Fault-Based Attack of RSA Authentication

Andrea Pellegrini,
Valeria Bertacco and Todd Austin
University of Michigan

DATE 10

UNIVERSITY OF MICHIGAN

Cryptography

- Public key cryptography is pervasive in our digital world

TPM Coprocessor

Online Banking

Entertainment Systems

Andrea Pellegrini
University of Michigan

2

Are these algorithms secure?

(i.e., cryptanalysis)

Attack the algorithm

by guessing key

13506641086599522334960
32162788059699388814756
05667027524485143851526
51060485953383394028715
05719094417982072821644
71551373680419703964191
74304649658927425623934
10208643832021103729587
25762358509643110564073
50150818751067659462920
55636855294….

time consuming:
> age of Universe

Attack the implementation

Side-channel

by monitoring side effects

- Timing [Brumley03]
- Power [Kocher99]

Fault-Based

a faulty processor may leak secrets

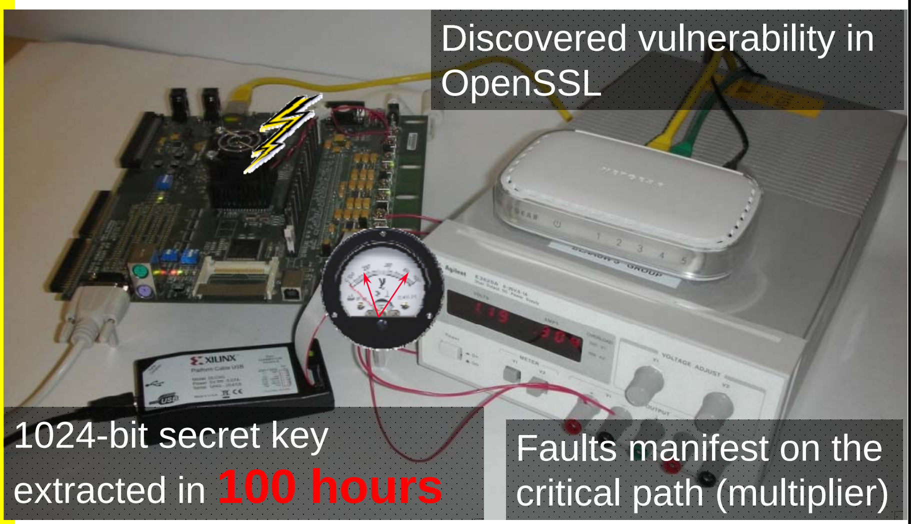
- Theoretical [Joye99, Boneh01, Wagner04]
- Demonstrated on simple components [Bar-El06]

Andrea Pellegrini
University of Michigan

3

Our Contribution

Fault-based attack on a complete system



Andrea Pellegrini
University of Michigan

4

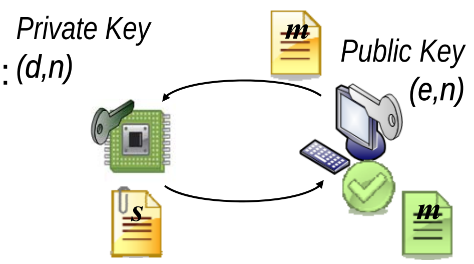
RSA Authentication

Correct behavior:

- Server challenge:

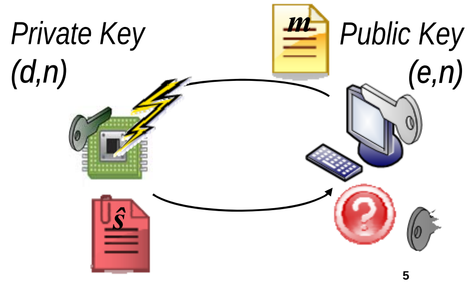
  s = m^d mod n
- Client verifies:

  m = s^e mod n



Faulty Server:

ŝ != m^d mod n



Andrea Pellegrini
University of Michigan

Computing: s=m^d mod n

Fall-back algorithm in OpenSSL

The algorithm partitions the exponent into windows:

d =1101 1011 0001…1101 1001 0101

```
s=1
for each window:
  for each bit in window: //4times
    s = (s * s) mod n
  s = (s * m^d[window]) mod n
return s
```

