Defeating with Fault Injection a Combined Attack Resistant Exponentiation

Benoit Feix
UL Security Lab, UK
XLIM, Limoges University

Alexandre Venelli
INSIDE Secure, France

Workshop COSADE 2013

Paris, 7 March 2013

Agenda

1. What is a combined attack?
2. Algorithm of Schmidt et al.
3. Fault attack
   1. On simplified algorithm
   2. On complete algorithm
4. Combined attack
5. Improved algorithm
6. Conclusion

What is a combined attack?

General principle

- Combines a fault attack with a leakage analysis
- Main goal: attack implementations resistant against fault and leakage analysis
- New implementations and new countermeasures often required

What is a combined attack?

Example on L2R exponentiation

Algorithm 1 Left-to-right multiply always exponentiation

Input: $x \in \mathbb{G}$ and $d = (d_{k-1}, \ldots, d_0)_2 \in \mathbb{N}$

Output: x^d

$R[0] \leftarrow 1$

$R[1] \leftarrow x$

$t \leftarrow 0$

for $i = k - 1$ to 0 **do**

  $R[0] \leftarrow R[0].R[t]$

  $t \leftarrow t \oplus d_i$

  $i \leftarrow i - 1 + t$

end for

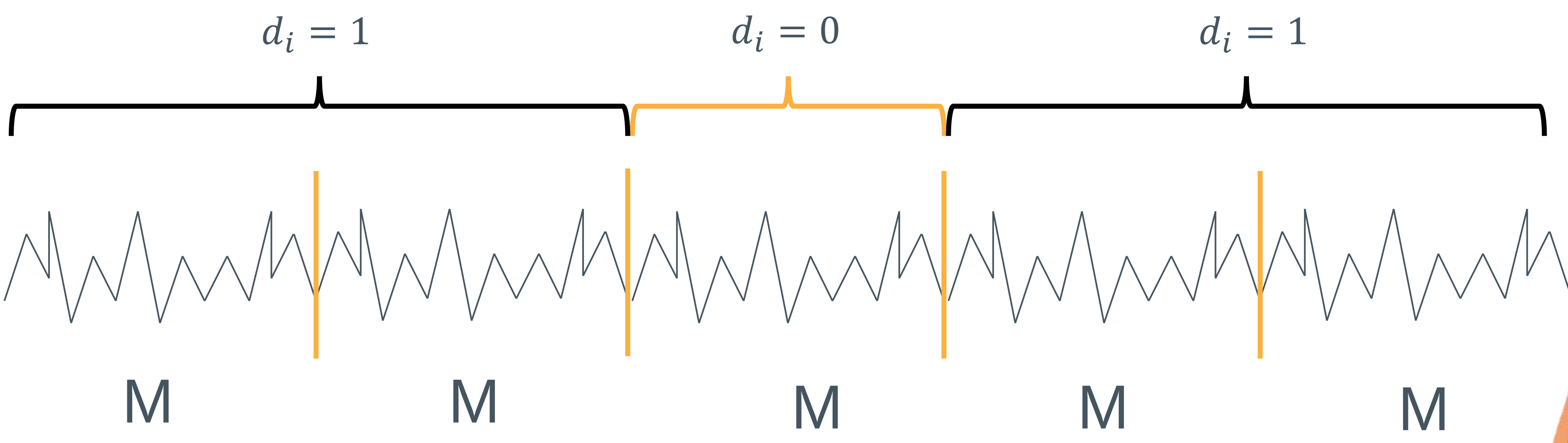
return $R[0]$

<u>Add:</u> classical fault checking mechanism

- *inverse operation calculation* or
- *doubling the calculation to verify equality of both*

What is a combined attack?

Example on L2R exponentiation



$$d = (101)_2$$

No SPA leakage, only multiplications

What is a combined attack?

