Fault attack on SQIsign

Jeonghwan Lee[1], Donghoe Heo[1], Hyeonhak Kim[1],
Gyusang Kim[1], Suhri Kim[2], Heeseok Kim[1], Seokhee Hong[1]

[1]Korea University

[2]Sungshin Women's University

PQCrypto 2024, University of Oxford

*We are from south ☺

Contents

SQIsign in a nutshell

SQIsign in a nutshell

SQIsign

- Is the only isogeny-based digital signature in candidates of NIST standardization process
- Is Fiat-Shamir heuristic-type digital signature
- Has the most compact keys and signature size
- Exploits the Deuring correspondence

j-invariant of a supersingular curve E	A maximal order in $B_{p,\infty}$ ($\mathcal{O} \cong End(E)$)
An isogeny $\varphi : E_s \to E_e$	An integral $(\mathcal{O}_s, \mathcal{O}_e)$-ideal
$\deg(\varphi)$	$nrd(I_\varphi)$
$\varphi : E_s \to E_e, \psi : E_s \to E_e$	Equivalent ideals $I_\varphi \sim I_\psi$
$\psi \circ \varphi : E_s \to E_i \to E_e$	$I_{\psi\circ\varphi} = I_\varphi \cdot I_\psi$
$[\varphi]_*\psi$, $[\psi]^*\varphi$	$[I_\varphi]_* I_\psi$, $[I_\psi]^* I_\varphi$

SQIsign in a nutshell

- **Key generation**

$$E_0 \xrightarrow{\boldsymbol{\tau_{sk}}} E_{pk} \qquad\qquad \mathcal{O}_0 \xrightarrow{\boldsymbol{I_{sk}}} \mathcal{O}_{pk}$$

SQIsign in a nutshell

- **Key generation**

E_0

$\mathcal{O}_0$

I_{sk}

Secret key

E_{pk}

$\mathcal{O}_{pk}$

Public key

SQIsign in a nutshell

- **Sign – The commitment phase**

$$
\begin{tikzcd}
 & E_0 \arrow[r, "\psi_{com}"] & E_{com} \\
E_{pk} & &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathcal{O}_0 \arrow[r, "I_{com}"] \arrow[dl, "I_{sk}"'] & \mathcal{O}_{com} \\
\mathcal{O}_{pk} & &
\end{tikzcd}
$$

SQIsign in a nutshell

- **Sign – The challenge phase**

$$
\begin{tikzcd}
 & E_0 \arrow[r, "\psi_{com}"] & E_{com} \arrow[d, "\varphi_{chl}"] \\
E_{pk} & & E_{chl}
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathcal{O}_0 \arrow[r, "I_{com}"] \arrow[dl, "I_{sk}"'] & \mathcal{O}_{com} \\
\mathcal{O}_{pk} & &
\end{tikzcd}
$$

SQIsign in a nutshell

- **Sign – The challenge phase**

$$
\begin{tikzcd}
 & E_0 \arrow[rr, "\psi_{com}"] & & E_{com} \arrow[d, "\varphi_{chl}"] \\
E_{pk} & & & E_{chl}
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathcal{O}_0 \arrow[dl, "I_{sk}"'] \arrow[d, "I^*_{chl}"] \arrow[rr, "I_{com}"] & & \mathcal{O}_{com} \\
\mathcal{O}_{pk} & \mathcal{O}^*_{chl} & &
\end{tikzcd}
$$

SQIsign in a nutshell

- **Sign – The challenge phase**

$$\begin{tikzcd}
 & E_0 \arrow[r, "\psi_{com}"] & E_{com} \arrow[d, "\varphi_{chl}"] \\
E_{pk} & & E_{chl}
\end{tikzcd}$$

$$\begin{tikzcd}
 & \mathcal{O}_0 \arrow[r, "I_{com}"] \arrow[dl, "I_{sk}"'] & \mathcal{O}_{com} \arrow[d, "I_{chl} = {[I_{com}]}_* I^*_{chl}"] \\
\mathcal{O}_{pk} & & \mathcal{O}_{chl}
\end{tikzcd}$$

SQIsign in a nutshell

- **Sign – The response phase**

$$\begin{tikzcd}
 & E_0 \arrow[r, "\psi_{com}"] & E_{com} \arrow[d, "\varphi_{chl}"] \\
E_{pk} & & E_{chl}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{O}_0 \arrow[r, "I_{com}"] \arrow[d, "I_{sk}"'] & \mathcal{O}_{com} \arrow[d, "I_{chl}"] \\
\mathcal{O}_{pk} \arrow[r, "J_{rsp} \coloneqq SigningKLPT(\bar{I}_{sk} I_{com} I_{chl})"'] & \mathcal{O}_{chl}
\end{tikzcd}$$

