Équipe de recherche commune

DE LA RECHERCHE À L'INDUSTRIE

SIKE: injection de fautes et contre-mesure sur la génération de clés

Élise Tasso (CEA)

Simon Pontié (CEA) simon.pontie@cea.fr

Nadia El Mrabet (MSE)

Luca De Feo (IBM)

Clément Gaine (CEA)

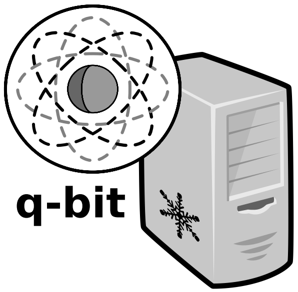
10 novembre 2021

Outline

1) Context
2) SIKE implementations
3) Known hardware attacks on SIKE
4) Ti's theoretical fault attack
5) Electromagnetic fault injection in a laboratory
6) Countermeasure and impacts

Tasso, É., De Feo, L., El Mrabet, N., & Pontié, S. (2021, October). Resistance of Isogeny-Based Cryptographic Implementations to a Fault Attack. In Constructive Side-Channel Analysis and Secure Design (COSADE) 2021.

Context: SIKE, a PQC KEM



Quantum computers have been shown to threaten classic asymmetric cryptography.

NIST Post Quantum Cryptography Standardization Contest for asymmetric cryptography algorithms (since 2016).

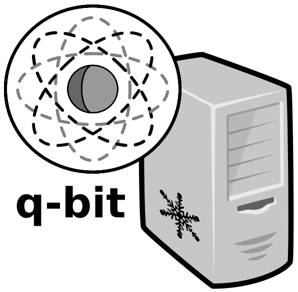
SIKE is one of the NIST round 3 alternate candidates for encryption and key encapsulation.

- The only one based on isogenies between elliptic curves

- Relatively slow: (intel CPU)
  x5.5 (9681 + 10343) kcycles for encapsulation + decapsulation vs
  (1862 + 1747) kcycles for the slowest among the other candidates at the lowest security level.

- Smallest public key size :
  ÷2 330 bytes (p434, uncompressed) vs
  672 bytes for the smallest key among the other candidates at the lowest security level

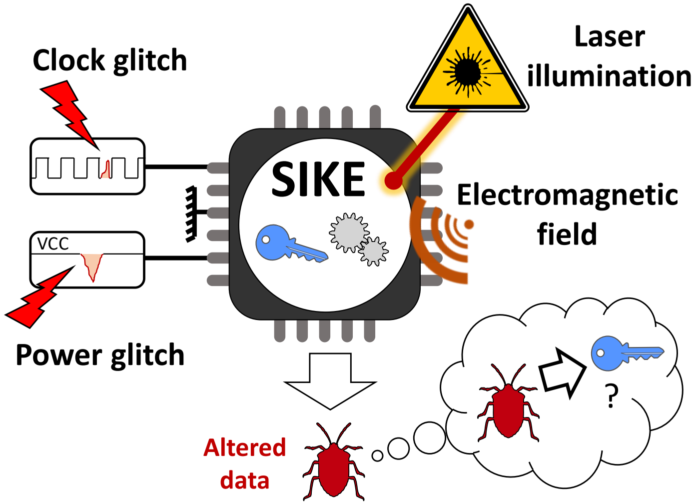
Context: SIKE, a PQC KEM



Quantum computers have been shown to threaten classic asymmetric cryptography.

NIST Post Quantum Cryptography Standardization Contest for asymmetric cryptography algorithms (since 2016).

SIKE is one of the NIST round 3 alternate candidates for encryption and key encapsulation.



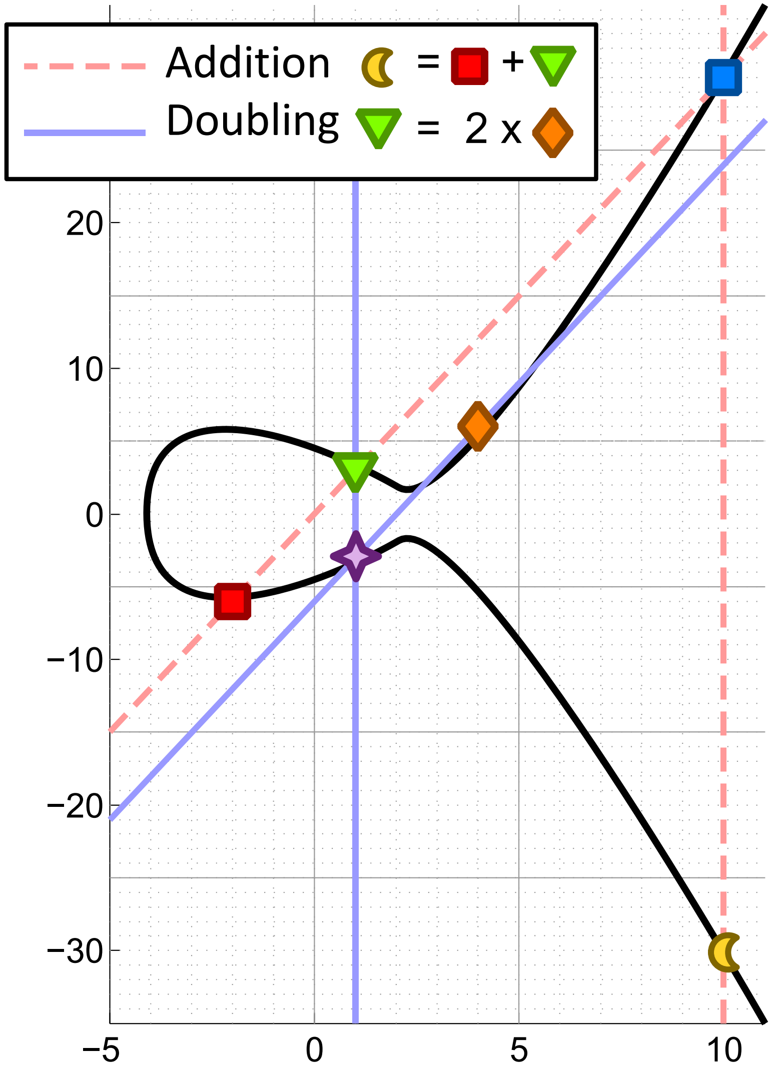
SIKE is believed to be mathematically secure, but physical attacks may exist depending on the implementation...

Is it possible to recover a secret with fault injection on a SIKE implementation?

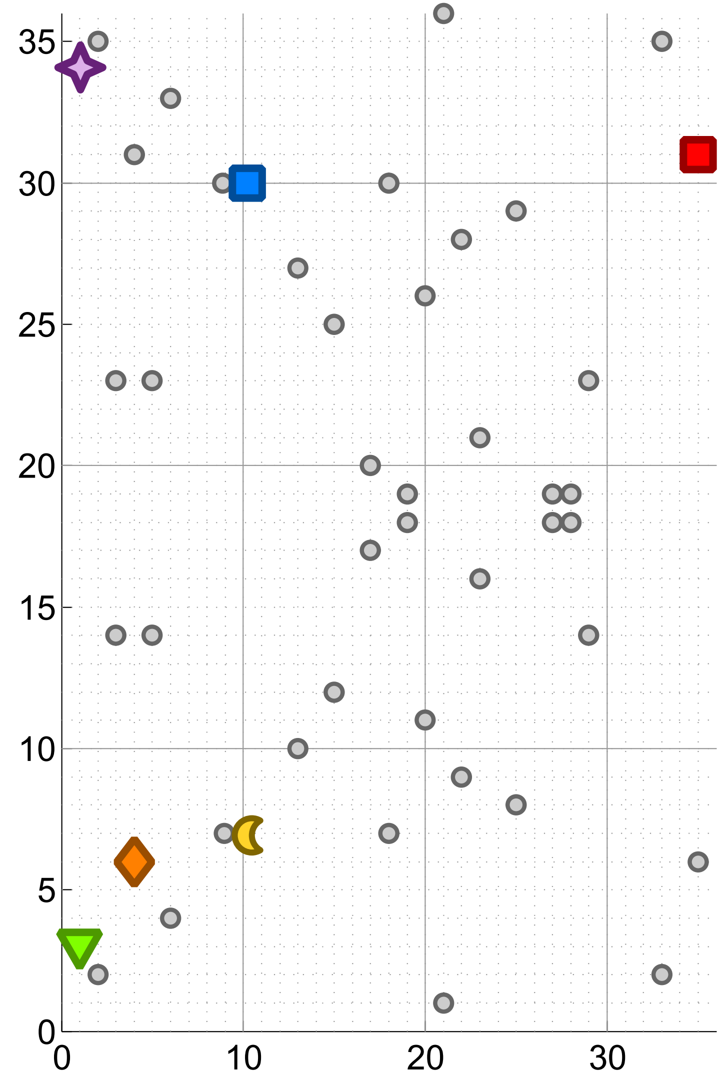
Which countermeasures can offer protection against these attacks?

