Attacks and Defenses

Dr. Falko Strenzke

fstrenzke@cryptosource.de

cryptosource

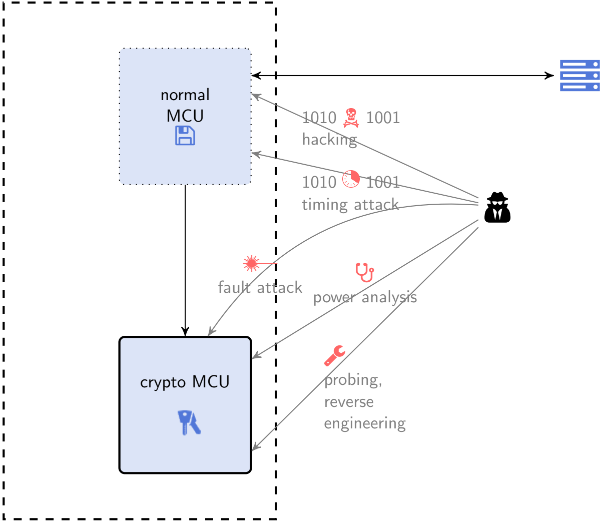
Cryptography. Security.

August 4, 2020

1. Attacks and Attack Resistance

2. Hardware Security Solutions

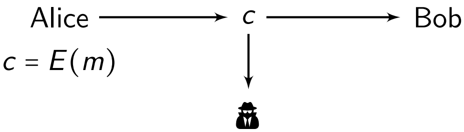
Attacks on Cryptographic Implementations



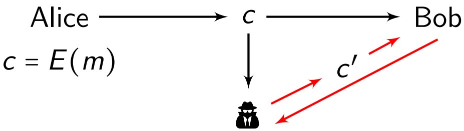
Software Attacks

- no physical access required – remote attacks
- “hacking”
- exploit
  - implementation flaws, e.g. buffer overflows
  - software fault attacks (decryption oracle attacks)
  - timing attacks
- scales well for attacker with low risk of detection
- some cryptography-specific software attacks exist

Decryption Oracle Attacks



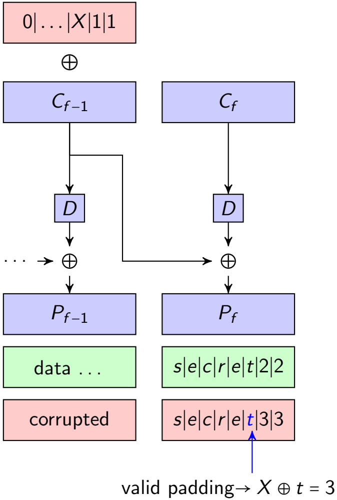
- classical attack scenario in cryptography: passive attacks



- Around 2000: active attacks

Padding Oracle Attacks

- CBC mode encrypts full multiples of the block length
- requires filling up of final block with padding bytes:
  - PKCS#7 Padding: E_k(<data> |4|4|4|4)
- Padding Oracle Attack:
  - Attacker manipulates CBC-encrypted ciphertext
  - triggers decryption
    - well-formed padding: no error
    - malformed padding: error indicated

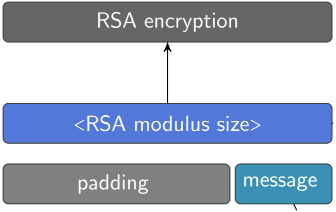


Symmetric Decryption Oracle Attacks in Practice

- Powerful attack which leads to total decryption of the plaintext
- Many vulnerabilities
  - SSL, IPsec: padding oracle (2002)
  - TLS: “Lucky 13” (2015), a timing attack variant
  - XML Encryption: application oracle (2011)
- ⚠ authenticity (MAC, signature) must be verified prior to decryption

Public-key Decryption Oracle Attacks in Practice

- PKCS#1 v1.5 encryption encoding for RSA

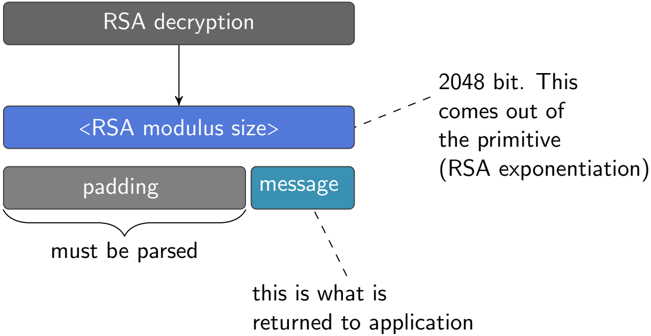


2048 bit. This must be
input into primitive
(RSA exponentiation)

this is what is
encrypted by application
(or a hash value
to be signed)

