Cryptanalysis of Two Protocols for RSA with CRT Based on Fault Infection

Sung-Ming Yen[1] and Dongryeol Kim[2]

[1] Dept of Computer Science and Information Engineering
National Central University, Taiwan, ROC
http://www.csie.ncu.edu.tw/~yensm/lcis.html

[2] Information Security Policy Division
Korea Information Security Agency, Korea

Outline :

1. Preliminary Background of CRT-based Cryptanalysis
2. Review: Two CRT-based RSA Computation Based on Fault Infection
3. Cryptanalysis of CRT-based RSA with Fault Infection
4. Conclusions

1. Introduction and Preliminary Background

RSA speedup with CRT

CRT-based fault attack

RSA Speedup with CRT

RSA speedup based on CRT:

- Given p, q, (n=p*q), d, and m,
  $S=m^d$ mod n can be sped up by

  $s_p=(m \bmod p)^{d \bmod (p-1)} \bmod p$

  $s_q=(m \bmod q)^{d \bmod (q-1)} \bmod q$

- Gauss's CRT recombination $S=CRT(s_p, s_q)$
  $[(s_p \times \underline{q \times (q^{-1} \bmod p)} + s_q \times \underline{p \times (p^{-1} \bmod q)}] \bmod n$
  $= [s_p \times X_p + s_q \times X_q] \bmod n$
- Garner's CRT recombination $S=CRT(s_p, s_q)$
  $s_q + [(s_p - s_q) \times \underline{(q^{-1} \bmod p)} \bmod p] \times q$