SQIsign in a nutshell

- **Sign – The response phase**

$$
\begin{tikzcd}
E_0 \arrow[r, "\boldsymbol{\psi_{com}}"] & E_{com} \arrow[d, "\boldsymbol{\varphi_{chl}}"] \\
E_{pk} \arrow[r, "\boldsymbol{\sigma_{rsp}}"'] & E_{chl}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{O}_0 \arrow[r, "\boldsymbol{I_{com}}"] \arrow[d, "\boldsymbol{I_{sk}}"'] & \mathcal{O}_{com} \arrow[d, "\boldsymbol{I_{chl}}"] \\
\mathcal{O}_{pk} \arrow[r, "\boldsymbol{J_{rsp}} \coloneqq SigningKLPT(\boldsymbol{\bar{I}_{sk} I_{com} I_{chl}})"'] & \mathcal{O}_{chl}
\end{tikzcd}
$$

Signature : $\boldsymbol{(\sigma_{rsp}, hint_1, hint_2)}$

SQIsign in a nutshell

- **Verify**
  - Recover E_{chl} from a part of a signature (σ_{rsp})

$$E_{pk} \xrightarrow[\boldsymbol{\sigma_{rsp}}]{\hspace{6em}} E_{chl}$$

SQIsign in a nutshell

- **Verify**
  - Recover E_{chl} from a part of a signature (σ_{rsp})
  - Recover E'_{com} using the torsion point information made **<u>deterministically from E_{chl}</u>** and $hint_1$ from a signature

$$
\begin{tikzcd}
 & E'_{com} \\
E_{pk} \arrow[r, "\boldsymbol{\sigma_{rsp}}"'] & E_{chl} \arrow[u, "\widehat{\boldsymbol{\varphi_{chl}}}"']
\end{tikzcd}
$$

SQIsign in a nutshell

- **Verify**
  - Recover E_{chl} from a part of a signature (σ_{rsp})
  - Recover E'_{com} using the torsion point information made **deterministically from E_{chl}** and $hint_1$ from a signature
  - Check whether the verifier generates the challenge isogeny correctly by using the **torsion information** made **from E'_{com}**, $\boldsymbol{hint_2}$ and **the hashing of the message and E'_{com}**

$$
\begin{tikzcd}
 & E'_{com} \\
E_{pk} \arrow[r, "\boldsymbol{\sigma_{rsp}}"'] & E_{chl} \arrow[u, "\widehat{\boldsymbol{\varphi_{chl}}}"']
\end{tikzcd}
$$

