Fault Induction & Bug Using

Seminar "Praktische Kryptoanalyse"

Dominik Bruhn

Institut für Kryptographie und Sicherheit

Januar 2010

Einleitung

- Wie kann es zu (Rechen-)fehlern in Geräten kommen?
- Wie können diese Fehler ausgenützt werden?
- Was kann getan werden um die Geräte und Programme besser zu schützen?

Definitionen

Fault Induction

- Ein Gerät soll angegriffen werden
- Physikalischer Zugang notwendig
- Veränderte äußere Parameter
- Recheneinheit liefert falsches Ergebniss

aus [Boneh, DeMillo and Lipton, 1997]

Versorgungspannung und Taktgeber

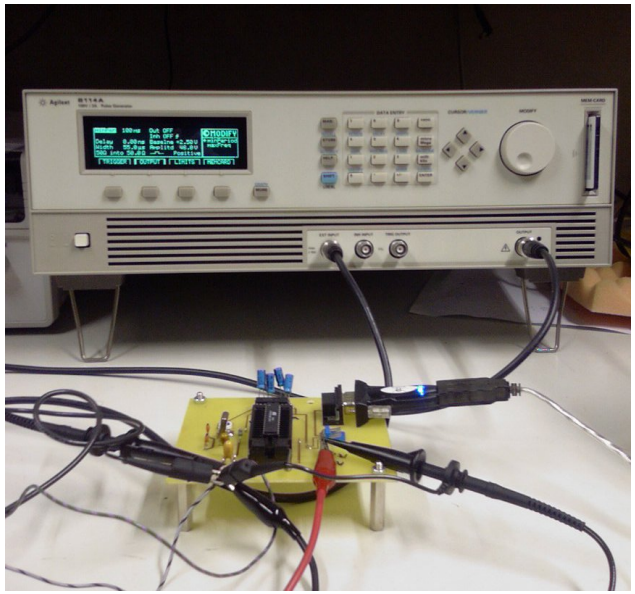
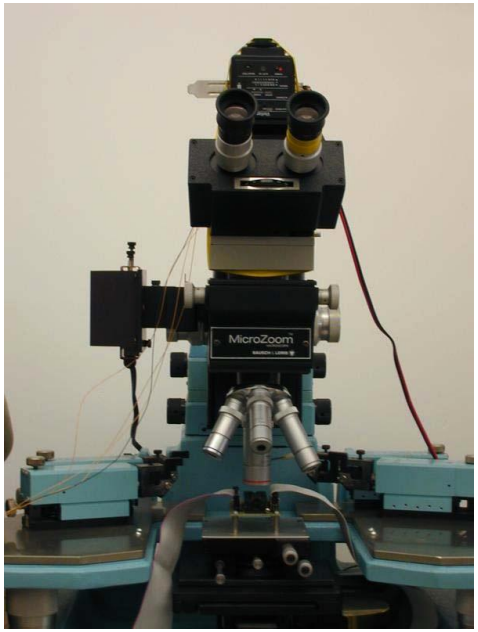
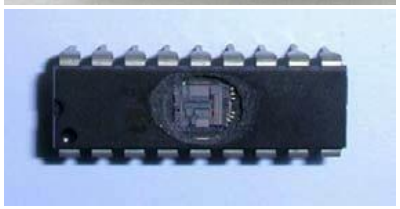


Abbildung: Versuchsaufbau aus [Kim and Quisquater, 2007]

- Erhöhen der Versorungungsspannung über die Grenzwerte
- Verändern der Taktfrequenz
- Überspringen von Befehlen und Verändern von Ergebnissen

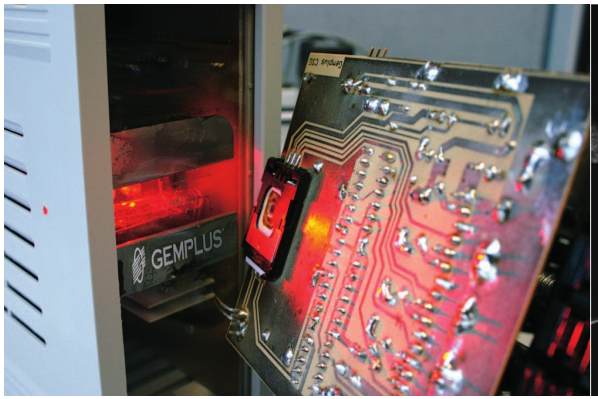
Licht und Laser





- Gehäuse des Chips muss geöffnet werden
- Ermöglicht Manipulationen des Speicherinhaltes

Abbildungen: [Skorobogatov and Anderson, 2003], [Naccache, 2005]



Bug Using

Definition

- Rechenfehler sind bereits im CPU vorhanden
- Kann aus der Ferne ausgenutzt werden
- Deterministisch

Mögliche Quellen

- Unbeabsichtige Fehler
- Beabsichtige Fehler

aus [Biham, Carmeli and Shamir, 2008]

RSA Erklärung - Dekodierung

zu berechnen

$K \equiv C^d \ mod \ N$

privater Schlüssel (d, N) *mit* $N = p * q$ $(p, q \ prim)$ *und* d *geeignet*

Verschlüsselter Nachricht C

RSA Erklärung - Dekodierung

zu berechnen

$K \equiv C^d \ mod \ N$

privater Schlüssel (d, N) *mit* $N = p * q$ $(p, q \ prim)$ *und* d *geeignet*

Verschlüsselter Nachricht C

Implementation mit dem CRT

Berechne stattdessen:

$$K_1 = C^d \ mod \ p \text{ und } K_2 = C^d \ mod \ q$$

es finden ist nun:

$$K = \beta * K_1 + \alpha * K_2 \ mod \ N$$

$\alpha = k * p$ und $\beta = l * q$ mit l, k geeignet gewählt.

