

Fault Resistant RSA Implementation

Author: Christophe GIRAUD

Speaker: Régis BEVAN

OBERTHUR CARD SYSTEMS

Overview

- Introduction
- Differential Fault Analysis (DFA)
- Simple Power Analysis (SPA)
- Previous SPA and DFA-proof RSA
- A new countermeasure
- Conclusion

Introduction

- RSA signature: compute $S_d = m^d \bmod p \cdot q$

To speed up, use Chinese Remainder Theorem (CRT):

$$S_{d_p} = m^{d \bmod p-1} \bmod p$$

$$S_{d_q} = m^{d \bmod q-1} \bmod q$$

$$S_d = ((S_{d_q} - S_{d_p}) \times p^{-1} \bmod q) \times p + S_{d_p} \bmod (pq)$$



Fault attacks on CRT-RSA

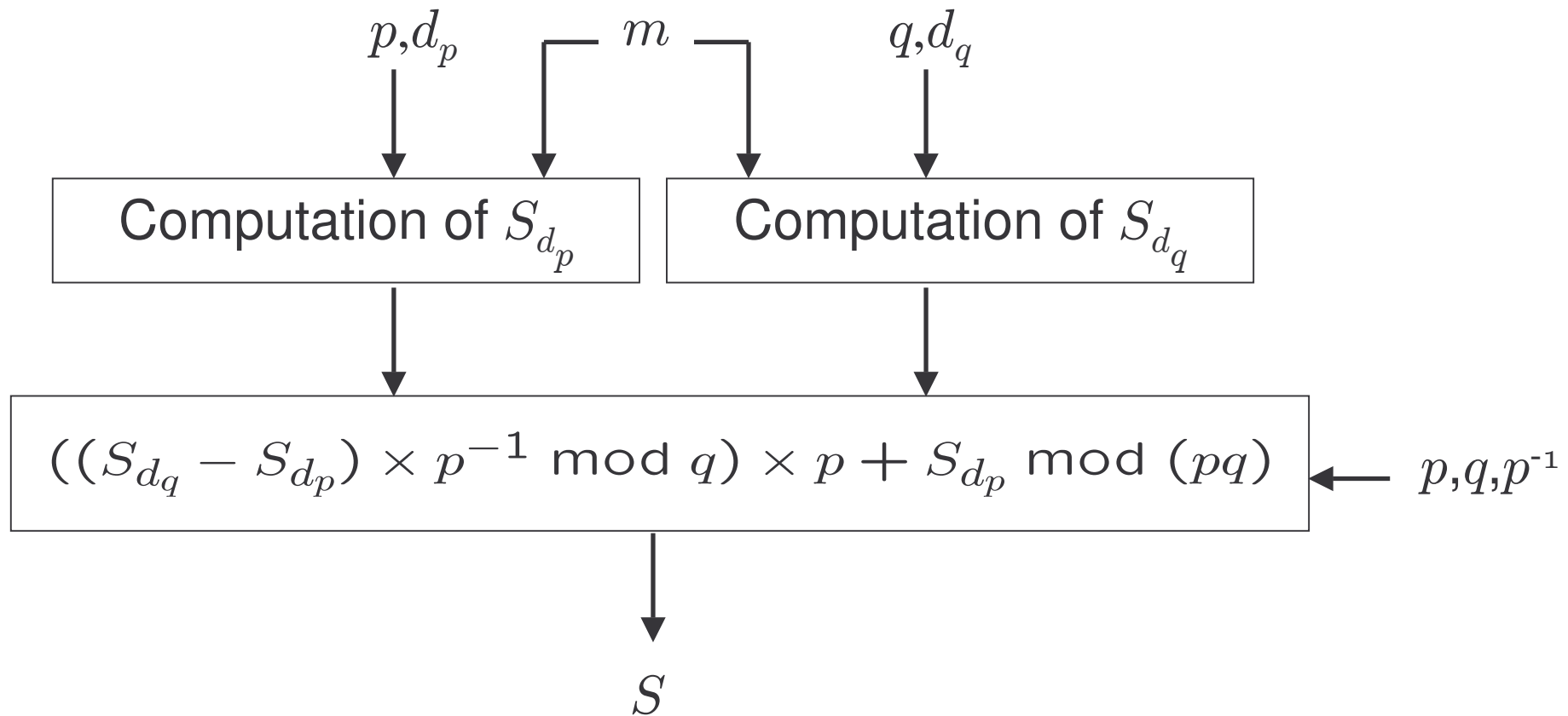
- Only 1 faulty signature \hat{S} to recover the secret key:

$$p \text{ or } q = \gcd(\hat{S}^{e-m} \bmod N)$$

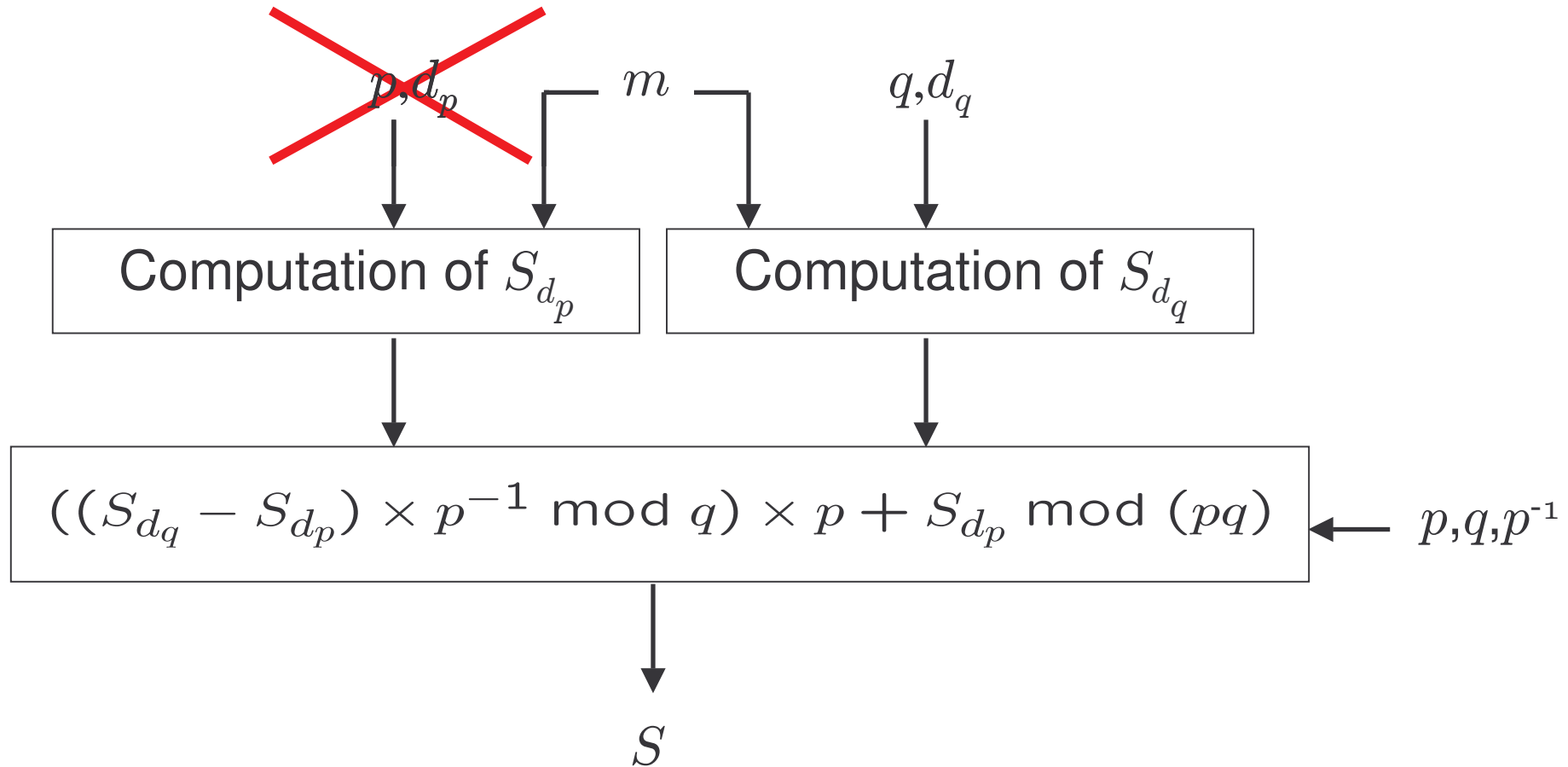
- Faulty signature \hat{S} “usable” :

$$\Leftrightarrow \begin{cases} \hat{S} \bmod p = S \bmod p \text{ and } \hat{S} \bmod q \neq S \bmod q \\ \text{or} \\ \hat{S} \bmod p \neq S \bmod p \text{ and } \hat{S} \bmod q = S \bmod q \end{cases}$$