Example on L2R exponentiation

Algorithm 1 Left-to-right multiply always exponentiation

Input: $x \in \mathbb{G}$ and $d = (d_{k-1}, \ldots, d_0)_2 \in \mathbb{N}$

Output: x^d

$R[0] \leftarrow 1$

$R[1] \leftarrow x$

$t \leftarrow 0$

for $i = k - 1$ to 0 **do**

  $R[0] \leftarrow R[0].R[t]$

  $t \leftarrow t \oplus d_i$

  $i \leftarrow i - 1 + t$

end for

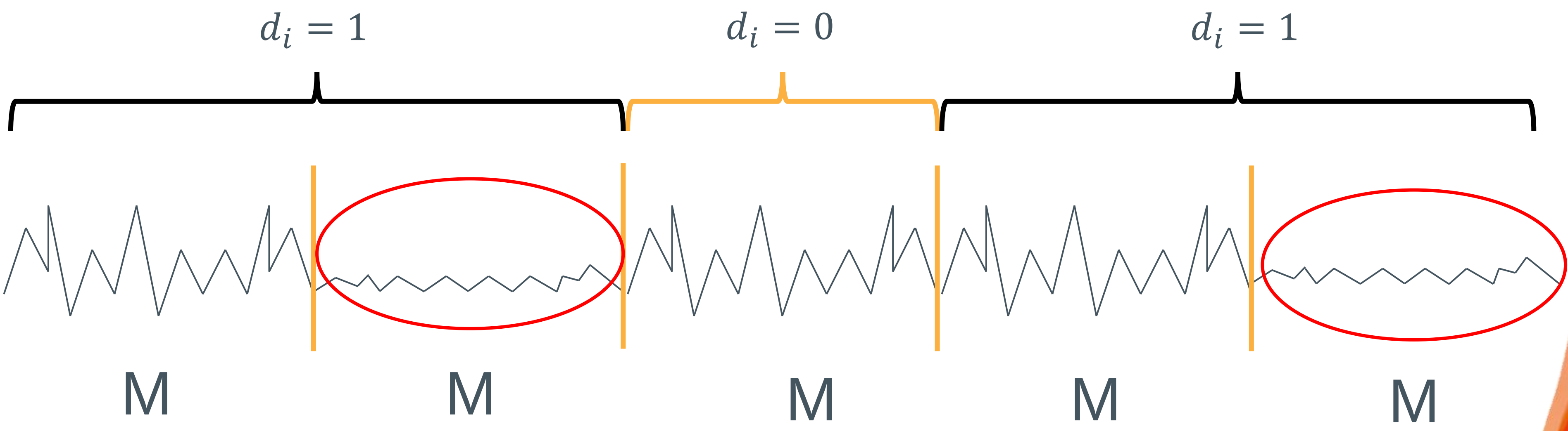
return $R[0]$

Skip instruction

Suppose $R[1] = 0$

What is a combined attack?

Example on L2R exponentiation



$$d = (101)_2$$

The use of the faulted register $R[1]$ is visible by SPA

What is a combined attack?

History on asymmetric

- **2007: Attack on atomic left-to-right exponentiation**
  - *Amiel et al. (FDTC)*
- **2010: Resistant algorithms for RSA and ECC**
  - *Schmidt et al. (LATINCRYPT)*
- **2011: Attack on scalar multiplication**
  - *Fan et al. (CHES)*
- **2012: Attack on prime generation**
  - *Vuillaume et al. (COSADE)*
- **2013: Attack on RSA-CRT**
  - *Barbu et al. (PKC)*

Agenda

1. What is a combined attack?
2. Algorithm of Schmidt et al.
3. Fault attack
    1. On simplified algorithm
    2. On complete algorithm
4. Combined attack
5. Improved algorithm
6. Conclusion

Algorithm of Schmidt et al.

General principle

- Add to (*potentially any*) SPA-resistant exponentiation
  - An infective computation method
  - An invariant system
- Link those two protections to strengthen the resistance

Algorithm of Schmidt et al.

Exponentiation

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod N r_2$
$R_1 \leftarrow i \cdot m \bmod N r_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
    $R_0 \leftarrow R_0 \cdot R_k \bmod N r_2$
    if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
        $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
    end if
    $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
    $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
    $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Algorithm of Schmidt et al.

