Introduction to fault attacks

BCRYPT course on embedded security and application
May 19, 2009

Chong Hee Kim
Jean-Jacques Quisquater
UCL Crypto Group
Contents

• What’s fault attacks?
• Requirements for fault attacks
• How to generate faults
  – Methods, types, models
• How to use faults to break cryptographic systems
  – RSA-CRT, AES
• How to make your code secure against fault attacks
What’s fault attacks?
What’s fault attack?

• Cryptography

The practice and study of hiding information

\[ P_{i_1} \oplus P_{i_2} \oplus \cdots \oplus C_{j_1} \oplus C_{j_2} \oplus \cdots = K_{k_1} \oplus K_{k_2} \oplus \cdots \]

\[ \Delta_Y = S(X) \oplus S(X \oplus \Delta_X) \]

Make

Break

Cryptanalysis
What’s fault attack?

- Cryptanalysis
What’s fault attack?

• Cryptanalysis

In practice, algorithms have to be implemented on real physical devices.
Requirements for fault attacks
Requirements for fault attacks

• Generate faults
Requirements for fault attacks

• Generate *right* faults
  – At the right time and on the right place
  – Not kill the chip but make it to work abnormally

• The generated fault must be exploitable
  – The attacker has to be able to extract some information about secret from the erroneous result/behavior
  – Depends on algorithms
How to generate faults

Methods, types, models
Methods of injection

• Glitch
• Temperature variation
• Light (Laser)
• Magnetic pulse
• …
Methods of injection

• Glitch
  – Variation in supply voltage or in the external clock
  – Cause instruction misinterpreting, omitting some instructions, data misread
  – The attacker should control the amplitude and duration of the glitch
  – The most common method used
    • Easy to apply since you do not need to consider a localization issue.
    • One main drawback since you cannot focus on specific part of the device.
  – Glitch detectors, DC filters to resist
Methods of injection

• Temperature variation
  – At the extreme temperature the device does not work properly.
  – Random modification of RAM cells or read and write threshold mismatch in NVMs may occur.
  – Temperature detectors
    • Most smart cards have temperature detectors but a mismatch may happen between the operating temperature of memory cells and detecting range of the detector.
Methods of injection

• Light (Laser)
  – Nowadays the most powerful attack.
  – Possible to choose the location of the attack.
  – All electric circuits are sensitive to light due to the photoelectric effects, the current caused by photons can be used to induce faults.
  – To succeed, the attacker should control the energy, wavelength, location, and emission time of the light.
Methods of injection

• Light (Laser)
  – Simple camera flash
    • Cheap but the penetration depth depends on the light wavelength and camera flash provides only visible wavelengths (white light)
    • difficult to control accurately
  – Laser
    • allows using several discrete wavelengths and targeting a very small area of the device.
    • Although most smart cards have light detectors and metal shields, the protection against laser attack that can attack very small areas is not easy.
    • nowadays a laser attack on the backside of the chip
Methods of injection

• Magnetic pulse
  – The magnetic field creates local currents on the surface of the component, which can generate a fault.
  – This attack can be performed with cheap materials, a needle wounded by a wire, and allows attacking small parts of the chip.
  – However except for the low cost, laser system seems to be more practical to use.
Types of faults

• Permanent faults vs. Transient faults
Types of faults

• Permanent faults
  – The value of a cell is definitely changed.
  – Either data (EEPROM or RAM) or code (EEPROM) can be damaged.
  – Very powerful when a secret key is changed.
  – Very difficult to make permanent faults on specific logical cells due to a memory ciphering and scrambled physical address of memory in modern smart card
Types of faults

• Transient faults
  – Provisional faults.
  – The circuit recovers its original behavior after reset or when the fault’s stimulus ceases.
  – A code execution or a computation can be disturbed.
    • A call to subroutine can be skipped, a test can be avoided, different executions can be executed, a wrong value can be fetched, or program counter can be modified.
Models of fault attacks

- Bit errors vs. byte errors
- Specific value error vs. random value error
- Static error vs. computational error
- Data error vs. control error
Models of fault attacks

• Bit errors vs. byte errors
  – The attacker can assume that he can change a value of one bit or a byte.
  – Usually byte error model is more practical since a byte is a basic level of storing data and a bus.
  – Easier to change a value of a byte than that of a bit.
  – Using a bit error model, all cryptosystem can be broken.
  – Inducing a bit error is very difficult with the current silicon technology.
Models of fault attacks

