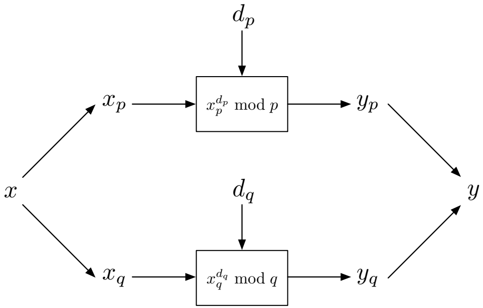
Fault Attacks and Countermeasures



Fault Attacks

- Safe-error attack was a type of fault attack
- Timely induce a fault into ALU during multiply operation at step i
- Check the output
  - If the result is incorrect (invalid signature or error notification), then the error was effective $\Rightarrow d_i = 1$
  - If the result is correct, then the multiplication was dummy (safe error) $\Rightarrow d_i = 0$
- Re-iterate the attack for another value of i
- It was introduced into the RSA when Square-and-Multiply-Always algorithm was used

Square-and-Multiply Algorithm

- The Square-and-multiply algorithm leaks information on private key

Input: m, $d = (d_{k-1}, \ldots, d_0)_2$, n
Output: $s = m^d \pmod{n}$
1. $R_0 \leftarrow 1$
2. For $i = k - 1$ downto 0
$R_0 \leftarrow R_0^2 \pmod{n}$
If $d_i = 1$ then $R_0 \leftarrow R_0 \cdot m \pmod{n}$
3. Return R_0

- It performs exponentiation left to right
- 2 Temporary variables R_0 and m
- Susceptible to SPA-type attacks

Square-and-Multiply-Always Algorithm

- One way to avoid leakage is to square and multiply at every step

Input: $m,\ d = (d_{k-1}, \ldots, d_0)_2,\ n$
Output: $s = m^d \pmod{n}$
1. $R_0 \leftarrow 1$; $R_1 \leftarrow 1$
2. For $i = k-1$ downto 0
$R_0 \leftarrow R_0^2 \pmod{n}$
$b \leftarrow 1 - d_i$; $R_b \leftarrow R_b \cdot m \pmod{n}$
3. Return R_0

- When $b = 1$ (i.e., $d_i = 0$), there is a dummy multiplication
- The power trace is a regular succession of squares and multiplies
- 3 Temporary variables: R_0, R_1 and m
- Not susceptible to SPA-type attacks
- Susceptible to **Safe-Error** attacks

Safe-Error Attack

- Timely induce a fault into ALU during multiply operation at step i
- Check the output
  - If the result is incorrect (invalid signature or error notification), then the error was effective $\Rightarrow d_i = 1$
  - If the result is correct, then the multiplication was dummy (safe error) $\Rightarrow d_i = 0$
- Re-iterate the attack for another value of i

Montgomery Powering Ladder

- Montgomery exponentiation algorithm

Input: $m,\ d = (d_{k-1}, \ldots, d_0)_2,\ n$
Output: $s = m^d \pmod{n}$
1. $R_0 \leftarrow 1\ ;\ \ R_1 \leftarrow m$
2. For $i = k-1$ downto 0
$b \leftarrow 1 - d_i\ ;\ R_b \leftarrow R_0 \cdot R_1 \pmod{n}$
$R_{d_i} \leftarrow R_{d_i}^2 \pmod{n}$
3. Return R_0

- This algorithm behaves regularly without dummy operations
- 2 Temporary variables: R_0 and R_1
- Not susceptible to SPA-type attacks
- Not susceptible to Safe-Error attacks

Square-and-Multiply Algorithm Example

- $e = 9 = (1001)_2$
- Square-and-Multiply Algorithm
- Start with $R_0 = 1$

i	d_i	Step 2a	Step 2b
3	1	$R_0 = R_0^2 = 1$	$R_0 = R_0 m = m$
2	0	$R_0 = R_0^2 = m^2$	
1	0	$R_0 = R_0^2 = m^4$	
0	1	$R_0 = R_0^2 = m^8$	$R_0 = R_0 m = m^9$