Context: Elliptic curve

$$y^2 = x^3 - 12.x + 20$$

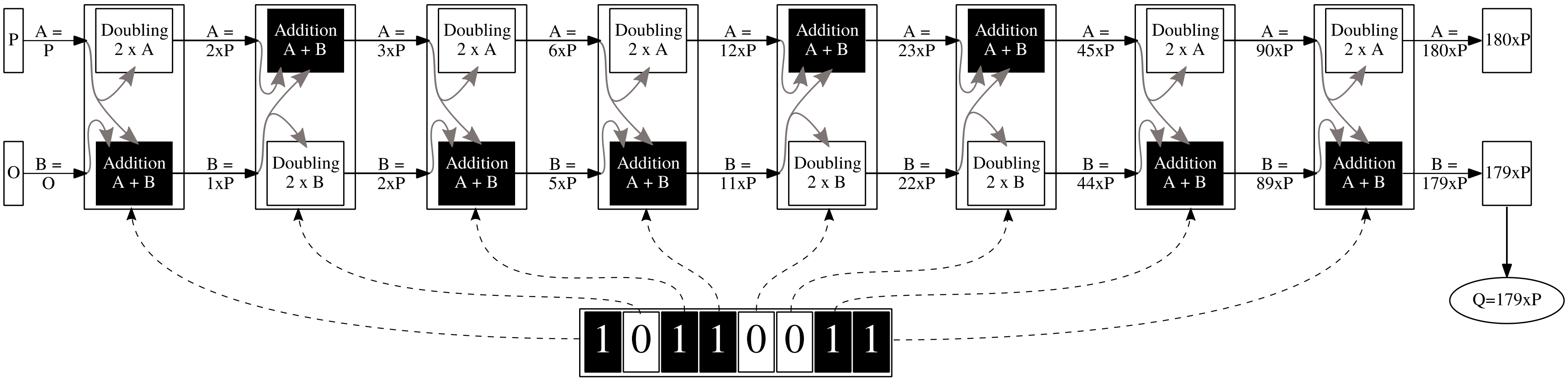


$$y^2 = x^3 - 12.x + 20 \ \ mod \ \ 37$$



Context: Elliptic curve cryptography

Example of a scalar multiplication computation with a small scalar

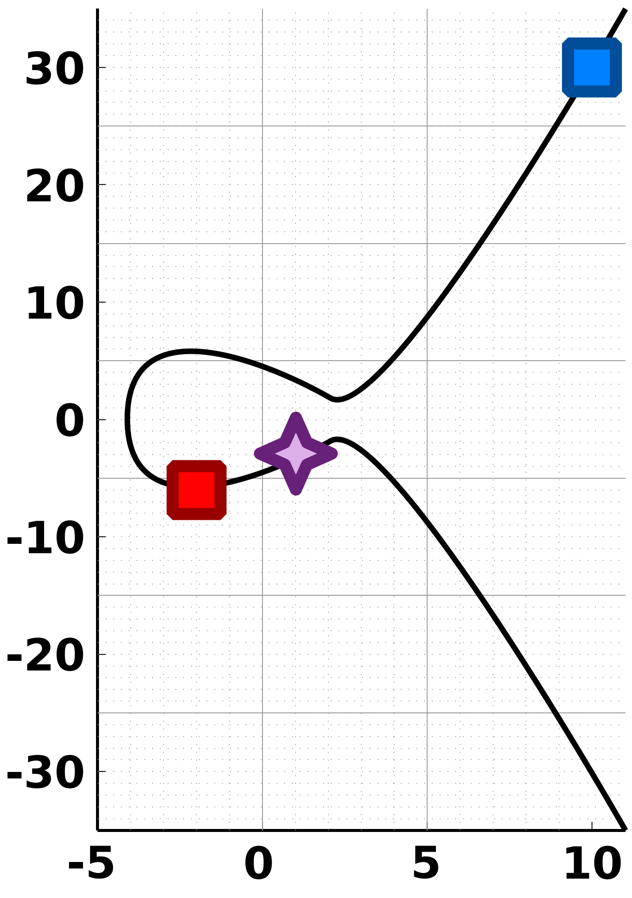


With large enough « k » and well chosen elliptic curve, it is hard to recover « k » from « P » and « Q = k x P ».

But supersingular curves are not well chosen.

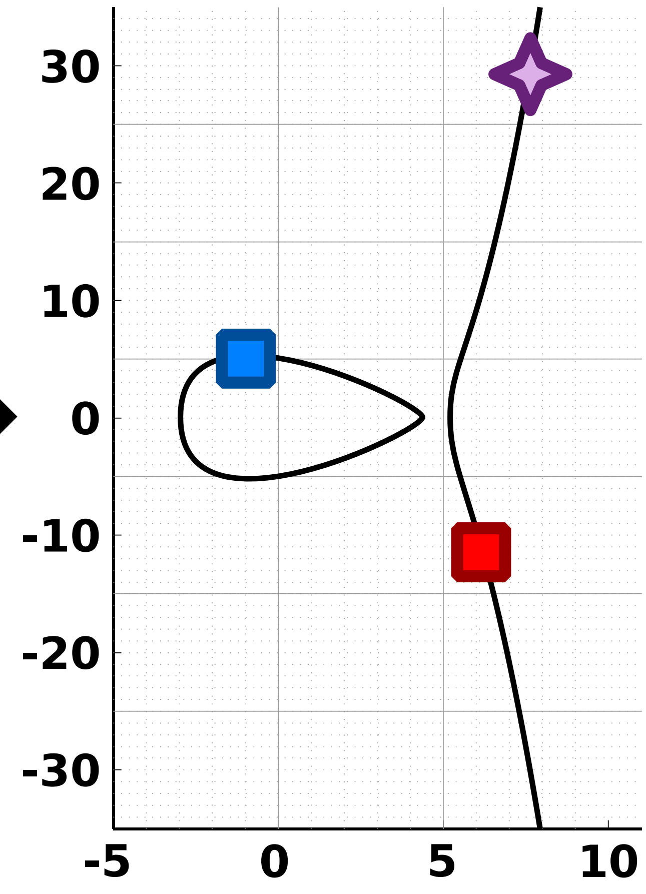
Context: Isogeny

$y^2 = x^3 + 12\,x^2 + x$



Isogeny

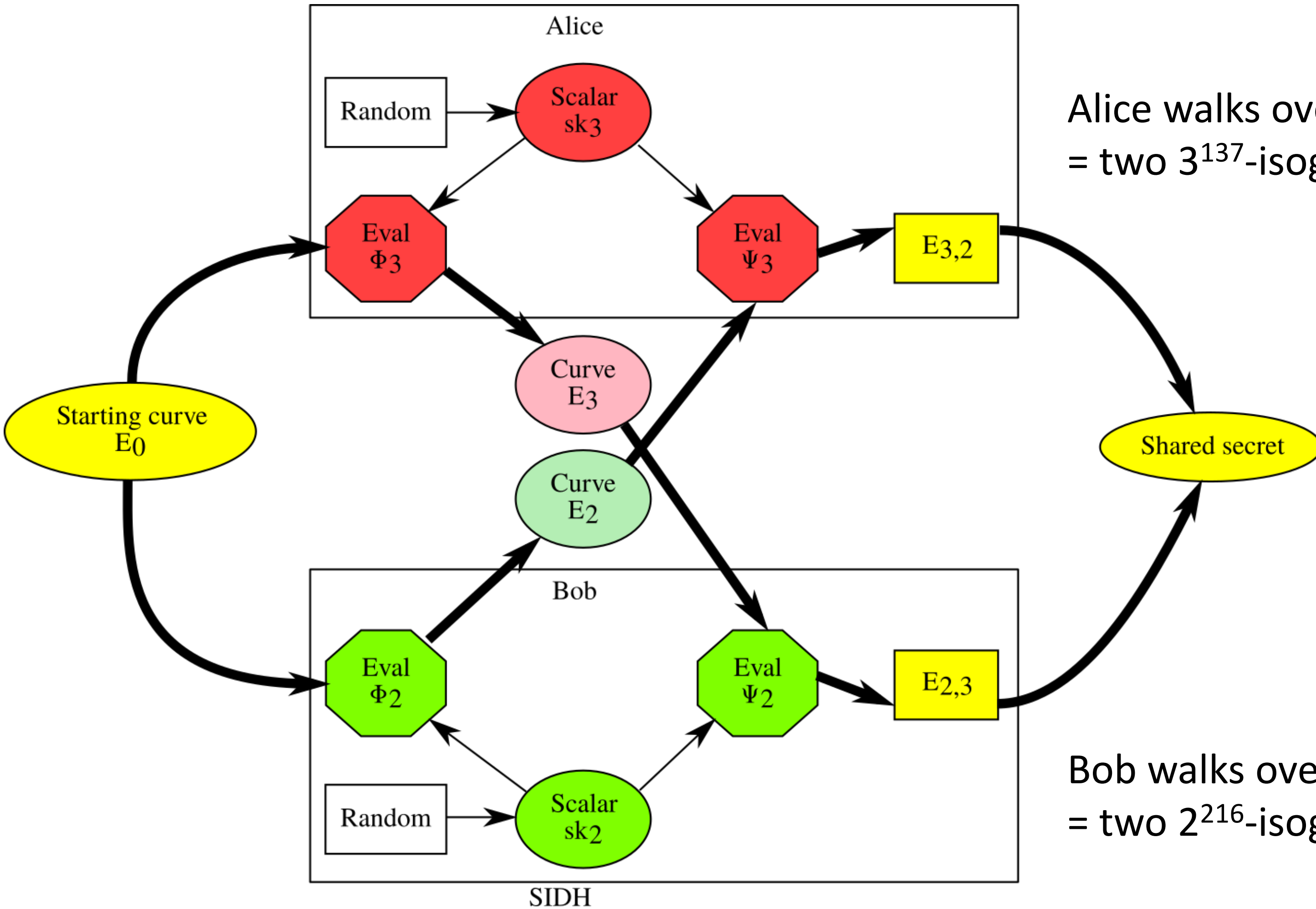
$y^2 = x^3 + (4124 + 123\,i)\,x^2 + x$



In SIKEp434, $(x, y) \in \mathbb{F}_{p^2},\ p = 2^{216}3^{137}$

Recovering the isogeny knowing the two curves is difficult if the isogeny order is large enough (isogeny order = the number of kernel elements) $(\sim 10^{65})$

SIDH: Supersingular isogeny Diffie-Hellman key exchange



Alice walks over the 3-isogeny graph
= two 3^{137}-isogenies

Bob walks over the 2-isogeny graph
= two 2^{216}-isogenies

SIDH: Supersingular isogeny Diffie-Hellman key exchange

How does Alice to compute φ and Ψ from the scalar ?

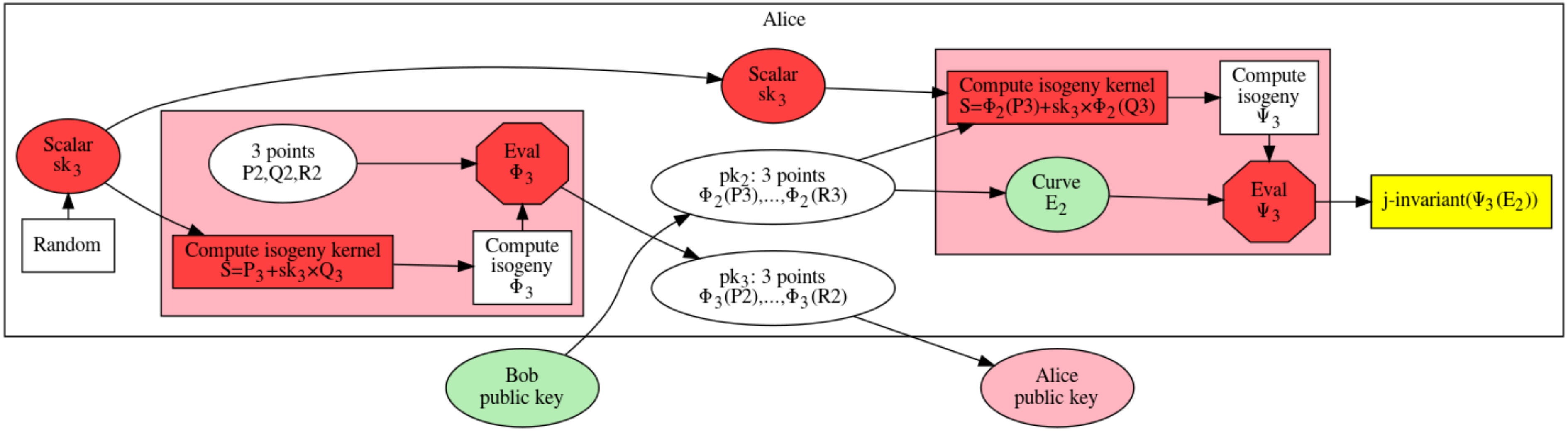
1. Alice computes the isogeny kernel generator S = P + scalar x Q

