CTF中常见签名——RSA与DSA

方向：Crypto

主讲：Vancleef

Oct 18, 2019

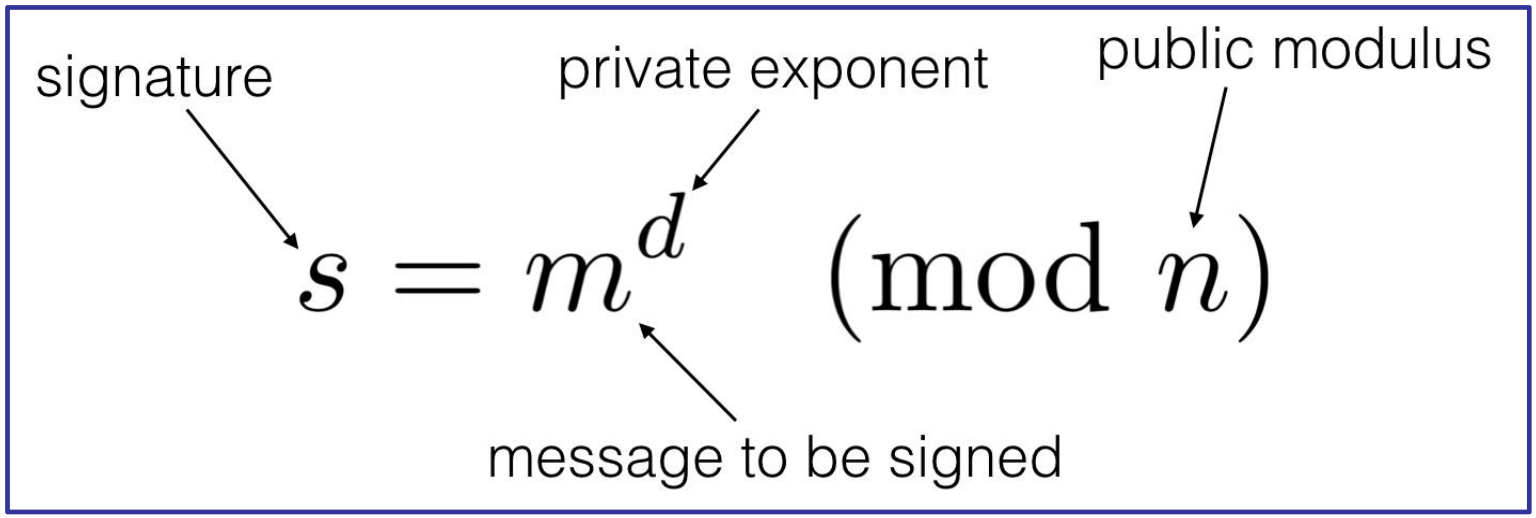
Outline

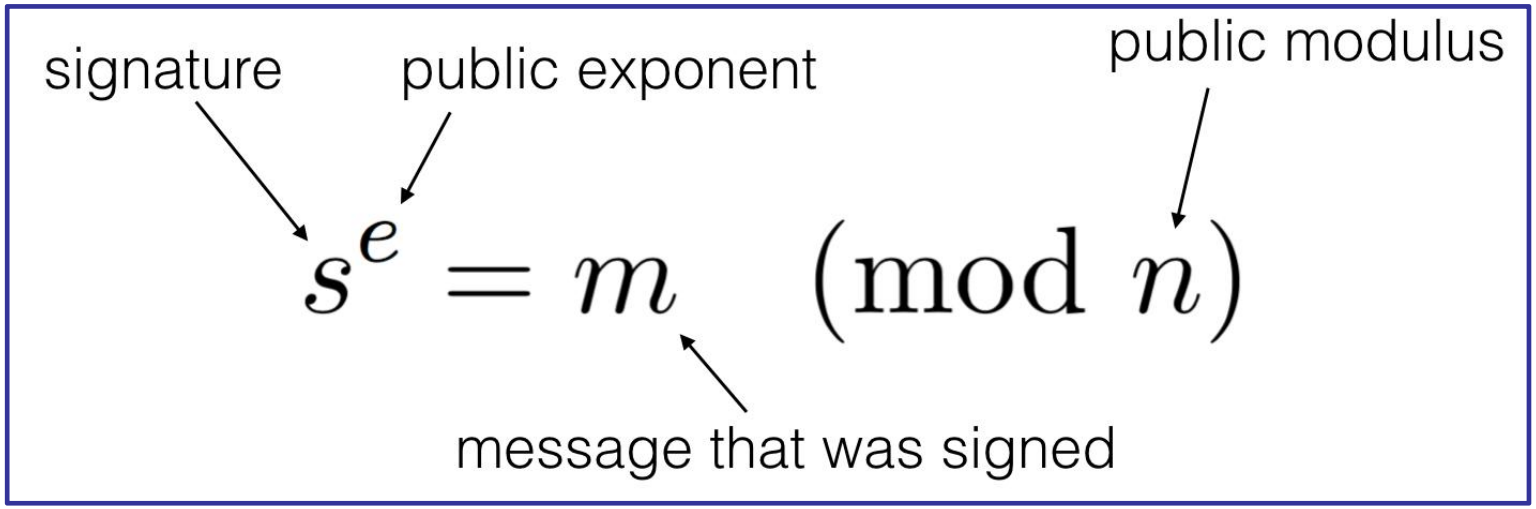
① RSA签名及其常见攻击方式

② DSA签名原理

③ DSA常见攻击方式

1.1 RSA签名


signature
private exponent
public modulus
$s = m^d \pmod{n}$
message to be signed


signature
public exponent
public modulus
$s^e = m \pmod{n}$
message that was signed

1.2 CRT

$$d_p = d \pmod{p-1}$$

$$d_q = d \pmod{q-1}$$

$$\left.\begin{aligned} s_1 &= m^{d_p} \pmod{p} \\ s_2 &= m^{d_q} \pmod{q} \end{aligned}\right\} \Longrightarrow s \pmod{pq}$$

使用CRT比不使用CRT理论上快四倍左右

1.3 Fault Attack

在应用中国剩余定理的RSA系统中，只要设备执行签名\加密时，满足以下三个条件[5]：

(1)签名\加密消息已知；

(2)在签名\加密时出现了一个错误；

(3)设备输出了错误的签名\密文。

就可能导致系统的大整数 n 被分解开。

```
Signature Box:
n =
109453943855659271832842942560403561756870609158731037775662385351197002786101553313741128891698293686816823489928849957259082253354725368827785717195435302172225437710056999830798824021337641459232307818433049338256971969203037801338219292565591596823843084807425759066355695095315215256626194447457000935219
e = 65537
test_message = "this is test message"
test_message_int = 664571392881790422435277591416285461838362142565

测试一：
signature1 =
953619475475780273988448736159108534292491968408752106666964593181078200344972769562754792402499522304771941765763108961736896311617405882022989641253113727125799424988449508447769415484409225769138287707210635058021299553385450267916788698880115379732518630688246390143101127911037937936022579596209773039591660
测试二：
signature2 =
530993803214208865927705472070113469909668731557692536624158033066940933
```

1.3 Fault Attack

$$\left.\begin{aligned} s_1 &= m^{d_p} \pmod{p} \\ \widetilde{s_2} &= m^{d_q} \pmod{q} \end{aligned}\right\} \Longrightarrow \widetilde{s} \pmod{pq}$$

$$\begin{cases} \widetilde{s}^e = m \pmod{p} \\ \widetilde{s}^e \neq m \pmod{q} \end{cases} \Longrightarrow \begin{cases} \widetilde{s}^e - m = 0 \pmod{p} \\ \widetilde{s}^e - m \neq 0 \pmod{q} \end{cases} \Longrightarrow \begin{cases} p \mid \widetilde{s}^e - m \\ q \nmid \widetilde{s}^e - m \end{cases}$$