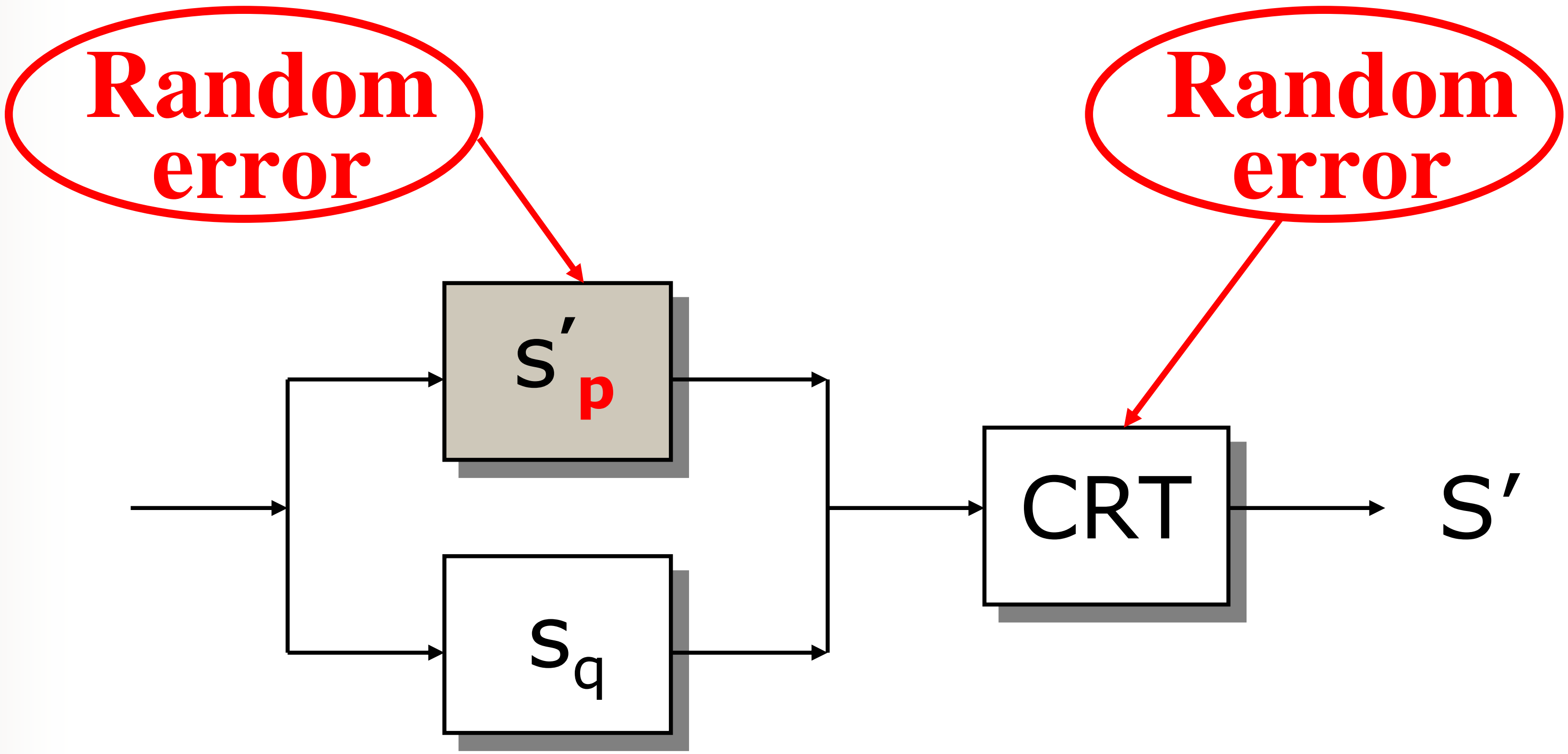
LCIS

CRT-based Fault Attack

Fault attack on the computation of s_p & s_q

- Given a faulty result of $S' = CRT(s'_p, s_q)$

  $q = gcd((S'^e - m) \bmod n, n)$



Shamir's Countermeasure

- Shamir's countermeasure (extend modulus then reduce modulus)

  $s_{pr}=m_{pr}^{d_{pr}} \bmod pr$

  $s_{qr}=m_{qr}^{d_{qr}} \bmod qr$

  where $m_{pr}= m \bmod pr$ & $d_{pr}= d \bmod \phi(pr)$

  and r is a random prime.

  - Output S only if $(s_{pr} \bmod r) = (s_{qr} \bmod r)$

  $S=CRT(s_p, s_q)$

  $=CRT(s_{pr} \bmod p, s_{qr} \bmod q)$

Other possible countermeasures:

(All need and strictly depend on the reliability of a comparison operation!)

- Compute S twice and compare the results
- Given $S = m^d \bmod n$, verify whether

$$m \;?\!= S^e \bmod n$$

Attack on Shamir's Method

- Possible attacks on the Zero flag!
  - Implementation of checking

    $(s_{pr} \bmod r)$ **=?** $(s_{qr} \bmod r)$
  - Implementation of "a ?= b"

    SUB a,b (or CMP a,b)

    JZ (jump if zero)

  It highly depends on the zero flag!

- Another reported CRT-based attack
  - The main **weakness:** It's assumed that correctness of s_{pr} and s_{qr} implies the correctness of both s_p and s_q

    where $s_p = s_{pr} \bmod p$

    possibly s'_p <-- $s_{pr} \bmod p$
  - The checking of whether

    $(s_{pr} \bmod r)$ **=?** $(s_{qr} \bmod r)$

    cannot detect the error in s'_p

Importance of CRT-based Attack

It has already been widely employed

But a single fault → total break down

- False alarm attack on RSA+CRT
  - may be initiated by any malicious attacker

  → **Denial of service** attack
- So, any potential CRT-based attack should be carefully considered

2. Review: Two CRT-based RSA Computation Based on Fault Infection

No fault-free decision procedure will be assumed in the countermeasure!

Fault Infective CRT Speedup

- No checking procedure will be assumed that should be fault free
- When a "random" error occurred in s_p (or s_q) it will influence computation of s_q (or s_p) or the overall computation of S (for example CRT(s'_p, s_q) or CRT(s_p, s'_q) is not accessible)

The CRT-1 Protocol

Parameter selection:

- $n = p \times q$ (usual key pair e & $d = e^{-1} \bmod \phi(n)$)
- additional key pair e_r & $d_r = e_r^{-1} \bmod \phi(n)$
  $d_r = d - r$ (r is a small integer)

The protocol:

- Compute $k_p = \lfloor m/p \rfloor$ & $\mathbf{k_q = \lfloor m/q \rfloor}$
  where $\lfloor x \rfloor$ means floor function
- Compute m^{d_r} mod n with CRT speedup