- Result: $R_0 = m^9$
- Total of 4 squarings and 2 multiplications

Square-and-Multiply-Always Algorithm Example

- $e = 9 = (1001)_2$
- Square-and-Multiply-Always Algorithm
- Start with $R_0 = 1$ and $R_1 = 1$

i	d_i	b	Step 2a	Step 2b
3	1	0	$R_0 = R_0^2 = 1$	$R_0 = R_0 m = m$
2	0	1	$R_0 = R_0^2 = m^2$	$R_1 = R_1 m = m$
1	0	1	$R_0 = R_0^2 = m^4$	$R_1 = R_1 m = m^2$
0	1	0	$R_0 = R_0^2 = m^8$	$R_0 = R_0 m = m^9$

- Result: $R_0 = m^9$
- Total of 4 squarings and 4 multiplications

Montgomery Powering Ladder Algorithm Example

- $e = 9 = (1001)_2$
- Montgomery Powering Ladder Algorithm
- Start with $R_0 = 1$ and $R_1 = m$

i	d_i	b	Step 2a	Step 2b
3	1	0	$R_0 = R_0 R_1 = m$	$R_1 = R_1^2 = m^2$
2	0	1	$R_1 = R_0 R_1 = m^3$	$R_0 = R_0^2 = m^2$
1	0	1	$R_1 = R_0 R_1 = m^5$	$R_0 = R_0^2 = m^4$
0	1	0	$R_0 = R_0 R_1 = m^9$	$R_1 = R_1^2 = m^{10}$

- Result: $R_0 = m^9$
- Total of 4 squarings and 4 multiplications

General Fault Attack Assumptions

- Precise bit errors
  - The attacker can cause a fault in a single bit
  - Full control over the timing and location of the fault
- Precise byte errors
  - The attacker can cause a fault in a single byte
  - Full control over the timing but only partial control over the location (e.g., which byte is affected)
- Unknown byte errors
  - The attacker can cause a fault in a single byte
  - Partial control over the timing and location of the fault
- Random errors
  - Partial control over the timing and no control over the location

Computing RSA Signature with CRT

- Computation of a signature $y = x^d \pmod{n}$ using CRT
- First we compute x_p, d_p and x_q, d_q using

$$
\begin{aligned}
x_p &= x \pmod{p} \quad \text{and} \quad d_p = d \pmod{p-1} \\
x_q &= x \pmod{q} \quad \text{and} \quad d_q = d \pmod{q-1}
\end{aligned}
$$

- Then we compute y_p and y_q using

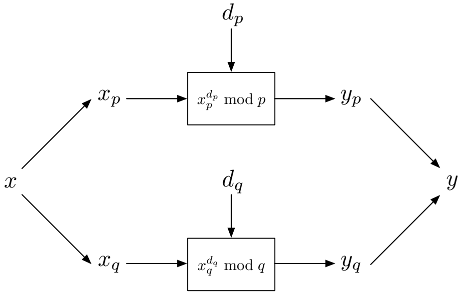
$$
\begin{aligned}
y_p &= x_p^{d_p} \pmod{p} \text{ with } d_p = d \pmod{p-1} \\
y_q &= x_q^{d_q} \pmod{q} \text{ with } d_q = d \pmod{q-1}
\end{aligned}
$$

Computing RSA Signature with CRT

- We then apply the CRT to compute $y = x^d \pmod{n}$

$$\begin{aligned} y &= \mathrm{CRT}(y_p, y_q \; ; \; p, q) \\ &= y_p + p \cdot [i_p \cdot (y_q - y_p) \bmod q] \end{aligned}$$

such that $i_p = p^{-1} \pmod{q}$



Computing RSA Signature with CRT

- We can prove this expression by reducing mod p

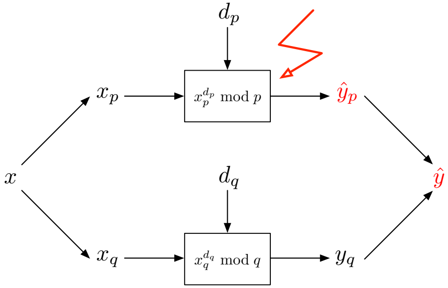
$$\begin{aligned} y &= y_p + p \cdot [i_p \cdot (y_q - y_p) \bmod q] \pmod{p} \\ &= y_p \pmod{p} \end{aligned}$$