   P⊥Q, order(P) = order(Q) = 3^{137}

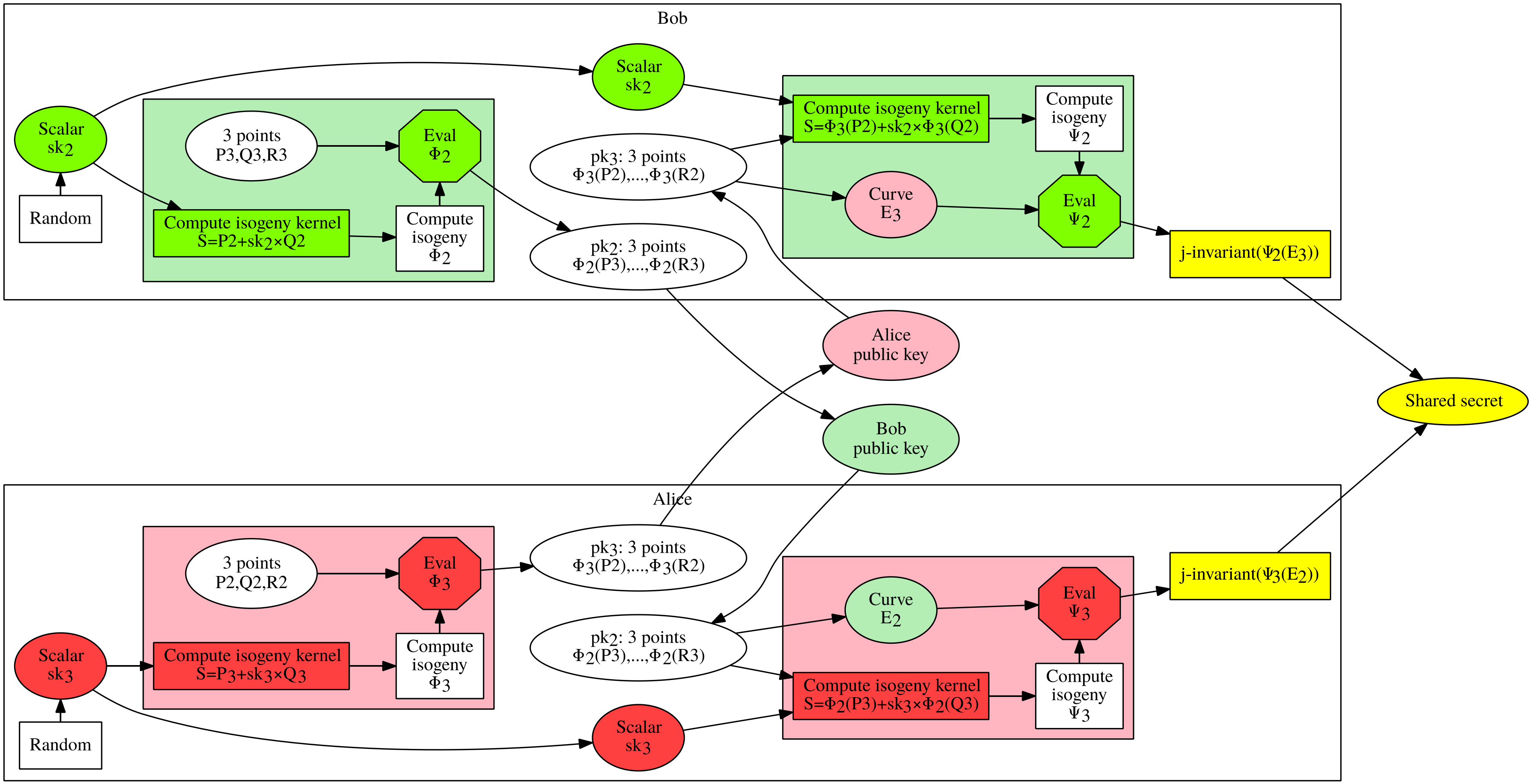
   For φ, P and Q are two fixed public points on the starting curve

   For Ψ, P and Q are image of these same points by the Bob secret isogeny.

2. Alice computes the isogeny map from its kernel generator



SIDH: Supersingular isogeny Diffie-Hellman key exchange

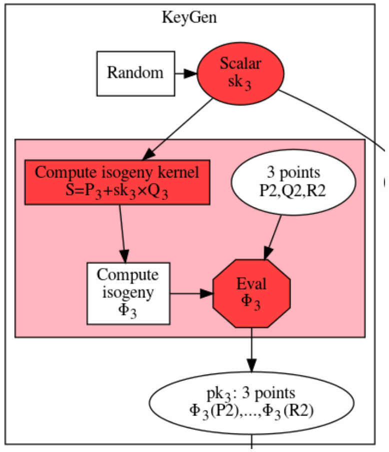


SIDH

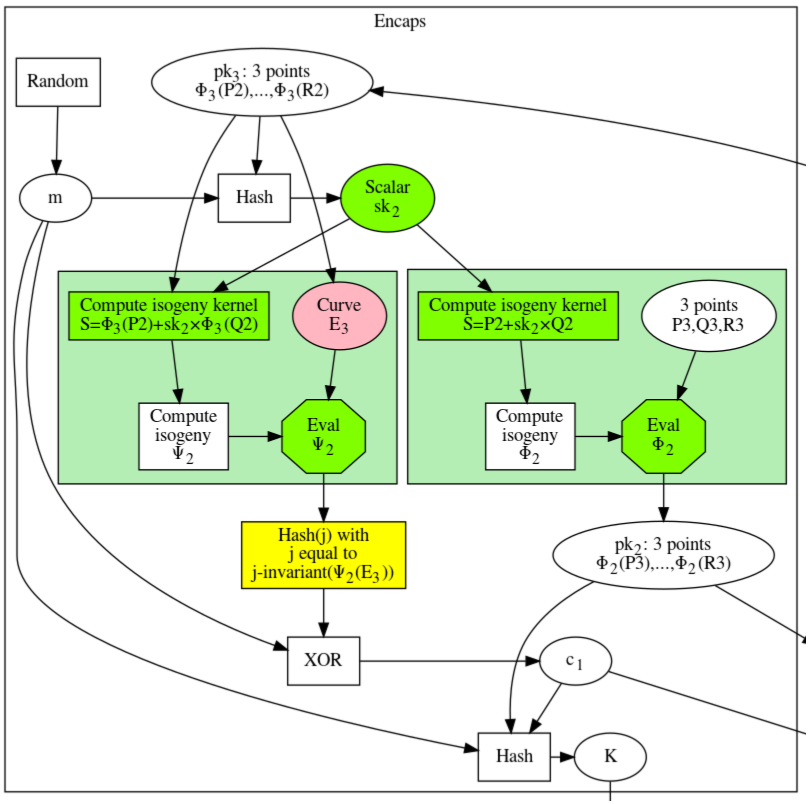
SIDH is mathematically insecure if one of the secret keys is static (Galbraith et al., 2016).
SIKE is mathematically secure in "semi-static mode".