  $s_p = A^{d_r \bmod (p-1)} \bmod p$ where $A = m \bmod p$

  $s_q = \hat{A}^{d_r \bmod (p-1)} \bmod q$

  where $\hat{A} = ((s_p^{e_r} \bmod p) + k_p \times p) \bmod q$
- Based on CRT

  $S = CRT(s_p, s_q) \times (\tilde{A}^r) \bmod n$

  where $\tilde{A} = (s_q^{e_r} \bmod q) + k_q \times q$

If the computation is fault free:

- Message reconstruction 1:

$$s_q = \hat{A}^{d_r \bmod (p-1)} \bmod q$$

where $\hat{A} = ((s_p{}^{e_r} \bmod p) + k_p \times p) \bmod q$

$= \mathbf{m} \bmod q$

- Message reconstruction 2:

$$S = CRT(s_p, s_q) \times (\tilde{A}^r) \bmod n$$

where $\tilde{A} = (s_q{}^{e_r} \bmod q) + k_q \times q$

$= \mathbf{m}$

The CRT-2 Protocol

Parameter selection:

- $n=p\times q$ (usual key pair e & $d=e^{-1} \bmod \phi(n)$)
- additional key pair e_r & $d_r=e_r^{-1} \bmod \phi(n)$
  $d_r=d-r$ (r is a small integer)

The protocol:

- Compute $k_p = \lfloor m/p \rfloor$ & $k_q = \lfloor m/q \rfloor$
- Compute m^{d_r} mod n with CRT speedup

  $s_p = A^{d_r \bmod (p-1)} \bmod p$ where $A = m \bmod p$

  $s_q = A^{d_r \bmod (p-1)} \bmod q$
- Based on CRT

  $S = CRT(s_p, s_q) \times (\hat{A}^r) \bmod n$

  where $\hat{A} = \lfloor (m_1 + m_2)/2 \rfloor$

  $m_1 = (s_p^{e_r} \bmod p) + k_p \times p$

  $m_2 = (s_q^{e_r} \bmod q) + k_q \times q$

If the computation is fault free:

- Message reconstruction:

$$S=CRT(s_p, s_q)\times(\hat{A}^r) \bmod n$$

where $\hat{A} = \lfloor (m_1+m_2)/2 \rfloor$

$$m_1=(s_p^{e_r} \bmod p)+k_p\times p$$

$$=\mathbf{m}$$

$$m_2=(s_q^{e_r} \bmod q)+k_q\times q$$

$$=\mathbf{m}$$

3. Cryptanalysis of CRT-based RSA with Fault Infection

Exploiting faults that usual CRT-based attack did not consider

Attack Exploiting Fault on Temporary Parameters

- Attacks exploit faults that usual CRT-based attack did not consider
  - Exploiting faults on temporary parameters that usual CRT speedup does NOT required
  - It has been overlooked previously

Attack on CRT-1 Protocol

- In the CRT-1 protocol:
  Suppose
  - k_p, s_p, and s_q are correct
  - but **k_q** becomes incorrect (when computed or accessed) k_q --> k_q'

  We got

  $S' = m^d + R*q \bmod n$ (R: random integer)

  - leads to **$q = \gcd((S'^e - m), n)$**

- It can be proven that fault on **k_p** disables the above attack

Attack on CRT-2 Protocol

- In the CRT-2 protocol:
  Suppose
  - k_p, s_p, and s_q are correct
  - but $\mathbf{k_q}$ becomes incorrect (when computed or accessed) k_q --> k_q'

  We got

  $S' = m^d + R*q \bmod n$ (R: random integer)

  - leads to <u>**$q = \gcd((S'^e - m), n)$**</u>

- Fault on $\mathbf{k_p}$ leads to <u>**$p = \gcd((S'^e - m), n)$**</u>

4. Conclusions

- **Basic consideration**:
  - Do not make unreasonable assumption, e.g., all the checking operations are error free
- **Important thing to remind again**:
  - Be careful about all CRT-based attack
    - ✓Explicit fault/attack
    - ✓Implicit fault/attack
  - The false alarm attack may lead to the "DoS" attack
- **One technical issue to notice**:
  - More "checking" operations being used will lead to a less reliable countermeasure
- **Open problem**:
  - Is error free checking operation necessary?
  - More research is still necessary