• Specific value error vs. random value error
  – The attacker can assume that he can change the value of a data into a specific or random value.
  – Mostly all zeros or ones are used for the specific values.
  – In general random value error is more easy to induce.
Models of fault attacks

• **Static error vs. computational error**
  - The attacker can assume that he can induce an error on a memory itself or during the computation of some operations.
    • Ex 1) the attack on RSA-CRT needs a faulty computation on one of the exponentiations.
    • Ex 2) DFA (Differential Fault Attack) on DSA assumes a flipping of a bit of the secret key stored in memory.
  - Usually computational error is easy to put into practice and change of a value stored in memory is not easy.
Models of fault attacks

• Data error vs. control error
  – Some iterations or operations can be skipped, jumped to another, etc by faults: control errors
  – Errors on the secret key, intermediate values, computations: data errors
  – Although the induction of a control error is more difficult, it is sometimes very powerful.
How to use faults to break cryptographic systems
History of fault attacks

- 1996/09, Boneh et al.  
  - RSA-CRT
- 1996/10, Biham and Shamir  
  - DES
- 1997, Bao et al.  
  - RSA, ElGamal
- 1997, Joye and Quisquater  
  - LUC, Demytko
- 2005, Blomer et al.  
  - ECC
- 2005, Naccache et al.  
  - DSA
History of fault attacks

• DFA on AES
  – 2003, Blomer and Seifert
  – 2003, Dusart et al.
  – 2003, Piret and Quisquater

• DFA on AES Key Schedule
  – 2004, Giraud
  – 2004, Chen and Yen
  – 2006, Peacham and Thomas
  – 2007, Takahashi et al.
  – 2008, Kim and Quisquater
## Algorithms and fault attacks

<table>
<thead>
<tr>
<th>Model</th>
<th>Location</th>
<th># of faulty results</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>Byte</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Anywhere among last 6 rounds</td>
<td></td>
</tr>
<tr>
<td>AES: State</td>
<td>Byte</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Anywhere between MixCol of 7th and 8th</td>
<td></td>
</tr>
<tr>
<td>AES: Key Schedule</td>
<td>Byte</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>9th round key scheduling</td>
<td></td>
</tr>
<tr>
<td>RSA-CRT</td>
<td>Size of modulus</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Anywhere during one of the exponentiations</td>
<td></td>
</tr>
<tr>
<td>RSA-SFM</td>
<td>Bit</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>Anywhere among 128 bytes</td>
<td></td>
</tr>
<tr>
<td>DSA</td>
<td>Bit</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>Anywhere among 20 bytes</td>
<td></td>
</tr>
<tr>
<td>ECDSA</td>
<td>Bit</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>Anywhere among 20 bytes</td>
<td></td>
</tr>
</tbody>
</table>
Fault attacks on RSA-CRT

- The best attack until now
  - 1996/09, Boneh et al.
  - 1 fault
RSA-CRT

• RSA-CRT (Chinese Remainder Theorem)
  – $N=p\cdot q$: RSA modulus, $p$ and $q$: large primes
  – $e \cdot d = 1 \mod (p-1)(q-1)$
  – $d_p = d \mod (p-1)$ and $d_q = d \mod (q-1)$
  – $l_q$: inverse of $q$ modulo $p$

  – Signature $S$ of a message $m$
    1. $S_p = m^{d_p} \mod p$
    2. $S_q = m^{d_q} \mod q$

  – $S = \text{CRT}(S_p, S_q) = S_p + q \cdot \{ (S_p - S_q) \cdot l_q \} \mod N$
Fault attack on RSA-CRT

- Signature $S$ of message $m$
  1. $S_p = m^{dp} \mod p$
  2. $S_q = m^{dq} \mod q$
  1. $S = \text{CRT}(S_p, S_q) = S_q + q \cdot \{(S_p - S_q) \cdot l_q \mod p\}$
Fault attack on RSA-CRT

– Signature $S$ of message $m$

1. $S_p = m^{dp} \mod p$ ← Fault attack

   $S_q = m^{dq} \mod q$

1. $S = \text{CRT}(S_p, S_q) = S_q + q \cdot \{(S_p - S_q) \cdot l_q \mod p\}$
Fault attack on RSA-CRT