Public-key Decryption Oracle Attacks in Practice

- PKCS#1 v1.5 encryption encoding for RSA



RSA decryption

<RSA modulus size>

padding

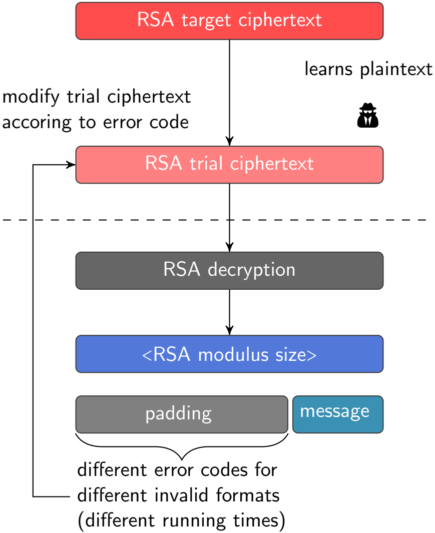
message

must be parsed

2048 bit. This comes out of the primitive (RSA exponentiation)

this is what is returned to application

Public-key Decryption Oracle Attacks in Practice



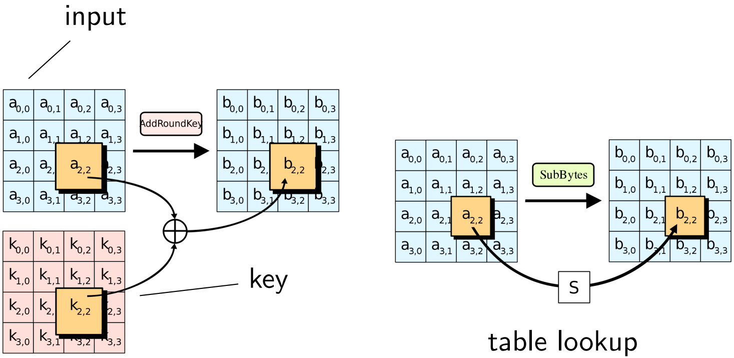
- 1993: RSA-PKCS#1 v1.5 encryption
- 1998: Bleichenbacher describes attack
  - decryption of ciphertext after many queries
- 2008: TLS 1.2 released
  - uses vulnerable PKCS#1 v1.5
  - specifies complicated countermeasures
- (2012: Attacks against XML Encryption)
- 2017: ROBOT ("Return Of Bleichenbacher's Oracle Threat")
  - many affected network devices

Timing Side-Channel Attacks

- Timing attacks are side-channel attacks
  - Trivial timing attack: byte-wise MAC comparison
  - Kocher 1996: Cryptographic timing attacks
  - Running time of RSA decryption is dependent on the private key
  - Many measurements and sophisticated statistical analysis may allow extraction of the private key

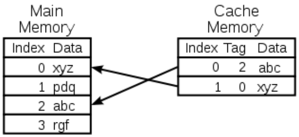
Cache-Timing Attacks on AES

- Efficient software implementations of AES use lookup tables for the SubBytes operation



- The indexing into the lookup table depends on a key byte
- $x = \mathrm{Table}[k_3 \oplus y]$ where y is a known input

CPU Cache



Main Memory

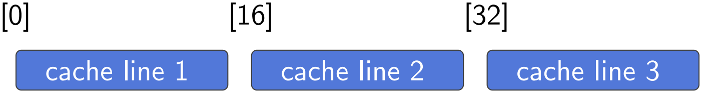
Index	Data
0	xyz
1	pdq
2	abc
3	rgf

Cache Memory

Index	Tag	Data
0	2	abc
1	0	xyz

Cache-Timing Attacks on AES

- The indexing into the lookup table depends on a key byte
- $x = \mathrm{Table}[k_3 \oplus y]$ where y is a known input