Invariant

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod N r_2$
$R_1 \leftarrow i \cdot m \bmod N r_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \cdot R_k \bmod N r_2$
  if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Idempotent element $i \in \mathbb{Z}_{N r_2}$ such that:

- $i \equiv 1 \bmod N$
- $i \equiv 0 \bmod r_2$

Algorithm of Schmidt et al.

Invariant

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod N r_2$
$R_1 \leftarrow i \cdot m \bmod N r_2$
$d \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
    $R_0 \leftarrow R_0 \cdot R_k \bmod N r_2$
    if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
        $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
    end if
    $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
    $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
    $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Idempotent element $i \in \mathbb{Z}_{Nr_2}$ such that:

- $i \equiv 1 \bmod N$
- $i \equiv 0 \bmod r_2$

i "mixed in" the registers R_0 and R_1

Algorithm of Schmidt et al.

Invariant

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod N r_2$
$R_1 \leftarrow i \cdot m \bmod N r_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \cdot R_k \bmod N r_2$
  if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Idempotent element $i \in \mathbb{Z}_{Nr_2}$ such that:

- $i \equiv 1 \bmod N$
- $i \equiv 0 \bmod r_2$

i “mixed in” the registers R_0 and R_1

Efficient test of integrity:

$R_0 \bmod r_2 \mathrel{?=} 0 \bmod r_2$
$R_1 \bmod r_2 \mathrel{?=} 0 \bmod r_2$

Algorithm of Schmidt et al.

Infective computation

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod Nr_2$
$R_1 \leftarrow i \cdot m \bmod Nr_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \cdot R_k \bmod Nr_2$
  if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Encode the exponent using
$\psi_\alpha : \mathbb{Z}_{r_2} \times \mathbb{Z}_{r_2} \rightarrow \mathbb{Z}_{r_2}$

- $\psi_\alpha(d^{(j)}) = (\alpha + N)^{-1} . d^{(j)} \bmod r_2$
- $\psi_\alpha^{-1}(\tilde{d}^{(j)}) = (\alpha + N) . \tilde{d}^{(j)} \bmod r_2$

Infective computation

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod N r_2$
$R_1 \leftarrow i \cdot m \bmod N r_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \cdot R_k \bmod N r_2$
  if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Encode the exponent using $\psi_\alpha : \mathbb{Z}_{r_2} \times \mathbb{Z}_{r_2} \rightarrow \mathbb{Z}_{r_2}$:

- $\psi_\alpha(d^{(j)}) = (\alpha + N)^{-1}.\, d^{(j)} \bmod r_2$
- $\psi_\alpha^{-1}(\tilde{d}^{(j)}) = (\alpha + N).\, \tilde{d}^{(j)} \bmod r_2$

If $\alpha = 0 \bmod r_2$
Correct decoding
Else
False decoding

Algorithm of Schmidt et al.

Link invariant and infective computation

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod N r_2$
$R_1 \leftarrow i \cdot m \bmod N r_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \cdot R_k \bmod N r_2$
  if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0+R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Encode the exponent using $\psi_\alpha : \mathbb{Z}_{r_2} \times \mathbb{Z}_{r_2} \rightarrow \mathbb{Z}_{r_2}$:

- $\psi_\alpha(d^{(j)}) = (\alpha + N)^{-1} . d^{(j)} \bmod r_2$
- $\psi_\alpha^{-1}(\tilde{d}^{(j)}) = (\alpha + N) . \tilde{d}^{(j)} \bmod r_2$

If $\alpha = 0 \bmod r_2$

Correct decoding

Else

False decoding

$\alpha := \mathrm{R}_0 + \mathrm{R}_1 \bmod \mathrm{r}_2$ is the invariant check

Algorithm of Schmidt et al.