- Signature $S$ of message $m$
  1. $S_p = m^{dp} \mod p$
  
  
  $S_q = m^{dq} \mod q$
  1. $S = \operatorname{CRT}(S_p, S_q) = S_q + q \cdot \{(S_p - S_q) \cdot I_q \mod p\}$
Fault attack on RSA-CRT

– Signature $S$ of message $m$
  1. $S_p = m^{dp} \mod p$
     \[ S_q = m^{dq} \mod q \quad \text{← Fault attack} \]
  1. $S = \text{CRT}(S_p, S_q) = S_q + q \cdot \{(S_p - S_q) \cdot l_q \mod p\}$
Fault attack on RSA-CRT

- Signature $S$ of message $m$
  1. $S_p = m^{dp} \mod p$
  2. $S_q = m^{dq} \mod q$
  1. $S = \text{CRT}(S_p, S_q) = S_q + q \cdot ((S_p - S_q) \cdot I_q \mod p)$
Fault attack on RSA-CRT

- Signature $S$ of message $m$
  1. $S_p = m^{dp} \mod p \leftarrow$ Fault attack
  2. $S_q = m^{dq} \mod q$
  3. $S = \text{CRT}(S_p, S_q) = S_q + q \cdot \{(S_p-S_q) \cdot l_q \mod p\}$
Fault attack on RSA-CRT

- Signature $S$ of message $m$
  1. $S_p = m^{dp} \mod p$ ← Fault attack
  2. $S_q = m^{dq} \mod q$
  1. $S = \text{CRT}(S_p, S_q) = S_q + q \cdot \{(S_p - S_q) \cdot l_q \mod p\}$
Fault attack on RSA-CRT

– Signature $S$ of message $m$
  1. $S_p = m^{dp} \mod p$ ← Fault attack

\[
S_q = m^{dq} \mod q 
\]

1. $S = \text{CRT}(S_p, S_q) = S_q + q \cdot \{(S_p - S_q) \cdot l_q \mod p\}$

• Find primes $p$ and $q$
  – By computing $\text{GCD}(S - S, N)$ or $\text{GCD}(S^e - m, N)$
    • Proof
      – $(S - S) \neq 0 \mod p$ and $(S - S) = 0 \mod q$
      – $(S - S) = k \cdot q$
Countermeasures

• Simple methods
  1. Compute signature twice and compare them
     • Drawback
       – Double computation time
       – Cannot avoid permanent errors

1. Verify signature with public exponent $e$
   • Device returns $S$ only when $S^e \equiv m \mod N$
   • Drawback
     – Too costly if $e$ is large
     – In some applications (e.g. Javacard), access of $e$ is not allowed during signature generation
Countermeasures

• Shamir’s (1997)
  – Use redundancy in computing $S_p$ and $S_q$
  – Check correctness before RSA combination
  – Let $r$ be a random $k$-bit integer (typically, $k=32$)

1. $S_p^* = m^d \mod (p \cdot r)$
   $S_q^* = m^d \mod (q \cdot r)$

2. $S = \text{CRT}(S_p^*, S_q^*) \mod N$ if $S_p^* \equiv S_q^* \mod r$,  
   $\text{error}$ otherwise.
Countermeasures

• Shamir’s generalizations (2001)
  – Joye et al. pointed out Shamir’s required $d$ which is not known in CRT. Only $d_p$ and $d_q$ are known in CRT.
  – Let $r_1, r_2$ be random $k$-bit integers.

1. \[
S_p^* = m^{d_p} \mod (p \cdot r_1), \quad s_1 = m^{d_p} \mod \varphi(r_1) \mod r_1 \\
S_q^* = m^{d_q} \mod (q \cdot r_2), \quad s_2 = m^{d_q} \mod \varphi(r_2) \mod r_2
\]

2. \[
\begin{cases} 
S = \text{CRT}(S_p^*, S_q^*) \mod N \text{ if } S_p^* \equiv s_1 \mod r_1 \text{ and } S_q^* \equiv s_2 \mod r_2 \\
\text{error otherwise.}
\end{cases}
\]
Countermeasures

• Aumüller et al.’s (2002)
  – Previous alg.’s cannot detect errors occurred during RSA combination
  – Check errors in each step
Countermeasures