- To prove it mod q, we note that $i_p \cdot p = 1 + M_1 \cdot q$ for some M_1

$$\begin{aligned} y &= y_p + (1 + M_1 \cdot q) \cdot (y_q - y_p) + M_2 \cdot q \\ y &= y_p + (1 + M_1 \cdot q) \cdot (y_q - y_p) + M_2 \cdot q \pmod{q} \\ &= y_p + y_q - y_p \pmod{q} \\ &= y_q \pmod{q} \end{aligned}$$

GCD Attack

- Assume that due to the induced fault, y_p is incorrectly computed



- The prime factor q can be obtained using the incorrect $\hat{y}$ as

$$\gcd(\hat{y}^e - x \bmod n,\ n) = q$$

GCD Attack – Proof

- If we had the correct y value, we would have $y^e = x \pmod{n}$
- Therefore, $\gcd(y^e - x \bmod n,\ n) = \gcd(0, n) = n$
- Due to incorrect $\hat{y}^e$ value we have $\hat{y}^e \neq x \pmod{n}$
- However, $\hat{y}$ is incorrect mod p but it is correct mod q

$$
\begin{aligned}
\hat{y}^e &= \hat{y}_p^e \neq y_p^e \neq x_p \pmod{p} \\
\hat{y}^e &= \hat{y}_q^e = y_q^e = x_q \pmod{q}
\end{aligned}
$$

GCD Attack – Proof

- Therefore

$$\begin{array}{rclcll} \hat{y}^e - x_p & = & \hat{y}_p^e - x_p \neq 0 & \pmod{p} & \iff & p \nmid (\hat{y}^e - x) \\ \hat{y}^e - x_q & = & \hat{y}_q^e - x_q = 0 & \pmod{q} & \iff & q \mid (\hat{y}^e - x) \end{array}$$

- This implies $\gcd(\hat{y}^e - x \bmod n,\ n) = q$
- Since $(\hat{y}^e - x)$ and n are both divisible by q, their GCD is q

GCD Attack Demonstration

- Let $p = 17$ and $q = 19$, which gives $n = p \cdot q = 323$ and $\phi(n) = (p-1) \cdot (q-1) = 288$
- Select $e = 23$, since $\gcd(e, \phi(n)) = \gcd(23, 288) = 1$
- Compute $d = e^{-1} \pmod{n}$, which gives $d = 263$
- We select $x = 100$
- We compute $y = x^d \pmod{n}$ as $y = 25$

GCD Attack Demonstration

- In order to apply CRT, we first compute

$$
\begin{array}{lclcl}
x_p = x \bmod p & \rightarrow & x_p = 100 \bmod 17 & \rightarrow & x_p = 15 \\
d_p = d \bmod (p-1) & \rightarrow & d_p = 263 \bmod 16 & \rightarrow & d_p = 7 \\
x_q = x \bmod q & \rightarrow & x_q = 100 \bmod 19 & \rightarrow & x_q = 5 \\
d_q = d \bmod (q-1) & \rightarrow & d_q = 263 \bmod 18 & \rightarrow & d_q = 11 \\
i_p = p^{-1} \bmod q & \rightarrow & i_p = 17^{-1} \bmod 19 & \rightarrow & i_p = 9
\end{array}
$$

GCD Attack Demonstration

- Mod p exponentiation:

$$y_p = x_p^{d_p} \pmod{p} \quad \rightarrow \quad y_p = 15^7 \bmod 17 \quad \rightarrow \quad y_p = 8$$