Additional check

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod N r_2$
$R_1 \leftarrow i \cdot m \bmod N r_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \cdot R_k \bmod N r_2$
  if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

If R_0 or R_1 is erased by fault
Corrupt the exponent

Check against the combined attack of Amiel et al. 2007

Output

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod Nr_2$
$R_1 \leftarrow i \cdot m \bmod Nr_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
    $R_0 \leftarrow R_0 \cdot R_k \bmod Nr_2$
    if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
        $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
    end if
    $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
    $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
    $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Returns (*possibly*) faulted results ☹

Agenda

1. What is a combined attack?
2. Algorithm of Schmidt et al.
3. Fault attack
    1. On simplified algorithm
    2. On complete algorithm
4. Combined attack
5. Improved algorithm
6. Conclusion

Fault attack

On simplified algorithm

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \dots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod Nr_2$
$R_1 \leftarrow i \cdot m \bmod Nr_2$
~~$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$~~
$[\tilde{d}^{(l-1)}, \dots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \dots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \cdot R_k \bmod Nr_2$
  if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \dots, \tilde{d}^{(0)}] \leftarrow [1, \dots, 1]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Simplified version: no exponent blinding

Fault attack

On simplified algorithm

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \dots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

```
r_1 ← random(1, 2^λ − 1)
r_2 ← random(1, 2^λ − 1)
i ← (r_2^{−1} mod N) · r_2
R_0 ← i · 1 mod N r_2
R_1 ← i · m mod N r_2
d̄ ← d + r_1 · φ(N)
[d̃^{(l−1)}, ..., d̃^{(0)}] ← [ψ_0(d̄^{(l−1)}), ..., ψ_0(d̄^{(0)})]
k ← 0
j ← bitlength(d̃) − 1
while j ≥ 0 do
    R_0 ← R_0 · R_k mod N r_2
    if (R_0 = 0) or (R_1 = 0) then
        [d̃^{(l−1)}, ..., d̃^{(0)}] ← [1, ..., 1]
    end if
    d̂ ← ψ^{−1}_{(R_0 + R_1 mod r_2)}(d̃^{(⌊j/W⌋)})
    k ← k ⊕ bit(d̂, j mod W)
    j ← j − ¬k
end while
c ← R_0 mod N
return c
```

Simplified version: no exponent blinding

Skip instruction

$\hat{d} = \psi_0^{-1}(1) = N \bmod r_2$

For the rest of the exponentiation!

Fault attack

On simplified algorithm

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod N r_2$
$R_1 \leftarrow i \cdot m \bmod N r_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \cdot R_k \bmod N r_2$
  if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Simplified version: no exponent blinding

Skip instruction

$\hat{d} = \psi_0^{-1}(1) = N \bmod r_2$

For the rest of the exponentiation!

Only W bits of $\hat{d}$ are used

Let $H = (N \bmod r_2) \bmod 2^W$

Fault attack

On simplified algorithm

- **Attacker knows v first bits of the exponent**
- **Fault u bits after in the loop**
- **Faulted exponent $\breve{d}_u$ of the result $\breve{S}_u$:**

$$\breve{d}_u = \underbrace{\sum_{i=\tilde{t}-v}^{\tilde{t}-1} 2^i.\tilde{d}_i}_{\text{Known part}} + \underbrace{\sum_{i=\tilde{t}-v-u}^{\tilde{t}-v-1} 2^i.\tilde{d}_i}_{u \text{ bits to retrieve}} + \sum_{i=0}^{\tilde{t}-v-u-1} 2^i.H_{(i \bmod W)}$$

Known part u bits to retrieve

with $\tilde{t}$ the bit size of the encoded exponent $\tilde{d}$

Fault attack

On simplified algorithm

- Faulted exponent $\breve{d}_u$:

$$\breve{d}_u = \underbrace{\sum_{i=\tilde{t}-v}^{\tilde{t}-1} 2^i . \tilde{d}_i}_{\text{Known part}} + \underbrace{\sum_{i=\tilde{t}-v-u}^{\tilde{t}-v-1} 2^i . \tilde{d}_i}_{u \text{ bits to retrieve}} + \sum_{i=0}^{\tilde{t}-v-u-1} 2^i . H_{(i \bmod W)}$$

Known part u bits to retrieve

- Guesses: u bits of d and W bits of H
- → Complete guessed result $S_g(u, H)$
- Validate guess by checking: $S_g(u, H) ?= \breve{S}_u$