1. $p' = p \cdot r$
   $d'_p = d_p + \text{random}_1 \cdot (p - 1)$
   $S'_p = m^{d'_p} \mod p'$
   if $\neg(p' \mod p \equiv 0 \land d'_p \mod (p - 1) \equiv d_p)$ then return error

   $q' = q \cdot r$
   $d'_q = d_q + \text{random}_2 \cdot (q - 1)$
   $S'_q = m^{d'_q} \mod q'$
   if $\neg(q' \mod q \equiv 0 \land q'_p \mod (q - 1) \equiv d_q)$ then return error

2. $S_p = S'_p \mod p$
   $S_q = S'_q \mod q$
   $S = \text{CRT}(S_p, S_q)$

3. if $\neg(S \mod p \equiv S_p \land S \mod q \equiv S_q)$ then return error

   $S_{pr} = S'_p \mod r$
   $d_{pr} = d'_p \mod (r - 1)$
   $S_{qr} = S'_q \mod r$
   $d_{qr} = d'_q \mod (r - 1)$
   if $(S_{pr}^{d_{qr}} = S_{qr}^{d_{pr}})$ then
   return $S$;
   else
   return error.
Countermeasures

• Infective computations
  – Yen et al. (2003)
    • Error detection based on decisional tests should be avoid.
    • If an error occurs in one of the exp., then it makes both 
      $S \neq S \mod p$ and $S \neq S \mod q$
    • Shown to be insecure by Yen and Kim (2004)
  – Blömer et al. (2003)
    • Shamir’s method + fault infective method
    • Shown to be insecure by Wagner (2004)
    • Variation by Blömer and Otto (2006)
Countermeasures

• Blömer et al. (2003)

1. \( S_p^* = m^{d_1} \mod (r_1 p) \),
   \( S_q^* = m^{d_2} \mod (r_2 q) \),

2. \( S^* = \text{CRT}(S_p^*, S_q^*) \mod (r_1 r_2 N) \),

3. 
   \( c_1 = (m - S^{*e_1} + 1) \mod r_1 \),
   \( c_2 = (m - S^{*e_2} + 1) \mod r_2 \),
   
   \( S = (S^*)^{c_1 c_2} \mod N \).
Countermeasures

• Ciet and Joye’s (2005)
  – Generalize Shamir’s + fault infective

Input: $m, p, q, d_p, d_q, I_q^*, r_1, r_2$
Output: $m^d \mod N$

1. $S_p^* \leftarrow m^{d_p} \mod p^*$ and $s_2 = m^{d_q} \mod \varphi(r_2) \mod r_2$,
2. $S_q^* \leftarrow m^{d_q} \mod q^*$ and $s_1 = m^{d_p} \mod \varphi(r_1) \mod r_1$,
3. $S^* \leftarrow S_q^* + q^* \cdot I_q^* \cdot (S_p^* - S_q^*) \mod p^*$,
4. $c_1 \leftarrow (S^* - s_1 + 1) \mod r_1$
5. $c_2 \leftarrow (S^* - s_2 + 1) \mod r_2$
6. $\gamma \leftarrow \lfloor (r_3 c_1 + (2^l - r_3)c_2) / 2^l \rfloor$
7. $S \leftarrow (S^*)^\gamma \mod N$
8. return $S$.

$p^* = r_1 p,$
$q^* = r_2 q,$
$q^* Inv = (q^*)^{-1} \mod p^*.$
Countermeasures

• Giraud’s (2005)
  – Use the fact that temporary variables \((a_0, a_1)\) are of the form \((m^\alpha, m^{\alpha+1})\) in Montgomery Ladder

\[
\begin{align*}
    a_0 & \leftarrow 1 \\
    a_1 & \leftarrow m \\
    \text{for } i \text{ from } n - 1 \text{ to } 0 \text{ do } \\
    a_{\bar{d}_i} & \leftarrow a_{\bar{d}_i} \cdot a_{d_i} \mod N \\
    a_{d_i} & \leftarrow a_{d_i}^2 \mod N \\
    \text{return } a_0 .
\end{align*}
\]
Countermeasures

- Giraud’s (2005)