SIKE: Supersingular Isogeny Key Encapsulation

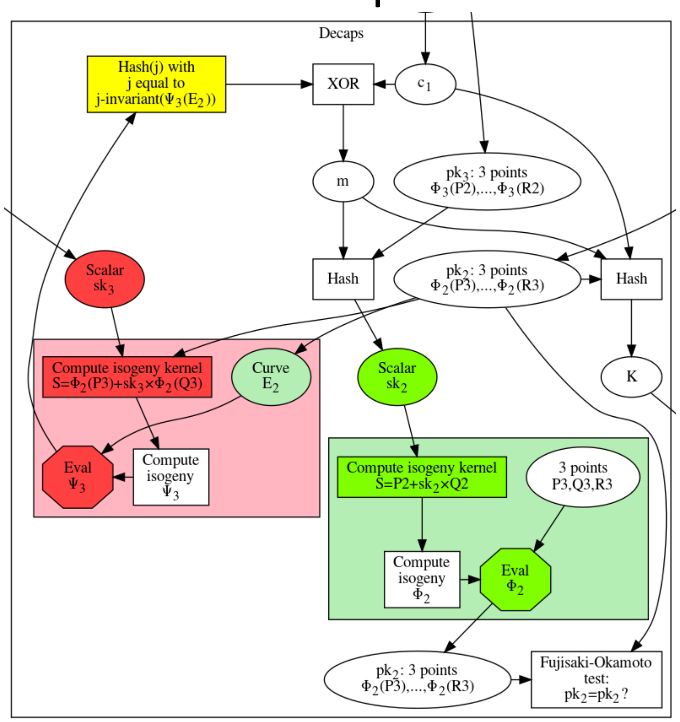
KeyGen



Encaps

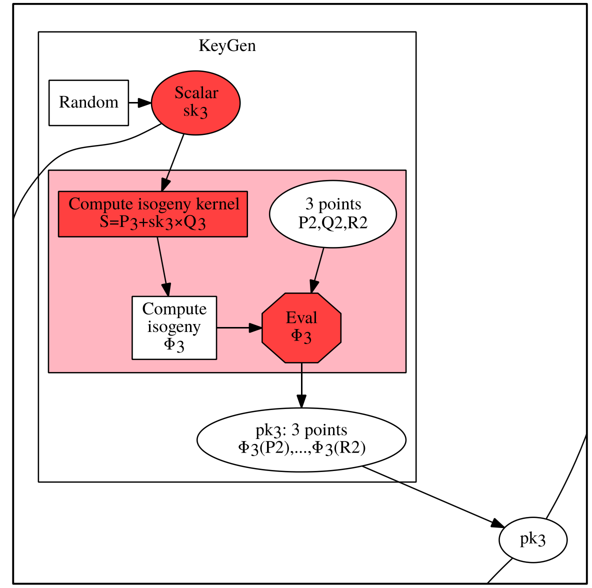


Decaps

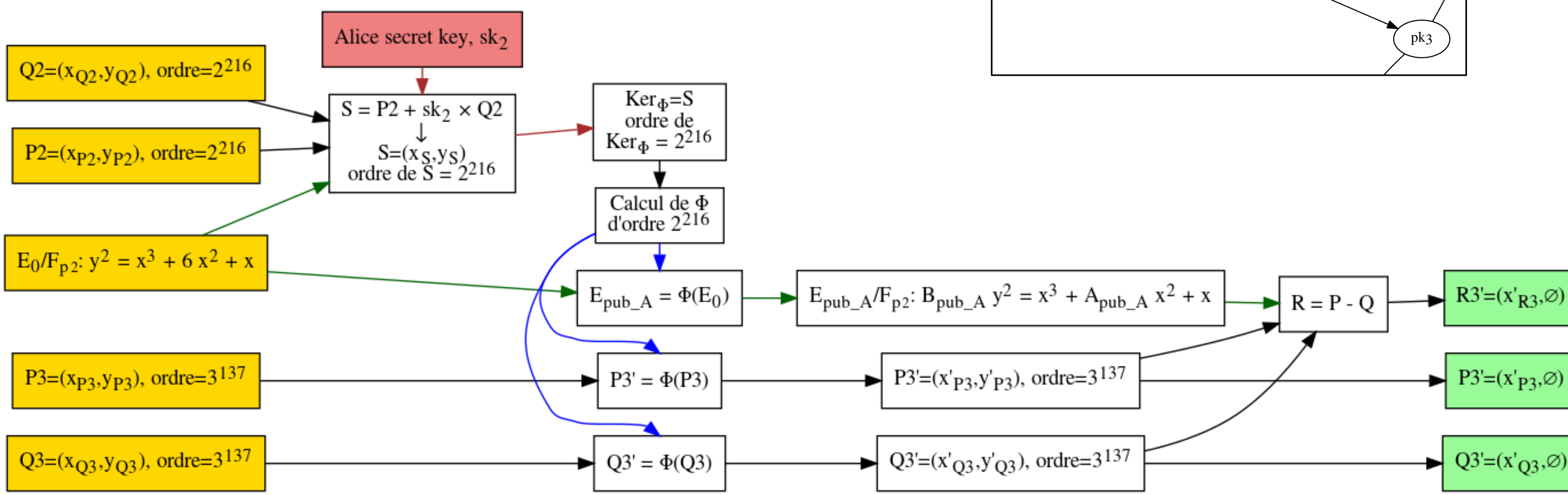


SIKE: Supersingular Isogeny Key Encapsulation

Key generation of SIKE

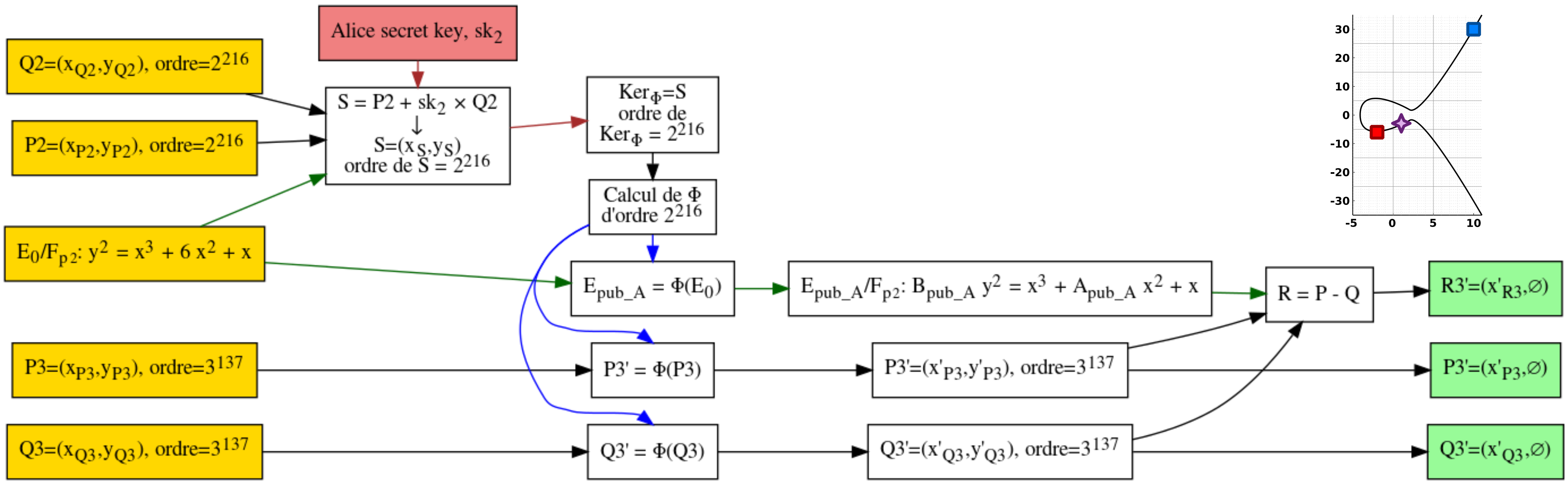


An isogeny kernel generator defines the isogeny

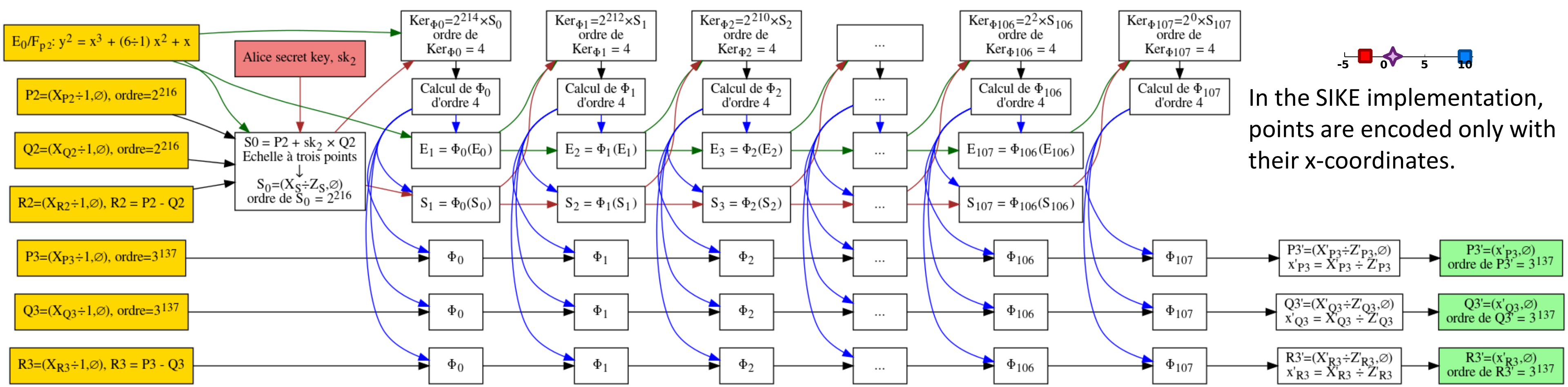


SIKEp434 Reference implementation, Round 3, Algorithm 42 = $isogen_2$ algorithm

SIKE: Supersingular Isogeny Key Encapsulation



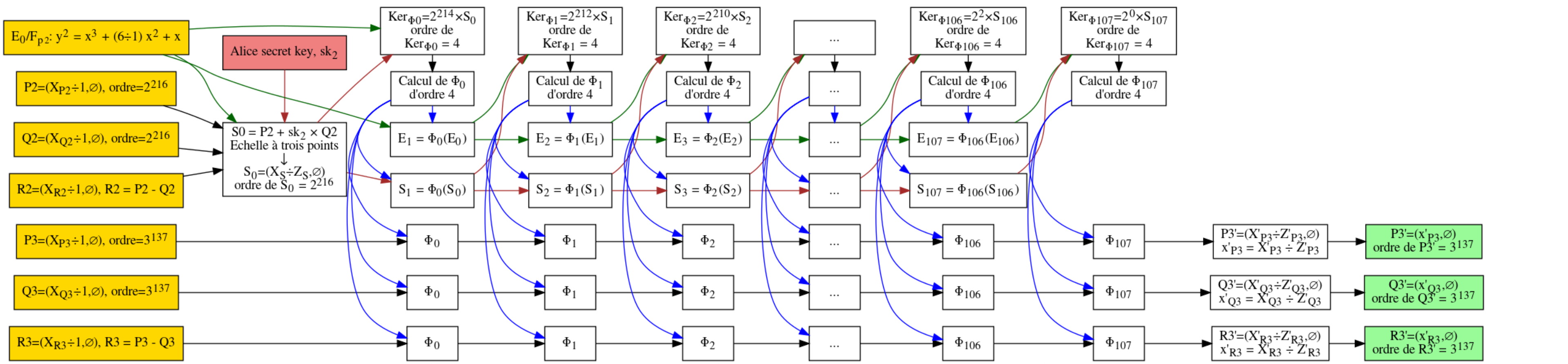
SIKEp434 Reference implementation, Round 3, Algorithm 42 = $isogen_2$ algorithm



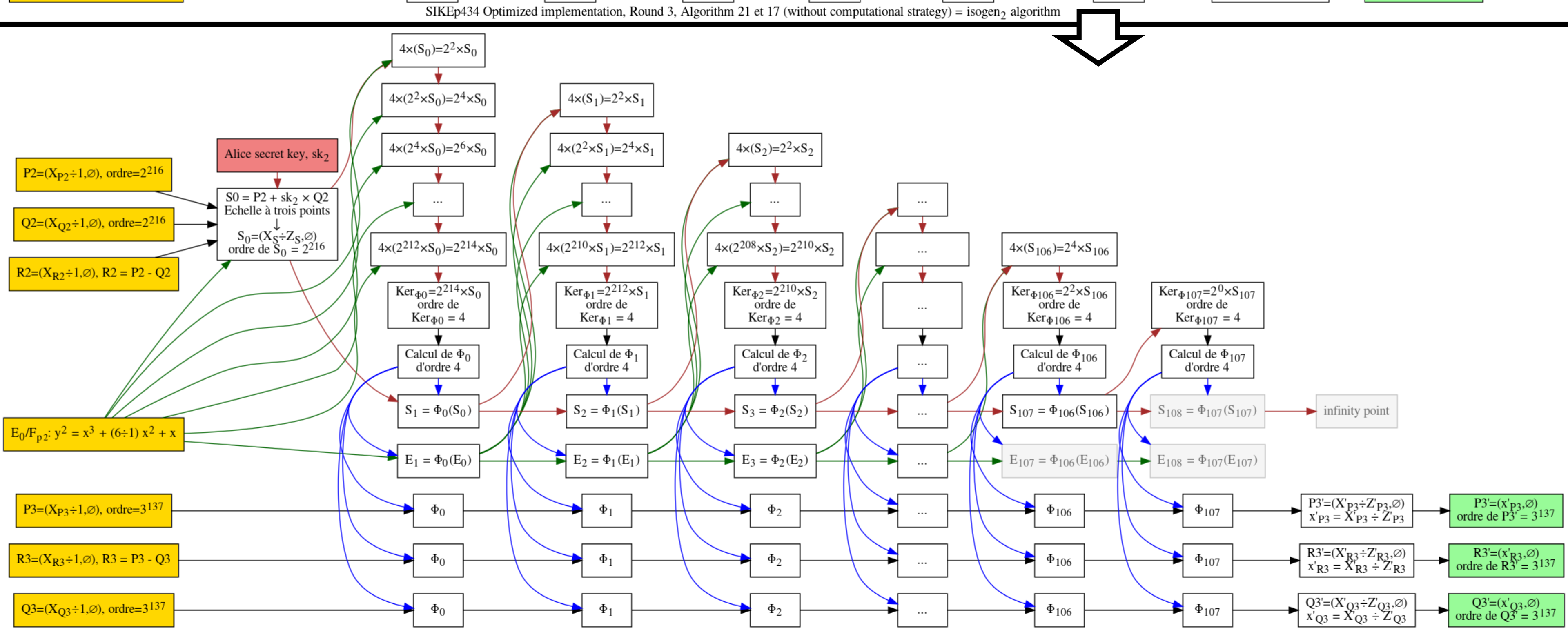
SIKEp434 Optimized implementation, Round 3, Algorithm 21 et 17 (without computational strategy) = $isogen_2$ algorithm

In the SIKE implementation, points are encoded only with their x-coordinates.

