Attacks against RSA signatures

Jean-Sébastien Coron

Université du Luxembourg

1 Fault attacks against RSA signatures

1. Implement the signature generation algorithm using the Chinese Remainder Theorem (CRT) using the Sage library. More precisely, to compute $s = m^d \bmod N$, compute
$$s_p = s \mod p = m^{d \bmod p-1} \mod p$$
and
$$s_q = s \mod q = m^{d \bmod q-1} \mod q$$
Recover $s \bmod N$ from s_p and s_q using the CRT.
2. Assume that an error occurs during the computation of s_p, that is, an incorrect value $s'_p \neq s_p$ is computed while s_q is correctly computed. Explain and implement how to recover the factorization of N from s, following the Bellcore attack [BDL97].
3. How could such error be detected ? Propose and implement a simple method to detect such error.

2 The Desmedt-Odlyzko attack

The goal is to implement the Desmedt-Odlyzko attack [DO85] described in the lecture, with the RSA signature scheme:
$$\sigma = H(m)^d \pmod N$$
for a hash function H of size k bits. The attack computes a forged signature as a multiplicative combination of existing signatures.

2.1 Signature scheme

We assume that we work with a public exponent $e = 3$. In that case, the key generation can be implemented as follows:

```
def keyGen(n=256):
  e=3
  while True:
    p=random_prime(2^(n//2));q=random_prime(2^(n//2))
    if gcd(e,(p-1)*(q-1))==1: break
  d=inverse_mod(e,(p-1)*(q-1))
  Nn=p*q
  return Nn,p,q,e,d
```

For simplicity the hash function can be computed as follows:

```
import hashlib

def sha1(s,digestsize=50):
  m = hashlib.sha1()
  m.update(s)
  return Integer(m.hexdigest(),base=16) % 2^digestsize
```

2.2 The attack

1. We generate the list $p_1, \ldots, p_\ell$ of the first ℓ primes, and we fix the smoothness bound $B = p_\ell$.
2. We find $\ell + 1$ messages m_i such that the $H(m_i)$ are B-smooth:
$$H(m_i) = p_1^{v_{i,1}} \cdots p_\ell^{v_{i,\ell}}$$
To detect smooth numbers among $H(m_i)$, one can use the `factor()` function from Sage.
3. We put the corresponding vector of exponent in the rows $\mathbf{M}_i$ of a $(\ell+1) \times \ell$ matrix $\mathbf{M}$:
$$\mathbf{M}_i = (v_{i,1} \bmod e, \ldots, v_{i,\ell} \bmod e)$$
4. Since we have $\tau = \ell + 1$ vectors of dimension ℓ and we are working modulo a prime $e = 3$, one vector must be a linear combination of the others. Such linear combination can be found by computing the first vector $\mathbf{u}$ of the left kernel of the matrix $\mathbf{M}$

```
u=Matrix(GF(3),M).left_kernel().matrix()[0]
```

This gives:
$$\sum_{i=1}^{\ell+1} u_i \cdot \mathbf{M}_i = 0 \pmod 3$$
We can then take the first $i^\star$ such that $u_{i^\star} \neq 0$. We can assume $u_{i^\star} = 2 \pmod 3$, otherwise we can let $\mathbf{u} \leftarrow -\mathbf{u} \pmod 3$. This enables to write:
$$\mathbf{M}_{i^\star} = \sum_{i \neq i^\star} u_i \cdot \mathbf{M}_i \pmod 3$$
This enables to write the linear relation on the exponents for all $1 \leq j \leq \ell$:
$$v_{i^\star,j} = \gamma_j \cdot e + \sum_{i \neq i^\star} u_i \cdot v_{i,j}$$
This gives the following multiplicative relation on the $H(m_i)$:
$$\begin{aligned} H(m_{i^\star}) &= \prod_{j=1}^{\ell} p_j^{v_{i^\star,j}} = \prod_{j=1}^{\ell} p_j^{\gamma_j \cdot e + \sum\limits_{i \neq i^\star} u_i \cdot v_{i,j}} = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j}\right)^e \cdot \prod_{j=1}^{\ell} \prod_{i \neq i^\star} p_j^{v_{i,j} \cdot u_i} \\ &= \left(\prod_{j=1}^{\ell} p_j^{\gamma_j}\right)^e \cdot \prod_{i \neq i^\star} \left(\prod_{j=1}^{\ell} p_j^{v_{i,j}}\right)^{u_i} = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j}\right)^e \cdot \prod_{i \neq i^\star} H(m_i)^{u_i} \end{aligned}$$
5. Writing
$$H(m_{i^\star}) = \delta^e \cdot \prod_{i \neq i^\star} H(m_i)^{u_i}, \quad \text{where } \delta := \prod_{j=1}^{\ell} p_j^{\gamma_j}$$
this gives a forgery. Namely, the attacker asks the signatures σ_i of m_i for $i \neq i^\star$ and forges the signature $\sigma_{i^\star}$ of $m_{i^\star}$:
$$\sigma_{i^\star} = H(m_{i^\star})^d = \delta \cdot \prod_{i \neq i^\star} \left(H(m_i)^d\right)^{u_i} \pmod N$$
$$\sigma_{i^\star} = \delta \cdot \prod_{i \neq i^\star} \sigma_i^{u_i} \pmod N$$

2.3 Testing the attack

To test the attack, one can use small parameters, for example `digestsize`=50, and a number of primes $\ell = 100$. It can be interesting to optimize the running time by varying ℓ for a fixed `digestsize`.

Implement the attack and compute experimentally the number of signatures needed as a function of the digest size, for small values of `digestsize`.

References

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 37–51, 1997.

[DO85] Yvo Desmedt and Andrew M. Odlyzko. A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes. In *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, pages 516–522, 1985.