

Fault-Based Attack of RSA Authentication

Andrea Pellegrini,
Valeria Bertacco and Todd Austin
University of Michigan



Cryptography

- Public key cryptography is pervasive in our digital world



TPM Coprocessor



Online Banking



Entertainment Systems

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-EI06]

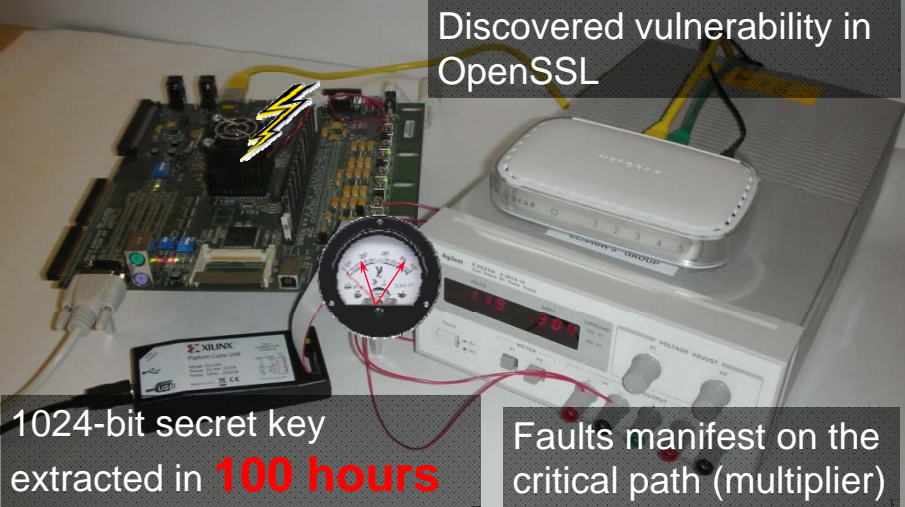


Andrea Pellegrini
University of Michigan

3

Our Contribution

Fault-based attack on a complete system



1024-bit secret key
extracted in **100 hours**

Discovered vulnerability in
OpenSSL

Faults manifest on the
critical path (multiplier)

Andrea Pellegrini
University of Michigan

4

RSA Authentication

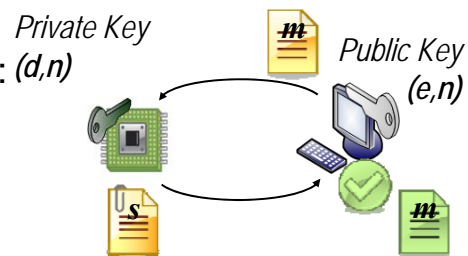
Correct behavior:

- Server challenge: (d,n)

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

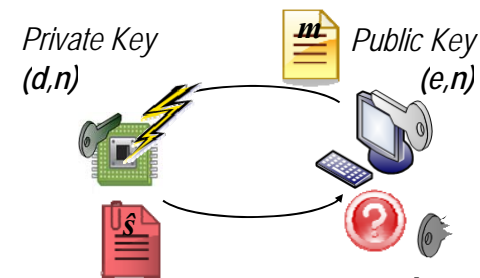
- Client verifies:

$$m = s^e \bmod n$$



Faulty Server:

$$\hat{s} \neq m^d \bmod n$$



Andrea Pellegrini
University of Michigan

5

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

Fall-back algorithm in OpenSSL

The algorithm partitions the exponent into windows:

$$d = \boxed{110110110001} \dots \boxed{110110010101}$$

```

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

6

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

$d = 214 = \underbrace{1101}_{\text{window 1}} \underbrace{0110}_{\text{window 2}}$

```


s=1    s=1
for each window:
  for each bit in window: //4times
    s = (s * s) mod n    s = (...(m1101)2)2)2
  s = (s * md[window]) mod n    s = m1101
return s    s = (...(m1101)2)2)2)m0110
    
```

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

Faulty Signature: $\hat{s} \neq m^d \pmod n$

$d = 214 = \underbrace{1101}_{\text{window 1}} \underbrace{0110}_{\text{window 2}}$

```

s=1    s=1
for each window:
  for each bit i  ndow: //4times
    s = (s * s) mod n     $\hat{s} = (\dots(m^{1101})^2)^2) \pm 2^f)^2$ 
  s = (s * md[window]) mod n    s = m1101
return s     $\hat{s} = (\dots(m^{1101})^2)^2) \pm 2^f)^2)^2 m^{0110}$ 
    
```

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

Retrieving the Private Key

- The attacker collects the faulty signatures

- The private key is recovered one window at the time

d = ~~X~~ ~~X~~ ~~X~~ ~~X~~

- The attacker checks its guess against the collected signatures

Andrea Pellegrini
University of Michigan 9

Retrieving the Private Key

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

d = d₃ ~~X~~ ~~X~~ X X

$$\hat{s} = (\dots (m^{d_3})^{16} m^{d_2})^2 \overset{+2f}{+2f} m^{d_1})^{16} m^{d_0}$$

Already known

Value?

Which Multiplication?

Which bit?

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

Andrea Pellegrini
University of Michigan 10

Physical Attack

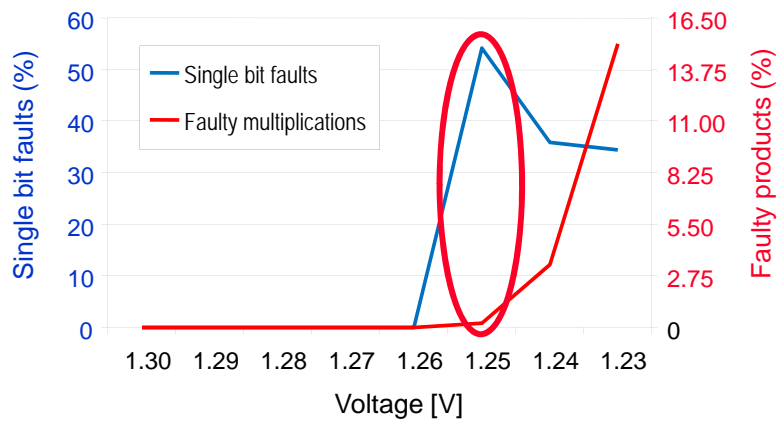


Andrea Pellegrini
University of Michigan

11

Fault Injection

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



Andrea Pellegrini
University of Michigan

12

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