SIKE: Supersingular Isogeny Key Encapsulation

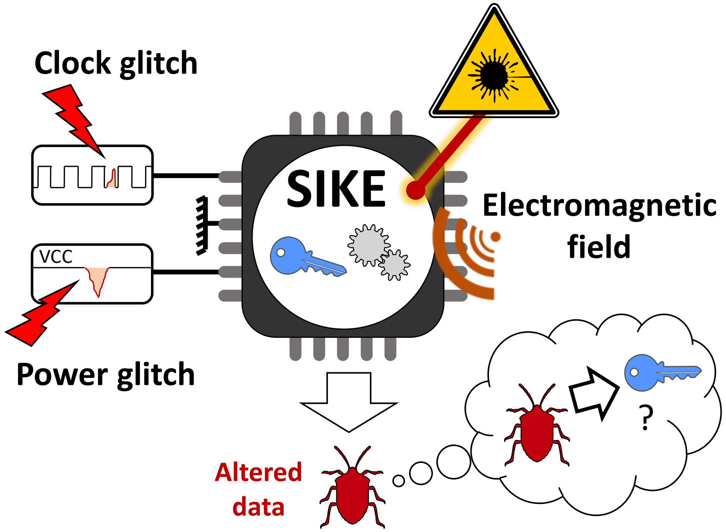


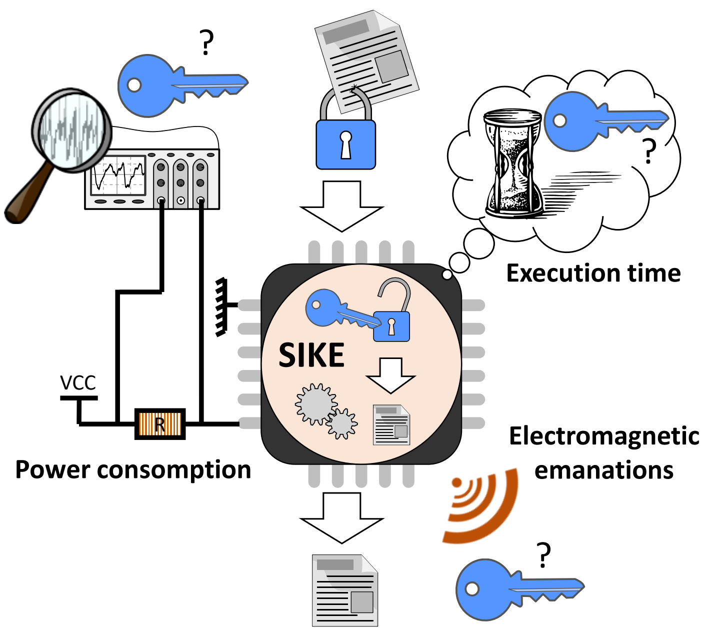
SIKEp434 Optimized implementation, Round 3, Algorithm 21 et 17 (without computational strategy) = isogen2 algorithm



SIKEp434 Optimized implementation, Round 3, Algorithm 21 et 17 (without computational strategy) = isogen2 algorithm

Known hardware attacks on SIKE





Side channel analysis

Theoretical	2017 Ti	Target isogeny
Simulated	2017 Gélin et al.	Target isogeny
Experimentally verified	This work	Target isogeny

Fault injection

Theoretical	2017 Koziel et al.	Target isogeny + scalar mult.
Simulated	X	
Experimentally verified	2018 Koppermann et al. 2020 Zhang et al. 2021 Genêt et al.	Target scalar mult.

SIKE key generation

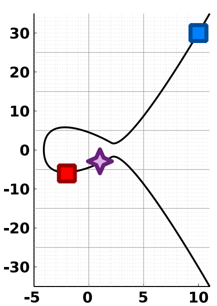
Focus on the SIKEp434 public key generation

Key generation input

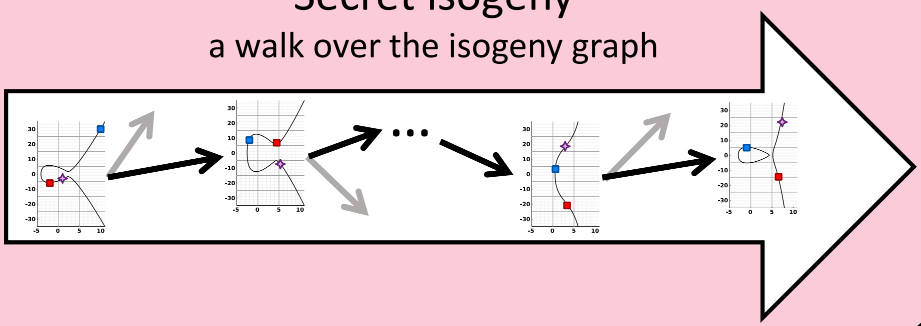
Secret scalar
randomly chosen in $[0, 2^{216}[$

Secret point
kernel of the secret isogeny

Starting elliptic curve and points on this curve

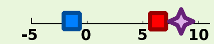


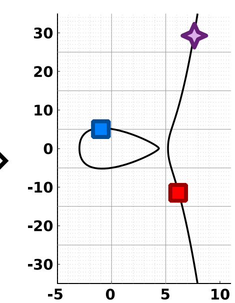
Secret isogeny
a walk over the isogeny graph



Key generation output

Public key
x-coordinate of 3 points

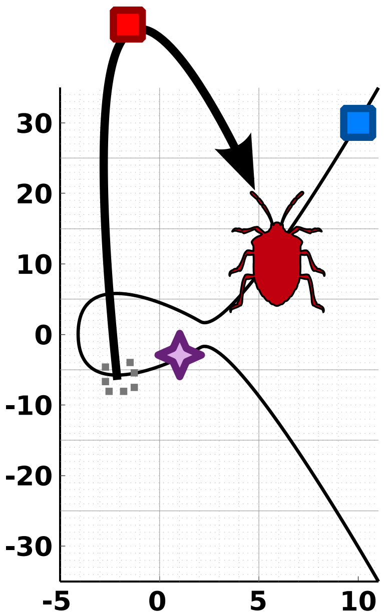




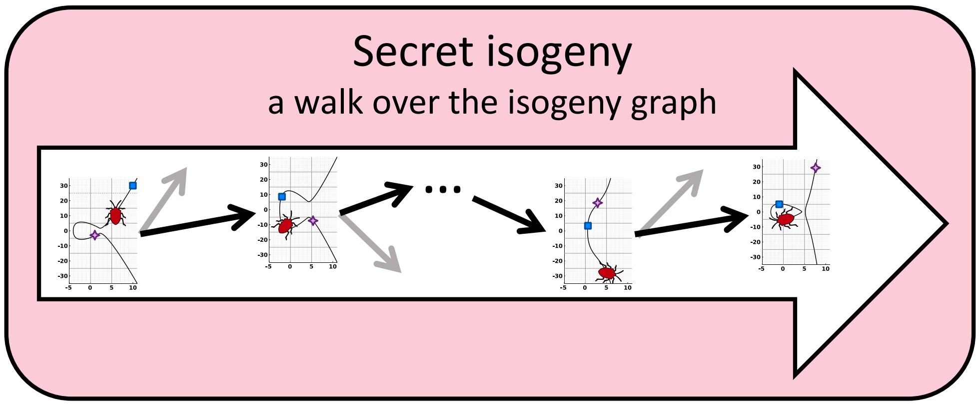
Attacker goal: recover the secret as a secret scalar, a secret point or a secret isogeny.

Ti's theoretical fault attack on isogeny-based cryptography

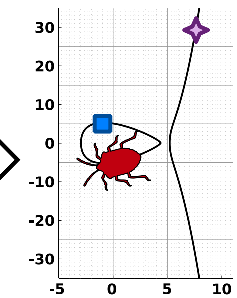
Theoretical fault model

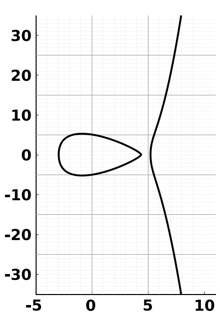


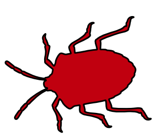
Altered point on the starting elliptic curve



Public elliptic curve with an altered point







Altered point on the public elliptic curve coloured by the secret

Correct public elliptic curve

Ti’s attack

If [bug] is a point on the starting curve.
If [bug] image is a point of order $2^i.3^j$ with i~216.

Recovered secret

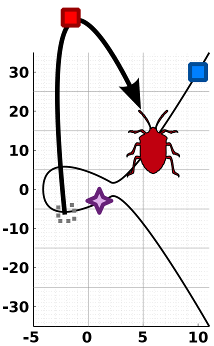
Ti, Y.B.: Fault attack on supersingular isogeny cryptosystems. In: International Workshop on Post-Quantum Cryptography. pp. 107-122. Springer (2017)

Ti's theoretical fault attack on isogeny-based cryptography

is fixed point on the starting curve (order = 3^{137}).

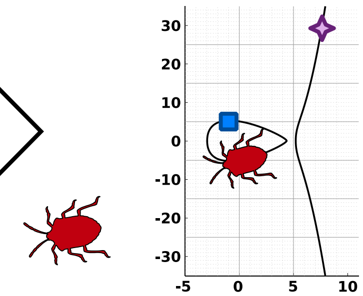
Fault injection

Becomes a random point on the starting curve (order = $2^i.3^j$, $0<=i<=216$, $0<=j<=137$).



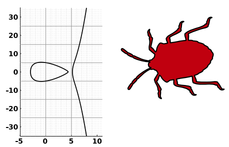
Secret isogeny (order = 2^{216})
a walk over the isogeny graph

Image of the random point by the secret isogeny is computed



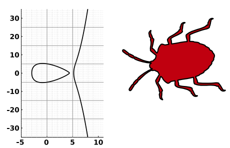
Should be a point of order = 3^{137} but will be a point of order = $2^k.3^j$ with $0<=k<=i<=216$

If k = 216, the output point is full (216/216) coloured by the secret



Ti's attack → Secret = a 2^{216}-isogeny

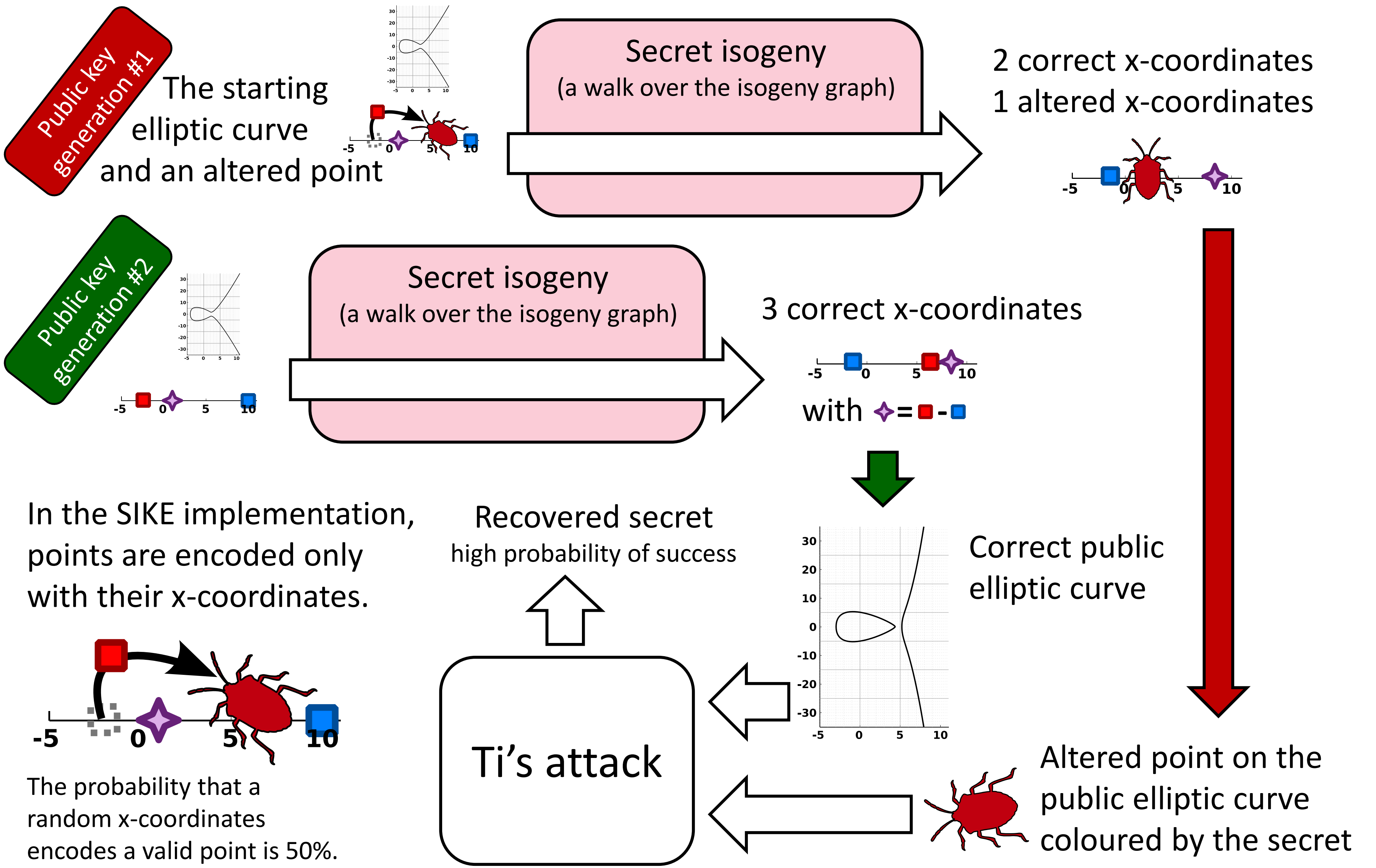
If 1 < k < 216, the output point is partially (k/216) coloured by the secret



