果记为 α_{ij} ， P_j 得到的结果记为 β_{ij} ，也就是满足

$$k_i \gamma_j = \alpha_{ij} + \beta_{ij}$$

具体演示（下面的 α_{ij}, β_{ij} 的值是满足要求情况下随意写的）如下：

```
k_1 * γ_2  =  α_{12} + β_{12}     =>    30  *  8   =  100 + 140
k_1 * γ_3  =  α_{13} + β_{13}     =>    30  *  16  =  120 + 360
k_2 * γ_1  =  α_{21} + β_{21}     =>    41  *  19  =  80  + 699
k_2 * γ_3  =  α_{23} + β_{23}     =>    41  *  16  =  117 + 539
k_3 * γ_1  =  α_{31} + β_{31}     =>    25  *  19  =  218 + 257
k_3 * γ_2  =  α_{32} + β_{32}     =>    25  *  8   =  30  + 170
```

定义 $\delta_i \triangleq k_i \gamma_i + \sum_{j \neq i} \alpha_{ij} + \sum_{j \neq i} \beta_{ji}$ ，则：

$$\begin{aligned}
\delta_1 &= k_1 \gamma_1 + \alpha_{12} + \alpha_{13} + \beta_{21} + \beta_{31} \\
&= 30 * 19 + 100 + 120 + 699 + 257 \\
&= 1746 \\
\delta_2 &= k_2 \gamma_2 + \alpha_{21} + \alpha_{23} + \beta_{12} + \beta_{32} \\
&= 41 * 8 + 80 + 117 + 140 + 170 \\
&= 835 \\
\delta_3 &= k_3 \gamma_3 + \alpha_{31} + \alpha_{32} + \beta_{13} + \beta_{23} \\
&= 25 * 16 + 218 + 30 + 360 + 539 \\
&= 1547
\end{aligned}$$

可以推出 $k\gamma = (\sum k_i) \cdot (\sum \gamma_i) = \sum \delta_i$ 。

Alice （即 P_1 ）公开 δ_1 给 Bob 和 Carol；而 Bob （即 P_2 ）公开 δ_2 给 Alice 和 Bob；Carol （即 P_3 ）公开 δ_3 给 Alice 和 Bob。这样，Alice/Bob/Carol 在不公开自己拥有的 k_i, γ_i 的情况下，都可以计算出 $k\gamma$ 了。

4.2.2.2. 第二次 MtA 协议（为求 s_i 做准备）

第二次 MtA 协议： P_i, P_j 对 k_i, w_j 使用 MtA，把 P_i 得到的结果记为 μ_{ij} ， P_j 得到的结果记为 ν_{ij} ，也就是满足

$$k_i w_j = \mu_{ij} + \nu_{ij}$$

具体演示（下面的 μ_{ij}, ν_{ij} 的值是满足要求情况下随意写的）如下：

```
k_1 * w_2  =  μ_{12} + ν_{12}     =>    30  *  (-876)  = -51000 + 24720
k_1 * w_3  =  μ_{13} + ν_{13}     =>    30  *  369   =  4000 + 7070
k_2 * w_1  =  μ_{21} + ν_{21}     =>    41  *  747   =  21452 + 9175
k_2 * w_3  =  μ_{23} + ν_{23}     =>    41  *  369   =  3413 + 11716
k_3 * w_1  =  μ_{31} + ν_{31}     =>    25  *  747   =  14737 + 3938
k_3 * w_2  =  μ_{32} + ν_{32}     =>    25  *  (-876)  = -30000 + 8100
```