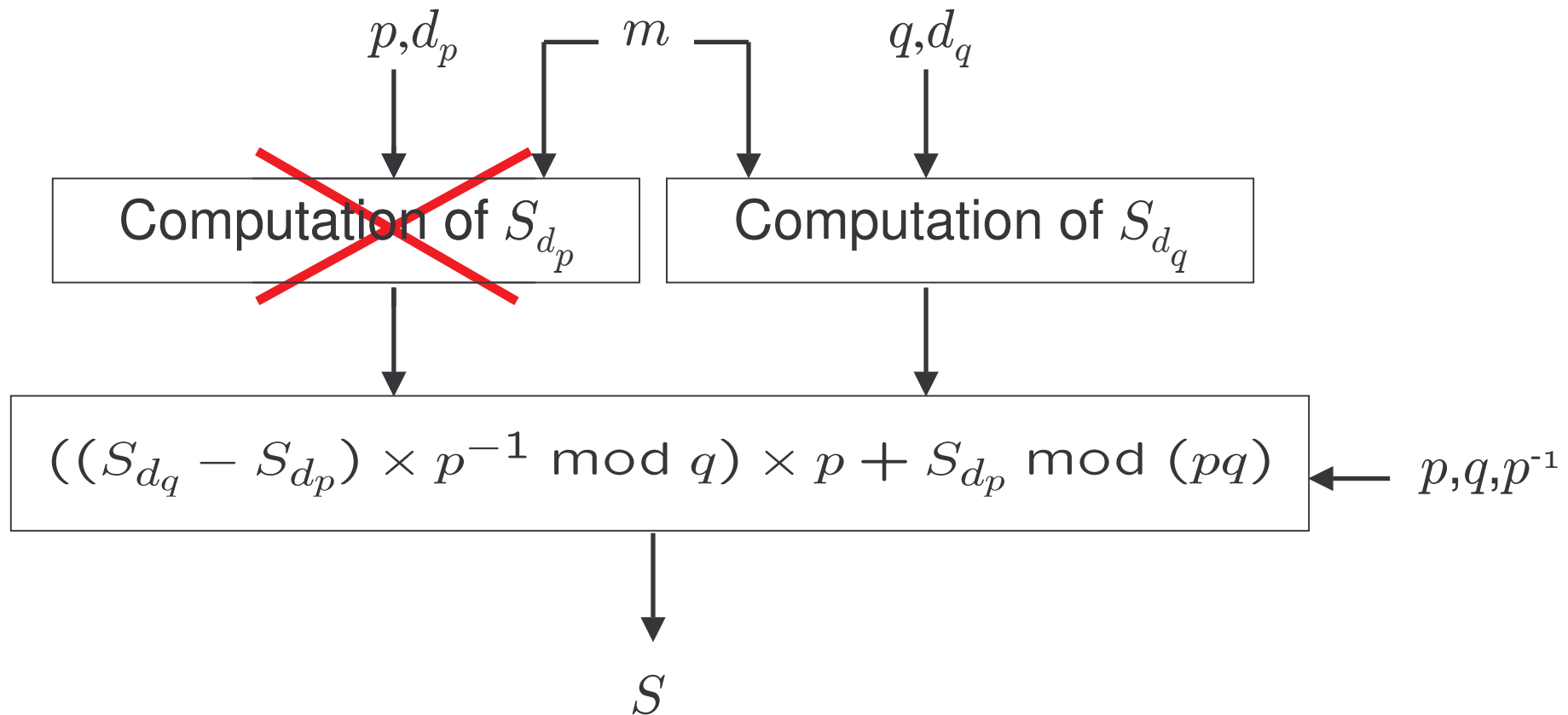
Ways to induce a “usable” fault



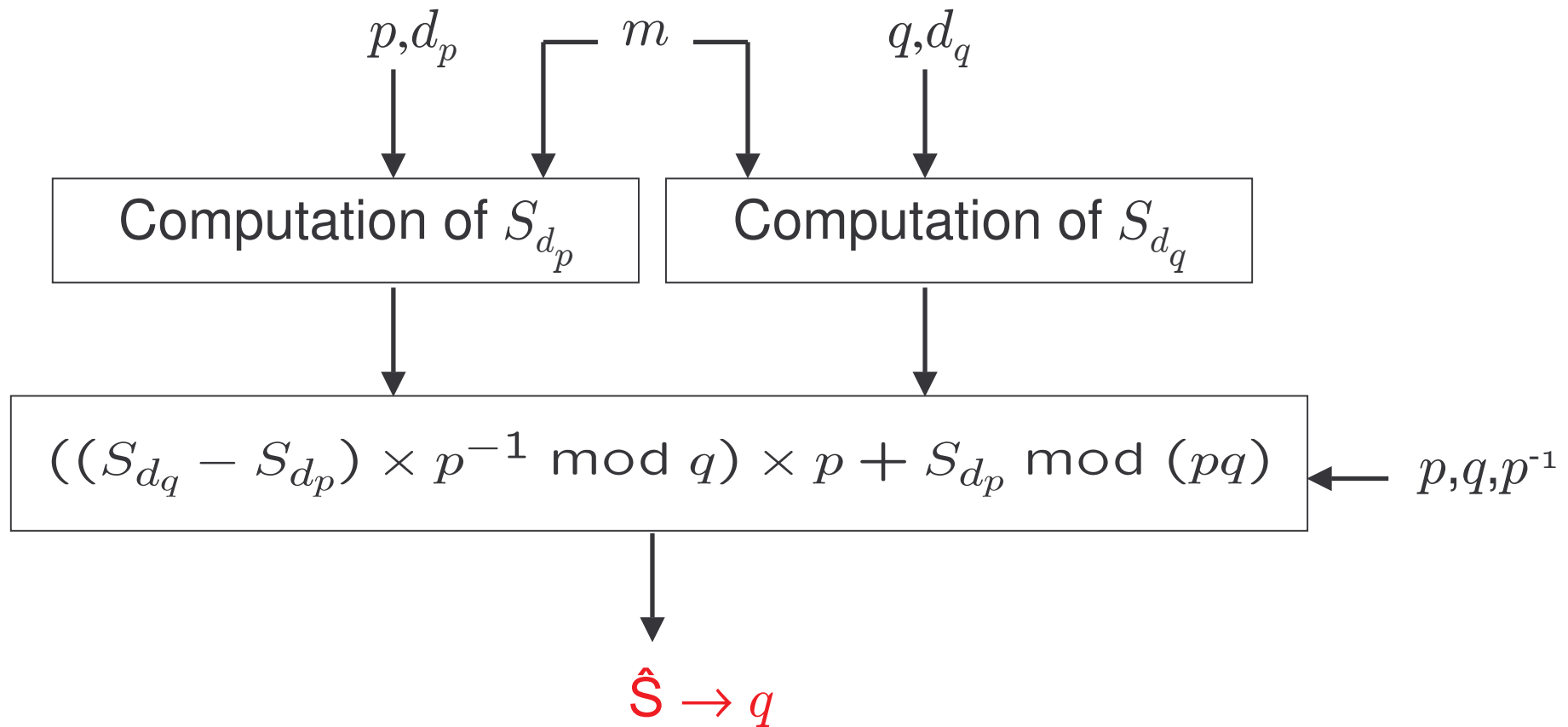
Ways to induce a “usable” fault



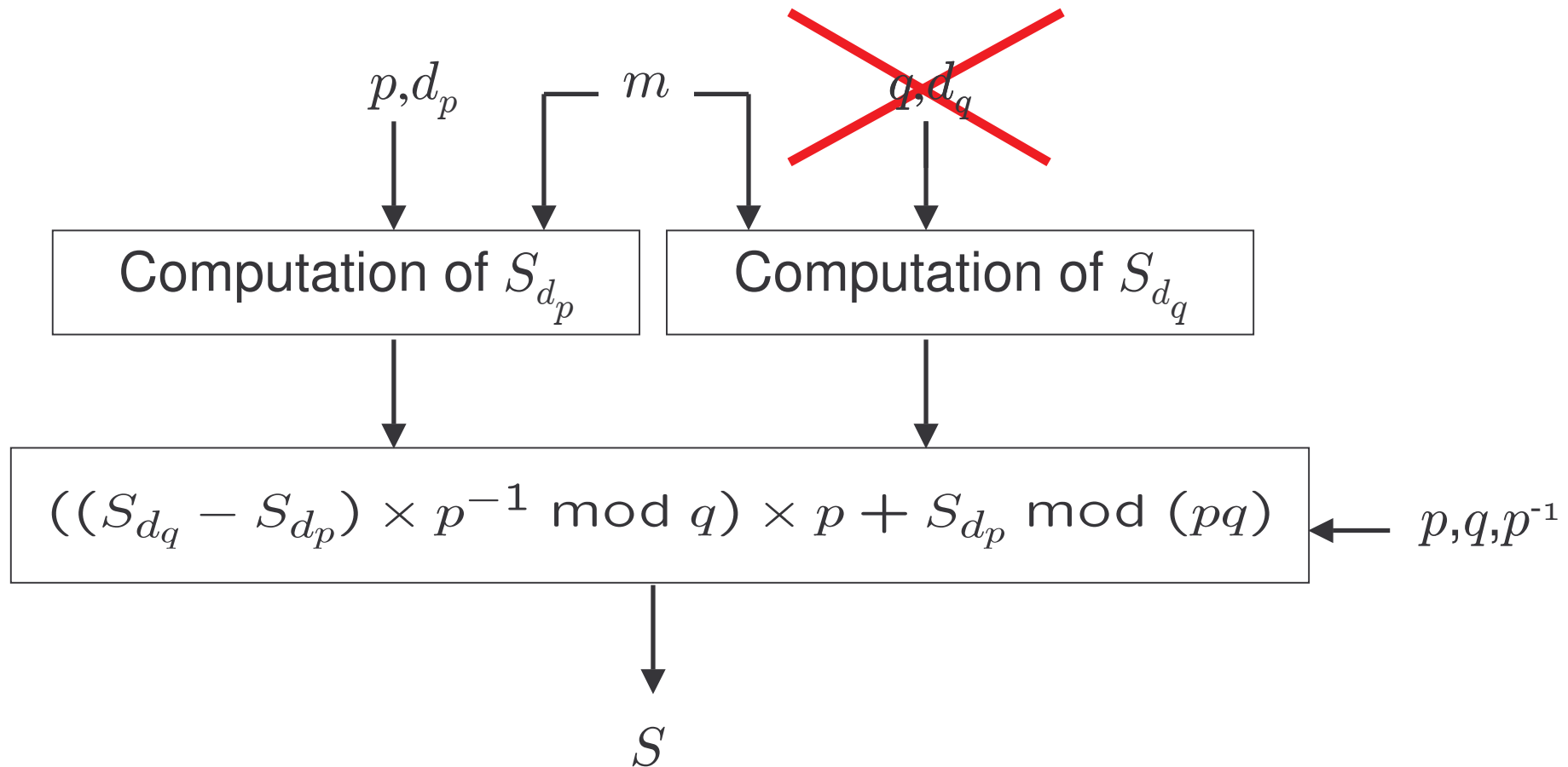
Ways to induce a “usable” fault



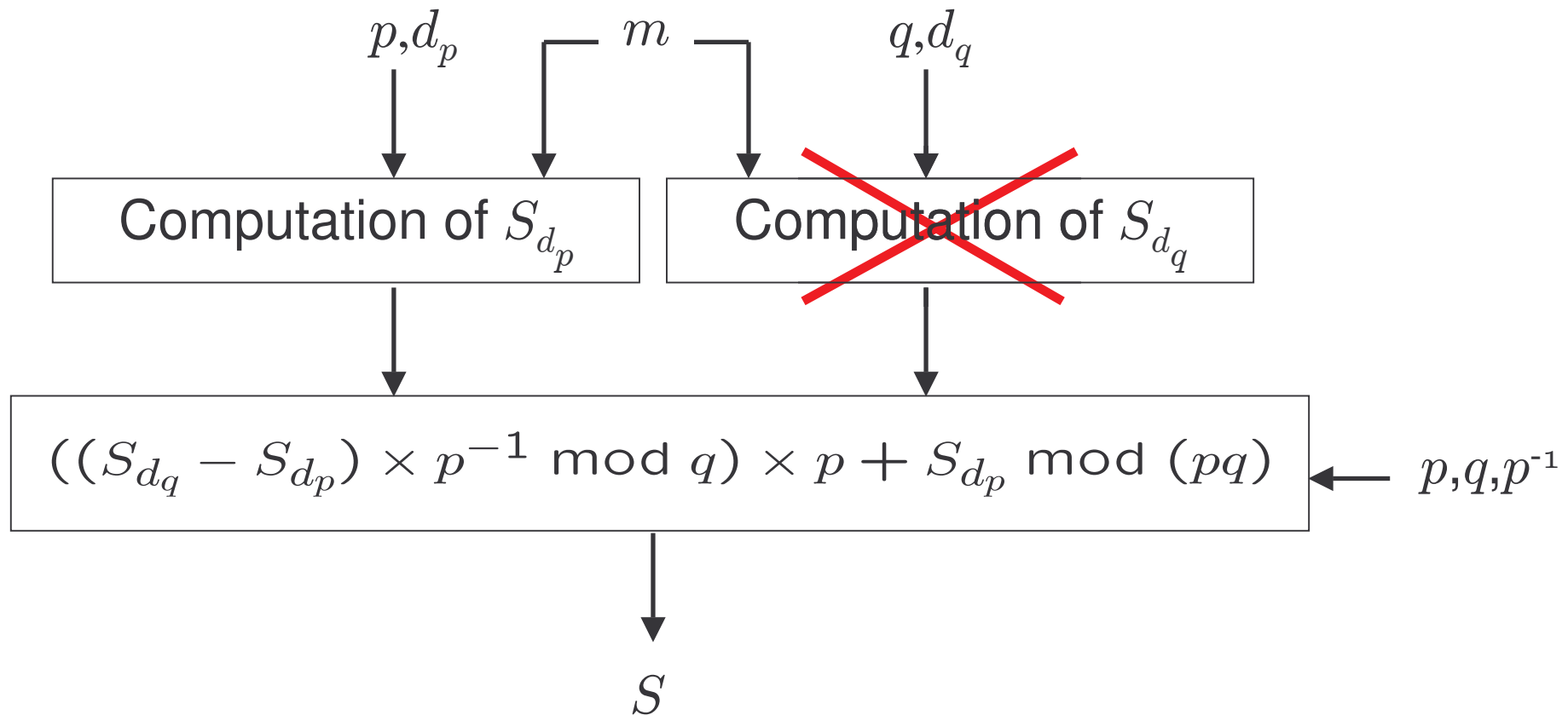
Ways to induce a “usable” fault



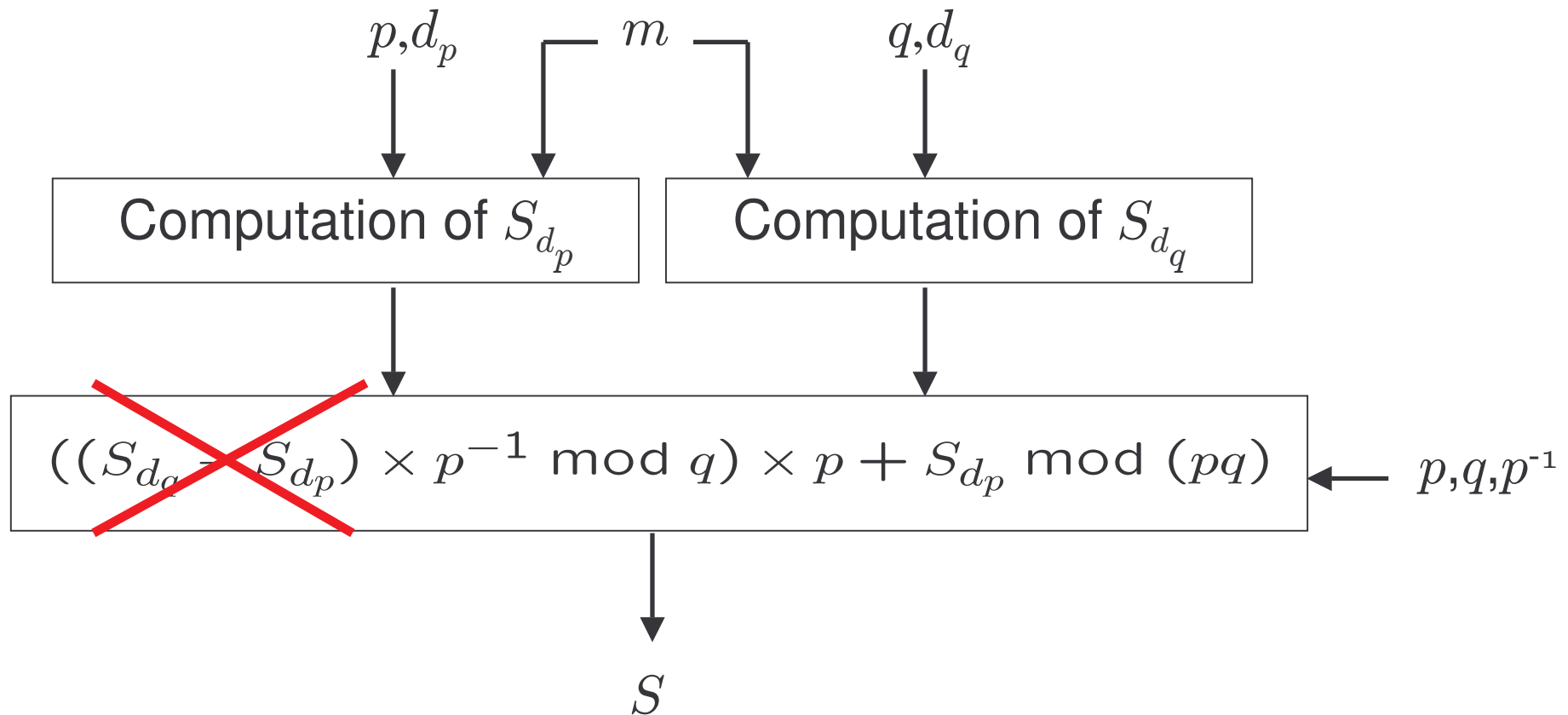
Ways to induce a “usable” fault



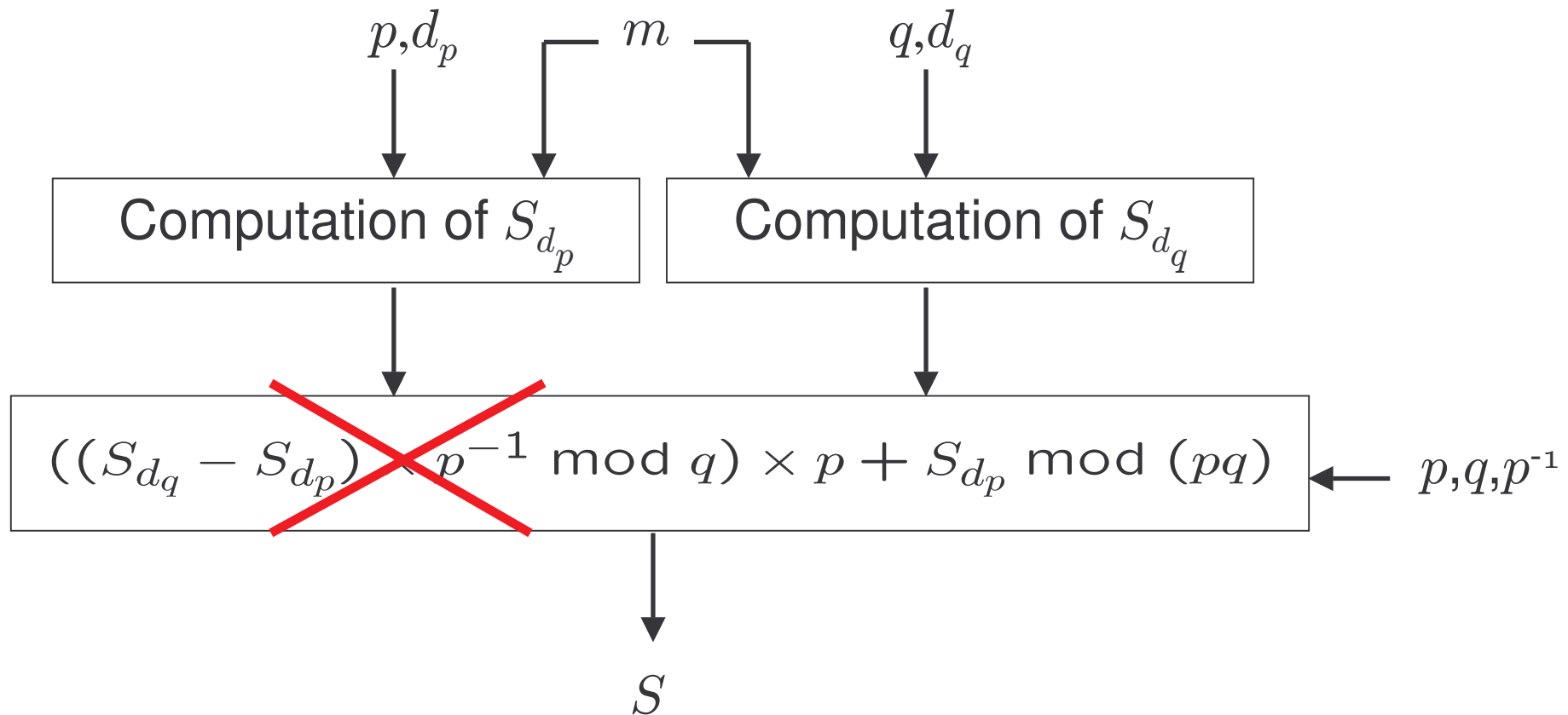
Ways to induce a “usable” fault



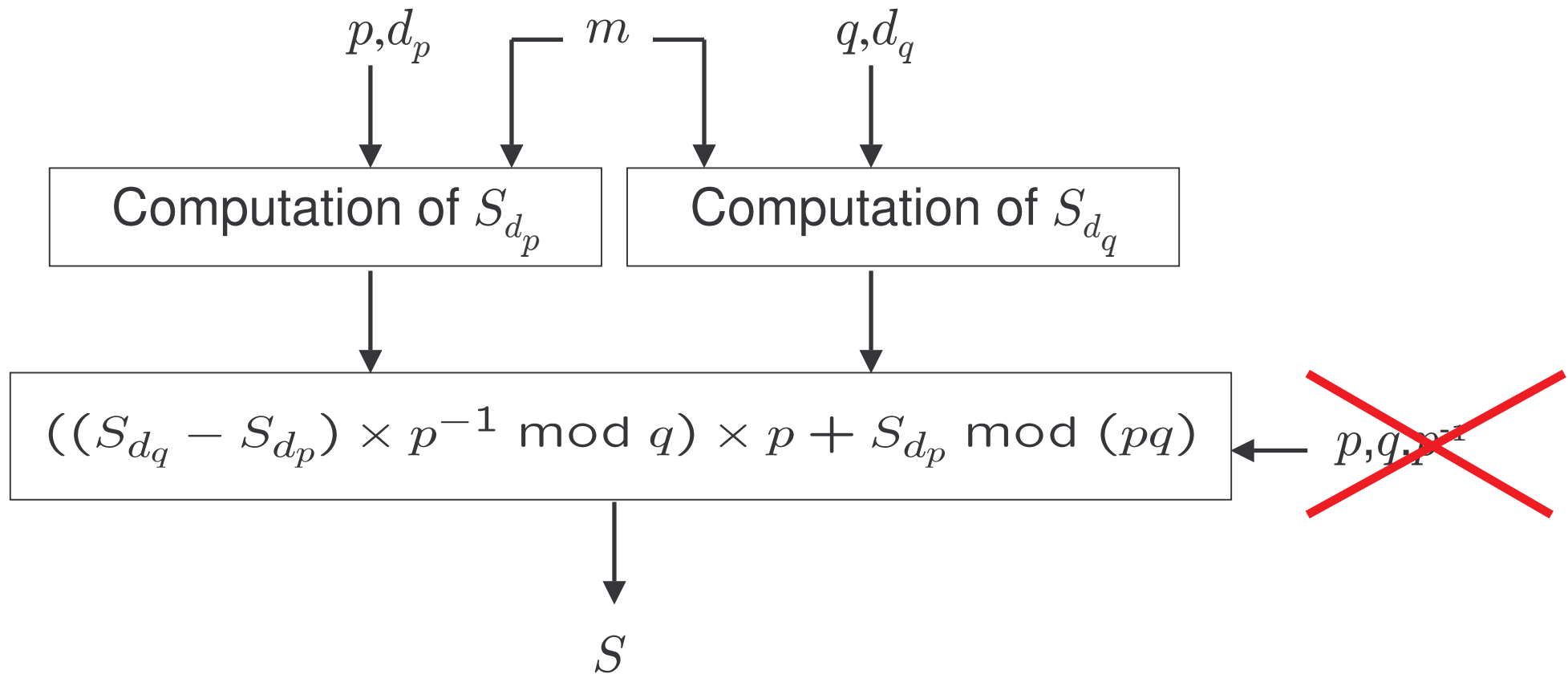
Ways to induce a “usable” fault



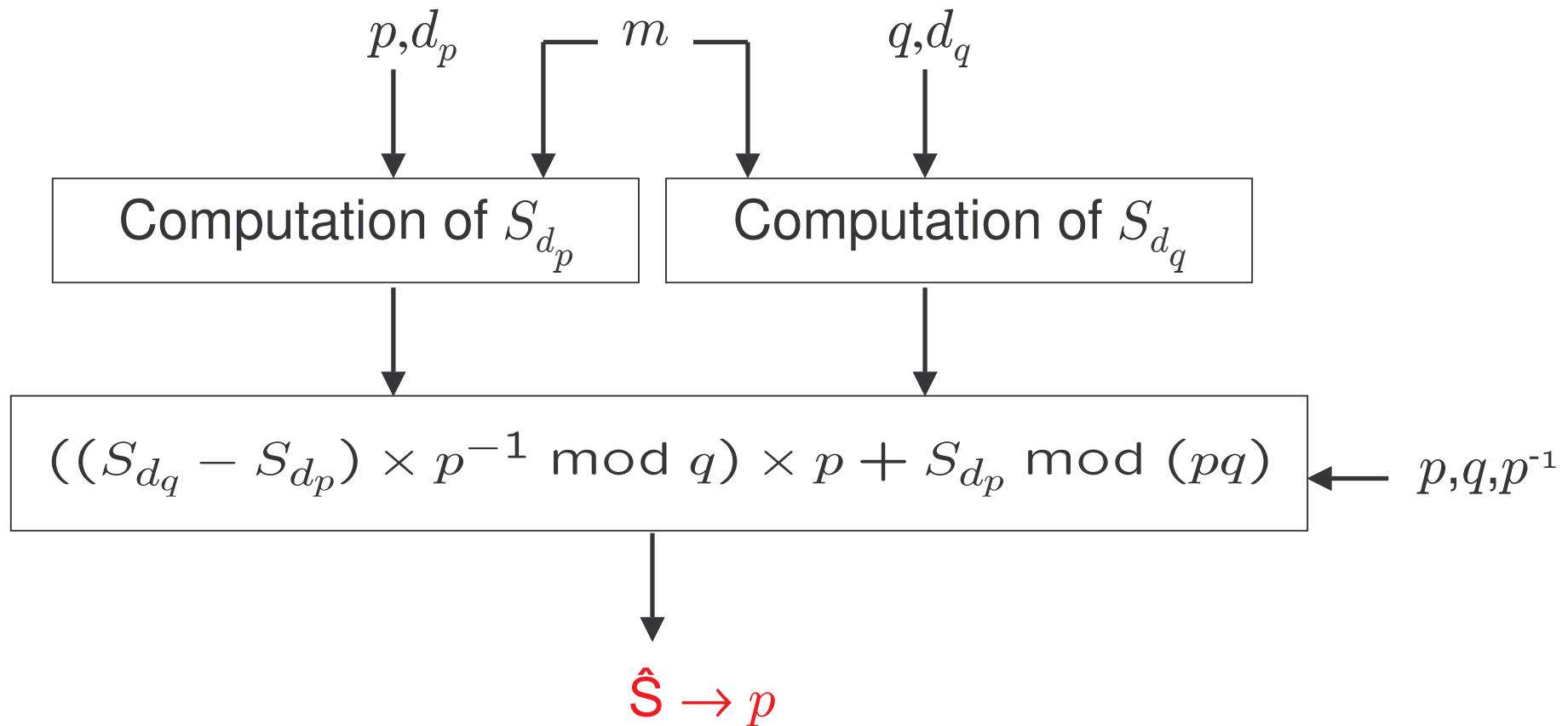
Ways to induce a “usable” fault



Ways to induce a “usable” fault



Ways to induce a “usable” fault



Former countermeasures

Former countermeasures

- Check with e

Former countermeasures

- Check with e
- Shamir (check)

Former countermeasures

- Check with e
- Shamir (check) → broken by Aumüller et al.

Former countermeasures

- Check with e
- Shamir (check) → broken by Aumüller et al.
- Aumüller et al. (check)

Former countermeasures

- Check with e
- Shamir (check) → broken by Aumüller et al.
- Aumüller et al. (check)
- Yen et al. (infective)

Former countermeasures

- Check with e
- Shamir (check) → broken by Aumüller et al.
- Aumüller et al. (check)
- Yen et al. (infective) → broken by Blömer et al. , Yen and Kim

Former countermeasures

- Check with e
- Shamir (check) → broken by Aumüller et al.
- Aumüller et al. (check)
- Yen et al. (infective) → broken by Blömer et al. , Yen and Kim
- Blömer et al. (infective)

Former countermeasures

- Check with e
- Shamir (check) → broken by Aumüller et al.
- Aumüller et al. (check)
- Yen et al. (infective) → broken by Blömer et al. , Yen and Kim
- Blömer et al. (infective) → broken by Wagner

Former countermeasures

- Check with e
- Shamir (check) → broken by Aumüller et al.
- Aumüller et al. (check)
- Yen et al. (infective) → broken by Blömer et al. , Yen and Kim
- Blömer et al. (infective) → broken by Wagner



Former countermeasures

- Check with e → not always available, overhead if e large
- Shamir (check) → broken by Aumüller et al.
- Aumüller et al. (check)
- Yen et al. (infective) → broken by Blömer et al. , Yen and Kim
- Blömer et al. (infective) → broken by Wagner

Former countermeasures

- Check with e → not always available, overhead if e large
- Shamir (check) → broken by Aumüller et al.
- Aumüller et al. (check) → broken by Blömer et al. , Yen and Kim
- Yen et al. (infective) → broken by Wagner
- Blömer et al. (infective)

Former countermeasures

- Check with e → not always available, overhead if e large
- Shamir (check) → broken by Aumüller et al.
- Aumüller et al. (check) → 20% overhead for a 1024-bit RSA
- Yen et al. (infective) → broken by Blömer et al. , Yen and Kim
- Blömer et al. (infective) → broken by Wagner

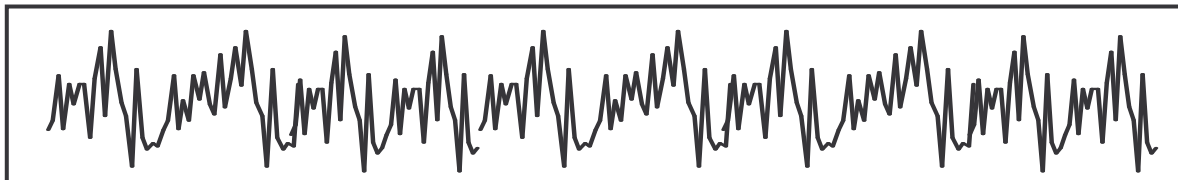


Simple Power Analysis against RSA

- Computation of S_{d_p} :
 $a_0 \leftarrow m \bmod p$
for i from $n - 2$ to 0 do
 $a_0 \leftarrow a_0^2 \bmod p$
 if($d_p[i] = 1$) then $a_0 \leftarrow m * a_0 \bmod p$
return(a_0)

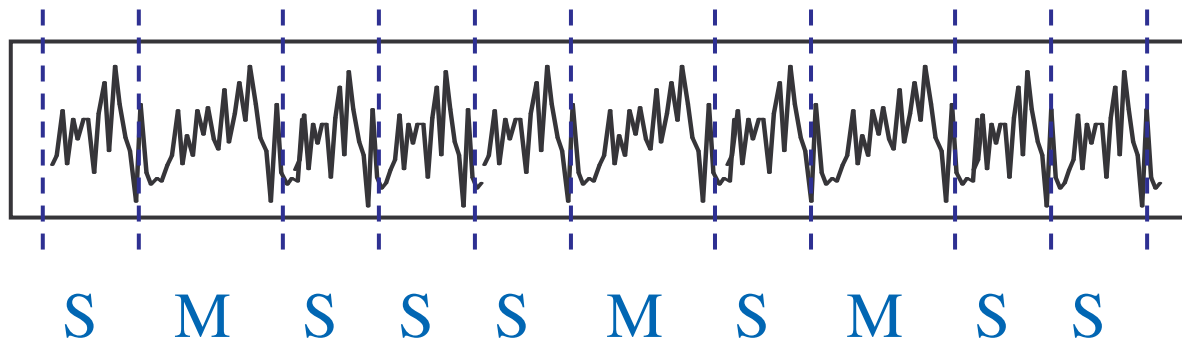
Simple Power Analysis against RSA

- Computation of S_{d_p} :
 $a_0 \leftarrow m \bmod p$
for i from $n - 2$ to 0 do
 $a_0 \leftarrow a_0^2 \bmod p$
 if($d_p[i] = 1$) then $a_0 \leftarrow m * a_0 \bmod p$
return(a_0)



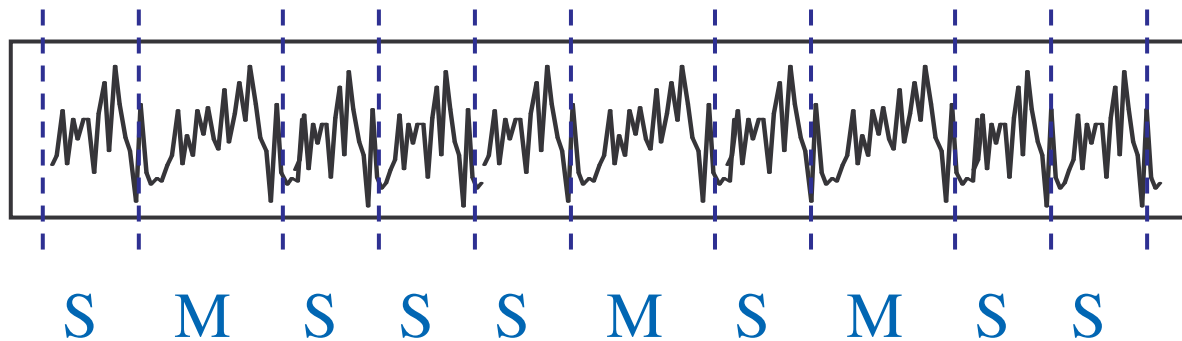
Simple Power Analysis against RSA

- Computation of S_{d_p} :
 $a_0 \leftarrow m \bmod p$
for i from $n - 2$ to 0 do
 $a_0 \leftarrow a_0^2 \bmod p$
 if($d_p[i] = 1$) then $a_0 \leftarrow m * a_0 \bmod p$
return(a_0)