Ti's attack → Bruteforce to recover a $2^{(216-k)}$-isogeny → Secret = 2^k-isogeny o 2^{216-k}-isogeny

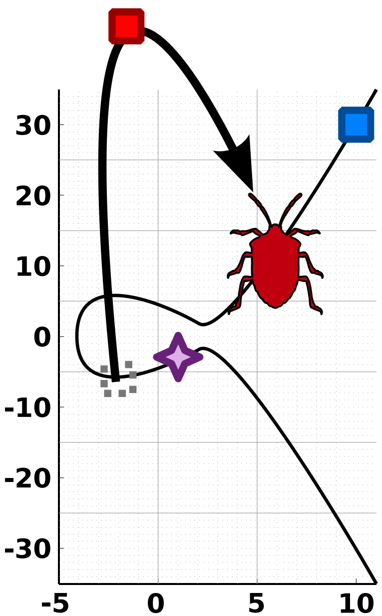
Ti, Y.B.: Fault attack on supersingular isogeny cryptosystems. In: International Workshop on Post-Quantum Cryptography. pp. 107-122. Springer (2017)

Why the attack requires 2 public keys generated from the same secret?



Fault injection in a laboratory

How to alter point on the starting elliptic curve ?



Starting points are loaded from program memory to data memory at beginning of key generation.

```
bl      0x404eb0 <init_basis>
ldr     x4, [x0] // from prgm memory
str     x4, [x1] // to   data memory
ldr     x4, [x0, #8]
str     x4, [x1, #8]
ldr     x4, [x0, #16]
str     x4, [x1, #16]
...
ldr     x1, [x0, #112]
str     x1, [x2]
ldr     x0, [x0, #120]
str     x0, [x3, #8]
...
ldr     x1, [x0, #320]
str     x1, [x3, #96]
ldr     x0, [x0, #328]
str     x0, [x3, #104]
ret
```

Loading of point (red square): first block of ldr/str instructions.
Loading of point (blue square): second block.
Loading of point (purple star): third block.

To alter the loading of starting points, we can disrupt the execution of one of the 84 consecutive `ldr` or `str` instructions.

With the skip instruction model, only 56/84 (66%) of these instructions are sensitive.

We use the ARMv8-A SIKE round 3 implementation: https://github.com/microsoft/PQCrypto-SIDH/tree/f43c9f74

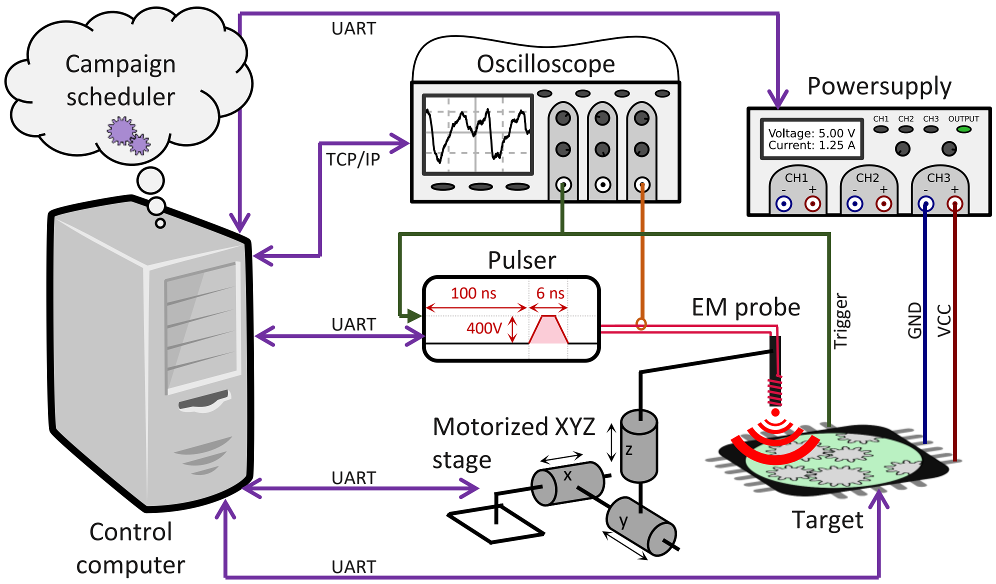
Electromagnetic fault injection (EMFI) in a laboratory: EMFI on SoC

How to alter the execution of one of the 84 consecutive `ldr` or `str` instructions ?

- A great timing precision is not necessary to perform this attack.
- We have therefore chosen to alter the algorithm execution on a SoC with EMFI.
- As SoC latency is difficult to predict, targeting a specific instruction is hard.

→ Chosen target is a SoC with four ARM Cortex-A53 cores (@1.2 GHz)

Our EMFI setup

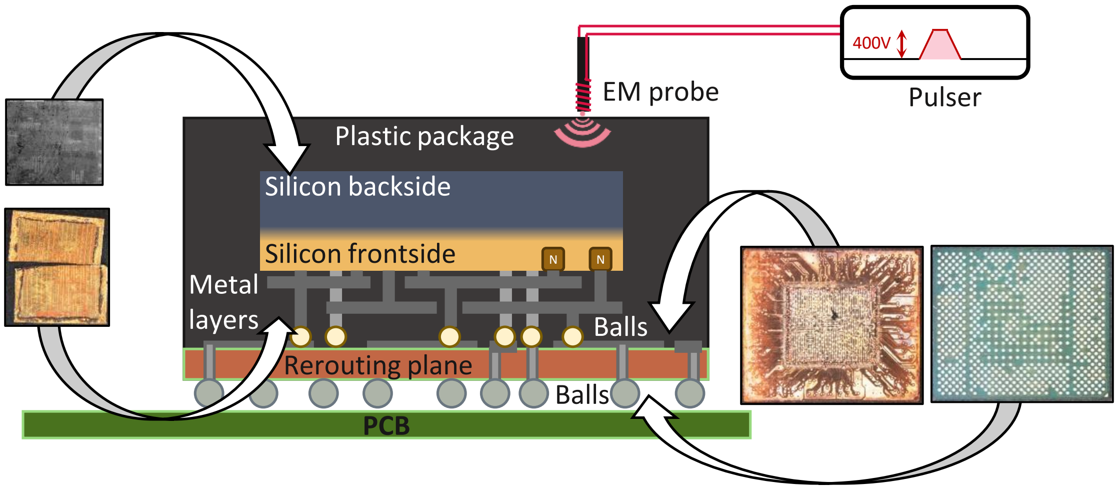


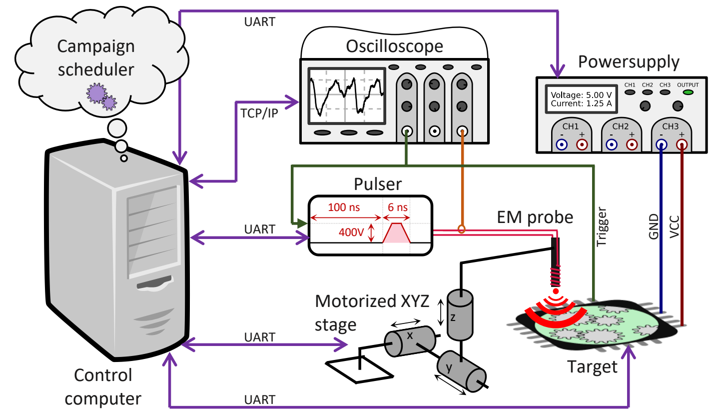
Two software codes under attack:

1) ARMv8-A dedicated code to tune the EMFI setup:
   - Search interesting probe location,
   - Search interesting amplitude range
2) ARMv8-A SIKE round 3 implementation to verify feasibility of the Ti's attack:
   - fixed probe location,
   - fixed pulse width,
   - Fine grain amplitude searching
   - Delay searching

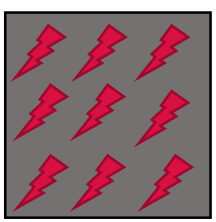
Gaine, C., Aboulkassimi, D., Pontié, S., Nikolovski, J.P., Dutertre, J.M.: Electromagnetic fault injection as a new forensic approach for SoCs. In: 2020 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1-6. IEEE (2020)

Electromagnetic fault injection (EMFI) in a laboratory: Where, When, What ?

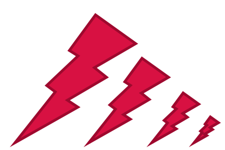




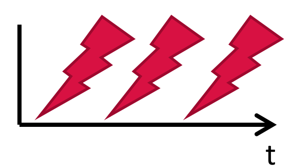
The silicium chip is in a flip-chip package



Where to fire ?



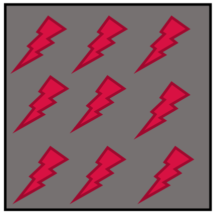
What power ?



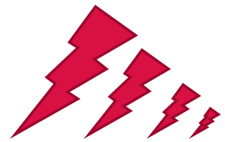
When to fire ?

Gaine, C., Aboulkassimi, D., Pontié, S., Nikolovski, J.P., Dutertre, J.M.: Electromagnetic fault injection as a new forensic approach for SoCs. In: 2020 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1-6. IEEE (2020)

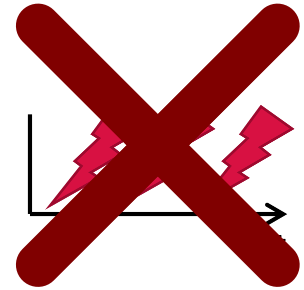
Electromagnetic fault injection (EMFI) in a laboratory: Where, When, What ?



Where to fire ?



What power ?