RSA Angriff

Korrekte Berechnung:

$$C = \beta * K_1 + \alpha * K_2 \ mod \ N$$

Fehler tritt bei der Berechnung von K_1 auf ($\hat{K_1} \neq K_1$):

$$\hat{C} = \beta * \hat{K_1} + \alpha * K_2 \ mod \ N$$

Es gilt nun:

$$ggT(C - \hat{C}, N) =$$

RSA Angriff

Korrekte Berechnung:

$$C = \beta * K_1 + \alpha * K_2 \ mod \ N$$

Fehler tritt bei der Berechnung von K_1 auf ($\hat{K_1} \neq K_1$):

$$\hat{C} = \beta * \hat{K_1} + \alpha * K_2 \ mod \ N$$

Es gilt nun:

$$ggT(C - \hat{C}, N) =$$
$$ggT(\beta * K_1 + \alpha * K_2 - (\beta * \hat{K_1} + \alpha * K_2), N) =$$

RSA Angriff

Korrekte Berechnung:

$$C = \beta * K_1 + \alpha * K_2 \ mod \ N$$

Fehler tritt bei der Berechnung von K_1 auf ($\hat{K_1} \neq K_1$):

$$\hat{C} = \beta * \hat{K_1} + \alpha * K_2 \ mod \ N$$

Es gilt nun:

$$ggT(C - \hat{C}, N) =$$
$$ggT(\beta * K_1 + \alpha * K_2 - (\beta * \hat{K_1} + \alpha * K_2), N) =$$
$$ggT(\beta * K_1 + \alpha * K_2 - \beta * \hat{K_1} - \alpha * K_2, N) =$$

RSA Angriff

Korrekte Berechnung:

$$C = \beta * K_1 + \alpha * K_2 \ mod \ N$$

Fehler tritt bei der Berechnung von K_1 auf ($\hat{K_1} \neq K_1$):

$$\hat{C} = \beta * \hat{K_1} + \alpha * K_2 \ mod \ N$$

Es gilt nun:

$$ggT(C - \hat{C}, N) =$$
$$ggT(\beta * K_1 + \alpha * K_2 - (\beta * \hat{K_1} + \alpha * K_2), N) =$$
$$ggT(\beta * K_1 + \alpha * K_2 - \beta * \hat{K_1} - \alpha * K_2, N) =$$
$$ggT(\beta * K_1 - \beta * \hat{K_1}, N) =$$

RSA Angriff

Korrekte Berechnung:

$$C = \beta * K_1 + \alpha * K_2 \ mod \ N$$

Fehler tritt bei der Berechnung von K_1 auf ($\hat{K_1} \neq K_1$):

$$\hat{C} = \beta * \hat{K_1} + \alpha * K_2 \ mod \ N$$

Es gilt nun:

$$ggT(C - \hat{C}, N) =$$
$$ggT(\beta * K_1 + \alpha * K_2 - (\beta * \hat{K_1} + \alpha * K_2), N) =$$
$$ggT(\beta * K_1 + \alpha * K_2 - \beta * \hat{K_1} - \alpha * K_2, N) =$$
$$ggT(\beta * K_1 - \beta * \hat{K_1}, N) =$$
$$ggT(\beta * (K_1 - \hat{K_1}), N) =$$

RSA Angriff

Korrekte Berechnung:

$$C = \beta * K_1 + \alpha * K_2 \ mod \ N$$

Fehler tritt bei der Berechnung von K_1 auf ($\hat{K_1} \neq K_1$):

$$\hat{C} = \beta * \hat{K_1} + \alpha * K_2 \ mod \ N$$

Es gilt nun:

$$ggT(C - \hat{C}, N) =$$
$$ggT(\beta * K_1 + \alpha * K_2 - (\beta * \hat{K_1} + \alpha * K_2), N) =$$
$$ggT(\beta * K_1 + \alpha * K_2 - \beta * \hat{K_1} - \alpha * K_2, N) =$$
$$ggT(\beta * K_1 - \beta * \hat{K_1}, N) =$$
$$ggT(\beta * (K_1 - \hat{K_1}), N) =$$
$$ggT(l * q * (K_1 - \hat{K_1}), p * q) = q$$

Code-Angriffe

```
1    b = answer_address
2    a = answer_length
3    if (a == 0) goto 8
4    transmit(*b)
5    b = b + 1
6    a = a - 1
7    goto 3
8    ....
```

aus [Anderson and Kuhn, 1998]

Gegenmaßnahmen

Auf Hardware-Ebene

- Schutzfunktionen
- Sensoren
- Redundante Auslegung

Auf Software-Ebene

- Überprüfen von Ergebnissen
- Doppelte Ausführung

Auf Algorithmen-Ebene

- Schutzfunktionen
- Bessere Implementationen

Fazit

- Großes Forschungsgebiet für die Industrie
- Interessant auch für die Forschung
- Breites Spektrum von angreifbaren Geräten

Literatur und Quellen


Dan Boneh and Richard A. DeMillo and Richard J. Lipton
On the Importance of Checking Cryptographic Protocols for Faults.


Eli Biham and Yaniv Carmeli and Adi Shamir
Bug Attacks


Chong Hee Kim and Jean-Jacques Quisquater
Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures and Ubiquitous Computing Systems


Sergei P. Skorobogatov and Ross J. Anderson
Optical Fault Induction Attacks


Naccache, D.
Finding faults [data security]


Ross Anderson and Markus Kuhn
Low cost attacks on tamper resistant devices