Simple Power Analysis against RSA

- Computation of S_{d_p} :
 $a_0 \leftarrow m \bmod p$
 for i from $n - 2$ to 0 do
 $a_0 \leftarrow a_0^2 \bmod p$
 if($d_p[i] = 1$) then $a_0 \leftarrow m * a_0 \bmod p$
 return(a_0)



$\Rightarrow d_p = 100110\dots$

SPA-resistant exponentiation

Simplest countermeasure:

“Square & Multiply always”

$$a_0 \leftarrow m \bmod p$$

for i from $n - 2$ to 0 do

$$a_0 \leftarrow a_0^2 \bmod p$$

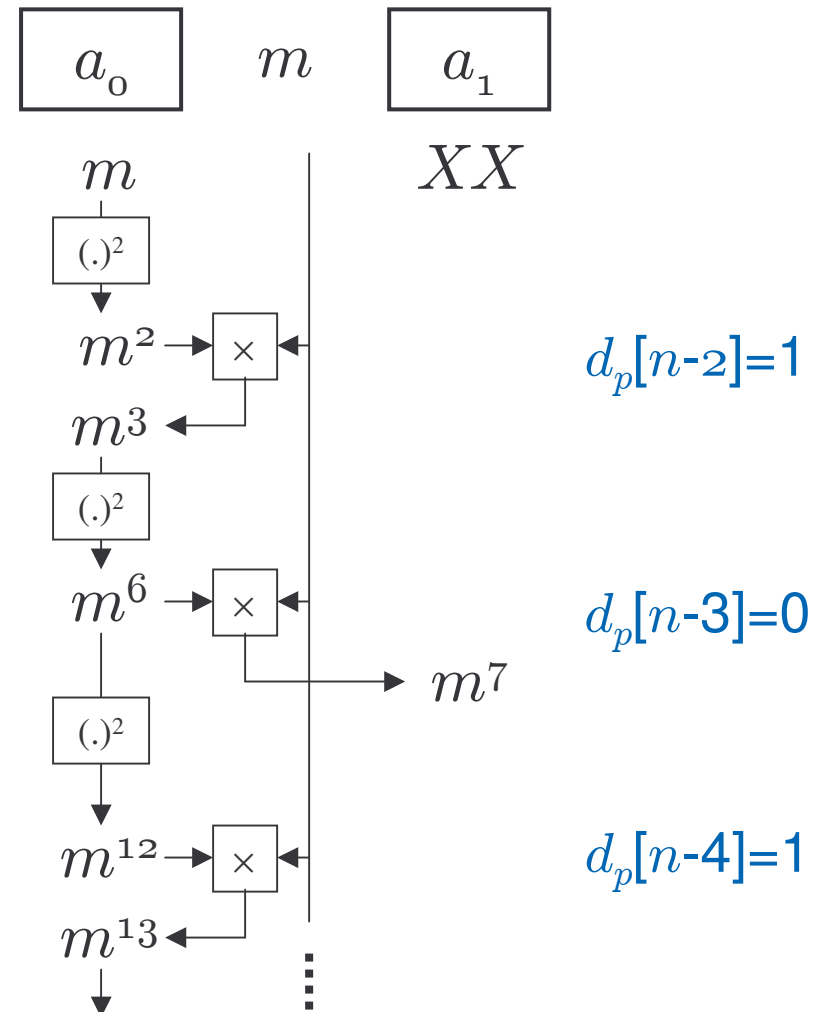
if ($d_p[i] == 1$)

$$a_0 \leftarrow m * a_0 \bmod p$$

else

$$a_1 \leftarrow m * a_0 \bmod p$$

return(a_0)



Safe-error Attack

Safe-error Attack

- “S & M always“

Safe-error Attack

- “S & M always“
⇒ SPA-proof

Safe-error Attack

- “S & M always“
⇒ SPA-proof
- “Check with e or
Aumüller et al.
method”

Safe-error Attack

- “S & M always“
⇒ SPA-proof
- “Check with e or
Aumüller et al.
method”
⇒ DFA-proof

Safe-error Attack

- “S & M always“
⇒ SPA-proof
- “Check with e or
Aumüller et al.
method”
⇒ DFA-proof
- “S & M always” +
“check with e or
Aumüller et al.
method”

Safe-error Attack

- “S & M always“

⇒ SPA-proof

- “Check with e or Aumüller et al. method”

⇒ DFA-proof

- “S & M always” + “check with e or Aumüller et al. method”

⇒ **NOT SECURE!**
Safe-errors attack
(Joye and Yen)

Safe-error Attack

- “S & M always“

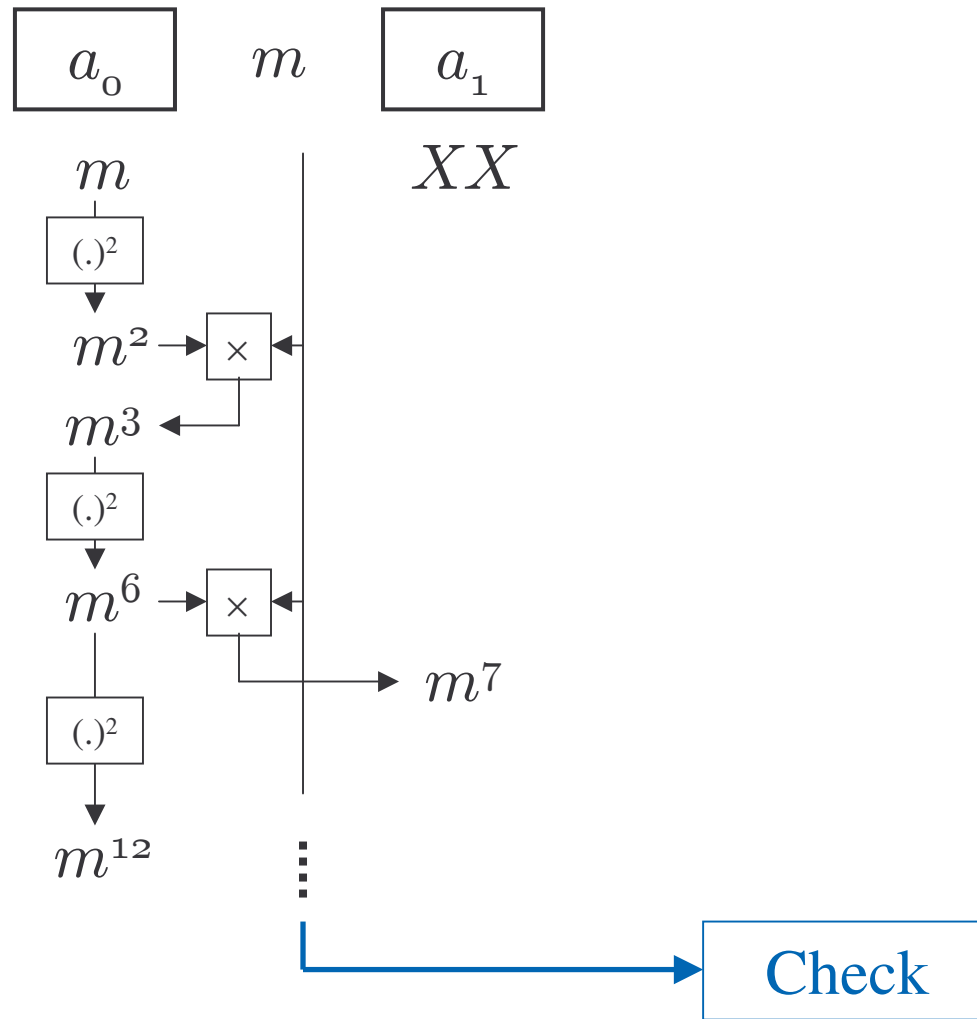
⇒ SPA-proof

- “Check with e or Aumüller et al. method“

⇒ DFA-proof

- “S & M always“ + “check with e or Aumüller et al. method“

⇒ NOT SECURE!
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

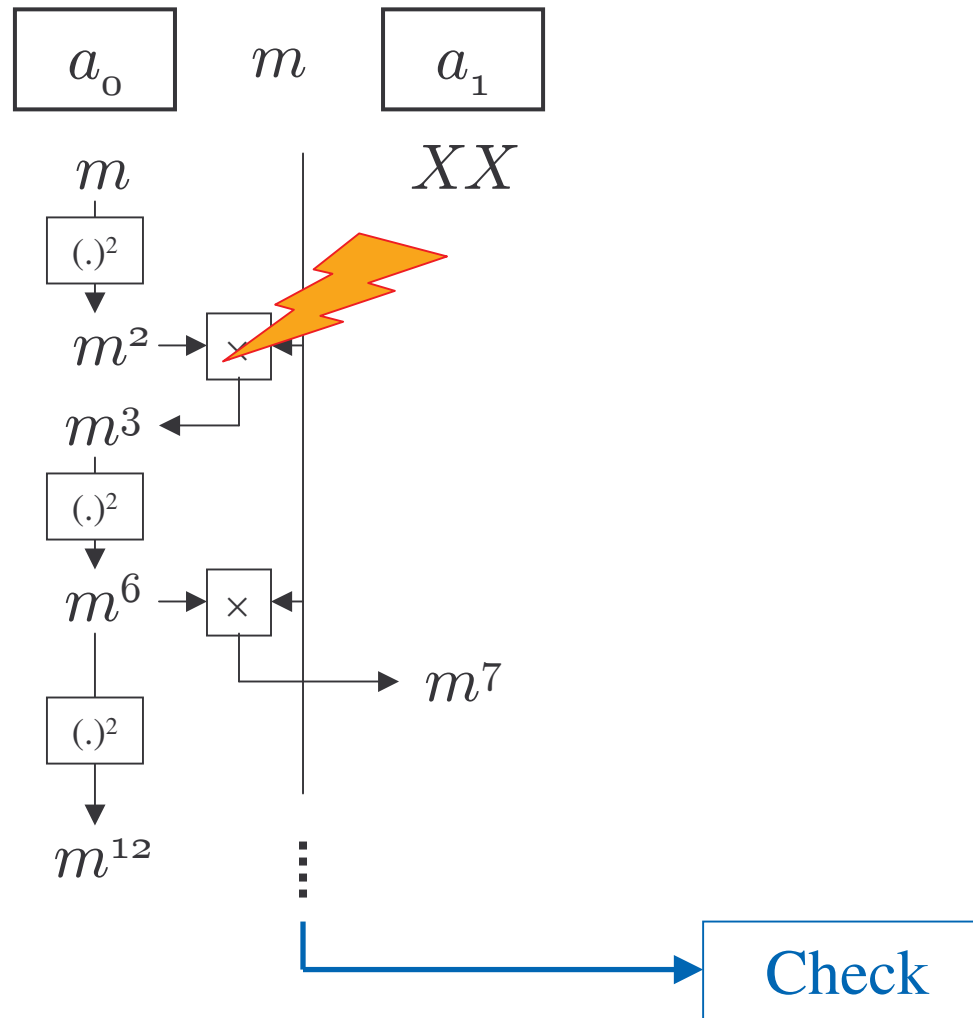
⇒ SPA-proof

- “Check with e or Aumüller et al. method”

⇒ DFA-proof

- “S & M always” + “check with e or Aumüller et al. method”

⇒ NOT SECURE!
 Safe-errors attack
 (Joye and Yen)



Safe-error Attack

- “S & M always“

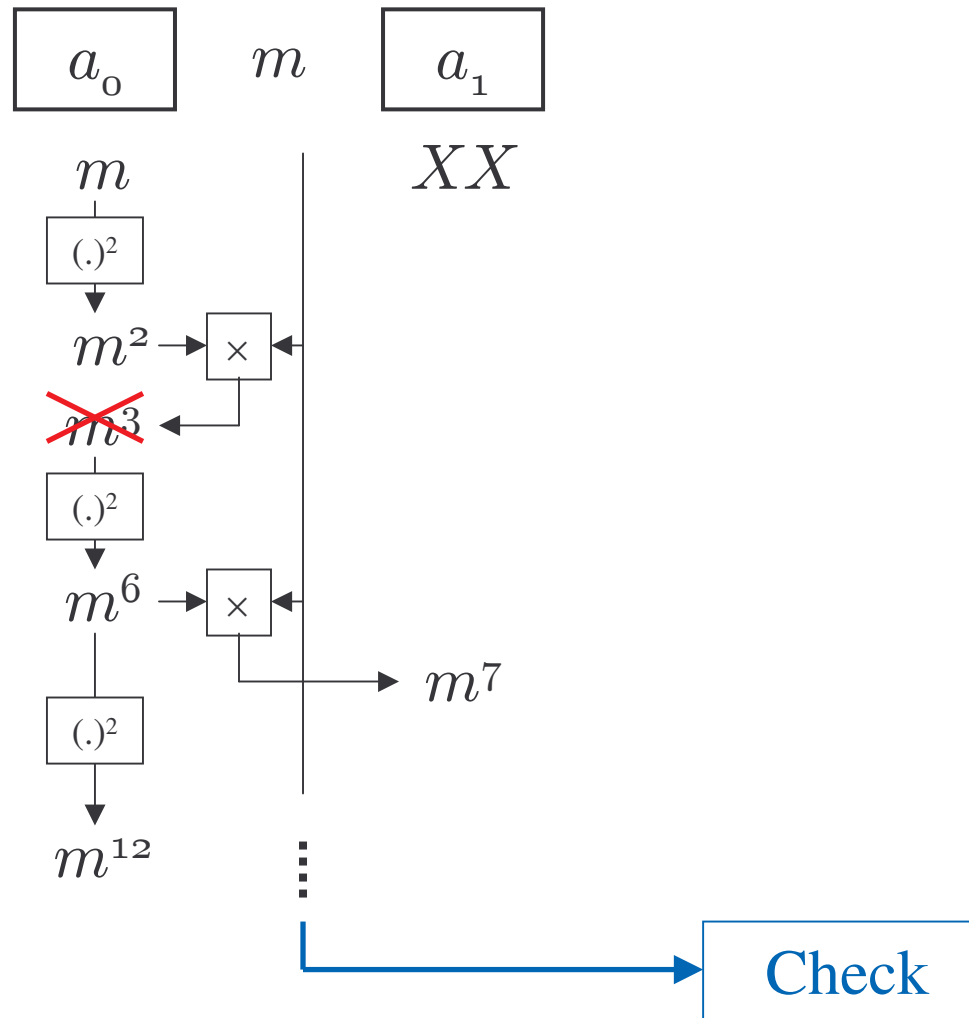
⇒ SPA-proof

- “Check with e or Aumüller et al. method”

⇒ DFA-proof

- “S & M always” + “check with e or Aumüller et al. method”

⇒ NOT SECURE!
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

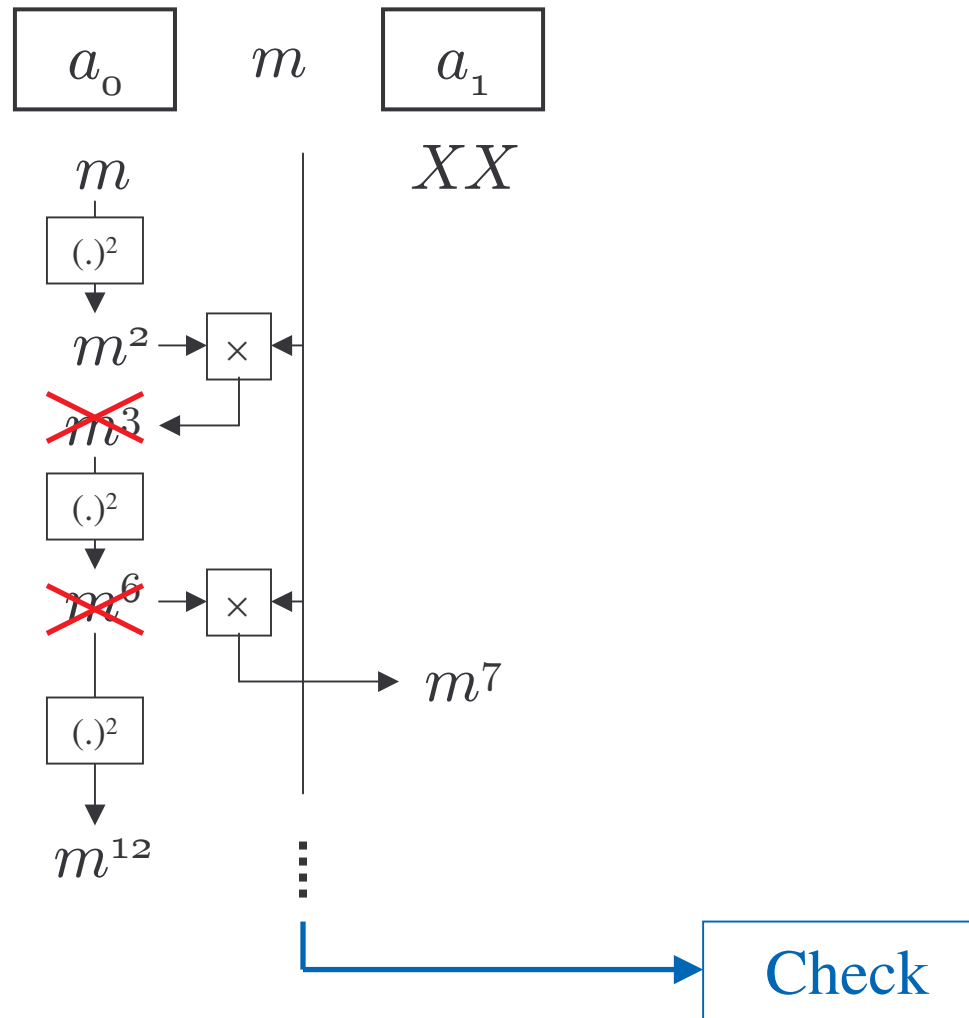
⇒ SPA-proof

- “Check with e or Aumüller et al. method“

⇒ DFA-proof

- “S & M always“ + “check with e or Aumüller et al. method“

⇒ **NOT SECURE!**
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

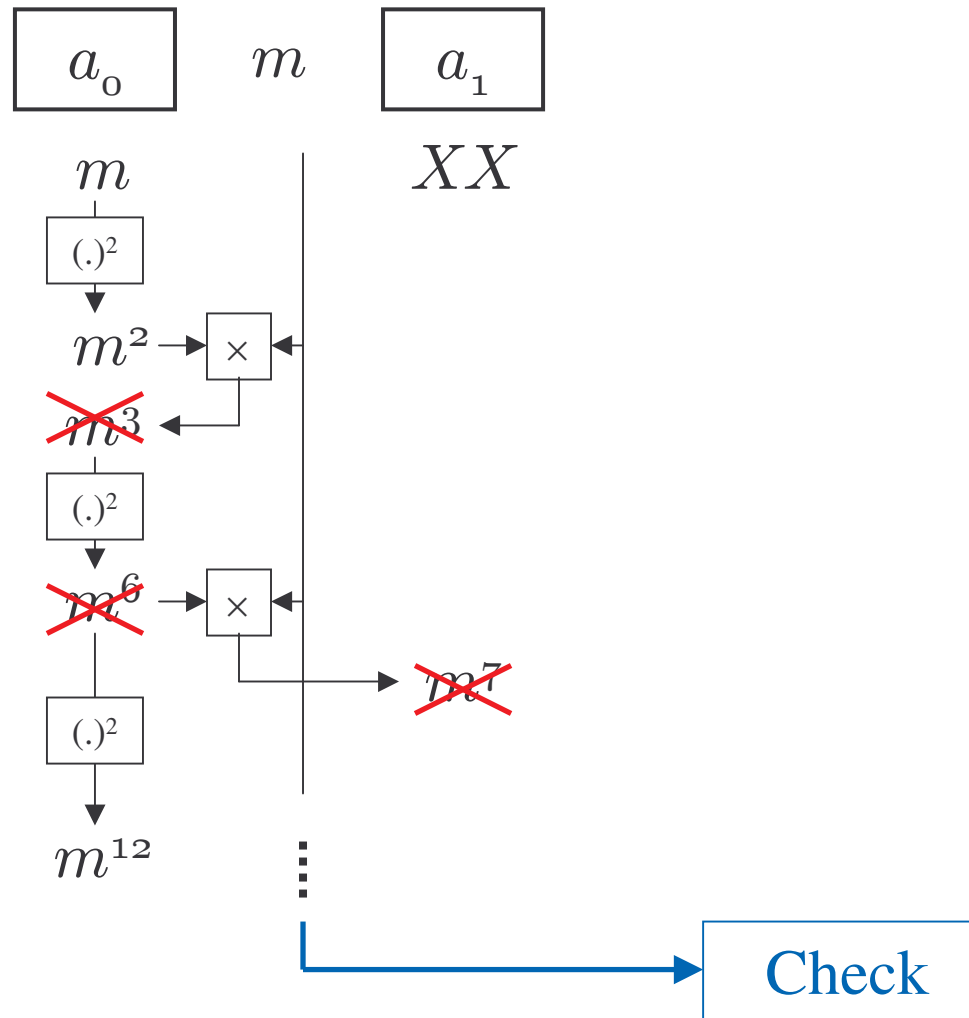
⇒ SPA-proof

- “Check with e or Aumüller et al. method“

⇒ DFA-proof

- “S & M always“ + “check with e or Aumüller et al. method“

⇒ NOT SECURE!
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

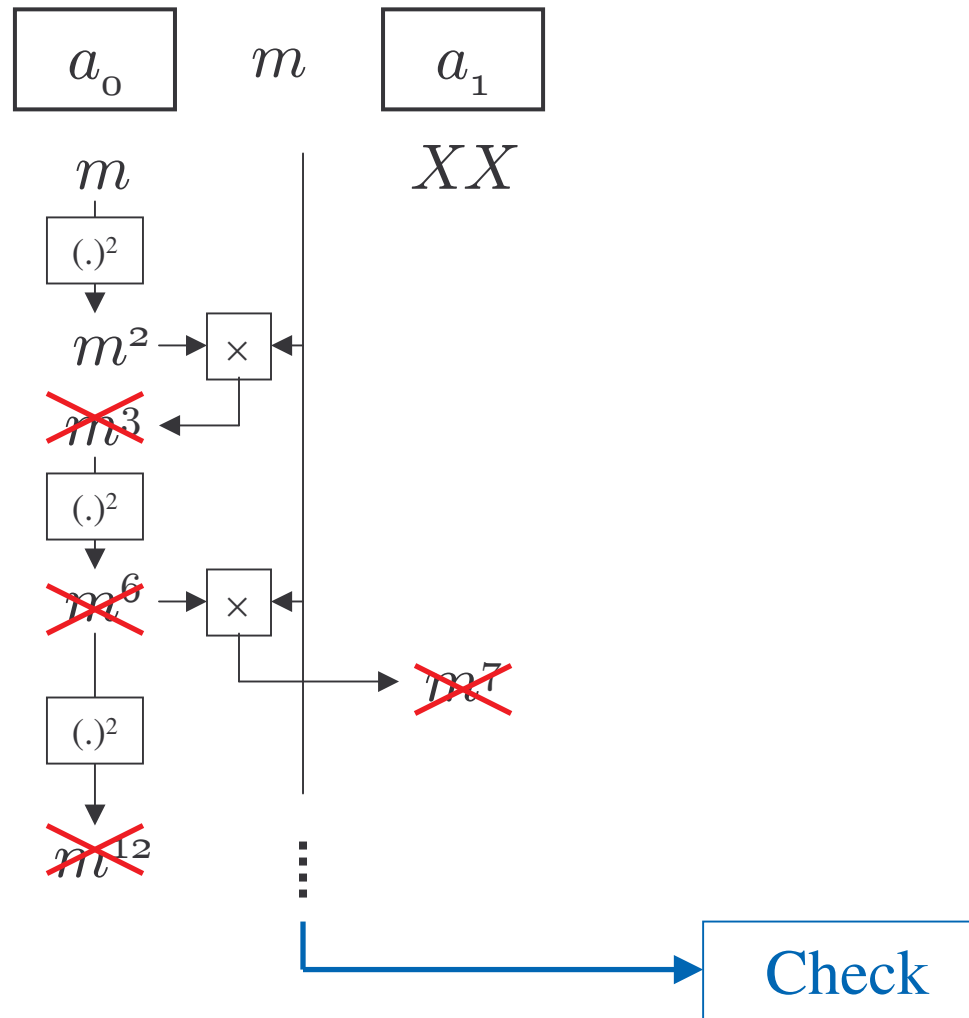
⇒ SPA-proof

- “Check with e or Aumüller et al. method”