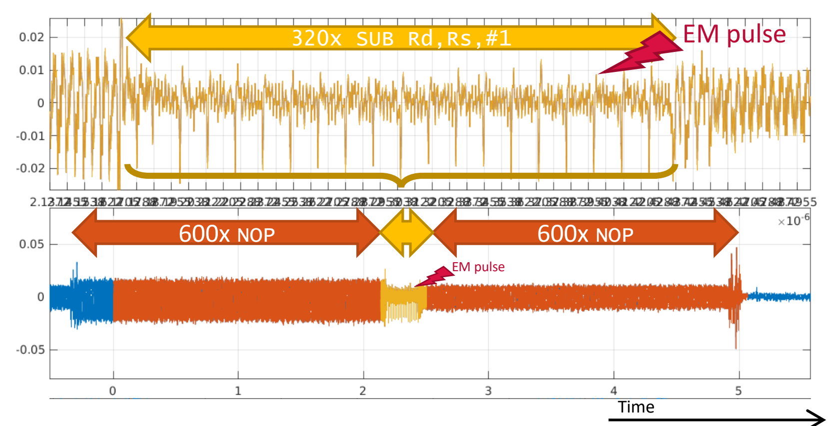


When to fire ?

A first software code to reduce dimensions of the exploration:

```
//Initialisation x28 = 368 = 0x170
mov x28, #0170
//Following sequences repeated 32 times
sub x19, x28, #0x1
sub x20, x19, #0x1
sub x21, x20, #0x1
...
sub x28, x27, #0x1
```

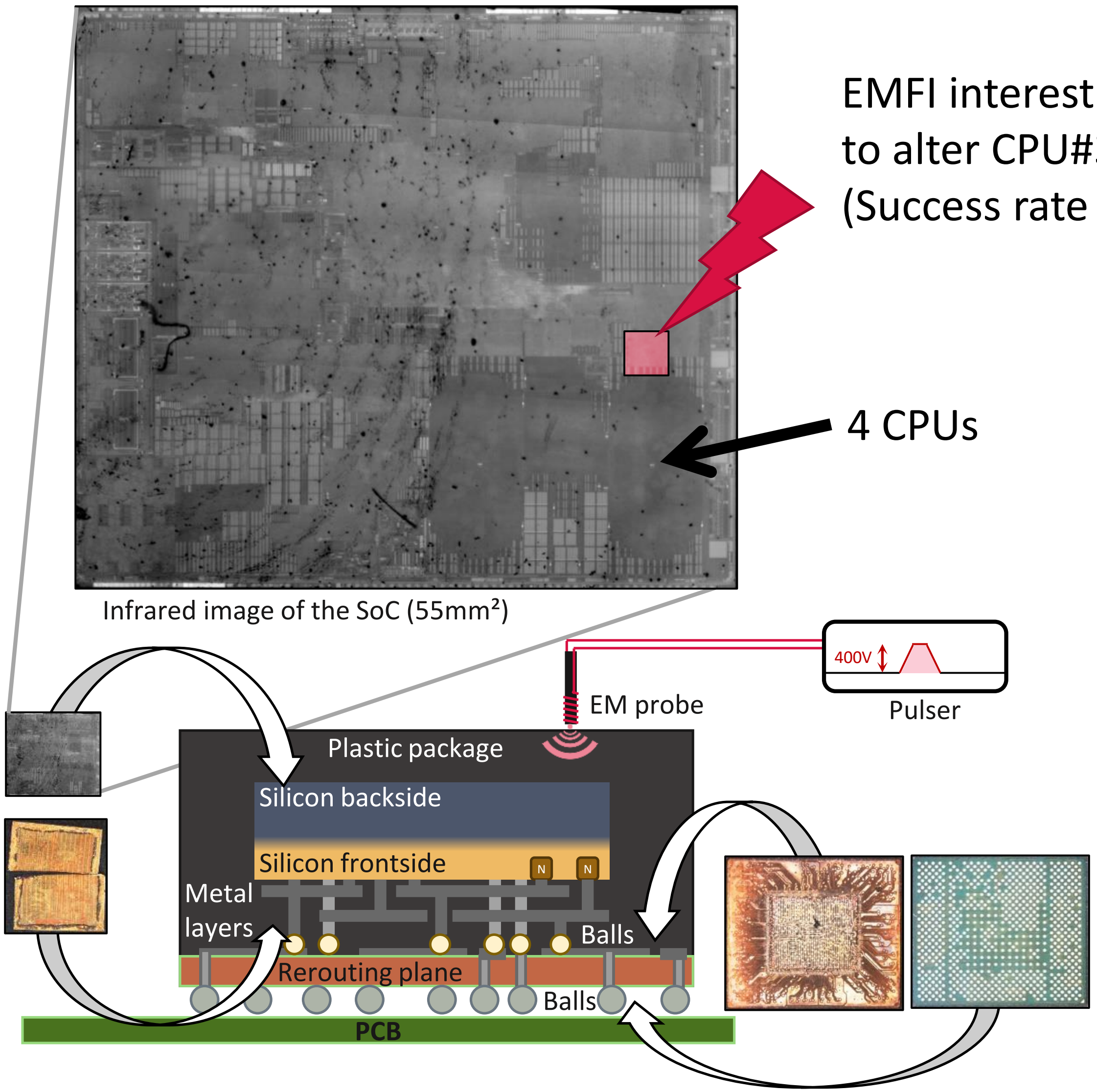
A successful altering of one execution of the 320 subtractions will be observable



SEMA traces of the code under attack

Gaine, C., Aboulkassimi, D., Pontié, S., Nikolovski, J.P., Dutertre, J.M.: Electromagnetic fault injection as a new forensic approach for SoCs. In: 2020 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1-6. IEEE (2020)

Electromagnetic fault injection (EMFI) in a laboratory: Where to fire ?



EMFI interesting area (0.4mm²) to alter CPU#3 execution (Success rate ≈ 30%)

4 CPUs

The pulse voltage must be around 400V for this SOC

The code under attack is one of the running applications on the Linux Yocto OS.

The target application is limited to one CPU.

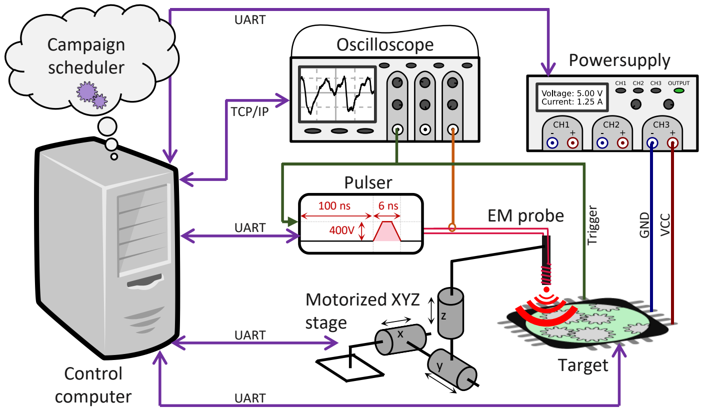
The SoC frequency is fixed (to avoid DVFS effect)

A ø750 µm probe

Gaine, C., Aboulkassimi, D., Pontié, S., Nikolovski, J.P., Dutertre, J.M.: Electromagnetic fault injection as a new forensic approach for SoCs. In: 2020 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1-6. IEEE (2020)

Electromagnetic fault injection (EMFI) in a laboratory: Where to fire ?

EMFI



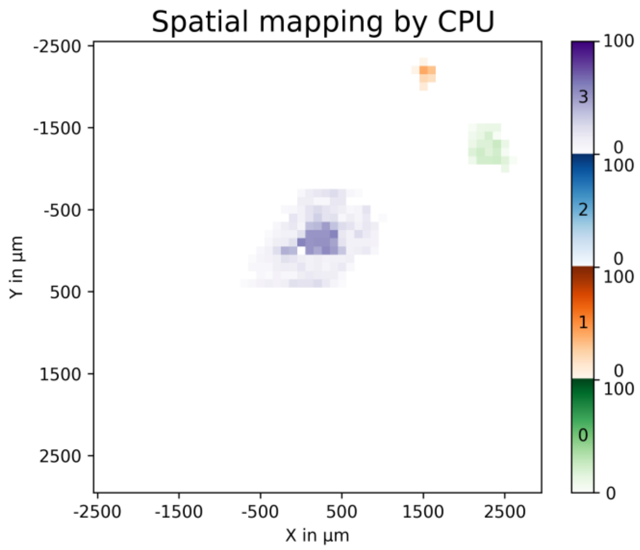
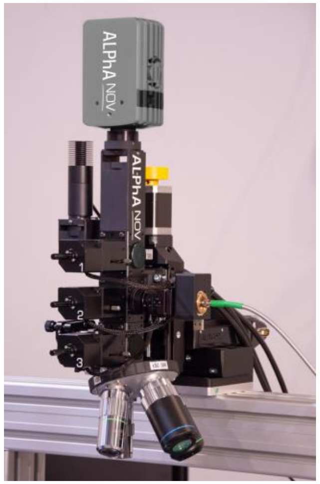
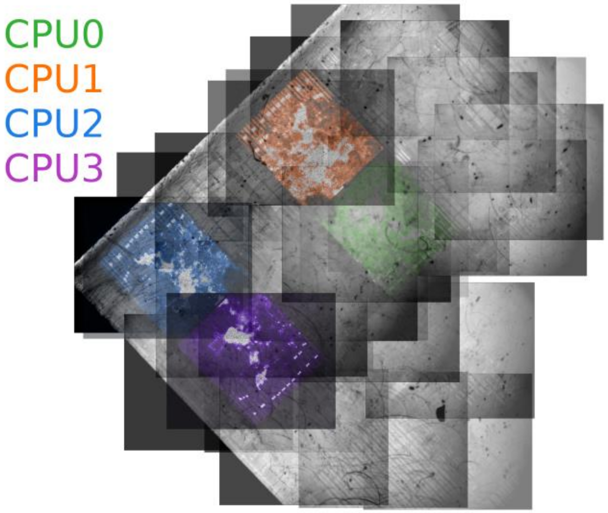


Photo-emission camera



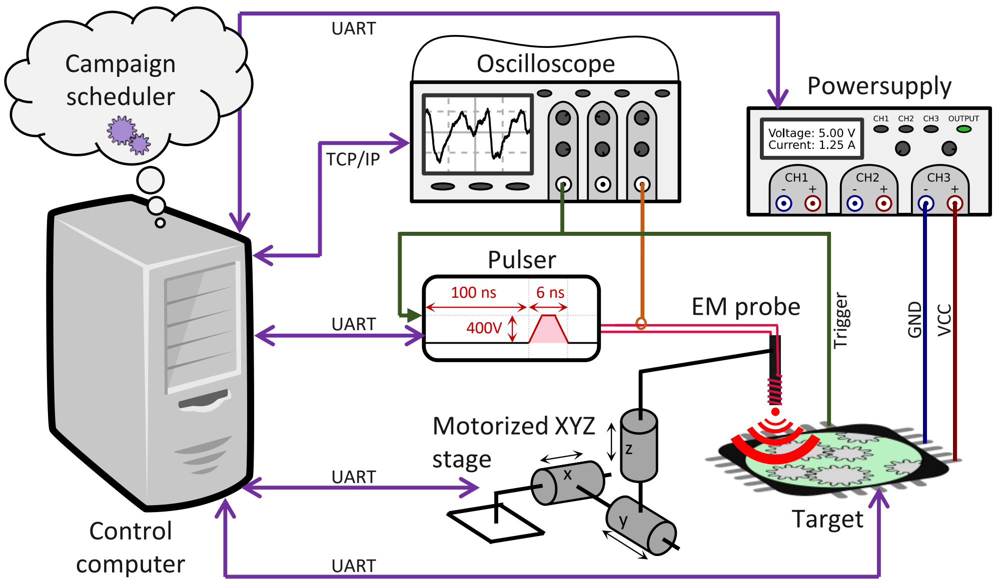


Electromagnetic fault injection (EMFI) in a laboratory: EMFI on SIKE implementation

How to alter the execution of one of the 84 consecutive `ldr` or `str` instructions ?