定义 $\sigma_i \triangleq k_i w_i + \sum_{j \neq i} \mu_{ij} + \sum_{j \neq i} \nu_{ji}$ ，则：

$$\begin{aligned}
\sigma_1 &= k_1 w_1 + \mu_{12} + \mu_{13} + \nu_{21} + \nu_{31} \\
&= 30 * 747 + (-51000) + 4000 + 9175 + 3938 \\
&= -11477 \\
\sigma_2 &= k_2 w_2 + \mu_{21} + \mu_{23} + \nu_{12} + \nu_{32} \\
&= 41 * (-876) + 21452 + 3413 + 24720 + 8100 \\
&= 21769 \\
\sigma_3 &= k_3 w_3 + \mu_{31} + \mu_{32} + \nu_{13} + \nu_{23} \\
&= 25 * 369 + 14737 + (-30000) + 7070 + 11716 \\
&= 12748
\end{aligned}$$

可以推出 $kx = (\sum k_i) \cdot (\sum w_i) = \sum \sigma_i$ 。

4.2.3. Phase 3 （求 r）

Alice/Bob/Carol 公开 δ_i ，这样，所有参与者都可以计算出 $k\gamma = \sum \delta_i = 1746 + 835 + 1547 = 4128$ ， $k\gamma$ 是求 r 所需要计算的东西，参考节 3.1 。

4.2.4. Phase 4 （求 r）

Alice/Bob/Carol 公开 g_i^{γ} ，这样可以求得 R ，从而求得 r 了，参考节 3.1 。

4.2.5. Phase 5 （求 s）

在第二次 MtA 协议后，Alice/Bob/Carol 各自计算出了 σ_i ，从而各自可以计算出 $s_i \triangleq mk_i + r\sigma_i$ ，再公开（不是直接公开，还有一些防止别人做恶的手段，这里省略）给大家，这样大家就可以求得 $s = s_1 + s_2 + s_3$ 了，参考节 3.2 。

5. GG18 VS. GG20

Rosario Gennaro 和 Steven Goldfeder 于 2020 年发表论文 One Round Threshold ECDSA with Identifiable Abort，被称为 GG20。它比 GG18 更优秀：

1. Rounds 数量更少；GG18 的 sign 过程需要 9 rounds，而 GG20 的 sign 过程只需要 7 rounds（前 6 个 rounds 和待签名的 msg 无关，可提前进行；最后一个 round 才和 msg 相关）。关于 GG20 的 7 rounds 中每个 round 的细节可参考论文 UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts 。
2. 可识别恶意参与者。GG18 协议，如果有恶意参与者，那么协议会终止，但在某些场景下却不知道谁是恶意参与者。在 GG20 协议中，只要默认失败，就可识别出是谁的责任。

6. 开源实现

GG18, GG20 的开源实现可以参考：

1. https://github.com/bnb-chain/tss-lib GG18 实现
2. https://gitlab.com/thorchain/tss/tss-lib GG20 实现
3. https://github.com/ZenGo-X/multi-party-ecdsa GG18 和 GG20 实现

7. 扩展阅读

7.1. Multi-Party Reshare（动态改变门限 t）

Distributed Key Generation 结束后，门限 t 就确定了，以后还可以改变吗？比如从 $(3, 4)$ 改为新门限设置 $(5, 7)$？答案是肯定的，不过在 GG18 论文中没有提到这一点。

具体做法是分两步：

1. 从 $(3, 4)$ 变为 3 人 Additive Sharing，参考节 8.2 ；
2. 把 3 人的 Additive Sharing 当作是 DKG（参考节 4.1）过程中的随机数 u_i （由于新门限的 n 值是 7，设置多出来的 4 个节点其 $u_i = 0$ 即可），再接着运行 DKG 即可得到新门限设置 $(5, 7)$ 的 DKG 结果。