- Mod q exponentiation:

$$y_q = x_q^{d_q} \pmod{q} \quad \rightarrow \quad y_q = 5^1 1 \bmod 19 \quad \rightarrow \quad y_q = 6$$

- The CRT gives y as

$$\begin{aligned} y &= y_p + p \cdot [i_p \cdot (y_q - y_p) \bmod q] \\ &= 8 + 17 \cdot [9 \cdot (6 - 8) \bmod 19] \\ &= 25 \end{aligned}$$

GCD Attack Demonstration

- Now assume that y_p was incorrectly computed as $\hat{y}_p$
- Instead of $y_p = 8$, we compute $\hat{y}_p = 10$ due to the induced fault
- This incorrect value $\hat{y}_p = 10$ would be used in the CRT computation

$$\begin{aligned} \hat{y} &= \hat{y}_p + p \cdot [i_p \cdot (y_q - \hat{y}_p) \bmod q] \\ &= 10 + 17 \cdot [9 \cdot (6 - 10) \bmod 19] \\ &= 44 \end{aligned}$$

- We would obtain an incorrect value $\hat{y} = 44$

GCD Attack Demonstration

- This result 44 is **incorrect** mod n since $44 \neq 100^{263} \pmod{323}$
- The correct result is $25 = 100^{263} \pmod{323}$
- This result 44 is **incorrect** mod p since $44 = 10 \pmod{17}$ since the correct result is $25 = 8 \pmod{17}$
- However, this result 44 is **correct** mod q since $44 = 6 \pmod{19}$ since the correct result is $25 = 6 \pmod{19}$

GCD Attack Demonstration

- This resulting incorrect value $\hat{y} = 44$ allows us to factor $n = p \cdot q$
- We obtain $q = \gcd(Q, n)$ such that $Q = \hat{y}^e - x \pmod{n}$

$$\begin{aligned} Q &= \hat{y}^e - x \pmod{n} \\ &= 44^{23} - 100 \pmod{323} \\ &= 228 \end{aligned}$$

$$\begin{aligned} q &= \gcd(Q, n) \\ &= \gcd(228, 323) \\ &= 19 \end{aligned}$$

Countermeasures Against GCD Attack

- Recomputation
  - It does not detect permanent errors
  - It doubles the computation time
- Verification
  - It may double the computation time
  - It requires the knowledge of e

Countermeasures Against GCD Attack

- Shamir's method
- Choose a small random r
- Compute $y_{rp} = x^{d \bmod \phi(rp)} \bmod rp$
- Compute $y_{rq} = x^{d \bmod \phi(rq)} \bmod rq$
- If $y_{rp} \neq y_{rq} \pmod{r}$, output ERROR and stop
- Output $y = \mathrm{CRT}(y_{rp} \bmod p,\ y_{rq} \bmod q)$

- Shamir's method requires the knowledge of d
- However, in CRT, only d_p and d_q are available

RSA Error Detection – The Standard Mode

- Compute $z = x^d \pmod{rn}$
- Compute $y_r = x^{d \bmod \phi(r)} \pmod{r}$
  (Note that r can be chosen prime, this $\phi(r) = r - 1$)
- If $y_r \neq z \pmod{r}$, output ERROR and stop
- Output $y = z \pmod{n}$

RSA Error Detection – The CRT Mode

- Compute $z_1 = x^{d_p} \pmod{r_1 p}$
- Compute $y_{r_1} = x^{d_p \bmod \phi(r_1)} \pmod{r_1}$
- If $y_{r_1} \neq z_1 \pmod{r_1}$, output ERROR and stop
- Compute $z_2 = x^{d_q} \pmod{r_2 q}$
- Compute $y_{r_2} = x^{d_q \bmod \phi(r_2)} \pmod{r_2}$
- If $y_{r_2} \neq z_2 \pmod{r_2}$, output ERROR and stop
- Output $y = \mathrm{CRT}(z_1 \bmod p, z_2 \bmod q)$