防止Fault Attack：
1. 冗余计算S1与S2；
2. 得到签名后程序先验证签名再输出。

Outline

① RSA签名及其常见攻击方式

② DSA签名原理

③ DSA常见攻击方式

2.1 DSA签名

密钥生成

1. 选择一个合适的哈希函数，目前一般选择 SHA1，当前也可以选择强度更高的哈希函数 H。
2. 选择密钥的长度 L 和 N，这两个值决定了签名的安全程度。在最初的 DSS（**Digital Signature Standard**）中建议 L 必须为 64 的倍数，并且$512 \leq L \leq 1024$，当然，也可以更大。N 必须大小必须不大于哈希函数 H 输出的长度。FIPS 186-3 给出了一些建议的 L 和 N 的取值例子：(1024, 160)，(2048, 224)，(2048, 256)，以及 (3,072, 256)。
3. 选择 N 比特的素数 q。
4. 选择 L 比特的素数 p，使得 p-1 是 q 的倍数。
5. 选择满足$g^k \equiv 1 \bmod p$的最小正整数 k 为 q 的 g，即在模 p 的背景下，ord(g)=q 的 g。即 g 在模 p 的意义下，其指数次幂可以生成具有 q 个元素的子群。这里，我们可以通过计算$g = h^{\frac{p-1}{q}} \bmod p$来得到 g，其中$1 < h < p - 1$。
6. 选择私钥 x，$0 < x < q$，计算$y \equiv g^x \bmod p$。

公钥为 (p,q,g,y)，私钥为 (x)。

2.1 DSA签名

签名

签名步骤如下

1. 选择随机整数数 k 作为临时密钥，$0 < k < q$。
2. 计算$r \equiv (g^k \bmod p) \bmod q$
3. 计算$s \equiv (H(m) + xr)k^{-1} \bmod q$

签名结果为 (r,s)。需要注意的是，这里与 Elgamal 很重要的不同是这里使用了哈希函数对消息进行了哈希处理。

验证

验证过程如下

1. 计算辅助值，$w = s^{-1} \bmod q$
2. 计算辅助值，$u_1 = H(m)w \bmod q$
3. 计算辅助值，$u_2 = rw \bmod q$
4. 计算$v = (g^{u_1}y^{u_2} \bmod p) \bmod q$
5. 如果 v 与 r 相等，则校验成功。

2.2 正确性推导

正确性推导

首先，g 满足 $g^k \equiv 1 \bmod p$ 的最小正整数 k 为 q。所以 $g^q \equiv 1 \bmod p$。所以 $g^x \equiv g^{x \bmod q} \bmod p$。进而

$$v = (g^{u_1} y^{u_2} \bmod p) \bmod q = g^{u_1} g^{xu_2} \equiv g^{H(m)w} g^{xrw} \equiv g^{H(m)w+xrw}$$