[0] [16] [32]

cache line 1 cache line 2 cache line 3

- repeated indexing into the same cache line: faster
- statistical analysis reveals key
- highly relevant for embedded systems with more deterministic timing behaviour
- (Note: cache-timing is used as a covert channel in Meltdown)

Timing Attack Countermeasures

- constant time implementations
  - no conditional branching based on secret values
  - hard to verify – interplay with compiler
  - does not help against other side channel attacks
- executing operations on randomly transformed inputs
- random delays
- specifically against cache-attacks:
  - cache warming
  - effectiveness depends on exact context

Physical Attacks

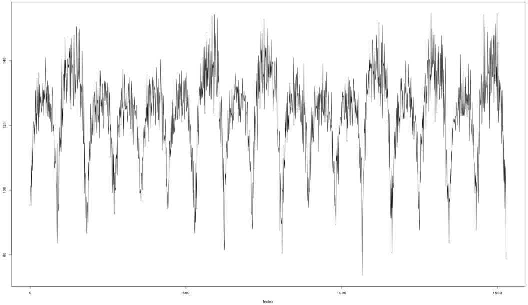
- scenario: attacker has (temporary) access to a device
  - a (stolen) smart card
  - “lunch-time” or “evil maid” attack
- attacker can trigger cryptographic operation
- perform measurements
- known in the smart card industry for decades

Power Analysis Attacks Basics

- Power Analysis Attacks
  - Power consumption of a CPU is dependent on
    - instruction type: higher for multiplication than addition
    - on the data: switching a register from 0x00...00 to $0xFF..FF$ requires more energy than to flipping a single bit

Simple Power Analysis against RSA

```
r = 1
for i = |d| down to 0
  r = r*r mod n
  if d[i] == 1
    r = r * m mod n
return r as c
```

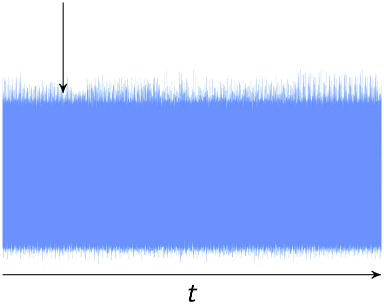


Courtesy of

SEGRIDS

Differential Power Analysis

- attack a single key byte in AES at a time
- $x = k_i \oplus y$
- y part of the input
- many different inputs with all 256 values of y
- measure power traces
- find points of greatest variation
- formulate hypotheses, e.g. $x = 0$ lowest / highest power consumption
- determine trace with lowest/highest power consumption → candidate for k_i
- repeat for all key bytes



Electromagnetic Emanation

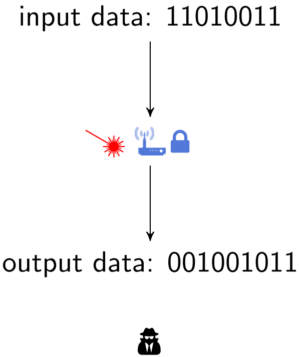
- measure electromagnetic emanation (EM) instead of power consumption
- directly on the chip
  - locate interesting functional block, e.g. register
  - measure EM emanation locally
- measurements from distance
  - less effective

Power/EM Analysis Attacks Countermeasures

- add random noise
- add random delays
- masking internal values
  - instead of $x = k_i \oplus y$
  - compute $x' = (m \oplus k_i) \oplus y$
- dual rail implementation: compensate differences
- shielding against EM emanation

Hardware Fault Attacks

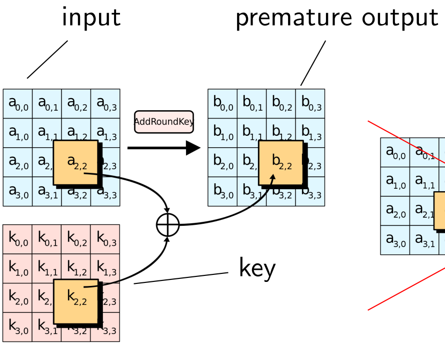
- Active attacks
- locate targeted functional unit on the chip
- use EM pulse or laser
- during a cryptographic operation
- effects
  - step over instruction
  - alter register values
- goals:
  - dump keys
  - dump intermediate values
  - evade security checks
- single run with low success probability
- many repetitions, automation