SQIsign in a nutshell

A deterministic SQIsign VS A randomized SQIsign

A deterministic SQIsign

$$E_0 \xrightarrow{\psi_{com}} E_{com}$$

The commitment isogeny is generated
by hashing of secret key and the message

Msg → *A deterministic SQIsign signing oracle* $(\boldsymbol{sk})$ → *Sign*

Msg → *A deterministic SQIsign signing oracle* $(\boldsymbol{sk})$ → *Sign*

A randomized SQIsign

$$E_0 \xrightarrow{\psi_{com}} E_{com}$$

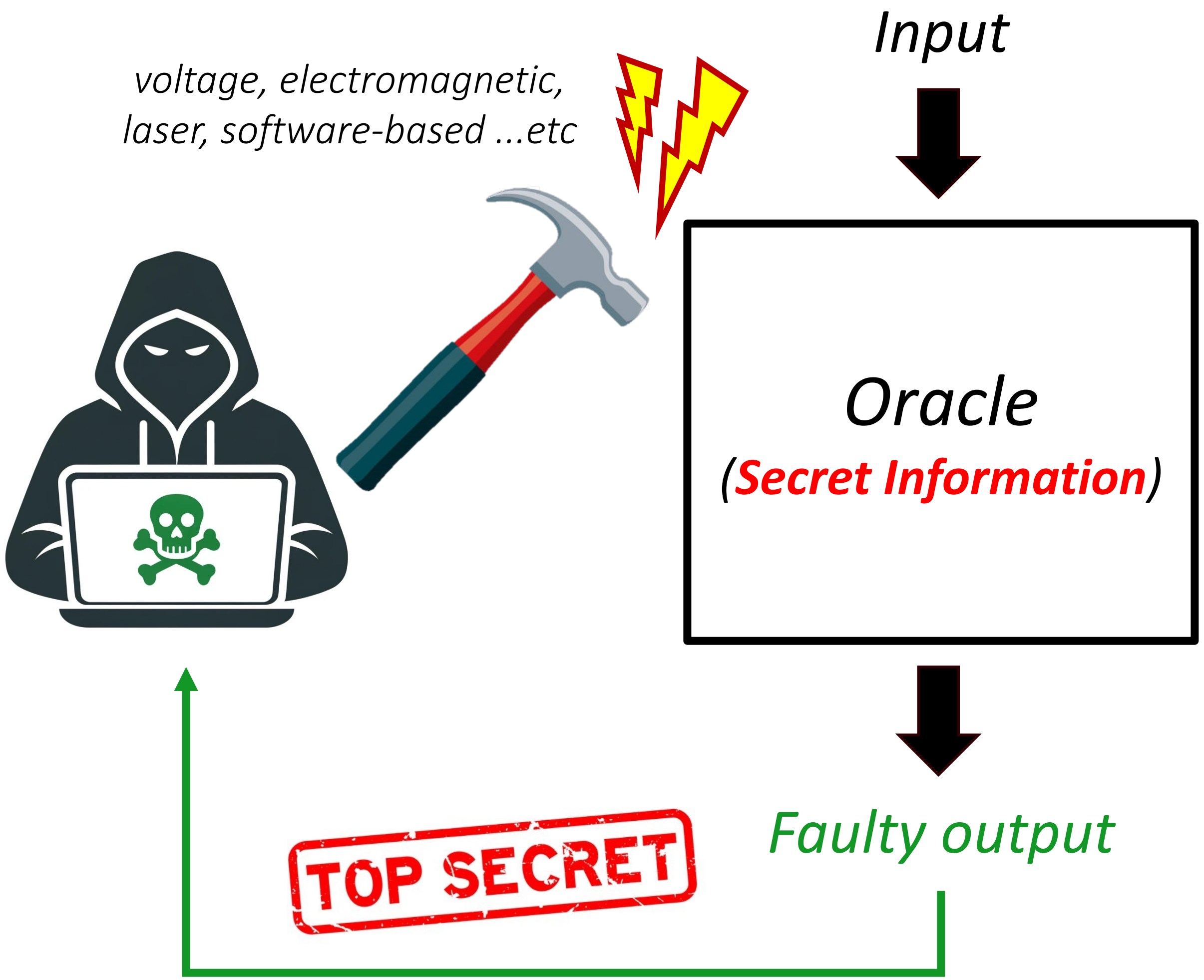
The commitment isogeny is generated
by using a random number

Msg → *A randomized SQIsign signing oracle* $(\boldsymbol{sk})$ → *Sign*

Msg → *A randomized SQIsign signing oracle* $(\boldsymbol{sk})$ → *Sign’*

Fault attack?

Fault attack



Fault attack

Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process

Updated October 2022 to reflect that IP statements can be accepted digitally.

4.B.4 Additional Security Properties While the previously listed security definitions cover many of the attack scenarios that will be used in the evaluation of the submitted algorithms, there are several other properties that would be desirable:

One such property where security and performance interact is resistance to side-channel attacks. Schemes that can be made resistant to side-channel attack at minimal cost are more desirable than those whose performance is severely hampered by any attempt to resist side-channel attacks. We further note that optimized implementations that address side-channel attacks (e.g., constant-time implementations) are more meaningful than those which do not. Finally, there are many different kinds of side-channel attacks, which require different kinds of access to the device being attacked. Attacks that can be carried out remotely, using only digital communications over a network, without physical access to the device being attacked, may be of special concern.

Fault attacks on lattice-based cryptography

Carry Your Fault: A Fault Propagation Attack on Side-Channel Protected LWE-based KEM

Suparna Kundu[1], Siddhartha Chowdhury[2], Sayandeep Saha[3], Angshuman Karmakar[1,4], Debdeep Mukhopadhyay[2] and Ingrid Verbauwhede[1]

[1] COSIC, KU Leuven, Belgium
[2] Indian Institute of Technology Kharagpur, India
[3] Université catholique de Louvain, Belgium
[4] Indian Institute of Technology Kanpur, India

Fault-Enabled Chosen-Ciphertext Attacks on Kyber

Julius Hermelink[1,2], Peter Pessl[1], and Thomas Pöppelmann[1]

[1] Infineon Technologies AG, Munich, Germany
{peter.pessl, thomas.poeppelmann}@infineon.com
[2] Research Institute CODE, Universität der Bundeswehr München, Munich, Germany julius.hermelink@unibw.de

Number "Not Used" Once - Practical fault attack on *pqm4* implementations of NIST candidates

Prasanna Ravi[12], Debapriya Basu Roy[3], Shivam Bhasin[1], Anupam Chattopadhyay[2], and Debdeep Mukhopadhyay[3]

[1] Temasek Laboratories, Nanyang Technological University, Singapore
[2] School of Computer Science and Engineering Nanyang Technological University, Singapore
[3] Indian Institute of Technology, Kharagpur

and so on...