Fault attack

On simplified algorithm

- **Attack retrieves u bits at a time**
- **Only possible if no exponent blinding**

- **Computational complexity:**

$$\mathcal{C} = \mathcal{O}\left(\frac{2^{(u+W)}.\,\tilde{t}}{u}\right)$$

- **Number of faults:**

$$\mathcal{F} = \mathcal{O}\left(\frac{\tilde{t}}{u}\right)$$

Fault attack

On simplified algorithm

- **Example of computational complexities for $u = 1$**

W	512 bits	1024 bits	2048 bits
8	$\mathcal{C} = 2^{18}$	$\mathcal{C} = 2^{19}$	$\mathcal{C} = 2^{20}$
16	$\mathcal{C} = 2^{26}$	$\mathcal{C} = 2^{27}$	$\mathcal{C} = 2^{28}$
32	$\mathcal{C} = 2^{42}$	$\mathcal{C} = 2^{43}$	$\mathcal{C} = 2^{44}$

- **Validated on PC using the GMP library**

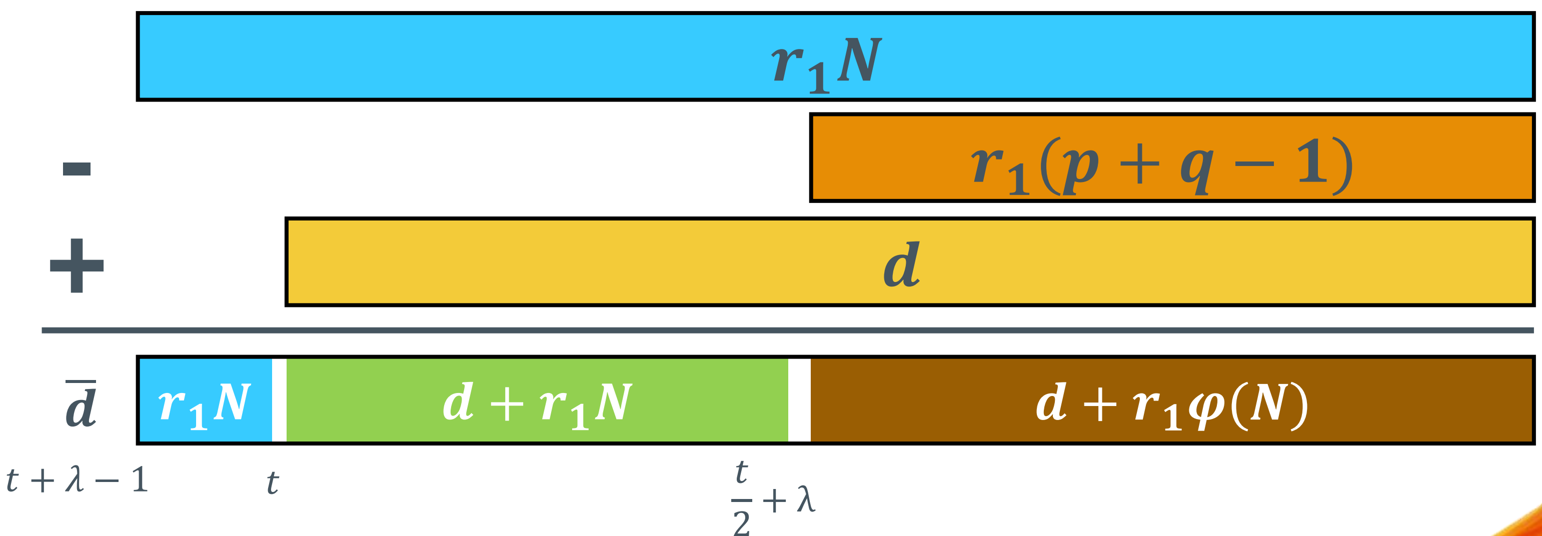
Agenda

Fault attack

On complete algorithm

- **Exponent blinding:** $\bar{d} = d + r_1\varphi(N) = d + r_1 N - r_1(p+q-1)$
- **Effect of the blinding:**

