RSA-CRT fault attack

How RSA signatures work

1. $p, q \in \mathbb{Z}_{2^{1024}}$ – two sufficiently large primes and modulus $n = pq$
2. private exponent $d \in \mathbb{Z}_{\varphi(n)}$, public exponent $e = d^{-1} \in \mathbb{Z}_{\varphi(n)}$.
3. If μ is the padding scheme (such as FDH, PFDH, PKCS #1 v1.5, PKCS #1 v2.5), the signature is $\sigma = (\mu(m))^d \pmod{n}$.
4. Verification $\sigma^e \pmod{n} = \mu(m)$.

Calculating in $\mathbb{Z}_n$ (if n is a 2048-bit integer) is slow.

```
$openssl speed rsa2048

... 1431 signatures per second,
... 51952 veritifactions per second
```

At least much slower, compared to calculating in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ separately. This gives 4 times performance increase. The Chinese Remainder Theorem (CRT) allows to speed-up computations.

$$\begin{cases} \sigma_p \equiv m^{d \bmod \varphi(p)} \pmod{p} \\ \sigma_q \equiv m^{d \bmod \varphi(q)} \pmod{q} \end{cases} \implies \sigma = CRT(\sigma_p, \sigma_q) \mod n \ .$$

Bellcore attack

$$\begin{cases} \sigma_p \equiv m^{d \bmod \varphi(p)} \pmod{p} \\ \hat{\sigma}_q \not\equiv m^{d \bmod \varphi(q)} \pmod{q} \end{cases} \implies \hat{\sigma} = CRT(\sigma_p, \hat{\sigma}_q) \mod n \ .$$

If an attacker manages to inject a fault so that she obtains two RSA signatures, one valid signature σ, and an invalid signature $\hat{\sigma}$, then

$$\begin{cases} (\sigma - \hat{\sigma}) \equiv 0 \pmod{p} \\ (\sigma - \hat{\sigma}) \not\equiv 0 \pmod{q} \end{cases} \implies \gcd(\sigma - \hat{\sigma}, n) = p \ .$$

This case can be reduced to just the knowledge of one faulty signature – the Boneh–DeMillo–Lipton attack.

Boneh–DeMillo–Lipton attack

$$\begin{cases} \sigma_p \equiv m^{d \bmod \varphi(p)} \pmod{p} \\ \hat{\sigma}_q \not\equiv m^{d \bmod \varphi(q)} \pmod{q} \end{cases} \implies \hat{\sigma} = CRT(\sigma_p, \hat{\sigma}_q) \mod n \ .$$

The attack is based on the observation that

$$\begin{cases} \hat{\sigma}^e \equiv m \pmod{p} \\ \hat{\sigma}^e \not\equiv m \pmod{q} \end{cases} \implies \begin{cases} \hat{\sigma}^e - m \equiv 0 \pmod{p} \\ \hat{\sigma}^e - m \not\equiv 0 \pmod{q} \end{cases} \implies \begin{cases} p | \hat{\sigma}^e - m \\ q \nmid \hat{\sigma}^e - m \end{cases} \implies \gcd(\hat{\sigma}^e - m, n) = p \ .$$

In some cases the attacker knows the message that is signed, i.e. attacks against TLS, where messages have pre–defined format, and the values for the required fields, necessary to reconstruct m, can be obtained by listening over the TLS handshake message exchange.

The attack works for any deterministic padding like FDH or PSS.

Seifert attack

The Bellcore and Boneh–DeMillo–Lipton attacks are fault injection attacks targeted against modular exponentiation. The Seifert attack targets modular reduction ($\bmod n$) instead.

$$\begin{cases} \sigma_p \equiv m^{d \bmod \varphi(p)} \pmod p \\ \sigma_q \equiv m^{d \bmod \varphi(q)} \pmod q \end{cases} \implies \hat{\sigma} = CRT(\sigma_p, \hat{\sigma}_q) \mod n \enspace .$$

The attack is executed using orthogonal latices and goes beyond the scope of this course – the details omitted.