又$s \equiv (H(m) + xr)k^{-1} \bmod q$ 且$w = s^{-1} \bmod q$ 所以

$$k \equiv s^{-1}(H(m) + xr) \equiv H(m)w + xrw \bmod q$$

所以$v \equiv g^k$。正确性得证。

DSA的安全性：
1.依赖于离散对数问题
2.ElGamal相关

Outline

① RSA签名及其常见攻击方式

② DSA签名原理

③ DSA常见攻击方式

3.1 常见攻击方法

① Known K

② Sahred K

③ Biased K

3.2 Known K

原理

如果知道了随机密钥 k，那么我们就可以根据$s \equiv (H(m) + xr)k^{-1} \bmod q$计算私钥 d，几乎攻破了 DSA。

这里一般情况下，消息的 hash 值都会给出。

$$x \equiv r^{-1}(ks - H(m)) \bmod q$$

3.3 Shared K

原理

如果在两次签名的过程中共享了 k，我们就可以进行攻击。

假设签名的消息为 m1,m2，显然，两者的 r 的值一样，此外

$$s_1 \equiv (H(m_1) + xr)k^{-1} \bmod q$$

$$s_2 \equiv (H(m_2) + xr)k^{-1} \bmod q$$

这里我们除了 x 和 k 不知道剩下的均知道，那么

$$s_1 k \equiv H(m_1) + xr$$

$$s_2 k \equiv H(m_2) + xr$$

两式相减

$$k(s_1 - s_2) \equiv H(m_1) - H(m_2) \bmod q$$

此时 即可解出 k，进一步我们可以解出 x。

相关题目：
2016 湖湘杯 DSA
2019 SUCTF DSA

3.4 Biased K —— Lattice

格在数学上至少有两种含义

- 定义在非空有限集合上的偏序集合 L，满足集合 L 中的任意元素 a，b，使得 a，b 在 L 中存在一个最大下界，和最小上界。具体参见 https://en.wikipedia.org/wiki/Lattice_(order)。
- 群论中的定义，是 R^n 中的满足某种性质的子集。当然，也可以是其它群。

目前关于格方面的研究主要有以下几大方向

1. 格中计算问题的困难性，即这些问题的计算复杂性，主要包括
    a. SVP 问题
    b. CVP 问题
2. 如何求解格中的困难性问题，目前既有近似算法，也有一些精确性算法。
3. 基于格的密码分析，即如何利用格理论分析一些已有的密码学算法，目前有如下研究
    a. Knapsack cryptosystems
    b. DSA nonce biases
    c. Factoring RSA keys with bits known
    d. Small RSA private exponents
    e. Stereotyped messages with small RSA exponents

3.4 Biased K —— Lattice

格定义

格是 m 维欧式空间 R^m 的 n $(m \geq n)$ 个线性无关向量$b_i(1 \leq i \leq n)$ 的所有整系数的线性组合，即 $L(B) = \{\sum_{i=1}^{n} x_i b_i : x_i \in Z, 1 \leq i \leq n\}$

这里 B 就是 n 个向量的集合，我们称

- 这 n 个向量是格 L 的一组基。
- 格 L 的秩为 n。
- 格 L 的位数为 m。

如果 m=n，那么我们称这个格式满秩的。

当然，也可以是其它群，不是 R^m。

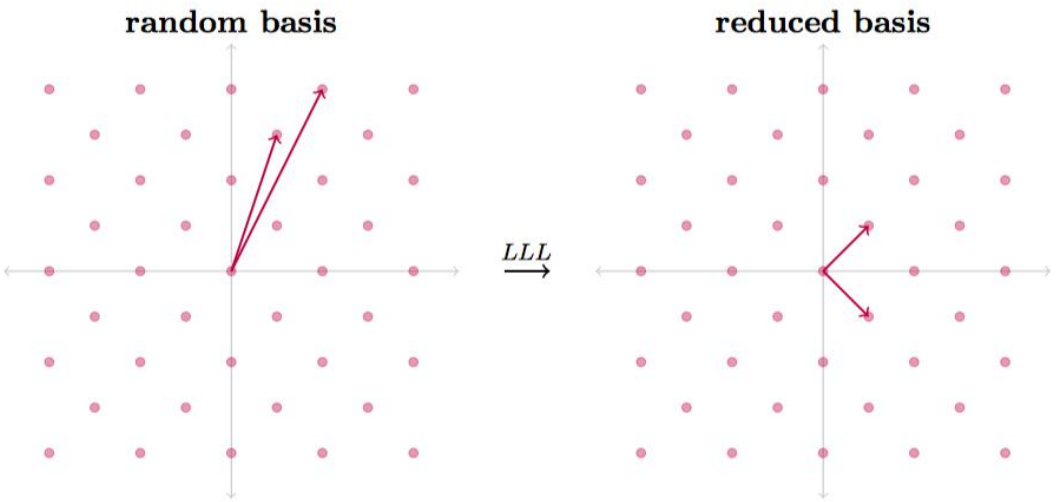
3.4 Biased K —— Lattice

Definition 1. *Let L be a lattice with a basis B. The δ-LLL algorithm applied on L's basis B produces a new basis of L : $B' = \{b_1, \ldots, b_n\}$ satisfying :*