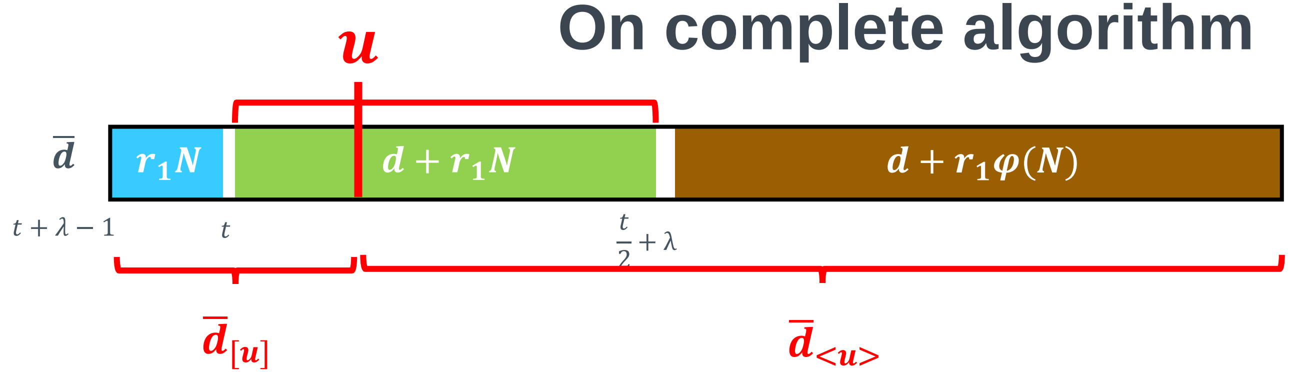
$$\bar{d} = \sum_{i=t}^{t+\lambda-1} 2^i.(r_1N)_i + \sum_{i=\frac{t}{2}+\lambda}^{t-1} 2^i.(d+r_1N)_i + \sum_{i=0}^{\frac{t}{2}+\lambda-1} 2^i.\left(d+r_1\varphi(N)\right)_i$$



Source: Berzati et al. CHES 2010

Fault attack

On complete algorithm



- <u>MSB part of d</u>
- Let $u \in \left[\frac{t}{2} + \lambda, t\right]$
- Let $\overline{d}_{[u]} = \sum_{i=t-u}^{t+\lambda-1} 2^i . \overline{d}_i$ and $\overline{d}_{<u>} = \sum_{i=0}^{t-u-1} 2^i . \overline{d}_i$

  $\overline{d}_i \rightarrow H_{(i \bmod W)}$

- Approximation of $\overline{d}_{[u]}$:

$$\overline{d}_{[u]} \approx \sum_{i=t-u}^{t+\lambda-1} 2^i . (d + r_1 N)_i$$

$$\approx d_{\text{known}} + \sum_{i=t-v-u}^{t-v-1} 2^i . d_i + \sum_{i=t-v-u}^{t+\lambda-1} 2^i . (r_1 N)_i + \text{carry}$$

Fault attack

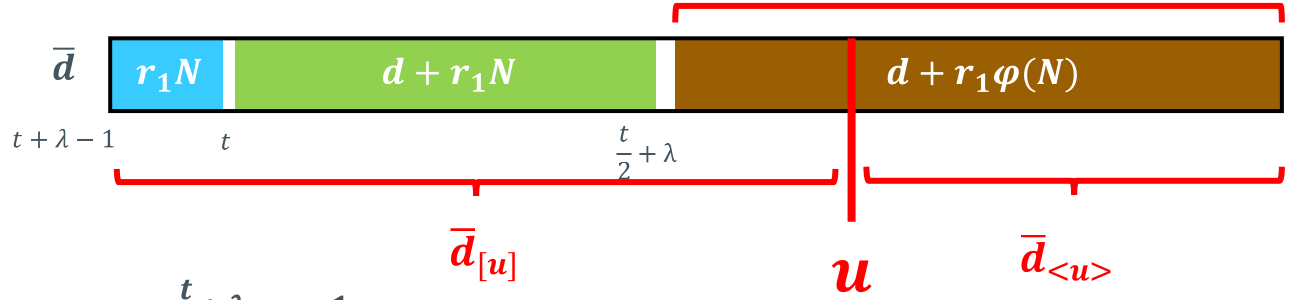
On complete algorithm

- <u>MSB part of d</u>
- Guesses on u bits of d, W bits of H and λ bits of r_1
- → Complete guessed exponent