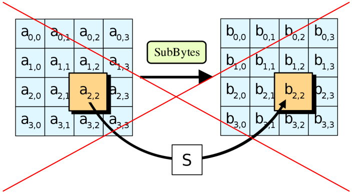
input data: 11010011

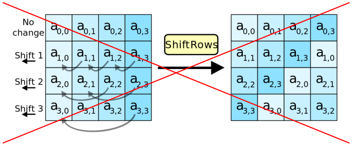
output data: 001001011

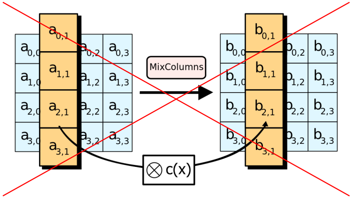
Example: Fault Attack against AES


input
premature output
AddRoundKey
key
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$
⊕

early termination enforced, “key ⊕ input” is dumped




No change
Shift 1
Shift 2
Shift 3
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$


MixColumns
⊗ c(x)
$a_{0,1}$ $a_{1,1}$ $a_{2,1}$ $a_{3,1}$
$b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{3,1}$

Countermeasures against Hardware Fault Attacks

- Redundant hardware layouts
- repeat operations and compare
- counter operations: verify encryption by decryption
- attack detection (and reaction)
- HW/SW checksums

Probing Attacks / Reverse Engineering

- Probing Attack / Reverse Engineering
- “there are no secrets in silicon”
- Chemical and mechanical removal of layers
- Analysing the gate structure
- Data extraction
- costly!
- Typical gains for the attacker
  - learning IP (firmware)
  - learning proprietary cryptographic algorithms
    - breaking them e.g. DECT (*)
  - learn system-wide master keys
  - find software bugs that allow remote exploitation

(*) `https://dedected.org/trac/raw-attachment/wiki/DSC-Analysis/FSE2010-166.pdf`

Hardware Security

- Security against physical attacks only with dedicated security modules
- a.k.a.
  - “security MCU”
  - “crypto chip”
  - “hardware security module”
  - “secure element”
- speed-up of cryptographic operations
- Typical features of security controllers
  - hardware random number generator
  - symmetric cryptographic engine (AES, Hash)
  - public-key support: modular arithmetic (RSA, ECC)
  - Fault attack and side-channel countermeasures
  - protection against probing attacks

Security Certifications

- FIPS 140-2 standard
  - NIST standard for the classification of cryptographic modules
  - Level 1 – no physical security measures
  - Level 2 – temper evidence
  - Level 3 – basic temper resistance
  - Level 4 – higher temper resistance
- Common Criteria (CC)
  - international standard for general security certification of IT components
  - complex methodology
  - Evaluation Assure Level (EAL) 1 - 7
    - EAL 3 “minimum”
    - EAL 7 high security
    - mostly EAL 3 - 5
    - influences evaluation methodology as well as physical resistance

Types of Security Controllers

- 🔒 “closed” cryptographic MCU
  - accessed via serial interface
  - typically supported features
    - key generation
    - secure key storage
    - execution of cryptographic operations
  - suitable for instance for device identification/authentication
- 💳 smart card controllers / secure elements
  - security controller with certified security OS
  - supports
    - secure file system
    - key management
    - cryptographic operations
    - sometimes custom JAVACard applications supported
  - fulfils high security requirements

Types of Security Controllers (2)

- </> “open” security controller with cryptographic coprocessor
  - shipped without OS
  - freely programmable
  - can run OS and/or application and perform security sensitive and cryptographic operations
  - usually high level of know-how required

Conclusion for Attacks and Defenses

- Types of Attacks
  - 💾 / 🕒 “Software Attacks” / Timing Attacks
    - remote attacks
    - scale well, can be automated!
    - precondition: vulnerable scheme or implementation
    - defense: sound implementation, countermeasures
  - 🩺 Passive “Hardware Attacks”
    - side channel attacks
    - precondition: control over device or at least proximity
    - defense: Hardware and software countermeasures (use of security controller)
  - ✳— Active “Hardware Attacks”
    - Fault Attacks
    - precondition: control over device
    - defense: security controller
  - 🔧 Probing Attacks / Reverse Engineering
    - recover secrets stored on the controller
    - commercial services exist for this
    - precondition: control over device
    - defense: security controller