Bellcore Fault Attacks and Defenses of CRT-RSA Signature Scheme

Xuan Tran and Naveen Kandakumar
Cryptography II
5/9/11
Outline

• Background
  – RSA as encryption scheme and signature scheme
  – CRT RSA speed up factor for signature scheme
  – Security requirements for signature schemes
• Different Attacks on RSA Signatures
  – Side-channel attacks
• CRT-RSA secured against fault attacks (past)
• New CRT-RSA resistant to powerful fault attacks
• Questions
RSA Encryption Scheme

- Bob’s encryption
  - Let \( n = p \times q \), where \( p,q \) are primes
  - \( a \times b \equiv 1 \pmod{(p-1)(q-1)} \)
  - public : \( n,a \)

- Alice who wants to send a message, \( m \) to Bob
  - \( s \equiv m^a \pmod{n} \)
  - Send \( s \) to Bob

- Bob computes \( m \)
  - \( m \equiv s^b \pmod{n} \)
RSA Signature Scheme

- Let $n = p \times q$, where $p,q$ are primes
- $a \times b \equiv 1 \pmod{(p-1)(q-1)}$
- Public key: $n,a$
- Private key: $p,q,b$
- $\text{Sig}_K(x), y = x^b \mod n$
- $\text{ver}_K(x,y) = \text{true},\ when \ x \equiv y^a \pmod{n}$
Proposed Bit Length by NIST

<table>
<thead>
<tr>
<th>Bits of security</th>
<th>Symmetric key algorithm</th>
<th>Comparable RSA key length</th>
<th>Comparable hash function</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>2TDEA*</td>
<td>1024</td>
<td>SHA-1</td>
</tr>
<tr>
<td>112</td>
<td>3TDEA</td>
<td>2048</td>
<td>SHA-224</td>
</tr>
<tr>
<td>128</td>
<td>AES-128</td>
<td>3072</td>
<td>SHA-256</td>
</tr>
<tr>
<td>192</td>
<td>AES-192</td>
<td>7680</td>
<td>SHA-384</td>
</tr>
<tr>
<td>256</td>
<td>AES-256</td>
<td>15360</td>
<td>SHA-512</td>
</tr>
</tbody>
</table>

- 2TDEA is 2-key triple DES

The following table is taken from NIST's Recommendation for Key Management [NIST-80057]. It shows the recommended comparable key sizes for symmetrical block ciphers (AES and Triple DES) and the RSA algorithm. That is, the key length you would need to use to have comparable security.


CRT - RSA

- $n = p \times q$
- $S_p = m^d \mod p$
- $S_q = m^d \mod q$
- Find $S$ (signed $m$) using either Gauss's or Garner's algorithm
  - **Gauss:** $S = (S_p \times q \times (q^{-1} \mod p) + S_q \times p \times (p^{-1} \mod q)) \mod n$
  - **Garner:** $S = S_q + q \times ((S_p - S_q) \times (q^{-1} \mod p) \mod p)$
Why CRT-RSA

• To speed up the processing time
• If $n \approx 2^{2L}$, $p,q \approx 2^L$, where $L$ is the bit length
• Using naïve way $(2CL)^3$
• Using CRT – $2(CL)^3$
  – Where $C$ is a constant
• Hence 4 times faster
Side-Channel Attacks

- Attack based on information gained from the physical implementation of a cryptosystem

- Side channel attacks make use of some or all of the information and try to recover the key
Timing attack

• A timing attack watches data movement into and out of the CPU, or memory, on the hardware running the cryptosystem or algorithm.

• Timing attacks can be applied to any algorithm that has data-dependent timing variation.
Timing attack (cont.)

- The number of samples needed to gain enough information are determined by the properties of the signal and the noise.

Bellcore Attack

- A random error occurs when computing $S_p$. This yields a faulty signature $S_p'$.
- $S_q$ is computed correctly.
- $n$ can be factorized
- $\gcd(((S')^e - m) \mod n, n) = q$
  (or)
- $\gcd(S' - S, n) = q$
Fault-based attack

• Occurrence of hardware faults
  – Transient hardware failures is a natural phenomenon

• This can be induced externally (or during manufacturing process)
Fault-based attack (cont.)

*A Pellegrini, V Bertacco and T Austin, University of Michigan, 2010
Fault-based attack (cont.)