- Validate guess by checking:

$$\breve{S}_u ?= m^{\overline{d}_{[u]}+\overline{d}_{<u>}} \bmod N$$

On complete algorithm



- <u>LSB part of d</u>
- Let $u \in \left[0, \frac{t}{2} + \lambda\right]$
- As previously, $\bar{d}_{<u>} = \sum_{i=0}^{\frac{t}{2}+\lambda-u-1} 2^i . H_{(i \bmod W)}$
- Approximation of $\bar{d}_{[u]}$:

$$\bar{d}_{[u]} \approx \sum_{i=\frac{t}{2}+\lambda-u}^{t+\lambda-1} 2^i . (d + r_1\varphi(N))_i$$

$$\approx d_{\text{known}} + \sum_{i=\frac{t}{2}+\lambda-v-u}^{\frac{t}{2}+\lambda-v-1} 2^i . \delta_i + \sum_{i=\frac{t}{2}+\lambda-v-u}^{t+\lambda-1} 2^i . (r_1 N)_i + \text{carry}$$

with $\delta_i = (d - (r_1(p + q - 1))_i$

Fault attack

On complete algorithm

- <u>LSB part of d</u>
- Guesses on u bits of d, W bits of H and λ bits of r_1
- → Complete guessed exponent

- Validate guess by checking:

$$\breve{S}_u ?= m^{\overline{d}_{[u]}+\overline{d}_{<u>}} \bmod N$$

- Here, we recover u bits of δ and not of d
- As d and $(p+q-1)$ are fixed values between exponentiations
- We can retrieve d by faulting multiple times at the instant u

On complete algorithm

- Computational complexity:

$$\mathcal{C} = \mathcal{O}\left(\frac{2^{(u+W+\lambda)}.t}{u}\right)$$

- Number of faults:

$$\mathcal{F} = \mathcal{O}\left(\frac{t}{u}\right)$$

- Size of r_2 does not impact the attack, only the size W

- Applicability of the attack depends on the size λ of r_1

Agenda

1. What is a combined attack?
2. Algorithm of Schmidt et al.
3. Fault attack
    1. On simplified algorithm
    2. On complete algorithm
4. Combined attack
5. Improved algorithm
6. Conclusion

Combined attack

Fault injection and differential side-channel

Algorithm 1 Schmidt *et al.* [20, Alg. 3] left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod Nr_2$
$R_1 \leftarrow i \cdot m \bmod Nr_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \cdot R_k \bmod Nr_2$
  if $(R_0 = 0)$ or $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [1, \ldots, 1]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Fault

→ Remove the exponent blinding:

- bypass (NOP) the call to this function

or

- bypass the multiplication by r_1

Fault injection and differential side-channel

- **Execute the calculation many times (k) on the attacked device**
  - **At each execution i**
    - Fault the step 6. execution
    - Acquire and store the side-channel trace C_i of the exponentiation
  - **Apply with these k curves the differential analysis from Amiel et al.**
    - *Distinguishing multiplications from squaring operations* – SAC 2008.
  - **Allow to recover the secret exponent d**

- ***Attack success depends essentially on the feasibility of the fault injection on the attacked hardware***

Combined attack

Fault injection and template analysis

- **Template pre-processing phase required on the attack device**
  - **Need to store many curves of**
    - The squaring operation $R_0 \times R_0$ with random values R_0
    - The multiplication operation $R_0 \times R_1$ with random values R_0 and R_1
- **Execute the calculation many times (u) on the attacked device**
  - **At each execution i**
    - Fault the step 6. execution
    - Acquire and store the side-channel trace C_i of the exponentiation
  - **Apply with these u curves the Template analysis from Hanley et al.**
    - *Using templates to distinguish multiplications from squaring operations -* International Journal of Information Security, 10. 2011.
  - **Allows to recover the secret exponent d**
- ***Attack success depends essentially on the feasibility of the fault injection on the attacked hardware***