⇒ DFA-proof

- “S & M always” + “check with e or Aumüller et al. method”

⇒ NOT SECURE!
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

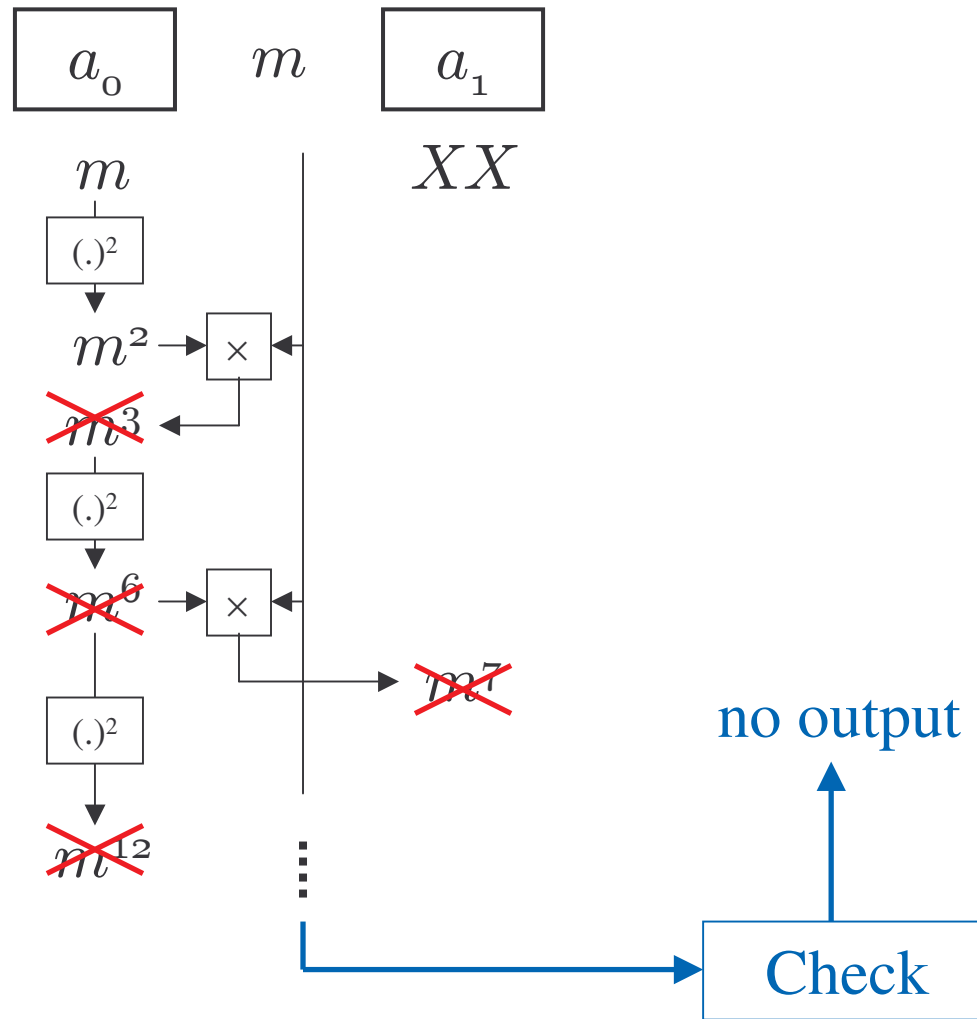
⇒ SPA-proof

- “Check with e or Aumüller et al. method”

⇒ DFA-proof

- “S & M always” + “check with e or Aumüller et al. method”

⇒ NOT SECURE!
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

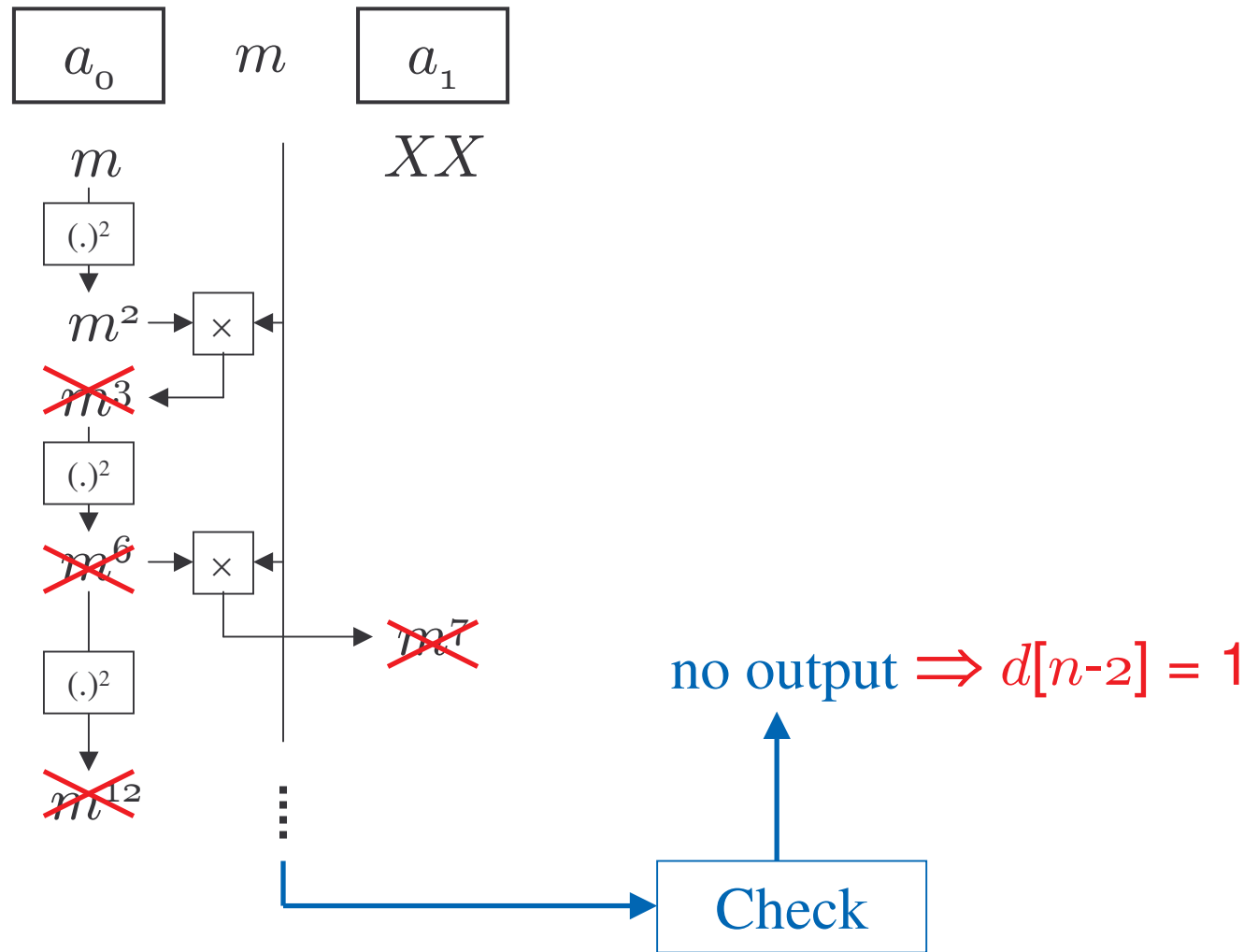
⇒ SPA-proof

- “Check with e or Aumüller et al. method“

⇒ DFA-proof

- “S & M always“ + “check with e or Aumüller et al. method“

⇒ NOT SECURE!
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

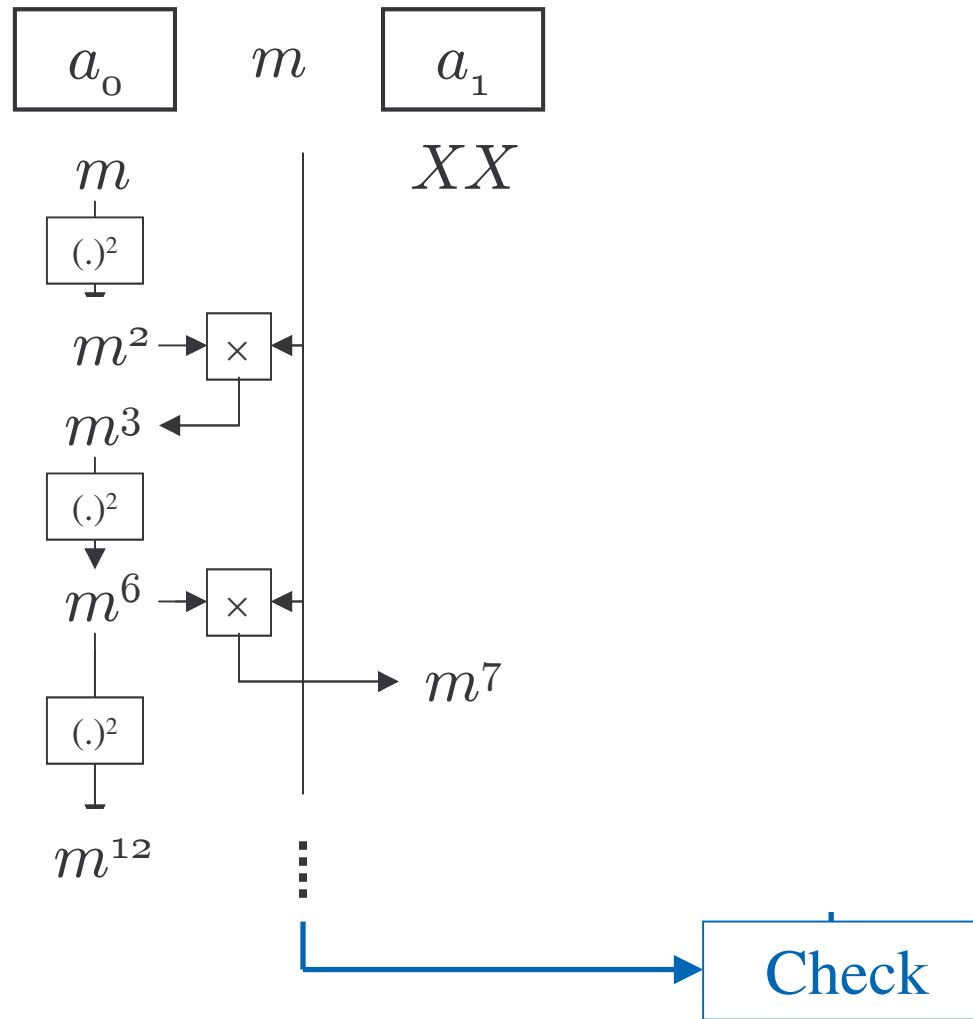
⇒ SPA-proof

- “Check with e or Aumüller et al. method“

⇒ DFA-proof

- “S & M always“ + “check with e or Aumüller et al. method“

⇒ **NOT SECURE!**
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

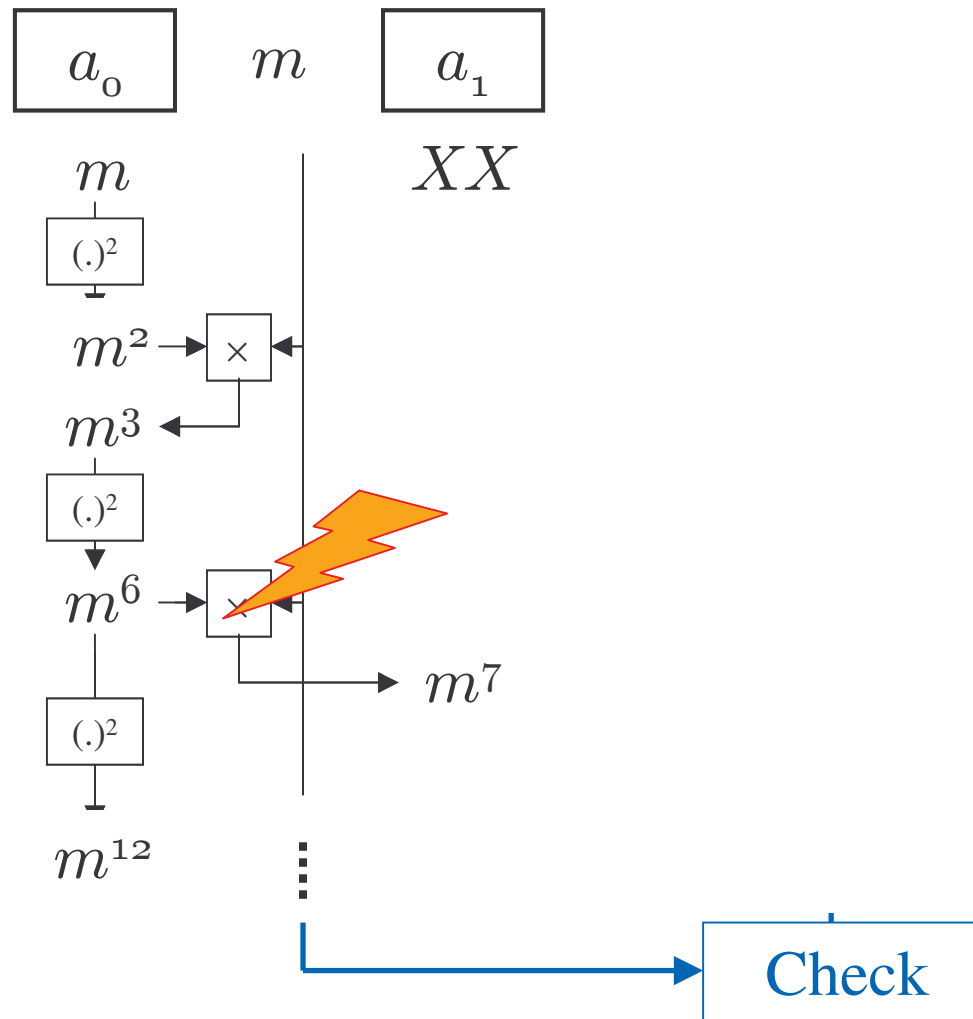
⇒ SPA-proof

- “Check with e or Aumüller et al. method“

⇒ DFA-proof

- “S & M always“ + “check with e or Aumüller et al. method“

⇒ NOT SECURE!
 Safe-errors attack
 (Joye and Yen)



Safe-error Attack

- “S & M always“

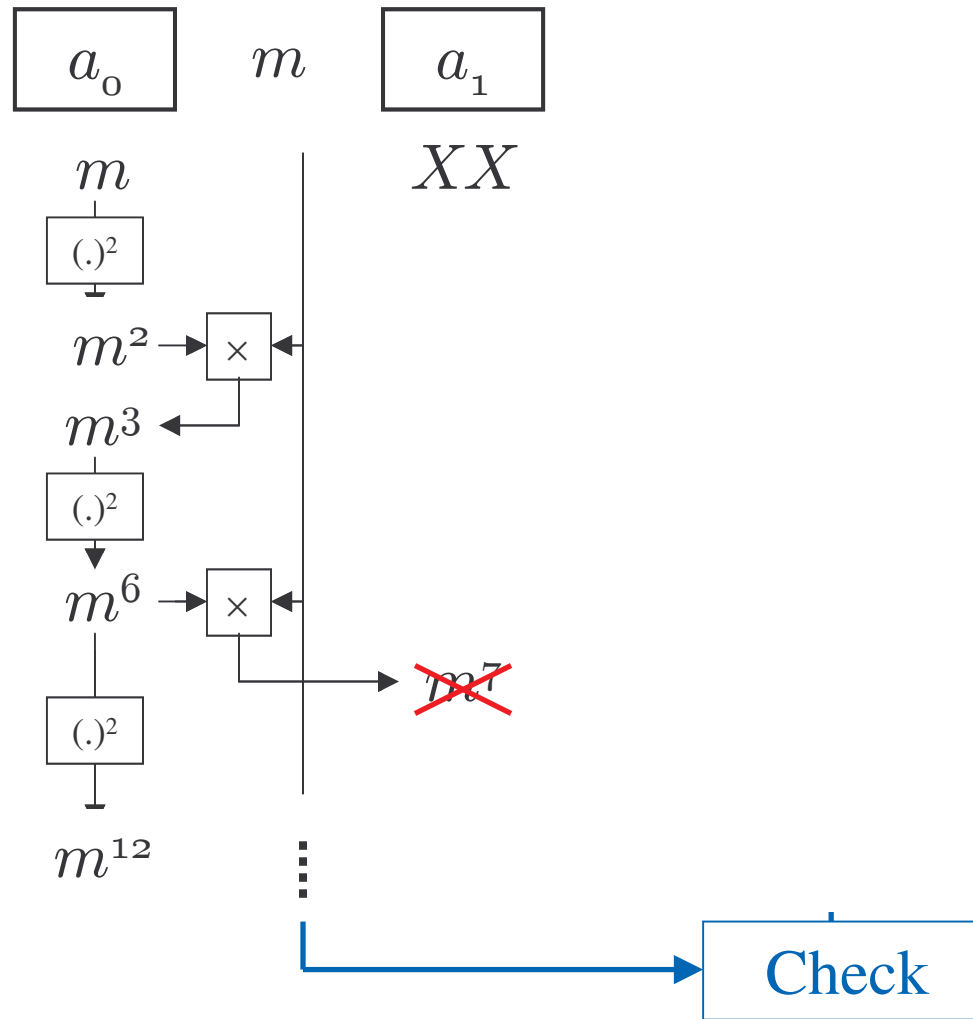
⇒ SPA-proof

- “Check with e or Aumüller et al. method“

⇒ DFA-proof

- “S & M always“ + “check with e or Aumüller et al. method“

⇒ NOT SECURE!
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

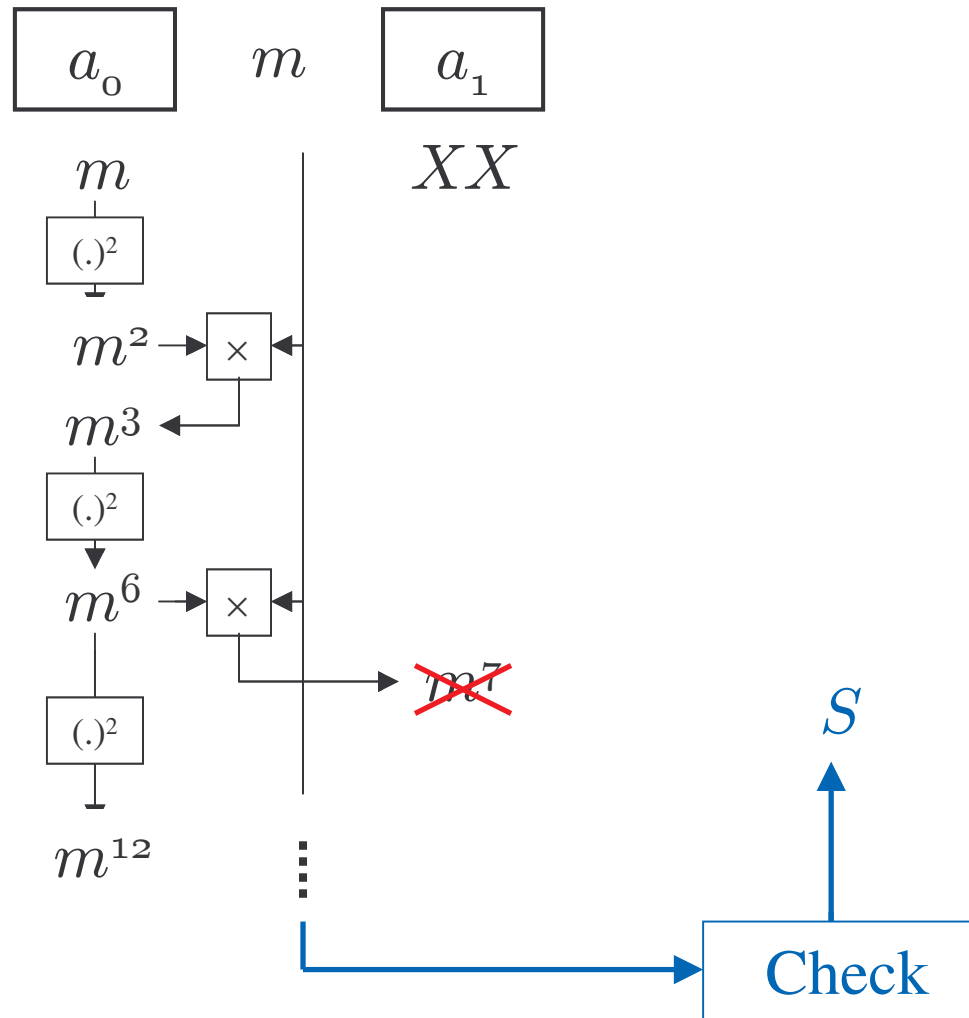
⇒ SPA-proof

- “Check with e or Aumüller et al. method“

⇒ DFA-proof

- “S & M always“ + “check with e or Aumüller et al. method“

⇒ **NOT SECURE!**
Safe-errors attack
(Joye and Yen)