Fault attack

Randomization

```
int a,b;
a = 1000;
b = a;
```

a : 1000
b : 684614321

Loop-abort

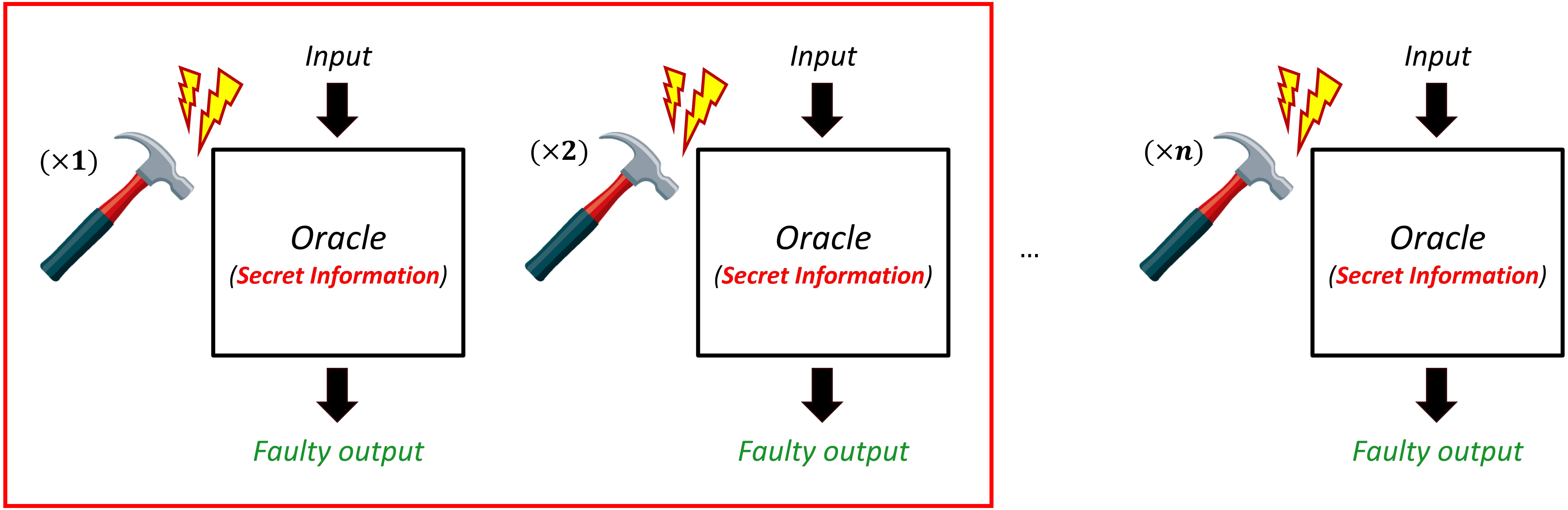
```
int i;
for (i=0;i<100;i++){
      ...
}
```

i : 351635410

aborted!

Fault attack

- n^{th} order fault attack



Fault vulnerabilities in SQIsign

Loop-aborts while computing the commitment isogeny

Algorithm 3 Odd-degree isogeny computation and the basis evaluation in lines 4 to 5 of Algorithm 1

Input: A domain curve E
Input: A generator $K^{\pm}$ of $E[p \pm 1] \cap ker(\phi)$, respectively
Input: A basis (P, Q) of $E[D]$ such that $gcd(D, deg(\phi)) = 1$
Output: An image curve E_c
Output: An evaluated basis $(\phi(P), \phi(Q))$