$$\forall\, 1 \leq j < i \leq n \text{ we have } |\mu_{i,j}| \leq \frac{1}{2} \tag{1}$$

$$\forall\, 1 \leq i < n \text{ we have } \delta \cdot \|\tilde{b}_i\|^2 \leq \|\mu_{i+1,i} \cdot \tilde{b}_i + \tilde{b}_{i+1}\|^2 \tag{2}$$

$$\text{with } \mu_{i,j} = \frac{b_i \cdot \tilde{b}_j}{\tilde{b}_j \cdot \tilde{b}_j} \text{ and } \tilde{b}_1 = b_1 \text{ (Gram-Schmidt)}$$



Property 1. *Let L be a lattice of dimension n. In polynomial time, the LLL algorithm outputs reduced basis vectors v_i, for $1 \leq i \leq n$, satisfying :*

$$\|v_1\| \leq \|v_2\| \leq \ldots \leq \|v_i\| \leq 2^{\frac{n(n-1)}{4(n+1-i)}} \cdot det(L)^{\frac{1}{n+1-i}}$$

3.4 Biased K —— HNP

Hidden number problem

HNP 的定义如下：

给定质数p、许多$t \in \mathbb{F}_p$ 以及每一个对应的$MSB_{l,p}(\alpha t)$，找出对应的α。

- $MSB_{l,p}(x)$ 表示任一满足 $|(x \mod p) - u| \leq \frac{p}{2^{l+1}}$ 的整数 u，近似为取$x \mod p$ 的l个最高有效位。

根据参考 3 中的描述，当$l \approx \log^{\frac{1}{2}} p$ 时，有如下算法可以解决 HNP：

我们可以将此问题转化为一个由该矩阵生成的格上的 CVP 问题：

$$\begin{bmatrix} p & 0 & \dots & 0 & 0 \\ 0 & p & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & \dots & p & 0 \\ t_1 & t_2 & \dots & t_n & \frac{1}{2^{l+1}} \end{bmatrix}$$

我们需要找到在格上离$\mathbf{u} = (u_1, u_2, \dots, u_n, 0)$ 最近的向量，所以在这里，我们可以采用 `Babai's nearest plane algorithm`。最终我们可以得到一组向量 $\mathbf{v} = (\alpha \cdot t_1 \mod p, \alpha \cdot t_2 \mod p, \dots, \frac{\alpha}{2^{l+1}})$，从而算出 α。

3.4 Biased K —— Babai's Algorithm

1 The Nearest Plane Algorithm

The algorithm has two main steps. First, it applies the LLL reduction to the input lattice. It then looks for an integer combination of the basis vectors that is close to the target vector t. This step is essentially the same as one inner loop in the reduction step of the LLL algorithm.

INPUT: Basis $B \in \mathbb{Z}^{m \times n}$, $t \in \mathbb{Z}^m$

OUTPUT: A vector $x \in \mathcal{L}(B)$ such that $\|x - t\| \leq 2^{\frac{n}{2}} \operatorname{dist}(t, \mathcal{L}(B))$

1. Run δ-LLL on B with $\delta = \frac{3}{4}$
2. $b \leftarrow t$

   for $j = n$ to 1 **do**

   $b \leftarrow b - c_j b_j$ where $c_j = \lceil \langle b, \tilde{b}_j \rangle / \langle \tilde{b}_j, \tilde{b}_j \rangle \rfloor$