- A great timing precision is not necessary to perform this attack.
- We have therefore chosen to alter the algorithm execution on a SoC with EMFI.
- As SoC latency is difficult to predict, targeting a specific instruction is hard.

Our EMFI setup



ARMv8-A SIKE round 3 implementation on a SoC with four ARM Cortex-A53 cores at a 1.2 GHz frequency with:

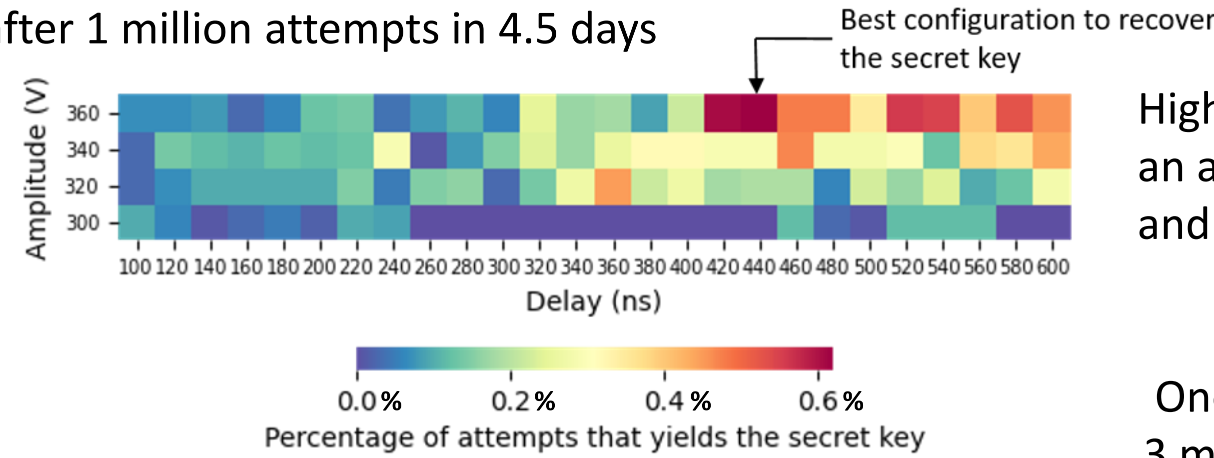
- fixed probe location,
- fixed pulse width,
- fixed SoC frequency and
- execution limited to one CPU.

Goal: find the best (amplitude,delay) configuration to recover the secret.

Gaine, C., Aboulkassimi, D., Pontié, S., Nikolovski, J.P., Dutertre, J.M.: Electromagnetic fault injection as a new forensic approach for SoCs. In: 2020 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1-6. IEEE (2020)

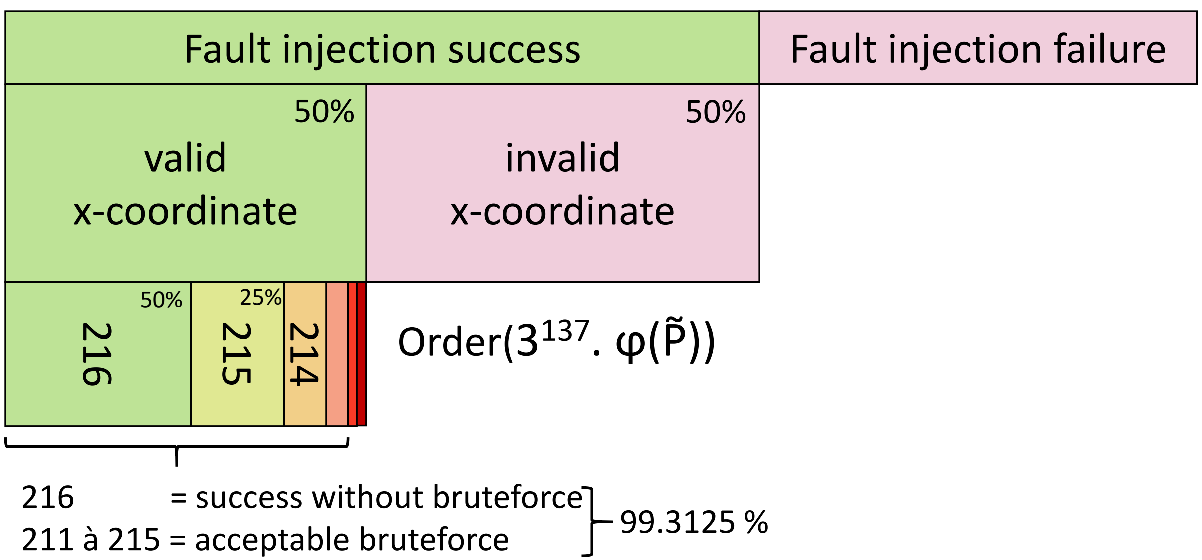
Results and conclusion

Results after 1 million attempts in 4.5 days



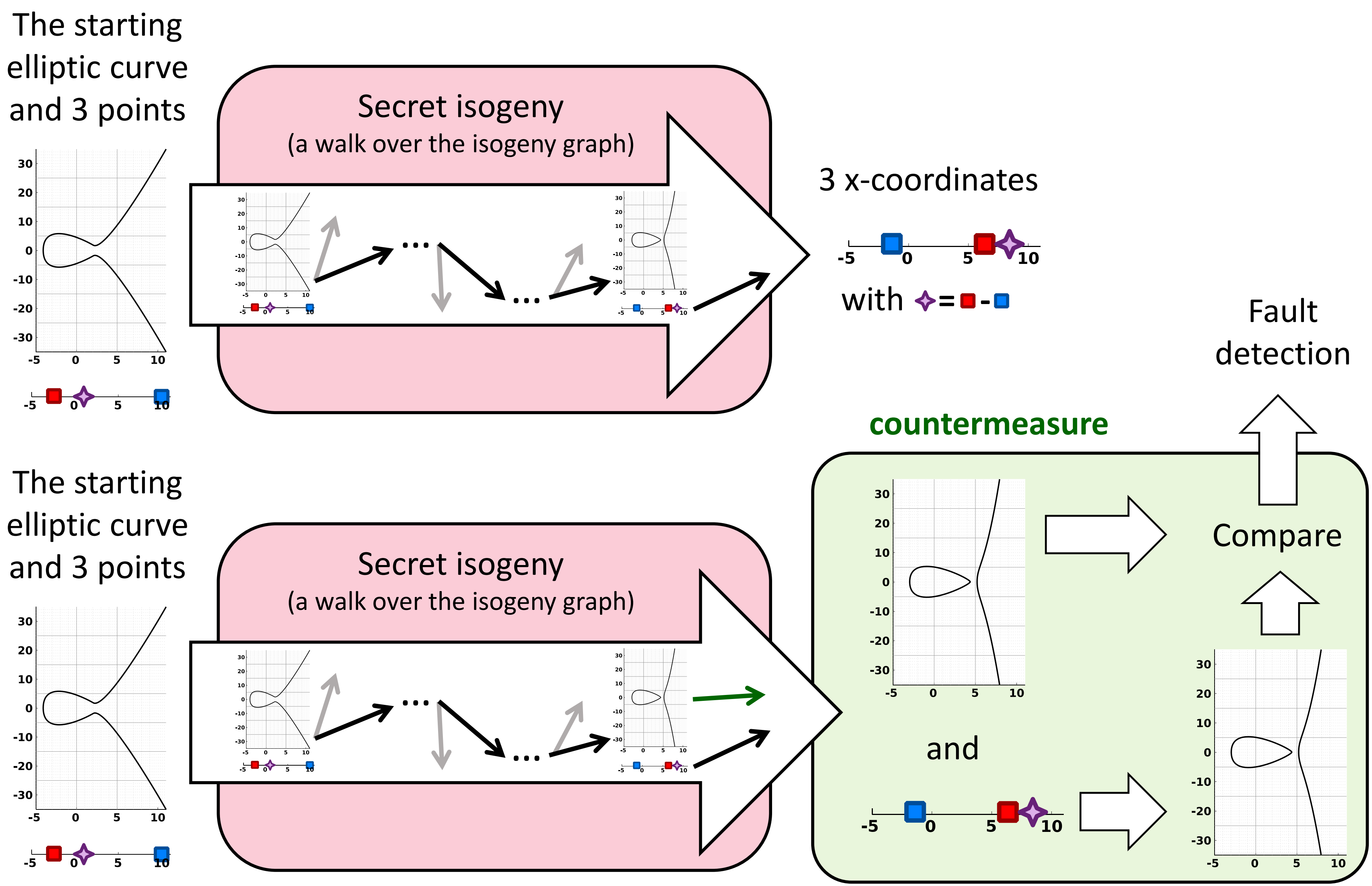
Highest success rate for an amplitude of 360 V and a delay of 440 ns : 0.62 %.

One secret is found every 3 minutes and 10 seconds.



Success probability = Fault injection success probability x 50 % x 99.3125 % = 0.62 %

Countermeasure based on existing redundancy



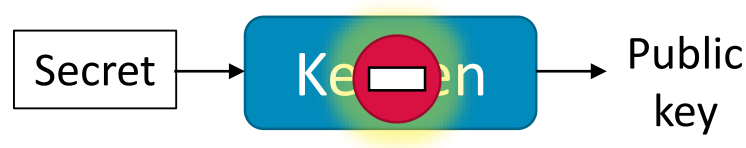
Countermeasure and impacts

Impacts

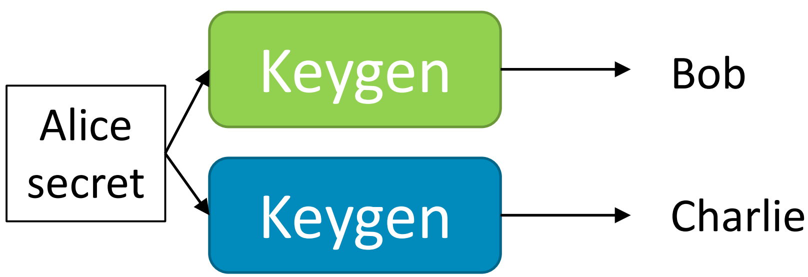
- SIKE is not broken, unless it is incorrectly implemented because generating twice the public key from the same secret is not compliant with the KEM API.

Generate the public key just after the secret generation

Never compute again the public key associated to the secret



- But preventing two public key generations is not possible in a multipartite key exchange.



Protection

- We propose a countermeasure that takes advantage of redundancy in SIKE's code and is cheap: there is a 1.5% overhead that can be further reduced.
- The probability to detect a fault is high: 1.67×10^{-261} for SIKEp434.

Tasso, É., De Feo, L., El Mrabet, N., & Pontié, S. (2021, October). Resistance of Isogeny-Based Cryptographic Implementations to a Fault Attack. In Constructive Side-Channel Analysis and Secure Design (COSADE) 2021.

Thanks for your attention

Simon Pontié

simon.pontie@cea.fr