SPA-FA-resistant exponentiation

Input: \( m, d = (d_{n-1}, \ldots, d_0), N \)
Output: \( m^d \mod N \)

\[
\begin{align*}
a_0 & \leftarrow m \\
a_1 & \leftarrow m^2 \mod N \\
\text{for } i \text{ from } n - 2 \text{ to } 1 \text{ do} \\
& \quad a_{d_i} \leftarrow a_{d_i} \cdot a_{d_i} \mod N \\
& \quad a_{d_i} \leftarrow a_{d_i}^2 \mod N \\
& \quad a_1 \leftarrow a_1 \cdot a_0 \mod N \\
& \quad a_0 \leftarrow a_0^2 \mod N \\
\text{if (Loop Counter } i \text{ not modified) \& (Exponent } d \text{ not modified) then} \\
& \quad \text{return } (a_0, a_1), \\
\text{else} \\
& \quad \text{return error.}
\end{align*}
\]
Countermeasures

• Giraud’s (2005)

Input: $m, p, q, d_p, d_q, I_q$
Output: $m^d \mod N$

1.1 $(S_p^*, S_p) \leftarrow$ SPA-FA-EXP$(m, d_p, p)$
1.2 $(S_q^*, S_q) \leftarrow$ SPA-FA-EXP$(m, d_q, q)$
2.1 $S^* \leftarrow$ CRT$(S_p^*, S_q^*)$
2.2 $S \leftarrow$ CRT$(S_p, S_q)$
2.3 $S^* \leftarrow m \cdot S^* \mod (p \cdot q)$
3.1 if $S^* = S$ & (Parameters $p$ and $q$ not modified) then
3.2 return $S$
3.3 else
3.4 return error
Countermeasures

• Still more countermeasures
  – CT-RSA 2009, Rivain, “Securing RSA against fault analysis by double addition chain exponentiation.”
Double faults attacks

• 2007, Kim and Quisquater

Step 1. Computation of two exponentiation
- Compute $S_p^*$ and $S_q^*$

Step 2. CRT combination
- Compute $S^* \leftarrow \text{CRT}(S_p^*, S_q^*)$

Step 3. Fault detection
- Return $\begin{cases} S \leftarrow f(S^*) & \text{if there is no error,} \\ \bot & \text{otherwise.} \end{cases}$

• Attack model
  – First fault on Step 1: Error on one of the exp.
  – Second fault just before Step 3: Skip Step3
Experimental results

- Generator for Glitch on supply voltage
- Circuit
- Normally running at 5V
- Glitch ~ 90V 50us
CRT combination & Fault detection

Fig. 1 RSA-CRT with CJ without faults

Glitch attack

Fault infection occurs

Fig. 2 Glitch attack during Sp* exp.
Fig. 3 Glitch attack both $Sp^*$ and $(S^*)^\gamma$

Glitch attack

2nd Glitch attack

Step 3 is skipped &
Return to main

CRT combination & Fault detection
Fault attacks on Block ciphers
Block cipher and Nonlinearity

- Each round
  - Key addition, SPN
- SPN
  - Substitution (non-linear)
  - Permutation
- Nonlinearity is important to resist DC, LC

General structure of block cipher
Fault attack models in Block cipher

- Fault is transmitted through linear part, only non-linear part is sufficient to consider.
- If we know $P_i$ and $P_{i+1}$, we can compute $K_i$.
- But, we know one of $P_i$ and $P_{i+1}$.
- Physical attack makes us to know partial info on one of them.

General structure of i-th round of block cipher:

(a) beginning rounds

$P_i \oplus K_i \rightarrow S$-box (nonlinear part)

$P_{i+1} = S[P_i] \oplus K_i$

(b) final rounds

$P_i \rightarrow S$-box (nonlinear part)

$S[P_i] \rightarrow K_i$

$P_{i+1} = S[P_i] \oplus K_i$
Fault attack models in Block cipher

Input: $P^i$ (or $P^{i+1}$)
Output: $K^i$
Procedure:
1. Gather partial information of $P^{i+1}$ (or $P^i$)
2. Figure out $K^i$ based on $P^i$ (or $P^{i+1}$) and partial information of $P^{i+1}$ (or $P^i$).
Fault attack models in Block cipher