• Implementation of RSA signature algorithm on OpenSSL, a widely used package for SSL encryption and authentication
• Last year, this attack was done on SPARC-based microprocessor running Linux and extracted 1024 bit private key in approximately 100 hours
Fault-based attack (cont.)
Countermeasures (past)

• Earlier countermeasures were based on inserting one or more conditional checks in the algorithm to verify the integrity of the values (A. Shamir, 1999, Method and apparatus for protecting public key schemes from timing and fault attacks)

• Introduced the concept of infective computation (Yen et al., 2003, RSA speedup with Chinese Remainder Theorem immune against hardware fault cryptanalysis)
Fault induced attacks

• Powerful attack
  – Able to inject bit-set-reset faults in the same execution step or across different steps
  – Able to control the algorithm clocking
  – Able to target specific small locations
  – Able to skip one or more intermediate instructions
  – Able to alter the result of conditional checks

• Ways to induce faults in a device while performing a cryptographic application
  – Varying temperature, the external clock, the supply voltage, or more accurately targeted faults my be injected with laser, X-ray or ion beams.
Overview of New CRT-RSA

• Performance and security
  – Allows easy randomization for each signature. No requirements on random values used to blind the moduli, the exponents or the message
  – Blinding computation does not require precomputation.
  – CRT recombination does not require computation of $q^{-1}$, i.e. the inverse of one of the blinded moduli.
  – Validation of the blinded signature is implicitly performed modulo $n$ after CRT recombination rather prior to it.

• If validation fails, the entire blinded signature is infected and the unblinding yields a random value modulo both $p$ and $q$.
• Infeasible for powerful attacker to precisely inject one or more faults aiming at modifying one signature without affecting the other one and mounting a gcd attack.
Countermeasures

• Quisquater and Couvreur suggested using CRT for more efficient computation, 1982

\[ s_p = m^{d_p} \mod p , \]
\[ s_q = m^{d_q} \mod q , \]
\[ s = \text{CRT}(s_p, s_q) , \]
\[ = s_q + q \cdot (i_q \cdot (s_p - s_q) \mod p) \]

• gcd attack a.k.a. Bellcore attack

\[ s_p^e \not\equiv m \pmod{p} \ , \]
\[ s_q^e \equiv m \pmod{q} \ , \]
\[ \gcd(s_q^e - m, n) = q \ , \]
Proposed Algorithm

Algorithm 1. Blinded CRT-RSA Algorithm

Input  
Private parameters: \( p, q, d_p, d_q, i_q = q^{-1} \bmod p \),
public parameters: \( n, e \),
message (padded and hashed): \( m \).

Output  Signature \( s \).
Proposed Algorithm (cont.)

1. Select random values $\rho_p, \rho_q, r_1, r_2, t$ and $\alpha$ at the beginning or right when needed.
2. $d'_p \leftarrow d_p + \rho_p(p-1)$ // mask the private exponents
3. $d'_q \leftarrow d_q + \rho_q(q-1)$
4. $p^* \leftarrow r_1 p$ // blind the moduli
5. $q^* \leftarrow r_2 q$
6. $b_p^* \leftarrow t^{\alpha e} \mod p^*$ // message blinding multipliers
7. $b_q^* \leftarrow t^{\alpha e} \mod q^*$
8. $m_{p^*} \leftarrow m \mod p^*$ // message components
9. $m_{q^*} \leftarrow m \mod q^*$
10. $\tilde{s}_{p^*} \leftarrow ((m_{p^*} \cdot b_{p^*})^{(d'_p-1)} \cdot m_{p^*}) \mod p^*$
    \quad // \equiv m_{p^*} \cdot t^{\alpha(1-e)} \mod p$
11. $\tilde{s}_{q^*} \leftarrow ((m_{q^*} \cdot b_{q^*})^{(d'_q-1)} \cdot m_{q^*}) \mod q^*$
    \quad // \equiv m_{q^*} \cdot t^{\alpha(1-e)} \mod q$
12. $\tilde{s} \leftarrow (\tilde{s}_{q^*} + q \cdot (i_q (\tilde{s}_{p^*} - \tilde{s}_{q^*}) \mod p^*)) \mod n$
    \quad // \equiv m^d \cdot t^{\alpha(1-e)} \mod n$
13. $b \leftarrow (b_{q^*} + q \cdot (i_q (b_{p^*} - b_{q^*}) \mod p^*)) \mod n$
    \quad // \equiv t^{\alpha e} \mod n$
14. $s \leftarrow \tilde{s} \cdot t^{\text{trunc}((m \cdot b + \alpha(e-1) - (\tilde{s} \cdot b)^e) \mod n)} \mod n$
    \quad // \equiv \tilde{s} \cdot t^{\alpha(e-1)} \equiv m^d \mod n$
15. Return($s$)
• Combined signature in a Garner-like form

\[ s = \{ s_{q^*} + q \left[ (q^{-1} \mod p) (s_{p^*} - s_{q^*}) \mod p^* \right] \} \mod n \]