Safe-error Attack

- “S & M always“

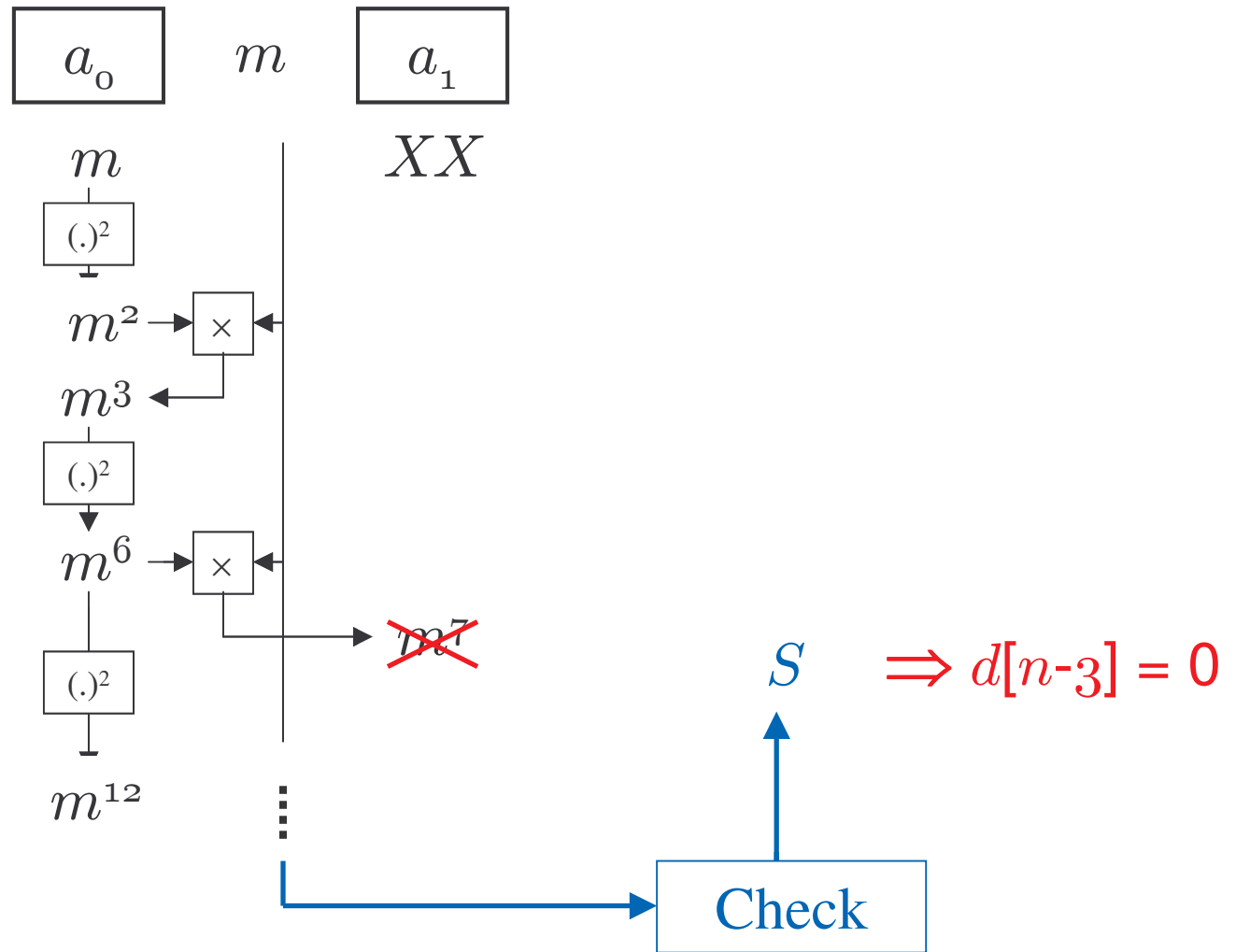
⇒ SPA-proof

- “Check with e or Aumüller et al. method“

⇒ DFA-proof

- “S & M always“ + “check with e or Aumüller et al. method“

⇒ NOT SECURE!
Safe-errors attack
(Joye and Yen)





Montgomery Ladder (ML)

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)

Montgomery Ladder (ML)

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{d_p[i]} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{d_p[i]}^2 \bmod p$$

return(a_0)

$$\boxed{a_0}$$

 m

$$\boxed{a_1}$$

 m^2

Montgomery Ladder (ML)

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

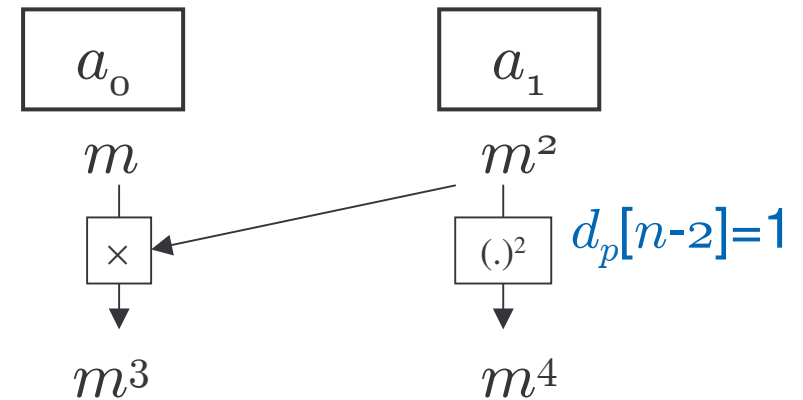
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



Montgomery Ladder (ML)

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

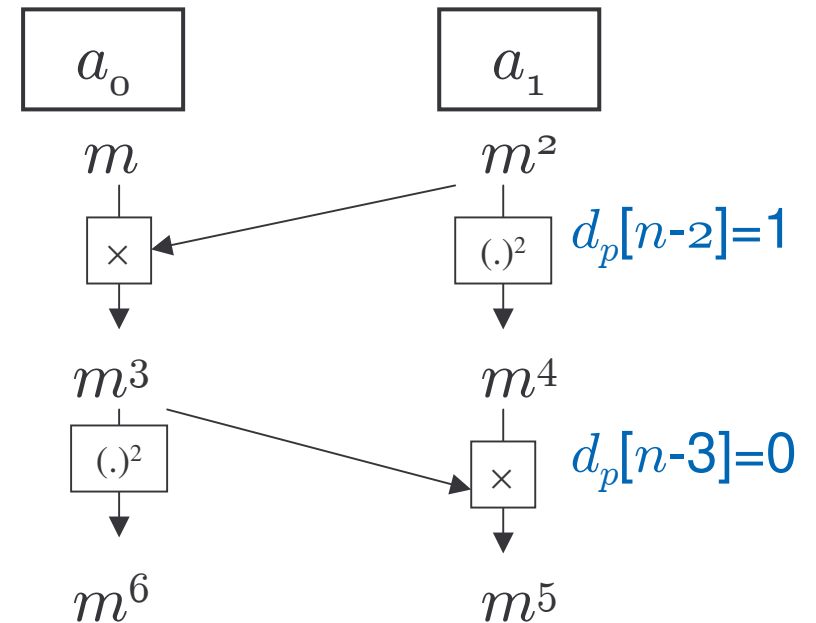
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



Montgomery Ladder (ML)

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

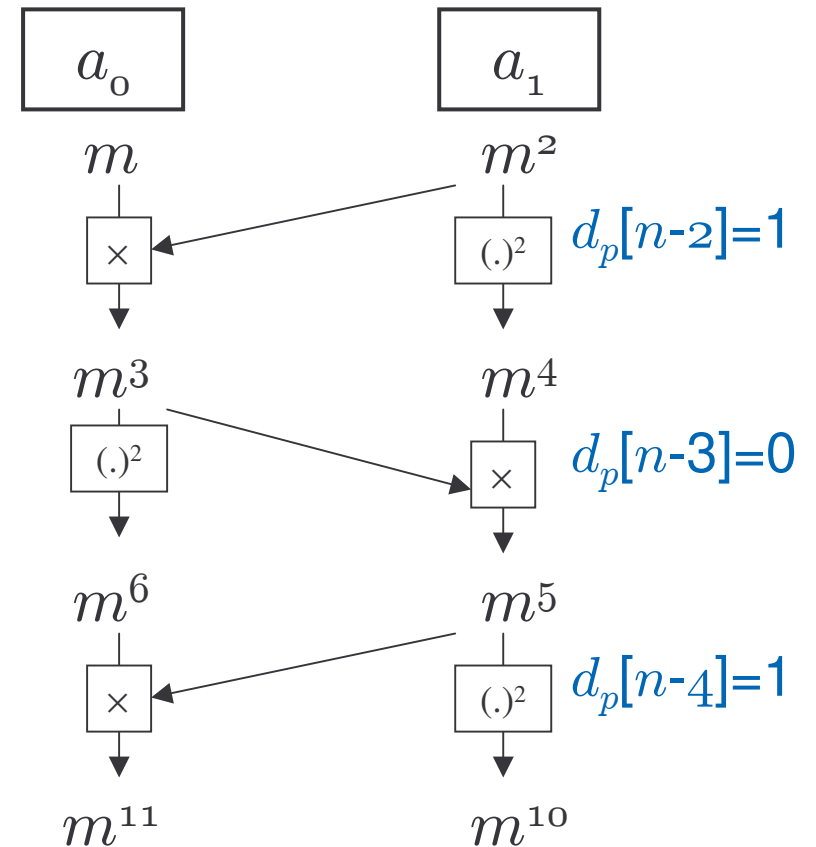
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



Montgomery Ladder (ML)

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

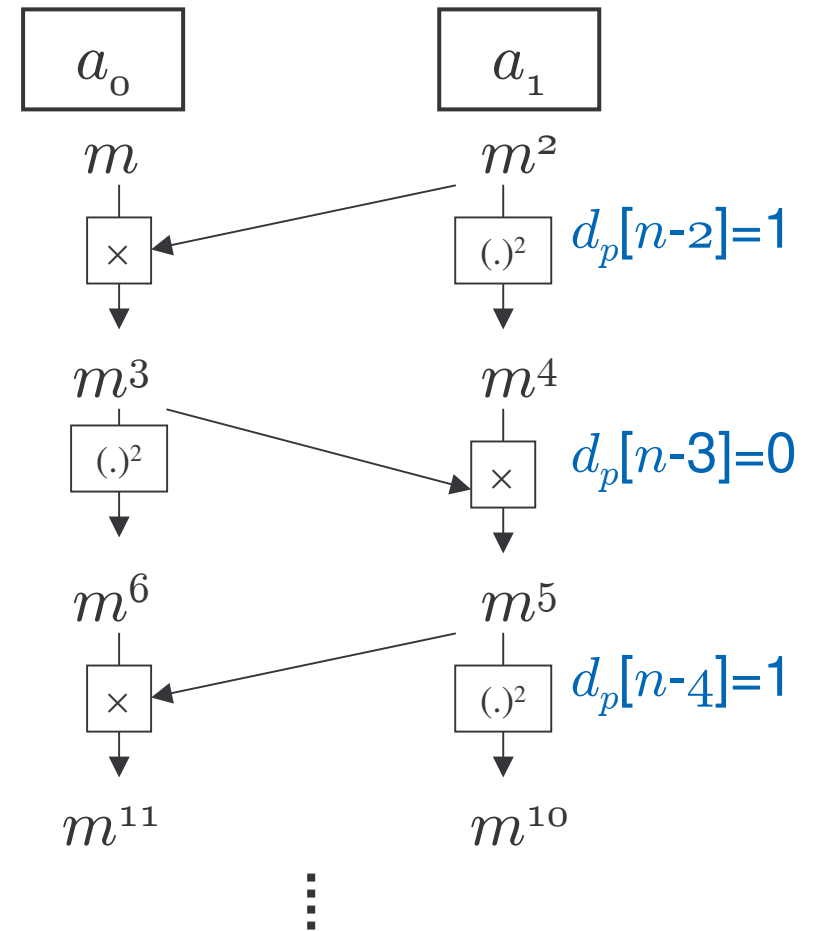
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



Montgomery Ladder (ML)

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

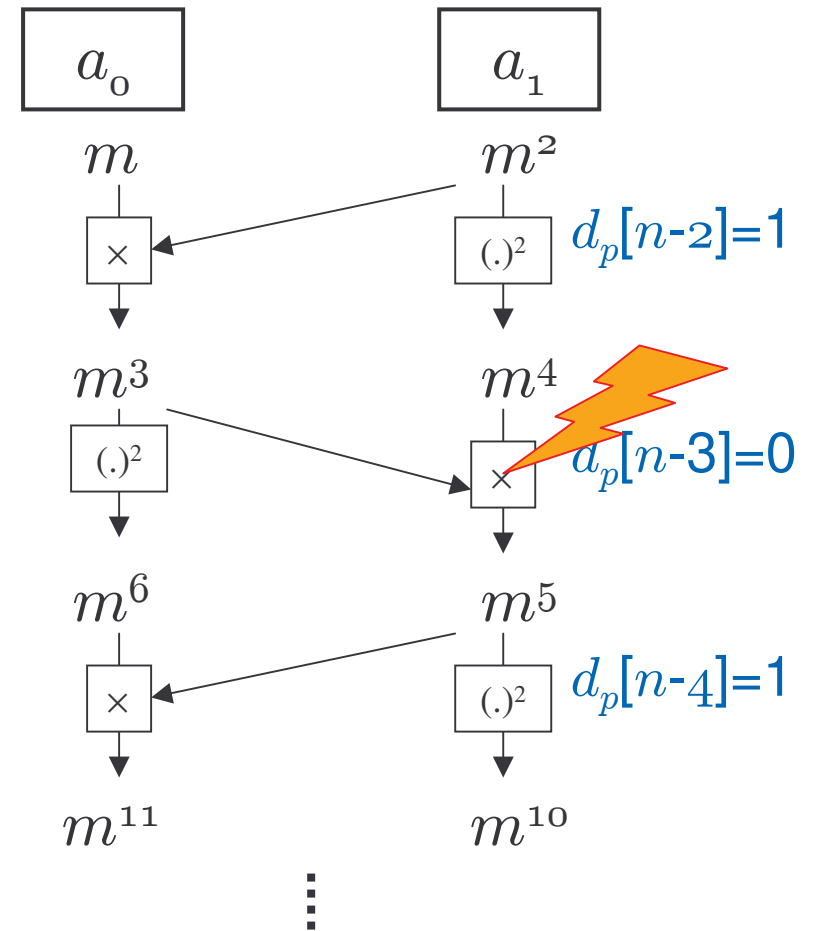
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



Montgomery Ladder (ML)

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

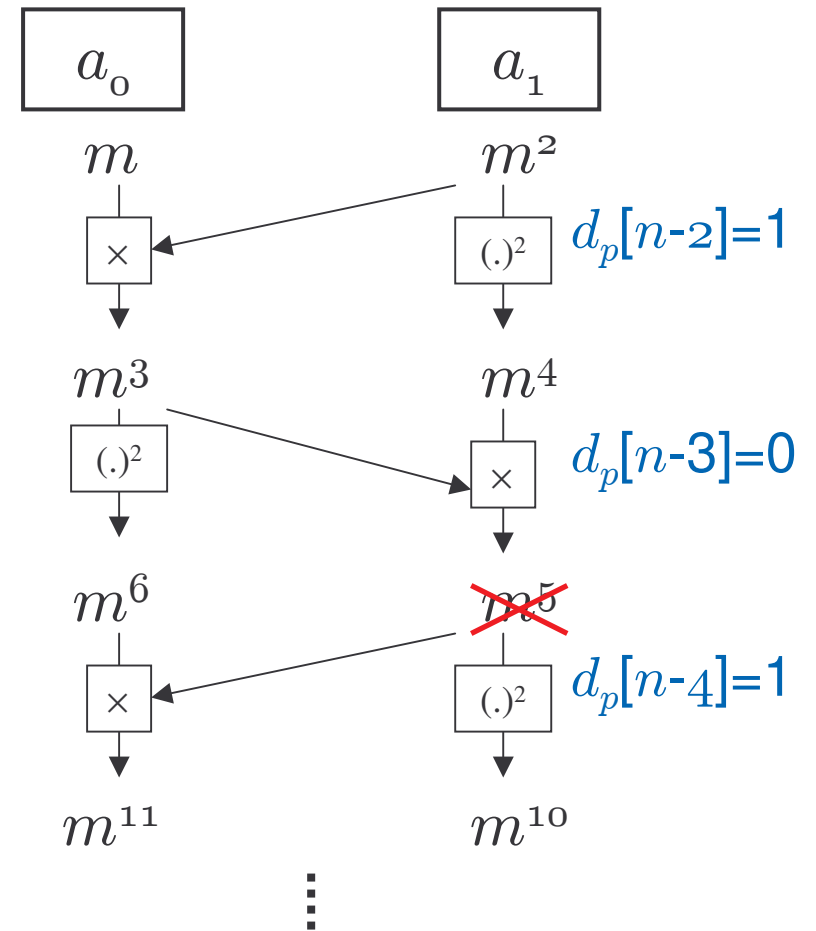
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



Montgomery Ladder (ML)

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

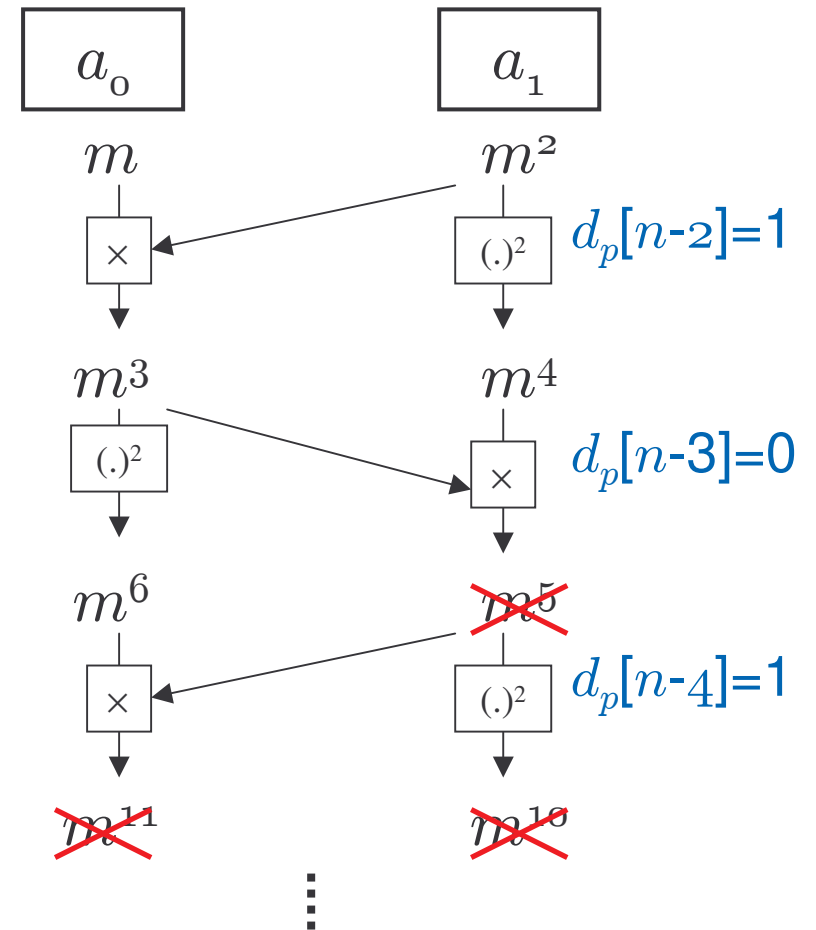
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



State of the art

- “Montgomery ladder” + “Check with e or Aumüller et al. method”
⇒ RSA implementation protected against SPA, DFA and Safe-errors
- “Check with e or Aumüller et al. method” → pb of performances

How to build a faster SPA-DFA proof RSA?

