TEA 算法

1. TEA 算法的基本参数

参数	说明
分组长度	64 位（2×32 位）
密钥长度	128 位（4×32 位）
加密轮数	通常 **32 轮（64 次 Feistel 迭代）**
运算方式	**Feistel 结构**（左右两部分交替加密）
魔数（delta）	**`0x9E3779B9`**（黄金比例的 32 位整数近似）

---

2. TEA 加密过程

（1）输入

明文（64 位）：分成两个 32 位无符号整数 **`v0`** 和 **`v1`**，即 **`v = (v0, v1)`**。
密钥（128 位）：分成四个 32 位无符号整数 **`k = (k0, k1, k2, k3)`**。

（2）加密伪代码

Python

```python
def tea_encrypt(v, k):
    v0, v1 = v[0], v[1]
    delta = 0x9E3779B9
    sum = 0
    for i in range(32):  # 32 轮加密
        sum += delta
        v0 += ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1)
        v1 += ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3)
    return (v0, v1)
```

关键运算解析：

`sum += delta`：每轮增加一个固定值 `delta = 0x9E3779B9`（用于增强混淆）。

`v0` 的更新：

`(v1 << 4) + k0`：`v1` 左移 4 位后加上 `k0`。

`(v1 + sum)`：`v1` 加上当前轮次的 `sum`。

`(v1 >> 5) + k1`：`v1` 右移 5 位后加上 `k1`。

最后用 **异或（XOR）** 混合这些值，并加到 `v0` 上。

`v1` 的更新：

类似 `v0`，但使用 `k2` 和 `k3`。

（3）加密示例

明文 `v = (0x01234567, 0x89ABCDEF)`

密钥 `k = (0xDEADBEEF, 0xCAFEBABE, 0x8BADF00D, 0x1BADB002)`

加密过程：

初始化 `sum = 0`。

进行 32 轮计算，每轮 `sum += delta`，并更新 `v0` 和 `v1`。

最终输出 `(v0, v1)` 即为密文。

3. TEA 解密过程

解密是加密的逆过程，`sum` 初始值为 `delta × 32`，并逐步减少 `delta`。

（1）解密伪代码

python

```python
def tea_decrypt(v, k):
    v0, v1 = v[0], v[1]
    delta = 0x9E3779B9
    sum = delta * 32  # 初始 sum = delta × 轮数
    for i in range(32):
        v1 -= ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3)
        v0 -= ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1)
        sum -= delta
    return (v0, v1)
```

关键运算解析：

`sum` 初始为 `delta × 32`，然后每轮减少 **`delta`**。
先更新 `v1`，再更新 `v0`（与加密顺序相反）。
使用 **减法（`-=`）** 而不是加密时的加法（**`+=`**）。