ElGamal cryptosystem

Public parameters are: group $G = \langle g \rangle, |G| = q$. Alice's private key: $x \in \mathbb{Z}_q$, public key (G, q, g, g^x), $g^x \in G$. To encrypt a message m to Alice, Bob selects an ephemeral key uniformly at random $y \in \mathbb{Z}_q$. The ciphertext is $(g^y, m \cdot g^{xy})$, where $g^{xy} \in G$ was obtained as $g^{xy} = (g^x)^y$. To decrypt, Alice uses her private key x to recover $g^{xy} \in G$ as $g^{xy} = (g^y)^x$. Then she finds the inverse $(g^{xy})^-1 = g^{-xy} \in G$, and computes $m \cdot g^{xy} \cdot g^{-xy} = m$.

Assuming that CDH assumption holds, the ElGamal encryption function is a one-way function. Assuming that DDH assumption holds, ElGamal is IND-CPA secure.

What is DDH assumption?

Distinguish DDH tuples (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^z), where $z \in \mathbb{Z}_{|G|}$ selected uniformly at random. This problem is believed to be hard, in certain groups.

How can we falsify DDH assumption?

Consider a challenge between an attacker A and the challenger C. The challenger uniformly at random selects integers $a, b \in \mathbb{Z}_{|G|}$, and a bit $b \in \{0, 1\}$, and if $b == 0$, then it sends (g^a, g^b, g^{ab}) to the adversary, if $b = 0$, then it randomly selects $z \in \mathbb{Z}_{|G|}$, and sends (g^a, g^b, g^z) to the adversary. The goal of the challenger is to output 1 if the adversary thinks (g^a, g^b, g^{ab}) was sent, and 0 otherwise.

Let F be an oracle of the function being studied, and let G be an oracle for an idealized function of that type. The adversary A is a probabilistic algorithm given F and G as input, and which outputs 1 or 0. The goal of A is to distinguish F from G based on making queries to the oracle it's given. Then the advantage of the adversary is

$$Adv(A) = |\Pr[A(F) = 1] - \Pr[a(G) = 1]|$$

The security assumption holds if

$$Adv(A) = |\Pr[A(F) = 1] - \Pr[a(G) = 1]| < \varepsilon \enspace ,$$

where ε is negligibly small value. If we manage to show that an adversay has a significant advantage, compared to random guessing, the cryptographic assumption is falsified, and thus does not hold.

The baseline – random guessing

Consider two random variables

$$X = \begin{cases} 1 & \text{if the challgenger has sent } (g^a, g^b, g^{ab}) \\ 0 & \text{if the challenger has send } (g^a, g^b, g^z) \end{cases} \qquad Y = \begin{cases} 1 & \text{if the adversary thinks he received } (g^a, g^b, g^{ab}) \\ 0 & \text{if the adversary thinks he received } (g^a, g^b, g^z) \end{cases}$$

Variables X and Y are independent, hence the domain of their joint distribution is

$$X \times Y = \{(0,0), (0,1), (1,0), (1,1)\} \ .$$

The adversary wins if her guess coincides with what did the challenger do. Hence

$$\begin{aligned} \Pr[A(G) = 1] &= \Pr[X = 0, Y = 0] + \Pr[X = 1, Y = 1] \\ &= \Pr[X = 0] \cdot \Pr[Y = 0] + \Pr[X = 1] \cdot \Pr[Y = 1] \\ &= \frac{1}{4} + \frac{1}{4} = \frac{1}{2} \ . \end{aligned}$$

By random guessing, in half of the cases the adversary will guess correctly. This is our baseline. To show that in certain groups the DDH assumption does not hold, we need to provide the description of the polynomial time algorithm A that would achieve significantly better results than by random guessing.