A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

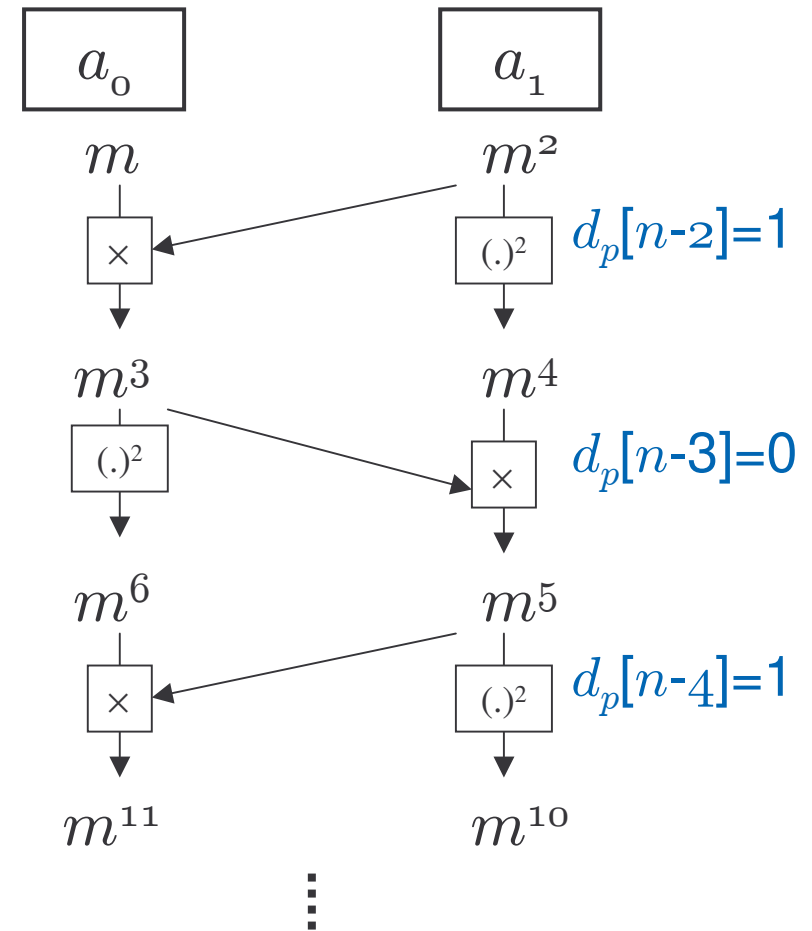
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

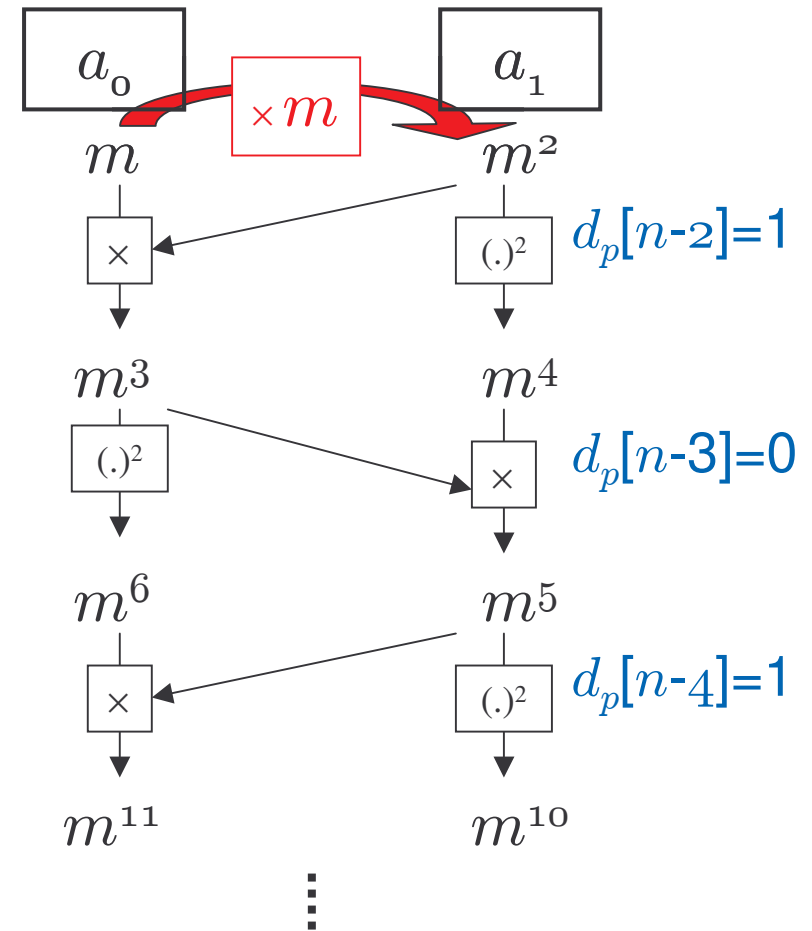
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

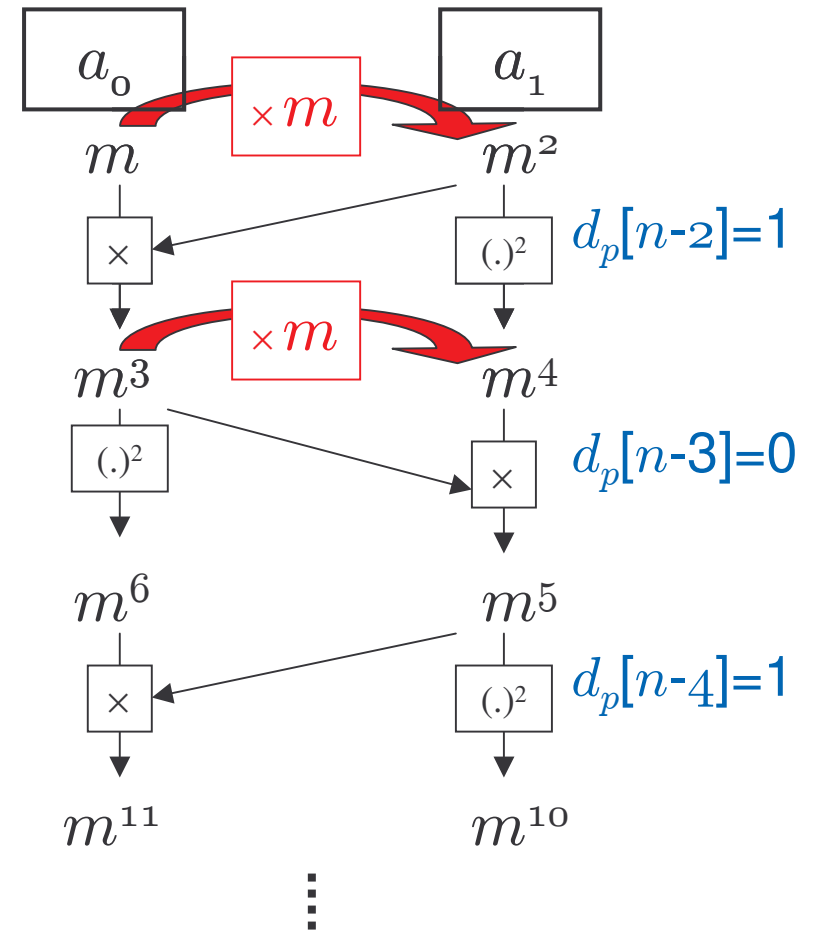
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

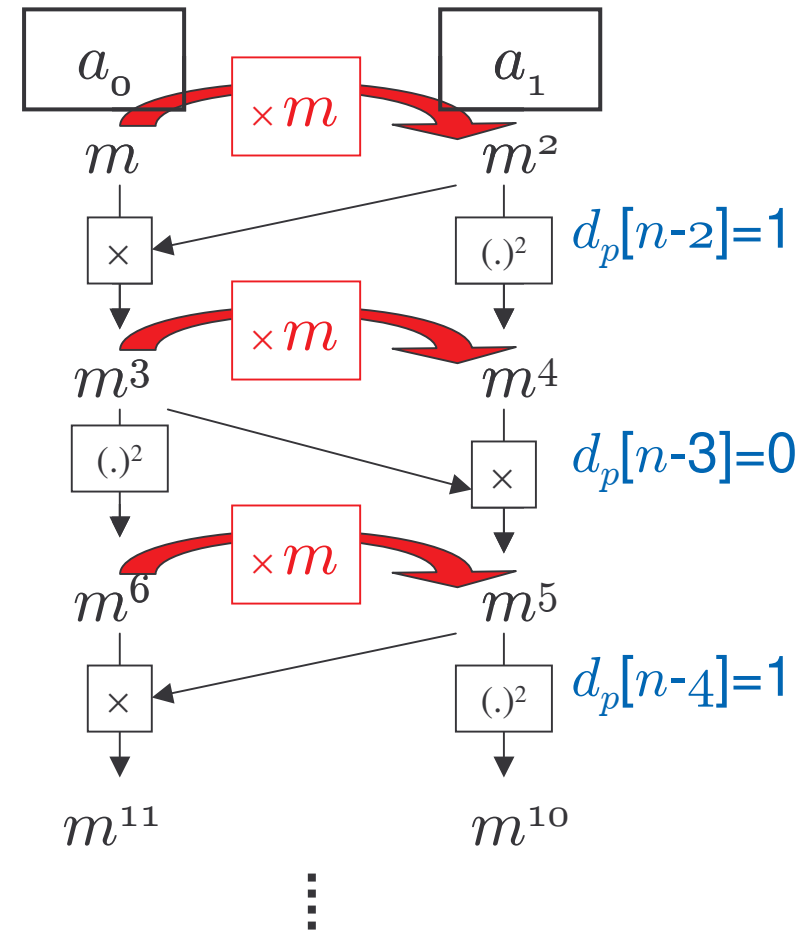
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

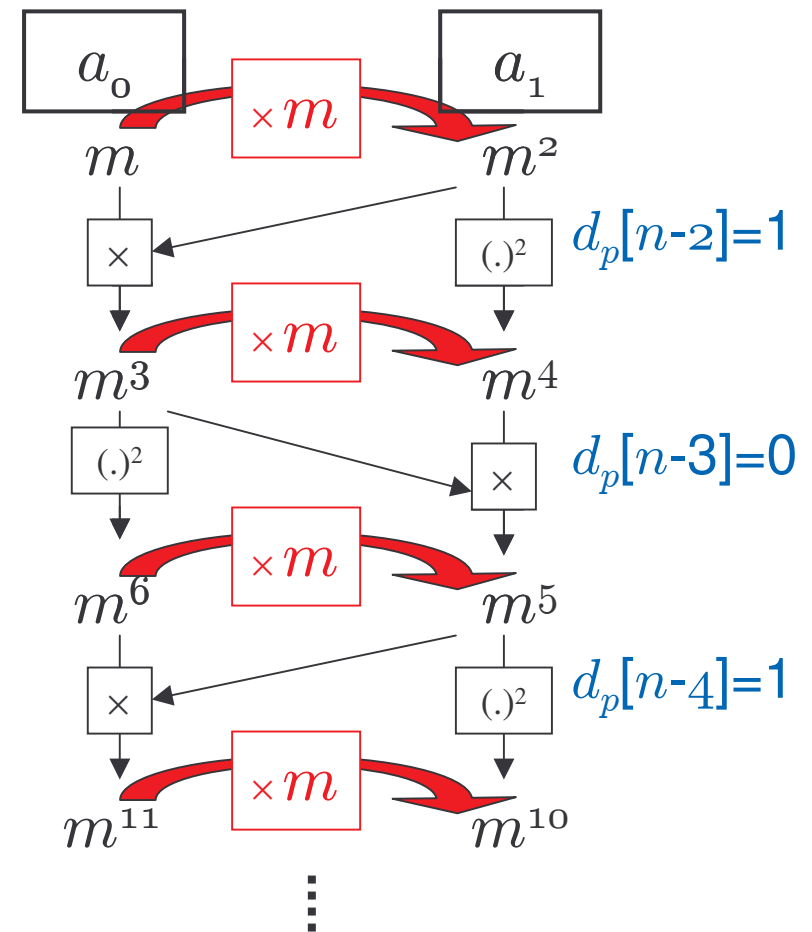
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

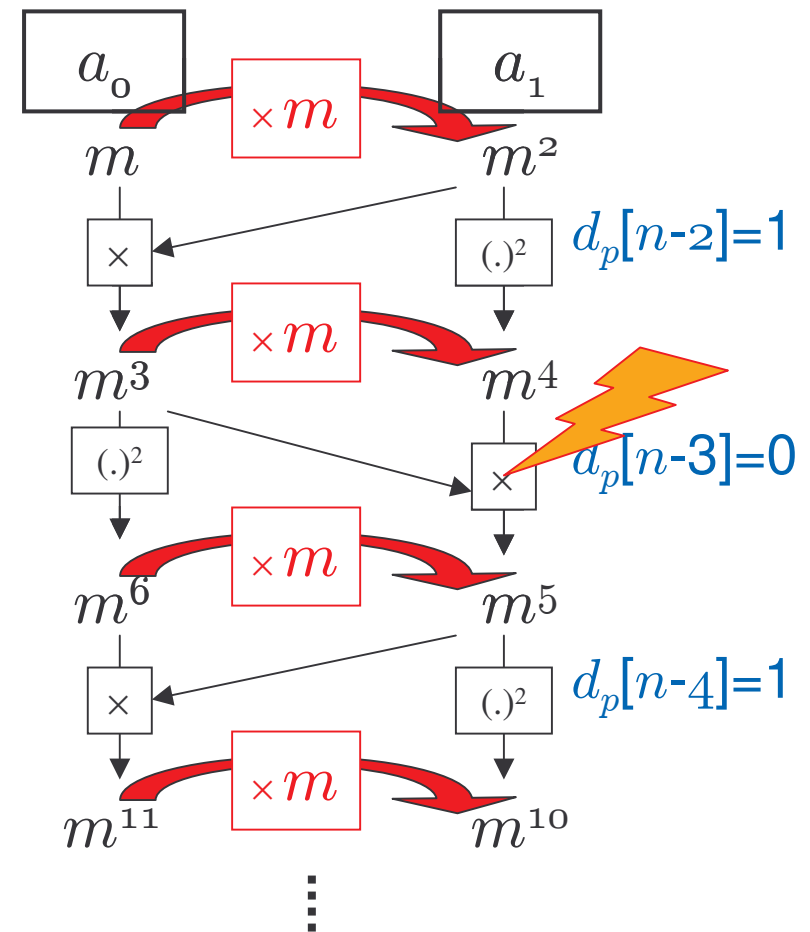
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

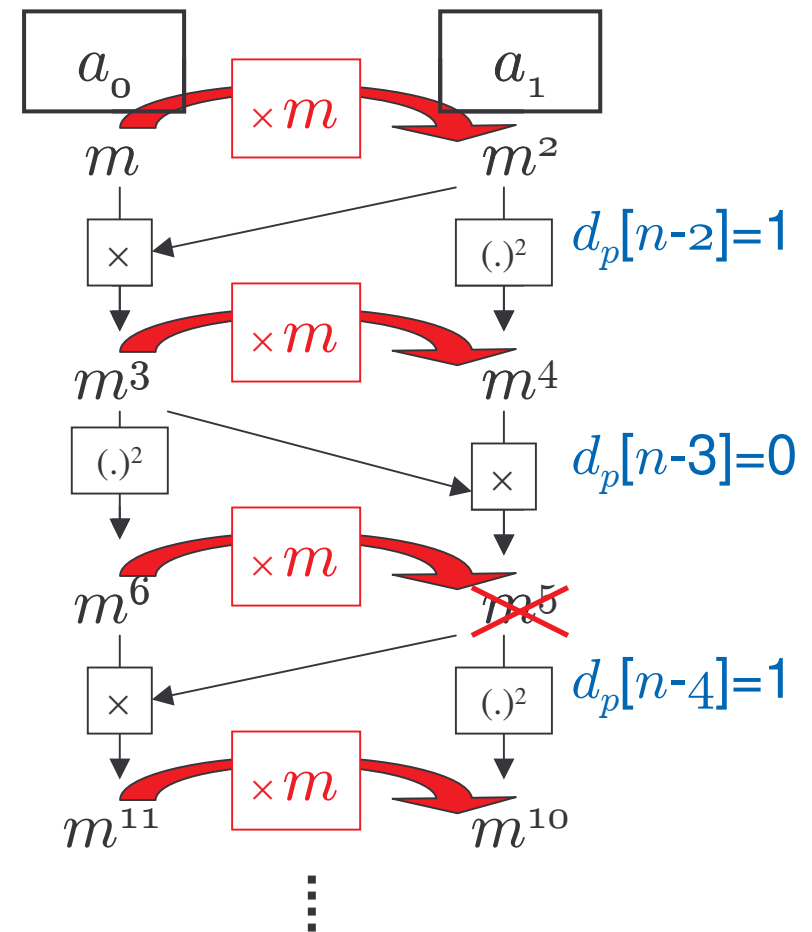
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

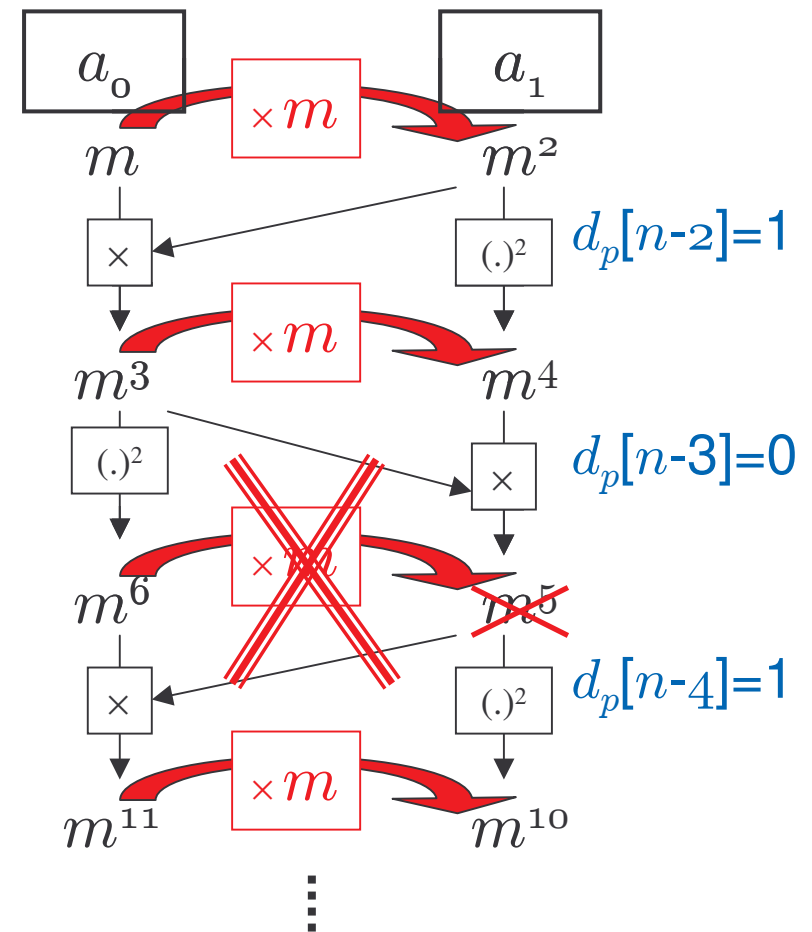
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