在论文 Attacking Threshold Wallets 的 3.1 节中介绍这个过程。

7.2. 其它 ECDSA 门限签名算法

除 GG18/GG20 外，还有其它一些 ECDSA 门限签名算法，可参考：A Survey of ECDSA Threshold Signing

8. Appendix

8.1. Multiplicative-to-Additive（MtA）协议

假设 Alice 有个秘密值 a ，而 Bob 有个秘密值 b ，下面介绍一个协议可以在 Alice 不泄露 a ，Bob 不泄露 b 的情况下，Alice 和 Bob 分别得到另外一个秘密值 α, β ，满足：

$$ab = \alpha + \beta \bmod q$$

这个协议称为乘法到加法的转换协议（MtA）。

图 3 是 MtA 协议的示意图，协议中用到了 Paillier 同态加密。

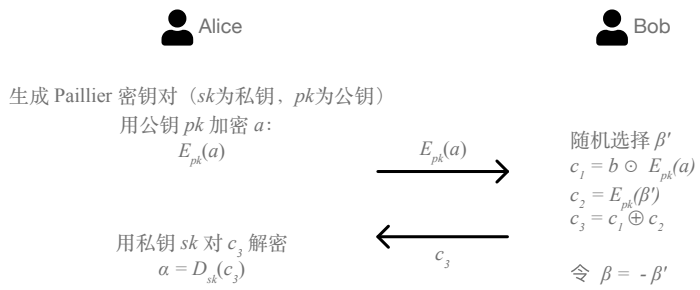

Figure 3: Multiplicative-to-Additive（MtA）协议

Alice 把 a 使用 Alice 公钥进行加密后，给 Bob；而 Bob 利用 Paillier 同态加密性质，把加密（当然也是 Alice 公钥加密）后的 $ab - \beta$ 给 Alice。这样 Alice 使用私钥解密后得到的值就是 α，它们会满足 $ab = \alpha + \beta \bmod q$ 。

8.2. 转换 (t,n) 秘密为 Additive Sharing

Alice/Bob/Carol 手头分别有部分秘密 $x_1 = 249, x_2 = 292, x_3 = 369$，它们是整体秘密 $x = 240$ 的 $(3, 4)$ Feldman-VSS 分享方案，现在想把它变为整体秘密 $x = 240$ 的 3 人 additive sharing 分享方案。也就是 Alice/Bob/Carol 通过计算分别得到另外一个部分秘密 w_1, w_2, w_3，要满足 $w_1 + w_2 + w_3 = 240$ 。

具体来说， w_i 可以通过拉格朗日系数乘以 x_i 得到：

$$
\begin{aligned}
w_1 = \lambda_1 \cdot x_1 = \frac{0-2}{1-2} \cdot \frac{0-3}{1-3} \cdot x_1 = 3 \cdot 249 = 747 \\
w_2 = \lambda_2 \cdot x_2 = \frac{0-1}{2-1} \cdot \frac{0-3}{2-3} \cdot x_2 = -3 \cdot 292 = -876 \\
w_3 = \lambda_3 \cdot x_3 = \frac{0-1}{3-1} \cdot \frac{0-2}{3-2} \cdot x_3 = 1 \cdot 369 = 369
\end{aligned}
$$

显然满足 $w_1 + w_2 + w_3 = 747 - 876 + 369 = 240$ 。

为什么 $w_1 + w_2 + w_3 = \lambda_1 \cdot x_1 + \lambda_2 \cdot x_2 + \lambda_3 \cdot x_3 = \lambda_1 \cdot f(1) + \lambda_2 \cdot f(2) + \lambda_3 \cdot f(3)$ 会恰好等于完整私钥 $f(0)$ 呢？这是因为已知 3 个点的值 $f(1) = 249, f(2) = 292, f(3) = 369$，其拉格朗日插值多项式为：

$$
L(x) = f(1) \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} + f(2) \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} + f(3) \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2}
$$

而 $f(0) = L(0) = \underbrace{f(1) \cdot \frac{0-2}{1-2} \cdot \frac{0-3}{1-3}}_{w_1} + \underbrace{f(2) \cdot \frac{0-1}{2-1} \cdot \frac{0-3}{2-3}}_{w_2} + \underbrace{f(3) \cdot \frac{0-1}{3-1} \cdot \frac{0-2}{3-2}}_{w_3}$