Agenda

1. What is a combined attack?
2. Algorithm of Schmidt et al.
3. Fault attack
    1. On simplified algorithm
    2. On complete algorithm
4. Combined attack
5. **Improved algorithm**
6. **Conclusion**

Improved algorithm

Algorithm 2 Improved Schmidt *et al.* left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod N r_2$
$R_1 \leftarrow i \cdot m \bmod N r_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$ (optional)
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
for $i = 0$ **to** $l - 1$ **do**
  $w_i \leftarrow \mathsf{random}(1, 2^W - 1)$
end for
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $r_3 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
  $R_2 \leftarrow R_k + r_3 \cdot N \bmod N r_2$
  $R_0 \leftarrow R_0 \cdot R_2 \bmod N r_2$
  if $(R_0 = 0)$ **or** $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [w_{l-1}, \ldots, w_0]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Replace constant 1 by different w-bit random values for infective operation on exponent

Improved algorithm

Algorithm 2 Improved Schmidt *et al.* left-to-right exponentiation.

Input: $d = (d_{t-1}, \ldots, d_0)_2, m \in \mathbb{Z}_N, N$ and block length W.
Output: $m^d \bmod N$

$r_1 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$r_2 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
$i \leftarrow (r_2^{-1} \bmod N) \cdot r_2$
$R_0 \leftarrow i \cdot 1 \bmod Nr_2$
$R_1 \leftarrow i \cdot m \bmod Nr_2$
$\bar{d} \leftarrow d + r_1 \cdot \varphi(N)$ (optional)
$[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [\psi_0(\bar{d}^{(l-1)}), \ldots, \psi_0(\bar{d}^{(0)})]$
for $i = 0$ **to** $l - 1$ **do**
  $w_i \leftarrow \mathsf{random}(1, 2^W - 1)$
end for
$k \leftarrow 0$
$j \leftarrow \mathsf{bitlength}(\tilde{d}) - 1$
while $j \geq 0$ **do**
  $r_3 \leftarrow \mathsf{random}(1, 2^\lambda - 1)$
  $R_2 \leftarrow R_k + r_3 \cdot N \bmod Nr_2$
  $R_0 \leftarrow R_0 \cdot R_2 \bmod Nr_2$
  if $(R_0 = 0)$ **or** $(R_1 = 0)$ **then**
    $[\tilde{d}^{(l-1)}, \ldots, \tilde{d}^{(0)}] \leftarrow [w_{l-1}, \ldots, w_0]$
  end if
  $\hat{d} \leftarrow \psi^{-1}_{(R_0 + R_1 \bmod r_2)}(\tilde{d}^{(\lfloor j/W \rfloor)})$
  $k \leftarrow k \oplus \mathsf{bit}(\hat{d}, j \bmod W)$
  $j \leftarrow j - \neg k$
end while
$c \leftarrow R_0 \bmod N$
return c

Replace constant 1 by different w-bit random values for infective operation on exponent

Replace squaring operation by multiplications:

$$R_0 \times R_0 \rightarrow R_0 \times (R_0 + Nr_3) \bmod Nr_2$$

→ Combined attacks cannot apply anymore.

Agenda

1. What is a combined attack?
2. Algorithm of Schmidt et al.
3. Fault attack
    1. On simplified algorithm
    2. On complete algorithm
4. Combined attack
5. Improved algorithm
6. **Conclusion**

Conclusion

We have presented two new attacks:

- First: a simple fault injection technique
  - Apply with and without the exponent blinding countermeasure
  - Allow to recover the secret exponent with few faulty ciphertexts
- Second: combined attacks
  - Fault injection and Amiel et al. differential analysis
  - Fault injection and Hanley et al. template analysis
- We have presented an improved version of the Schmidt et al. algorithm that thwarts those attacks.

Thanks for your attention …