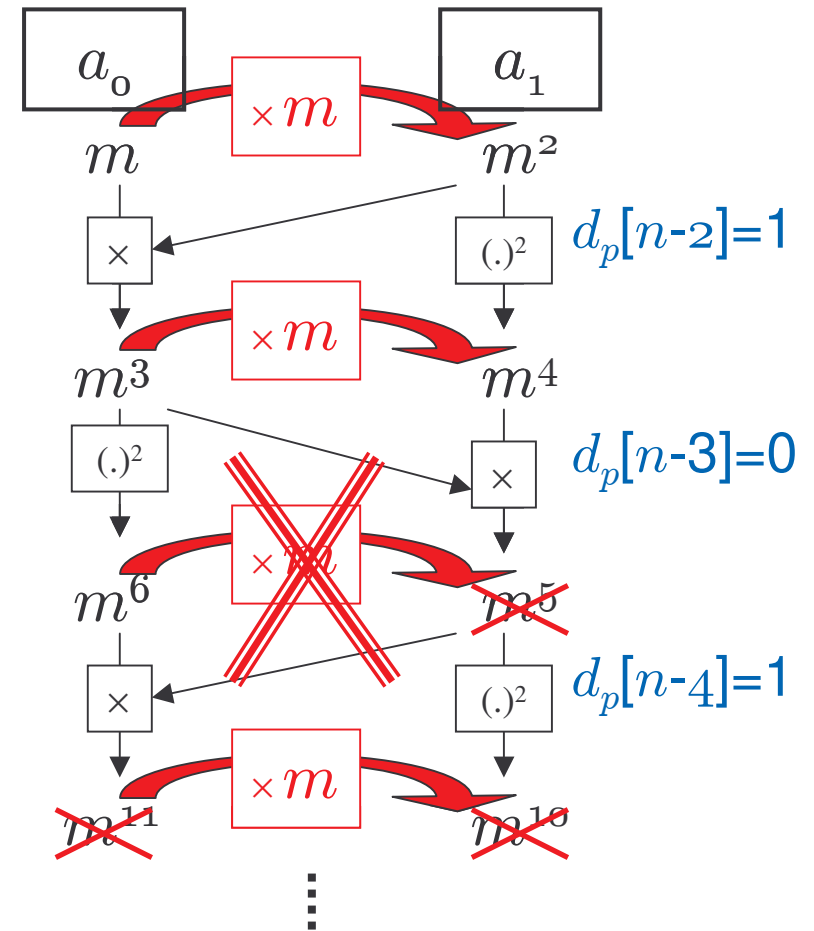
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{d_p[i]} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{d_p[i]}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of S_{d_p} :

$$a_0 \leftarrow m \bmod p$$

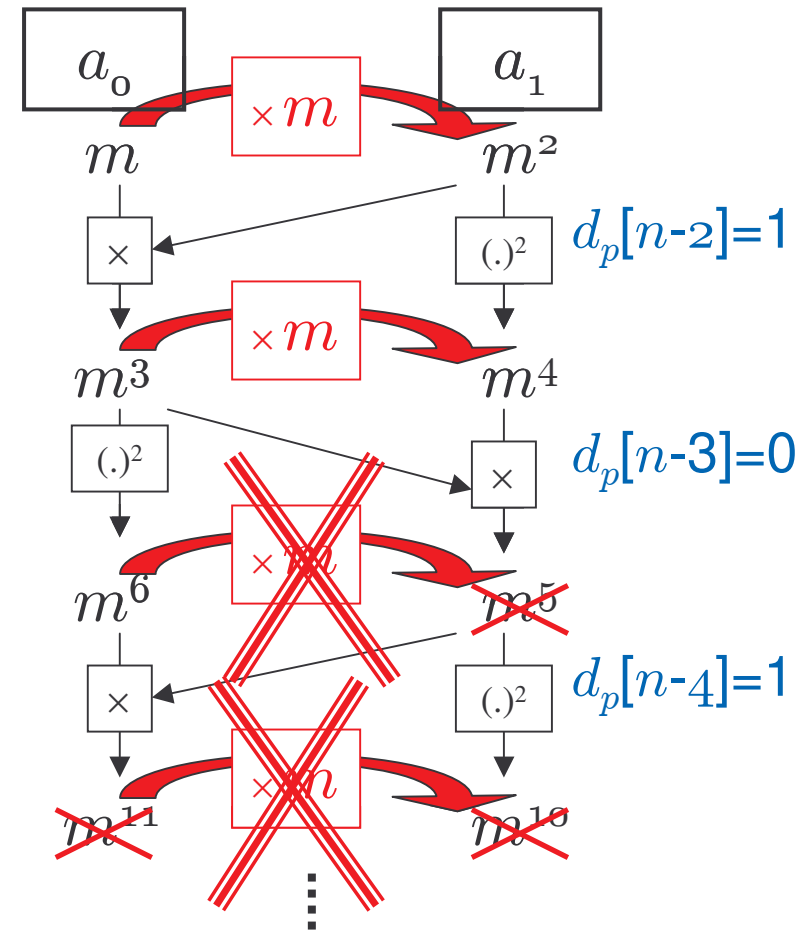
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{d_p[i]} \leftarrow a_0 * a_1 \bmod p$$

$$a_{d_p[i]} \leftarrow a_{d_p[i]}^2 \bmod p$$

return(a_0)



A very interesting property

- Computation of (S_{d_p}, S_{d_p-1}) :

$$a_0 \leftarrow m \bmod p$$

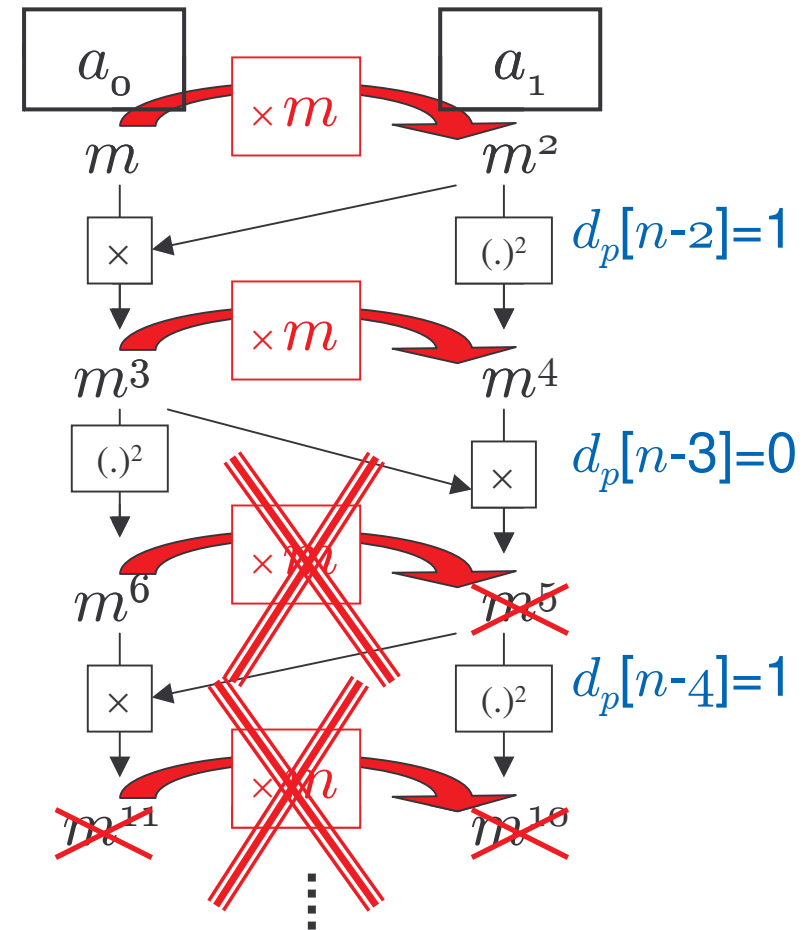
$$a_1 \leftarrow m^2 \bmod p$$

for i from $n - 2$ to 0 do

$$a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$$

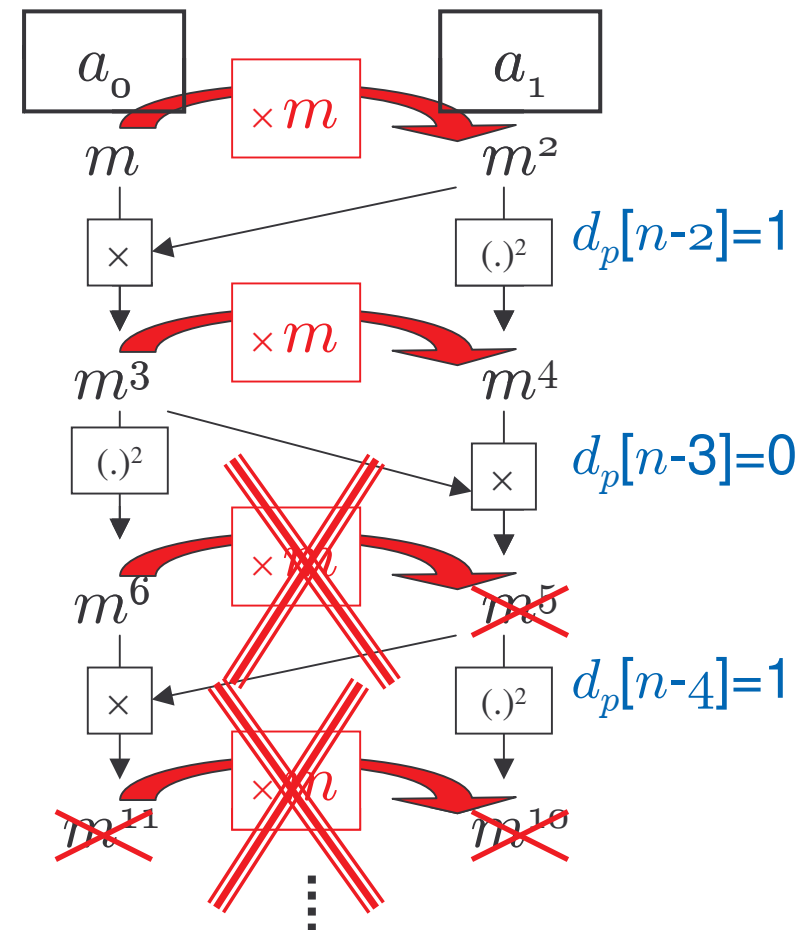
$$a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$$

return(a_0)

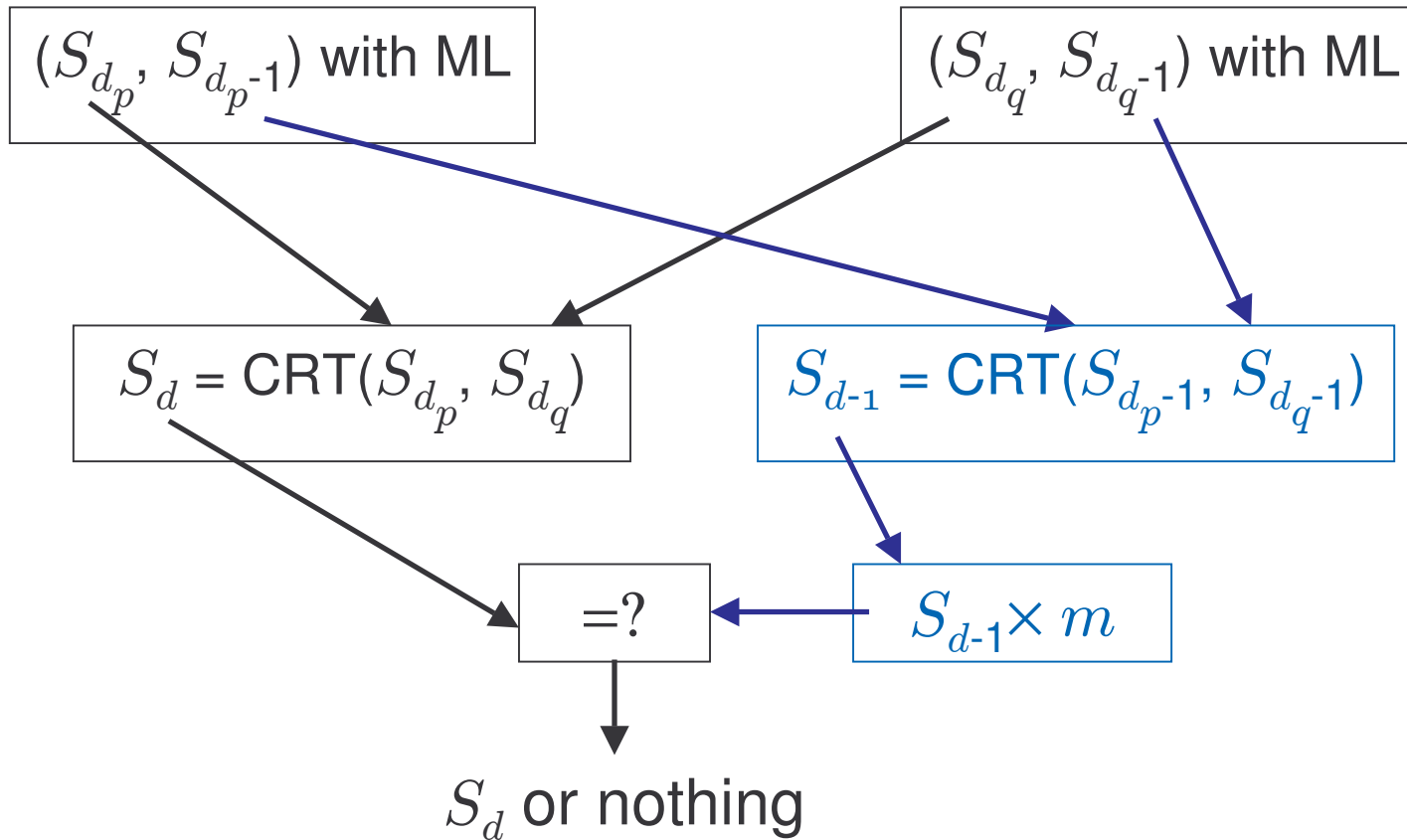


A very interesting property

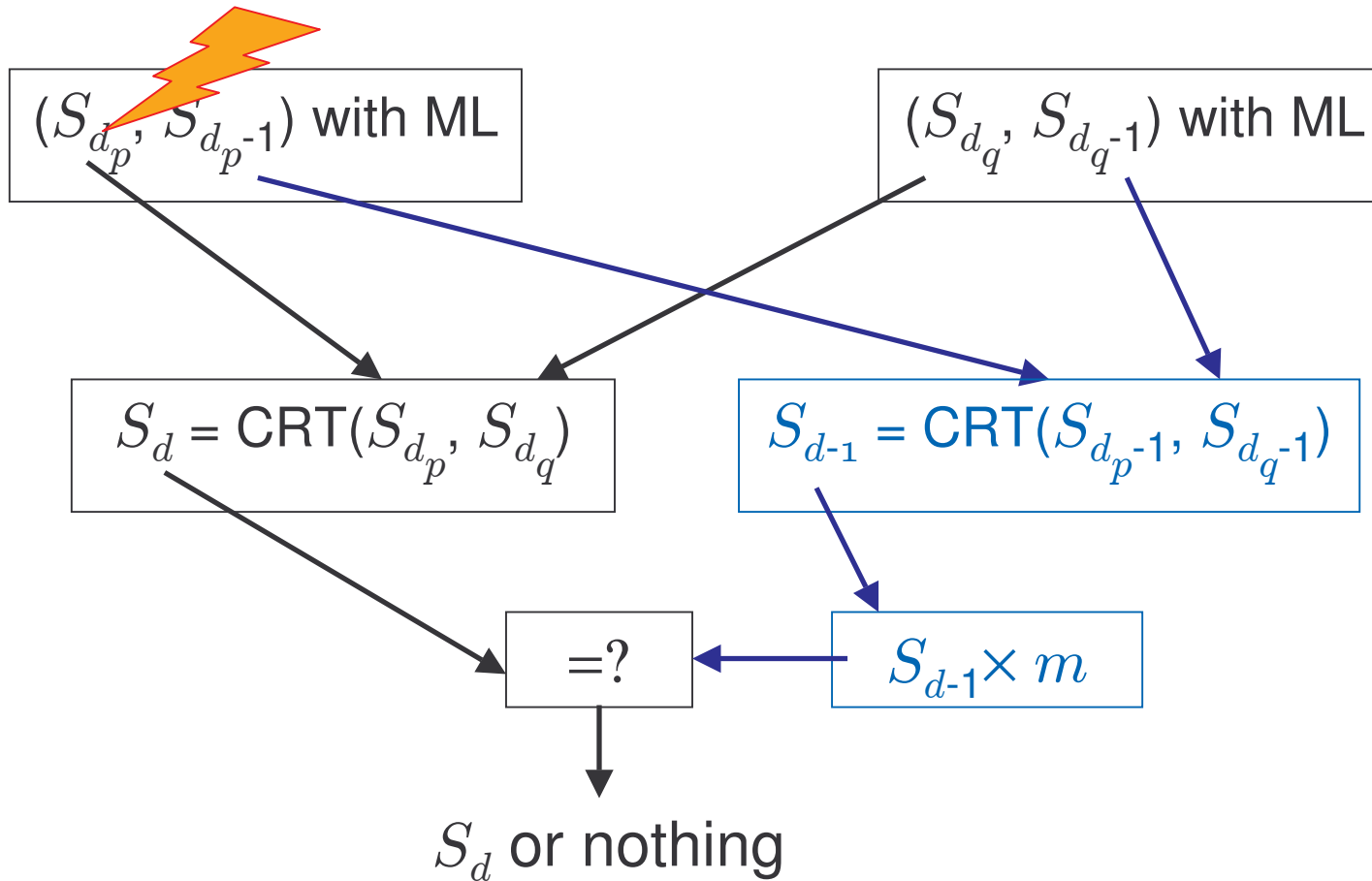
- Computation of (S_{d_p}, S_{d_p-1}) :
 - $a_0 \leftarrow m \bmod p$
 - $a_1 \leftarrow m^2 \bmod p$
 - for i from $n - 2$ to 1 do
 - $a_{\overline{d_p[i]}} \leftarrow a_0 * a_1 \bmod p$
 - $a_{d_p[i]} \leftarrow a_{\overline{d_p[i]}}^2 \bmod p$
 - $a_1 \leftarrow a_0 \times a_1 \bmod p$
 - $a_0 \leftarrow a_0^2 \bmod p$
 - return((a_1, a_0))