Andrea Pellegrini
University of Michigan

Computing: $s=m^d \bmod n$

d=214= 1101 0110

window 1 window 2

```
s=1
for each window:
  for each bit in window: //4times
    s = (s * s) mod n
  s = (s * m^d[window]) mod n
return s
```

s=1

$s= (\cdots(m^{1101})^2)^2)^2)^2$

$s= m^{1101}$

$s= (\cdots(m^{1101})^2)^2)^2)^2)m^{0110}$

$$s = (\cdots(m^{1101})^2)^2)^2)^2)m^{0110}$$

Andrea Pellegrini
University of Michigan

Faulty Signature: $\hat{s} != m^d \bmod n$

d=214= 1101 0110

window 1 window 2

```
s=1
for each window:
  for each bit in window: //4times
    s = (s * s) mod n
  s = (s * m^d[window]) mod n
return s
```

s=1

$\hat{s} = (\cdots(m^{1101})^2)^2) \pm 2^f)^2)^2$

$s= m^{1101}$

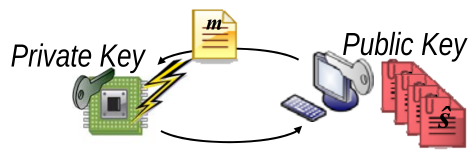
$\hat{s} = (\cdots(m^{1101})^2)^2) \pm 2^f)^2)^2)m^{0110}$

$$\hat{s} = (\cdots(m^{1101})^2)^2) \pm 2^f)^2)^2)m^{0110}$$

Andrea Pellegrini
University of Michigan

Retrieving the Private Key

- The attacker collects the faulty signatures



- The private key is recovered one window at the time

$d = d_3\, d_2\, d_1\, d_0$ (each window overlaid with X)

- The attacker checks its guess against the collected signatures

Andrea Pellegrini
University of Michigan

Retrieving the Private Key

- The private key is recovered one window at the time, guessing where and when the fault hits

$d = d_3\; d_2\; X\; X$

$$\hat{s} = (\cdots(m^{d_3})^{16})m^{d_2})^2)^2 \pm 2^f)^2)^2)m^{d_1})^{16}m^{d_0}$$

Already known (d_3) · Value? (d_2) · Which Multiplication? · Which bit? ($\pm 2^f$)

- Extend the window if no signature can confirm the value of the guess

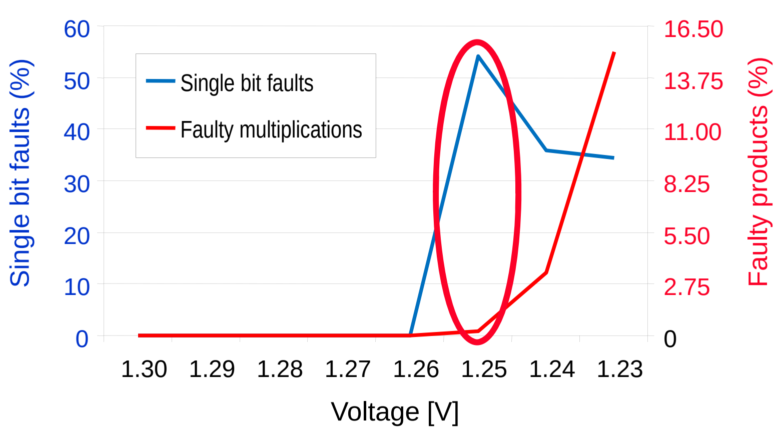
Andrea Pellegrini
University of Michigan

Physical Attack

Andrea Pellegrini
University of Michigan

Fault Injection

- A corrupted signature leaks data only if one multiplication was corrupted by a single bit flip



Andrea Pellegrini
University of Michigan

Physical Attack

Andrea Pellegrini
University of Michigan

13

Conclusions

- **Faults can leak vital private key data**
- Fault-based attack devised for OpenSSL Fixed Window Exponentiation algorithm
- Attack demonstrated on a complete physical Leon3 SPARC system
- **Future work: general software fix using “blind”ing (patch for OpenSSL sent to developers)**

Andrea Pellegrini
University of Michigan

14