- 2 partial info. on the output (or input) of S-box
  - The value
  - The difference
1) The value of the output of S-box

$$S[P^i \oplus K^i] = P^{i+1}.$$ 

• Several S-boxes (4 or 8 bits input/output)
• We can find a byte of the key in 8*8 S-box,
  – By exhaustive search over all 256 candidates
  – 128 candidates left, if one bit of $P^{i+1}$ is known.
    • With several different $P^i$, we can find a byte
      – Average 9 pairs
• How can we get info. of the output of S-box?
  – Microprobing
  – “Stuck-at” faults
2) The diff. of the outputs of S-box

\[ S[P_1^i \oplus K^i] = P_1^{i+1}, \]
\[ S[P_2^j \oplus K^i] = P_2^{i+1}. \]

\[ S[P_1^i \oplus K^i] \oplus S[P_2^j \oplus K^i] = P_1^{i+1} \oplus P_2^{i+1}. \]

- If we know the output difference \( \Delta_O = P_1^{i+1} \oplus P_2^{i+1} \)
  - We can compute \( K^i \)

- How can we get the output difference?
  - Exact value
  - A set of possible values
2) The diff. of the outputs of S-box

• Exact value
  – It is not easy to know $\Delta O$ as we don’t know $P_{1i+1}$, $P_{2i+2}$
  – Use side channel info.
    • We cannot to guess the value of $P_{1i+1}$ and $P_{2i+2}$, but we can check if $P_{1i+1} \neq P_{2i+2}$ with side channel info.
    • Induce faults on $P_{2i+2}$, we have $P_{2i+2} + e$
    • Try to fi $\Delta O$ $P_{1i+1}$ that is the same with $P_{2i+2} + e$, which makes $\Delta O = e$. 
2) The diff. of the outputs of S-box

• A set of possible values

\[ S[P^i] \oplus K^i = P^{i+1} \]

– Induce faults on the input \( P^i \), we have

\[ S[P^i] \oplus K^i = P^{i+1} \]

– We know \( P^{i+1} \) and \( P^{i+1} \)

– The number of possible differences = \( 2^m - 1 \) (m: size of plaintext)

– If we change only a byte, we have \((2^8-1)^\lceil \frac{m}{8} \rceil\) possible diff. We can reduce the possible candidates of \( K^i \) by

\[ \frac{(2^8 - 1) \times k}{2^m - 1} \]
Fault attacks on Block ciphers

<table>
<thead>
<tr>
<th>Partial info.</th>
<th>Value</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct</td>
<td>Probing</td>
<td>Direct difference</td>
</tr>
<tr>
<td>indirect</td>
<td>Stuck-at</td>
<td>A set of differences</td>
</tr>
</tbody>
</table>
Fault attacks on AES

• Advanced Encryption Standard (Rijndael)
  – Successor of DES
  – Invented by Joan Daemen and Vincent Rijmen

• Enrique Zabala’s animation
  – Available at http://www.math.ntu.edu.tw/~jmchen/
Fault attacks on AES State

• The best attack until now
  – Piret and Quisquater

• We assume
  – Random fault is injected in a byte between 7\textsuperscript{th} and 8\textsuperscript{th} round Mixed column.
8 round

Sub bytes
Shift rows
MixCol

9 round

Sub bytes
Shift rows
MixCol

255 differences

$256^d (255/255^d)$ of keys survive

After N pairs, $256^d (255^3)^N$ candidates remain

255 differences

After 2 pairs, only 1 candidate remains

255 differences
8 round

9 round

10 round
Fault attacks on AES Key Schedule

• The best attack until now
  – Kim and Quisquater, 2008
  – Basic attack
    • Retrieve 32 bits with 2 pairs, 1 fault
  – Improved attack
    • Retrieve 96 bits with 2 pairs, 1 fault injection point
    • Retrieve 128 bits with 4 pairs, 2 fault injection points
Fault Model

• We assume
  – Random fault is injected in 9th round AES Key Scheduling process
  – Some bytes of the first column of 9th round key are corrupted
Basic Attack

• Assumption

  – One byte of 1st column of 9th round key is corrupted

\[ a = K_{0,0}^9 \text{ XOR } K_{0,0}^9 \]
Basic Attack

256^4 (255/255^4) of keys survive
After N pairs, 256^4 (255^-3)^N candidates remain
After 2 pairs, only 1 candidate remains