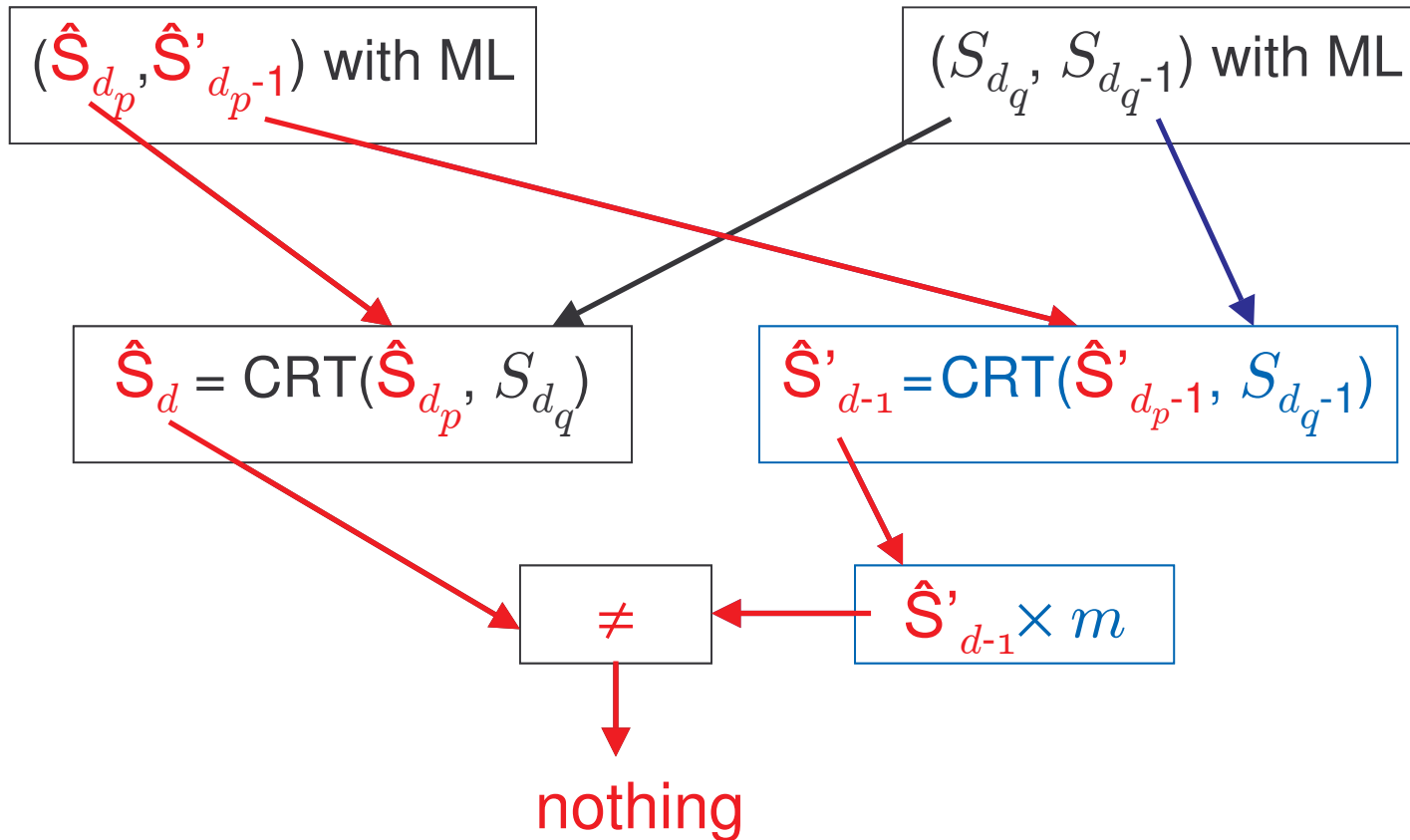
New countermeasure



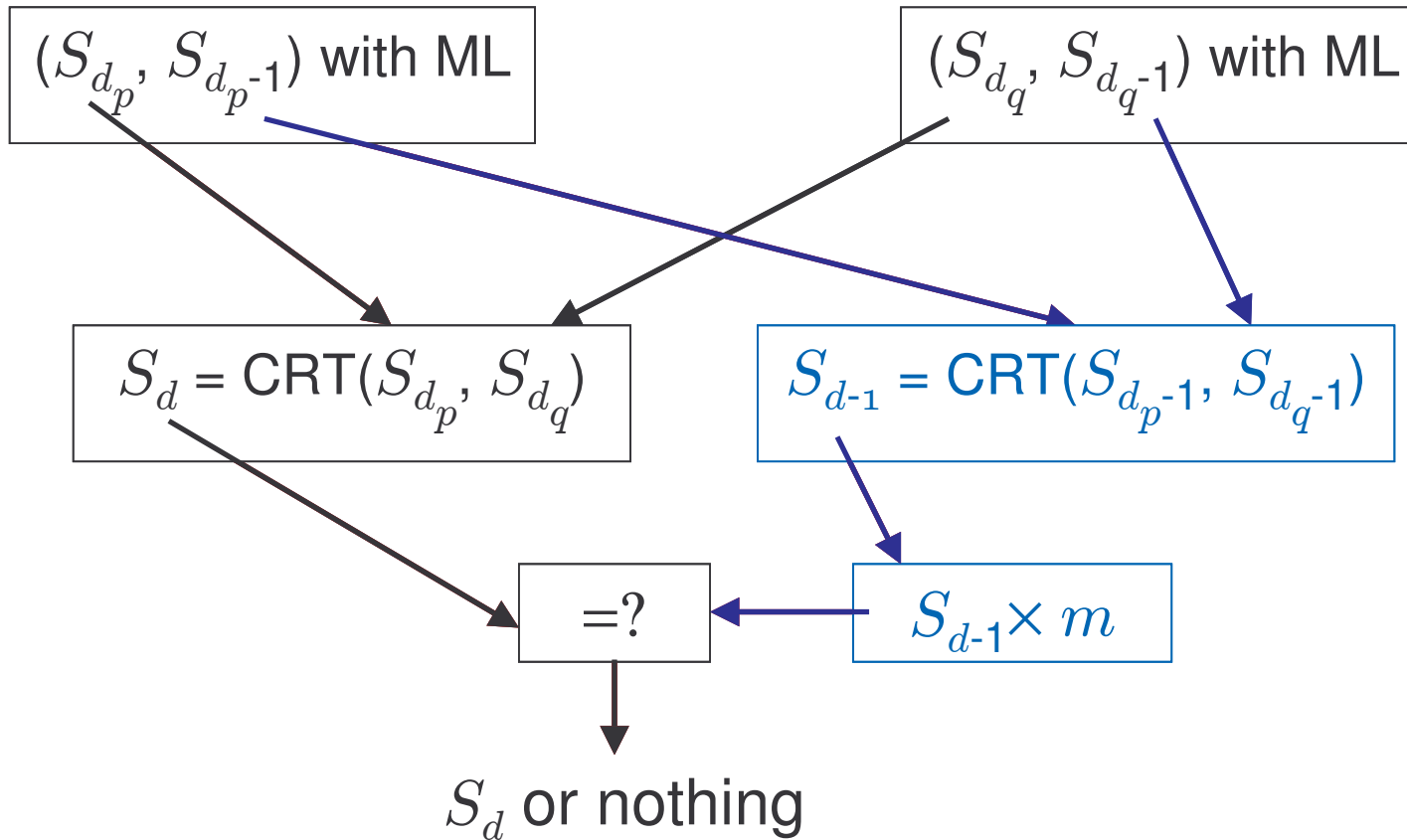
New countermeasure



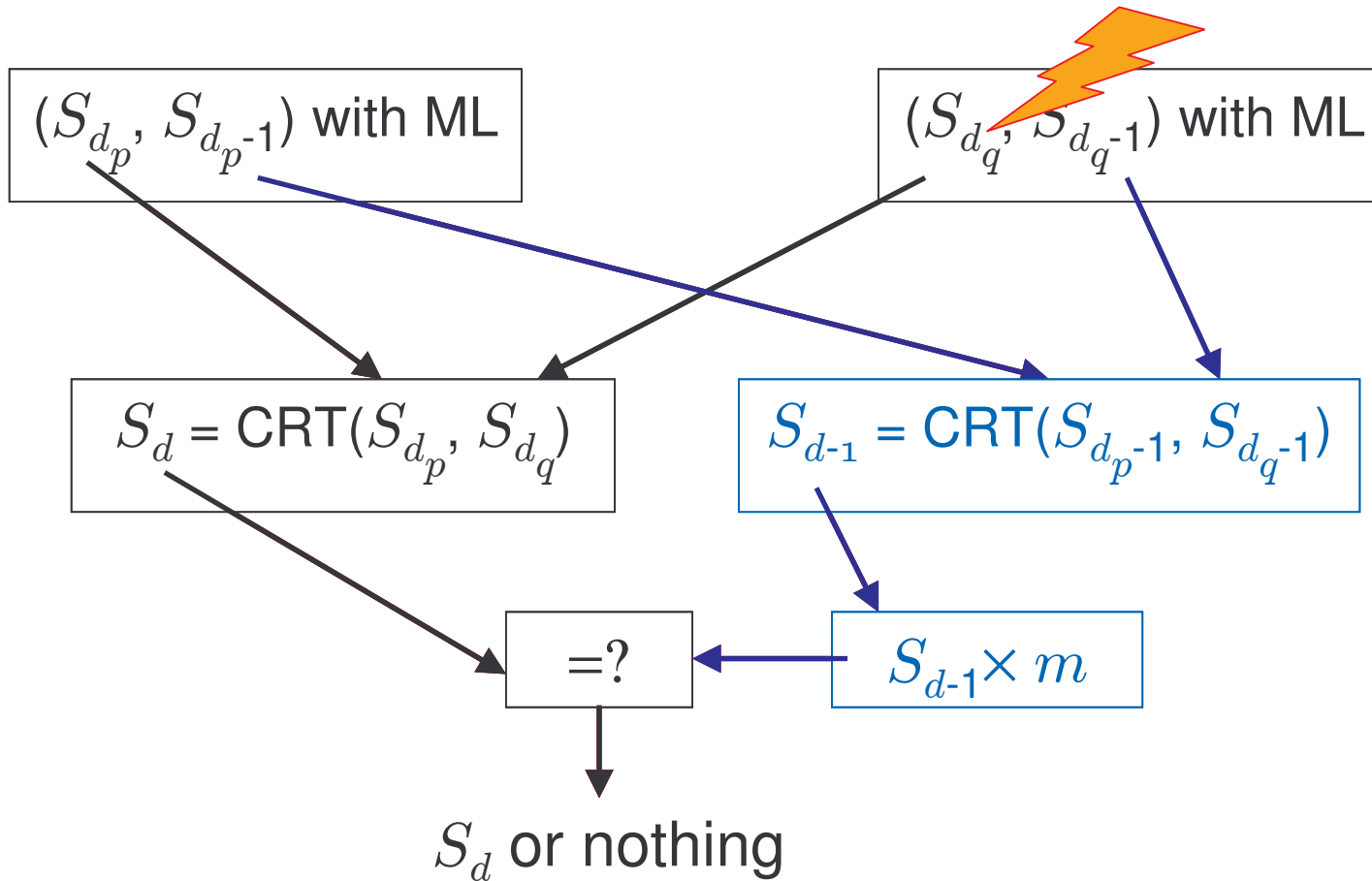
New countermeasure



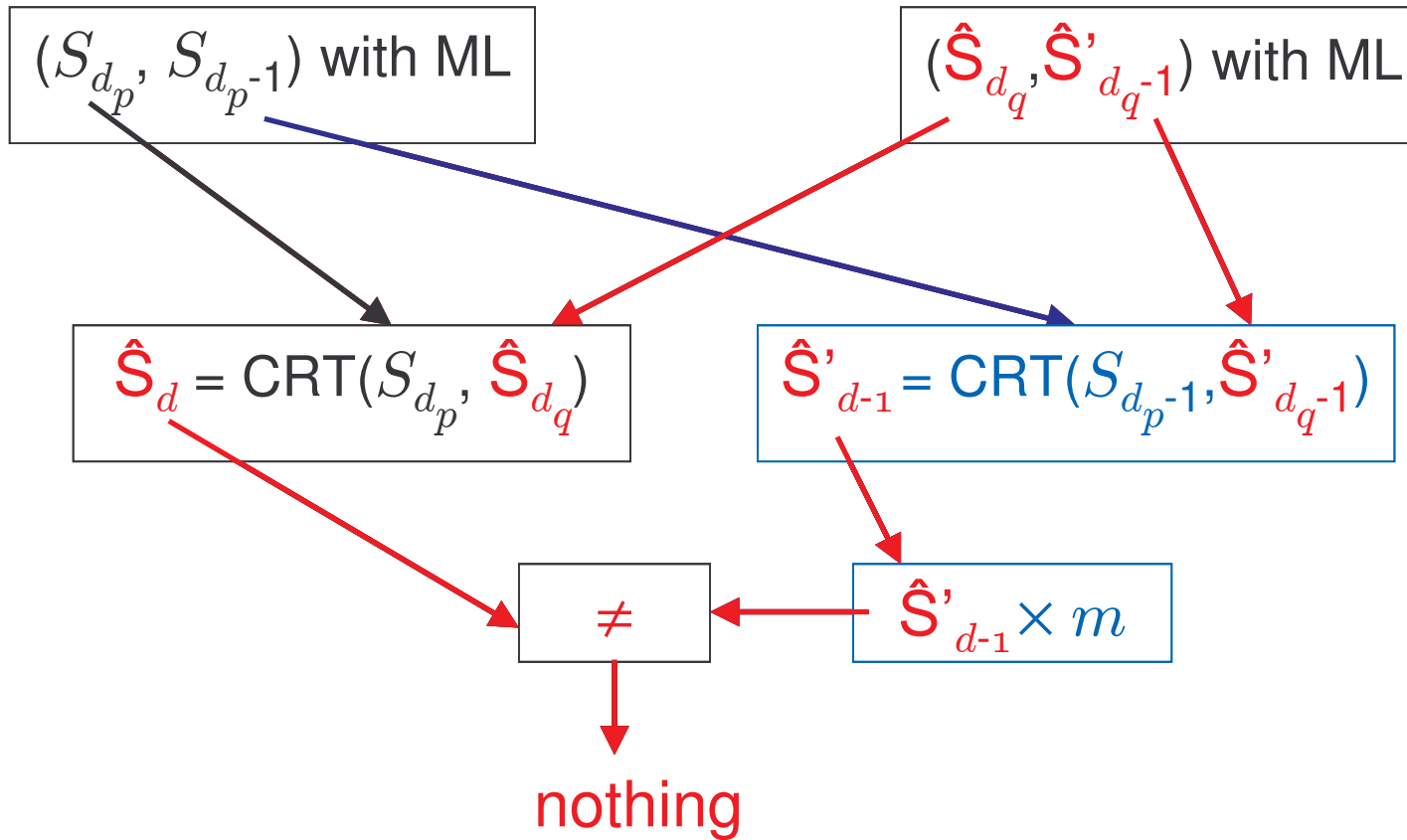
New countermeasure



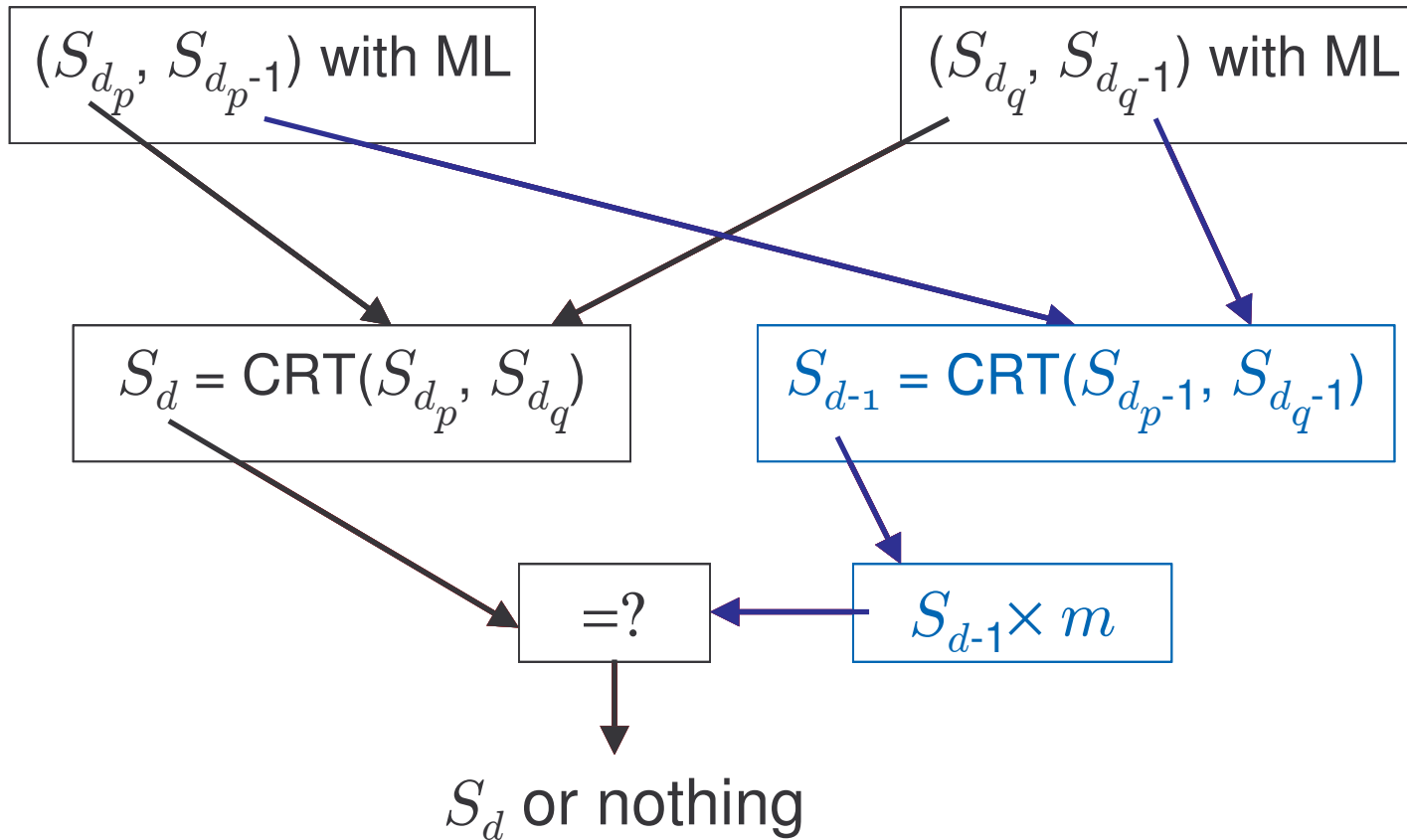
New countermeasure



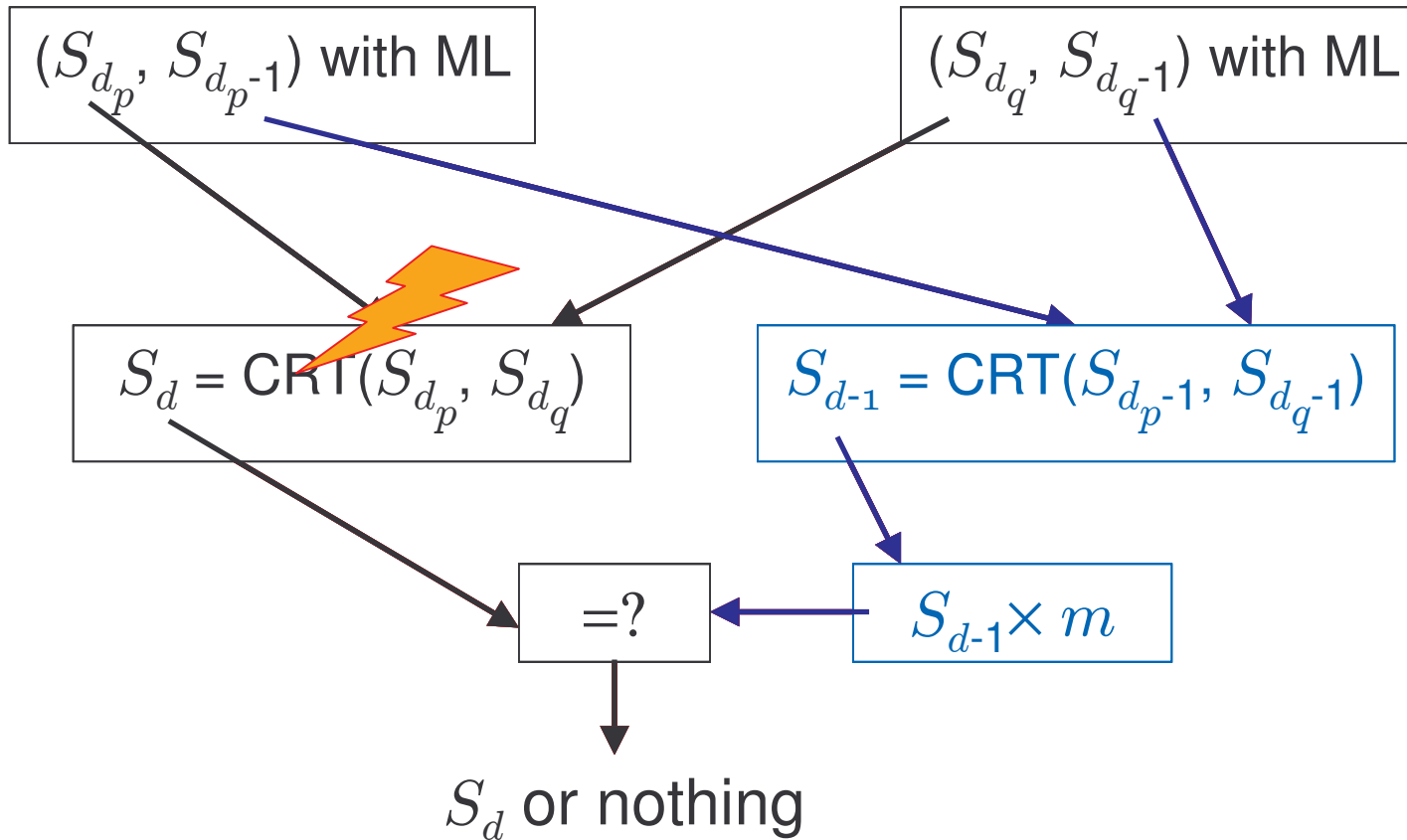
New countermeasure



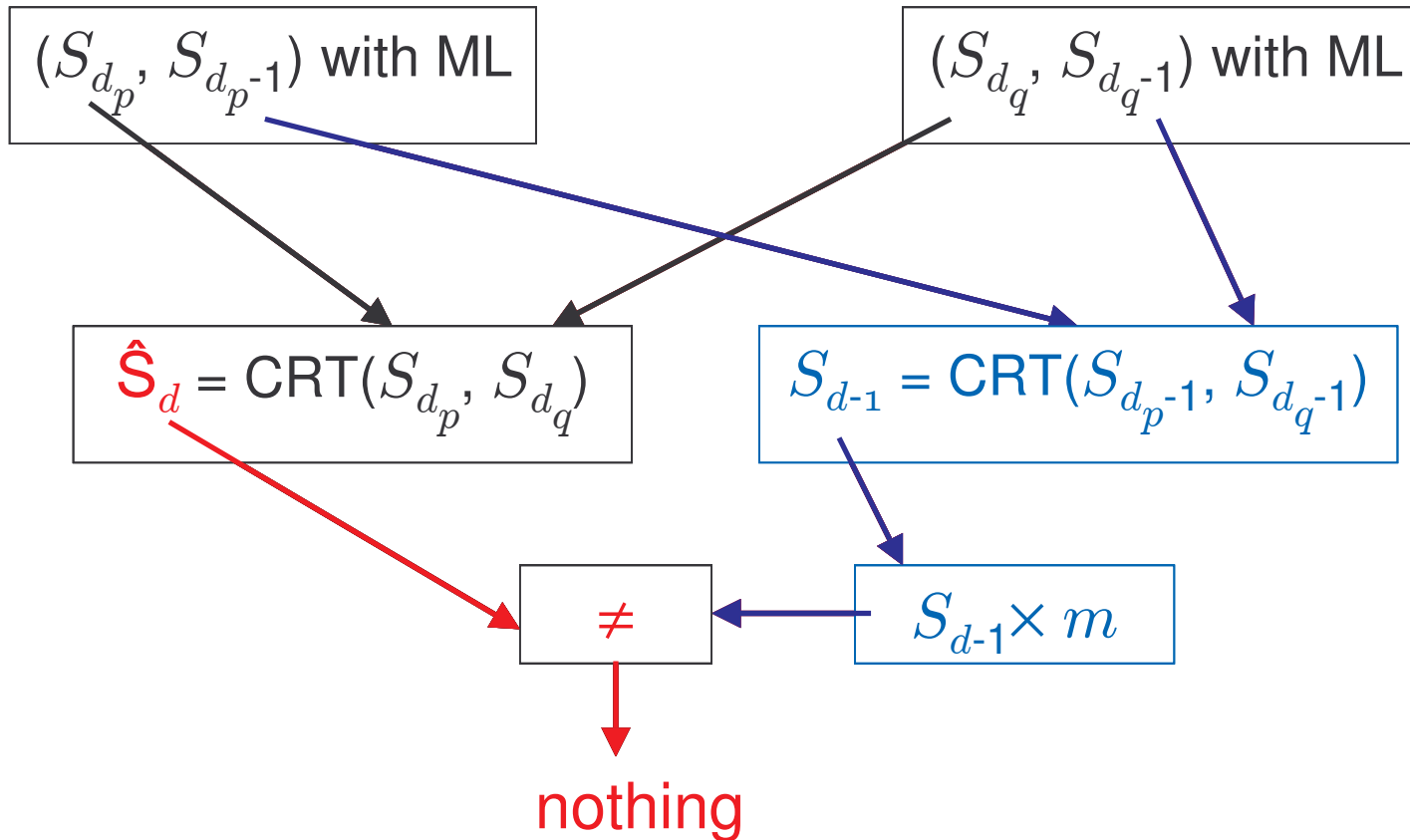
New countermeasure



New countermeasure



New countermeasure



Some details

Remark :

e not available \Rightarrow no computation to check the link between the secret parameters

Solution:

- ◆ add checksum to $p, q, p^{-1} \bmod q, d_p, d_q$
- ◆ check the integrity of the parameters used before outputting the results

Conclusion

Conclusion

- A new SPA-DFA-proof CRT RSA implementation
 - ◆ No need of e
 - ◆ No extra parameters to store
 - ◆ Only 3 more modular multiplications \Rightarrow the fastest method

Conclusion

- A new SPA-DFA-proof CRT RSA implementation
 - ◆ No need of e
 - ◆ No extra parameters to store
 - ◆ Only 3 more modular multiplications \Rightarrow the fastest method
- Based on:
 - ◆ Check of the coherence between 2 data
 - ◆ Check of the integrity of the secret parameters

Conclusion

- A new SPA-DFA-proof CRT RSA implementation
 - ◆ No need of e
 - ◆ No extra parameters to store
 - ◆ Only 3 more modular multiplications \Rightarrow the fastest method
- Based on:
 - ◆ Check of the coherence between 2 data
 - ◆ Check of the integrity of the secret parameters

- Full paper available at c.giraud@oberthurcs.com