DDH is easy in $\mathbb{Z}_p^\times$

In prime order groups $\mathbb{Z}_p^\times$, an attacker can calculate the Legendre symbol of g^a and g^b and thus reveal the Legendre symbol of g^{ab}. It can detect with probability 1 the true negative cases when

$$\left(\frac{g^{ab}}{n}\right) \neq \left(\frac{g^z}{n}\right) \ .$$

For prime p, the Legendre symbol

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} = \begin{cases} 1 & \text{if } a \text{ is a quadratic residue modulo } p \text{ and } a \not\equiv 0 \pmod{p}, \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p, \\ 0 & \text{if } a \equiv 0 \pmod{p}. \end{cases}$$

$a \in \mathbb{Z}_p^\times$ is a quadratic residue modulo p if there exists $x \in \mathbb{Z}_p^\times$ such that $x^2 \equiv a \pmod{p}$.

If g^a is a quadratic residue, it means that a is even, the same with g^b. The relation between redusiocity of g^a, g^b, g^{ab} is given in Table 1. Since we can detect true-negative results based on the expected redusiocity of g^{ab}, to model the interaction between the adversary and the challenger, consider 3 random variables.

$$X = \begin{cases} 0 & \text{if } g^a \text{ is a square residue,} \\ 1 & \text{if } a^a \text{ is a square non-residue} \end{cases} \qquad Y = \begin{cases} 0 & \text{if } g^b \text{ is a square residue,} \\ 1 & \text{if } a^b \text{ is a square non-residue} \end{cases}$$

$$Z = \begin{cases} 0 & \text{if } g^z \text{ is a square residue,} \\ 1 & \text{if } a^z \text{ is a square non-residue} \end{cases}$$

Table 1: Redusiocity of g^{ab}.

g^a	g^b	g^{ab}
residue	residue	residue
non–residue	residue	residue
residue	non–residue	residue
non–residue	non–residue	non–residue

The variables X, Y, Z are independent, hence

$$X \times Y \times Z = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)\} \ .$$

In the events $(0,0,1),(0,1,1),(1,0,1),(1,1,0)$ the adversary can with probability 1 say that $g^c \neq g^{ab}$. The probability of each of these events is 1/8. Apart from that, in the other cases, the best the adversary can do is to randomly select 0 or 1 hoping that her result would match. In approximately half of these cases the adversary is expected to win by random guessing. Hence, the probability that an adversary will guess the outcome correctly is

$$\Pr[A(F)=1] = 4 \cdot \frac{1}{8} + 4 \cdot \frac{1}{16} = \frac{3}{4} \ .$$

The probability that the adversary will guess incorrectly is $\frac{1}{4}$. The advantage of the adversary is then

$$Adv(A) = |\Pr[A(F)=1] - \Pr[a(G)=1]| = |\frac{3}{4} - \frac{1}{2}| = \frac{1}{4} \ ,$$

which is not a negligible value ε. Therefore, the DDH assumption does not hold in groups $\mathbb{Z}_p$, since the adversary gains significant advantage in distinguishing g^{ab} from a random element of G by calculating the Legendre symbols for g^a and g^b.

But the DDH assumption holds in subgroups of k-residues, since every element is a residue and the advantage of using the Legendre symbol is lost.

DDH is easy in elliptic curves over $GF(p)$ with small embedding degree

Indeed, elliptic groups over $GF(p)$ are groups with pairings (bilinear maps $G_A \times G_B \to G_T$), and if the embedding degree is small (i.e., supersingular elliptic curves), then these bilinear maps are efficiently computable. Observe that

$$c \cdot e(P,P) = e(P,cP) = e(aP,bP) = ab \cdot e(P,P) \ .$$

If $e(P,cP) = e(aP,bP)$, then $c = ab$. This way the adversary can always distinguish between tuples (P, aP, bP, cP) and (P, aP, bP, abP). If supersingular elliptic curves with high embedding degree are used, twisted elliptic curves or hyperelliptic curves, the computation of a pairing function is difficult there, and DDH holds in these elliptic groups.