   Output $t - b$

这个算法严谨的的正确性证明可以搜索有关文献，此处只说一下大概的思想。

3.4 Biased K —— Babai's Algorithm

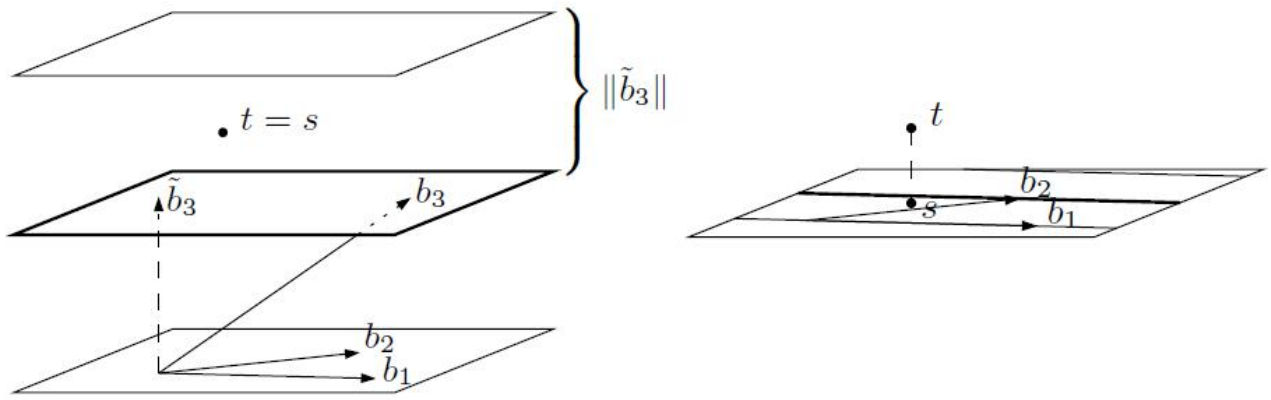


Figure 1: The nearest plane algorithm for a rank 3 lattice and the resulting rank 2 instance. The chosen hyperplanes are thicker.

1. Let s be the projection of t on $\mathrm{span}(b_1, \ldots, b_n)$.
2. Find c such that the hyperplane $c\tilde{b}_n + \mathrm{span}(b_1, \ldots, b_{n-1})$ is as close as possible to s.
3. Let $s' = s - cb_n$. Call recursively with s' and $\mathcal{L}(b_1, \ldots, b_{n-1})$. Let x' be the answer.
4. Return $x = x' + cb_n$.

3.4 Biased K —— Application

So let's say you know the low ℓ bits of k. (This paper describes one way of recovering those bits.) Then you can write k as

$$k = a + 2^\ell b$$

In other words, you know $a \in [0, 2^\ell - 1]$. For simplicity, let $a = 0$. Then the equation for s in the ECDSA signature (r, s) becomes

$$s = (h + rx) \cdot (2^\ell b)^{-1}$$

where h is your hashed message and x is your private key, and everything is modulo q, the order of your base point. Rewrite this as

$$xr \cdot (2^\ell s)^{-1} = -h \cdot (2^\ell s)^{-1} + b$$

Define $t \equiv r \cdot (2^\ell s)^{-1}$ and $u \equiv -h \cdot (2^\ell s)^{-1}$ and you have

$$xt = u + b$$

Remembering that $0 < b < q/2^\ell$, you have

$$xt - u < q/2^\ell$$

3.4 Biased K —— Application

Given a message μ signed with the nonce k, the congruence

$$\alpha r(k) \equiv s(k,\mu)k - h(\mu) \pmod{q},$$

can be rewritten for $s(k,\mu) \neq 0$ as:

$$\alpha r(k) 2^{-\ell} s(k,\mu)^{-1} \equiv \left(a - s(k,\mu)^{-1} h(\mu)\right) 2^{-\ell} + b \pmod{q}. \quad (1)$$

Now define the following two elements

$$t(k,\mu) = \left\lfloor 2^{-\ell} r(k) s(k,\mu)^{-1} \right\rfloor_q,$$

$$u(k,\mu) = \left\lfloor 2^{-\ell} \left(a - s(k,\mu)^{-1} h(\mu)\right) \right\rfloor_q,$$