• Final reduction mod n is performed, s is a correct modulo both p and q, computing the above is equivalent to \( m^d \mod n \)

\[
\begin{align*}
    s &\equiv (s_{q^*} \mod p) \\
    &\quad + (q \mod p)(q^{-1} \mod p) [s_p - (s_{q^*} \mod p)] \pmod{p} \\
    &\equiv (s_{q^*} \mod p) + [s_p - (s_{q^*} \mod p)] \pmod{p} \\
    &\equiv s_p \pmod{p}
\end{align*}
\]
We have,

- \( f(m, e, s, t, \alpha) = \text{trunc}((mb + \alpha(e-1)-(sb)^e) \mod n) \), where \( b \equiv t^{\alpha e} \mod n \)
- Function that truncates its input to a bit length no less than that of \( \alpha(e-1) \). If no fault occurs, then the following holds,

\[
\begin{align*}
(\tilde{s} b)^e & \equiv (m^d t^{\alpha(1-e)} t^{\alpha e})^e \pmod{n} \\
& \equiv (m^d t^{\alpha})^e \pmod{n} \\
& \equiv mb \pmod{n}
\end{align*}
\]

- Wagner’s framework
  - Suggest modeling the state of device \( s_i \), as the content of the registers and memory, the set of possible states is \( S \)

\[
\langle x, k \rangle = s_0 \leadsto s_1 \leadsto \cdots \leadsto s_n = \langle y \rangle
\]

- \( x \), represents the collection of inputs, \( k \) secret values and \( y \) are outputs
- Fault attack is specified by the tuple \((x, \rightarrow_1, \ldots, \rightarrow_n)\). Faulty computation is represented as:

\[
\langle x, k \rangle = s_0 \leadsto s_1 \rightarrow_1 s_1 \leadsto s_2 \rightarrow_2 s_2 \cdots \leadsto s_n \rightarrow_n s_n = \langle y \rangle
\]
Attacks on Algorithm

• If the attacker injects a fault in making $d_p' \neq d_p \pmod{p-1}$, then the scheme is not affected

\[
\begin{align*}
    d'_p &\rightarrow_1 d'_p \neq d_p \\
    \leadsto \quad \tilde{s}_p^* &\neq \tilde{s}_p^* \\
    \leadsto \quad \tilde{s} &\equiv m^d t^{\alpha(1-e)} \pmod{n} \\
    \leadsto \quad (\tilde{s}b)^e &\equiv mb \pmod{n} \\
    \leadsto \quad s &\equiv \tilde{s}t^{\text{trunc}(mb-mb)+\alpha(e-1)} \pmod{n} \\
    &\equiv m^d t^\beta \pmod{n} \\
    &\equiv m^{dq} t^\beta \pmod{q} \\
    &\neq s_q^* \pmod{q}
\end{align*}
\]
Attacks on Algorithm

- If the attacker injects a fault in $m_p^*$

\[
\begin{align*}
    m_p^* & \rightarrow 1 \quad m_p^* \not\equiv m_p \pmod{p} \\
    \implies \quad \tilde{s}_p^* & \not\equiv \tilde{s}_p^* \pmod{p} \\
    \implies \quad \tilde{s} & \equiv m^d t^{\alpha(1-e)} \pmod{n} \\
    \implies \quad (\tilde{s}b)^e & \equiv mb \pmod{n} \\
    \implies \quad s & \equiv m^d t^\beta \pmod{n} \\
    & \equiv m^{d_q} t^\beta \pmod{q} \\
    & \not\equiv s_q^* \pmod{q}
\end{align*}
\]
Performance Evaluation

- **Recommended number of bits**

<table>
<thead>
<tr>
<th>Description</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1, r_2$</td>
<td>32</td>
</tr>
<tr>
<td>$\rho_p, \rho_q$</td>
<td>32</td>
</tr>
<tr>
<td>$\alpha$ (if $e = 2^{16}+1$)</td>
<td>15</td>
</tr>
<tr>
<td>$\alpha e_t$ is a random value</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Proposed algorithm compares to original CRT-RSA**
  - 1024 bits: overhead of 12.5%
  - 2048 bits: overhead of 6.25%

$^1$mask of private exponents
References


“Mr. Osborne, may I be excused? My brain is full.”