255^4 possible difference \( \Delta = C \text{ XOR } C^* \)
Basic Attack

9th round

Sub bytes → Shift rows → MixCol

9th subkey

State 0

$2^{16}$ candidates for $K^{10}_{0,1}$ & $K^{10}_{0,3}$

Guess $K^{10}_{0}$ & $K^{10}_{0}$

→ $2^{64}$ candidates

→ Impractical

10th round

State 1

Sub bytes → Shift rows

State 2

State 3
Basic Attack

• 2 pairs \((C,C^*)\) and \((D,D^*)\)
  • Fault on \(C^*\) does not need to be same with that on \(D^*\)
  • Denote \(a_1 = K_{9,0}^9 \text{ XOR } K_{9,0}^9\) for \((C,C^*)\)
  • \(a_2 = K_{9,0}^9 \text{ XOR } K_{9,0}^9\) for \((D,D^*)\)
1. Compute candidate for \((K_{10,1}^{10}, K_{10,3}^{10}, a_1, a_2)\)
Basic Attack

2. Compute candidate for \((K^{10}_{10,1}, K^{10}_{10,2}, K^{10}_{10,3}, a_1, a_2)\)

For \(2^8\) candidates for \(K^{10}_{10,2}\) & \(K^*\) from Step 1
Basic Attack

3. Compute candidate for \( (K_{10}^{0,0}, K_{10}^{0,1}, K_{10}^{0,2}, K_{10}^{0,3}, a_1, a_2) \)

9th round

10th round

For \( 2^8 \) candidates for \( K_{10}^{0,0} \) & \( K^* \) from Step 2
Basic Attack

• Finally we have
  – One correct key for \((K_{0,0}^{10}, K_{0,1}^{10}, K_{0,2}^{10}, K_{0,3}^{10})\)
  – Faulty values \((a_1, a_2)\)
  – Simulation results
    • Less than 0.5 second on 3.2GHz Pentium 4 PC
Basic Attack

9th round

Sub bytes → Shift rows → MixCol → State 0

10th round

Sub bytes → Shift rows → State 1

State 2 → State 3
Basic Attack

• Finally we have
  – One correct key for \((K_{10,0}^{10}, K_{10,1}^{10}, K_{10,2}^{10}, K_{10,3}^{10})\)
  – Faulty values \((a_1, a_2)\)
  – Simulation results
    • Less than 0.5 second on 3.2GHz Pentium 4 PC

• With 8 pairs
  – We can find 128 bits
Improved Attack

- Consider errors on consecutive bytes

(a) Fault injection on $K_{6,0}^3$
Improved Attack

- We can find 96 bits with 2 pairs
- Last 32 bits
  - Another faults on $K_{3,0}^9$ with 2 pairs
    - Total 4 pairs for 128 bits
    - 2.3 seconds by simulation
  - Exhaustive search for 32 bits
How to make your code secure against fault attacks
Secure codes against fault attacks

• Give some examples for SW development secure against fault attacks
  – There is NO single method to prevent all fault attacks completely.
  – Present some rules to reduce the possibility of fault attacks
Secure codes against fault attacks

• Avoid conditional check
  – From the viewpoint of low-level implementation of this decision procedure, it often totally relies on the status of the zero flag of a processor.
  – The zero flag is a bit of the status register in a processor.
  – So, if an attack can induce a random fault into the status register, then conditional jump instruction may perform falsely.
Secure codes against fault attacks

• Do not Initialize constant = 0x00

  Initialize A=0x00
  Compute B1
  If errors:
    A += 1
  Compute B2
  If errors:
    A += 1

  ← Fault attack

A = 0xA5
Secure codes against fault attacks

- Program flow

Compute A

IF no errors

Yes

Execute B

No

Output Error

Compute A

If Errors?

Output Errors

Else

Execute B

Routine Success:

Execute B

Compute A

If Errors?

Output Errors

Else

Goto Routine Success

skip
Secure codes against fault attacks

• Program flow

\begin{align*}
& A = 0xA5 \\
& \text{Compute C1} \\
& A += c1 \\
& \text{Compute C2} \\
& A += c2 \\
& \text{Compute C3} \\
& A += c3 \\
& \text{Check A}
\end{align*}
Secure codes against fault attacks

- Bait between important parts

Compute $C$

$A += c_1$

Compute $C_1$

Compute $C_2$
Secure codes against fault attacks

• I/O
  – Check output length, input address

  for i less than output_length:
  output bytes

  – Use Watchdog timer
Secure codes against fault attacks

• COPY
  – Source, destination address check.
  – Contents check
    • Integrity check
Thank you!