Let $deg(\phi) = p_1^{e_1} \cdot p_2^{e_2} \cdots p_n^{e_n}$ for $n \in \mathbb{Z}^+$ such that $p_i | (p+1)$ for $1 \le i \le h$ and $p_i | (p-1)$ for $h < i \le n$
$E_{1,0} := E$
$P_{1,0}, Q_{1,0} := P, Q$
Evaluate isogenies with kernel in $E[p+1]$
for $i \in \{1, ..., h\}$ **do**
  $K_i^+ := [p_{i+1}^{e_{i+1}} \cdot p_{i+2}^{e_{i+2}} \cdots p_h^{e_h}]K^+$
  for $j \in \{1, ..., e_i\}$ **do**
    $K_{i,j}^+ := [p_i^{e_i - j}]K_i^+$
    Compute isogeny $\phi_{i,j}$ of $deg(\phi_{i,j}) = p_i$ corresponding to the kernel $K_{i,j}^+$
    $E_{i,j} := \phi_{i,j}(E_{i,j-1})$, $K^+ := \phi_{i,j}(K^+)$, $K^- := \phi_{i,j}(K^-)$
    $P_{i,j}, Q_{i,j} := \phi_{i,j}(P_{i,j-1}), \phi_{i,j}(Q_{i,j-1})$
  end for
end for
Evaluate isogenies with kernel in $E[p-1]$
for $i \in \{h+1, ..., n\}$ **do**
  $K_i^- := [p_{i+1}^{e_{i+1}} \cdot p_{i+2}^{e_{i+2}} \cdots p_n^{e_n}]K^-$
  for $j \in \{1, ..., e_i\}$ **do**
    $K_{i,j}^- := [p_i^{e_i - j}]K_i^-$
    Compute isogeny $\phi_{i,j}$ of $deg(\phi_{i,j}) = p_i$ corresponding to the kernel $K_{i,j}^-$
    $E_{i,j} := \phi_{i,j}(E_{i,j-1})$, $K^- := \phi_{i,j}(K^-)$
    $P_{i,j}, Q_{i,j} := \phi_{i,j}(P_{i,j-1}), \phi_{i,j}(Q_{i,j-1})$
  end for
end for
$E_c := E_{n,e^n}$
$\phi(P), \phi(Q) := P_{n,e^n}, Q_{n,e^n}$
return $E_c, (\phi(P), \phi(Q))$

$$E_0 \xrightarrow{\psi_{com}} E_{com}$$

$$deg(\psi_{com}) = p_1^{e_1} \cdot p_2^{e_2} \cdots p_n^{e_n}$$

Loop-aborts while computing the commitment isogeny

Algorithm 3 Odd-degree isogeny computation and the basis evaluation in lines 4 to 5 of Algorithm 1

Input: A domain curve E
Input: A generator $K^{\pm}$ of $E[p \pm 1] \cap ker(\phi)$, respectively
Input: A basis (P, Q) of $E[D]$ such that $gcd(D, deg(\phi)) = 1$
Output: An image curve E_c
Output: An evaluated basis $(\phi(P), \phi(Q))$

Let $deg(\phi) = p_1^{e_1} \cdot p_2^{e_2} \cdots p_n^{e_n}$ for $n \in \mathbb{Z}^+$ such that $p_i | (p+1)$ for $1 \leq i \leq h$ and $p_i | (p-1)$ for $h < i \leq n$
$E_{1,0} := E$
$P_{1,0}, Q_{1,0} := P, Q$
Evaluate isogenies with kernel in $E[p+1]$
for $i \in \{1, ..., h\}$ **do**
  $K_i^+ := [p_{i+1}^{e_{i+1}} \cdot p_{i+2}^{e_{i+2}} \cdots p_h^{e_h}]K^+$
  for $j \in \{1, ..., e_i\}$ **do**
    $K_{i,j}^+ := [p_i^{e_i - j}]K_i^+$
    Compute isogeny $\phi_{i,j}$ of $deg(\phi_{i,j}) = p_i$ corresponding to the kernel $K_{i,j}^+$
    $E_{i,j} := \phi_{i,j}(E_{i,j-1})$, $K^+ := \phi_{i,j}(K^+)$, $K^- := \phi_{i,j}(K^-)$
    $P_{i,j}, Q_{i,j} := \phi_{i,j}(P_{i,j-1}), \phi_{i,j}(Q_{i,j-1})$
  end for
end for
Evaluate isogenies with kernel in $E[p-1]$
for $i \in \{h+1, ..., n\}$ **do**
  $K_i^- := [p_{i+1}^{e_{i+1}} \cdot p_{i+2}^{e_{i+2}} \cdots p_n^{e_n}]K^-$
  for $j \in \{1, ..., e_i\}$ **do**
    $K_{i,j}^- := [p_i^{e_i - j}]K_i^-$
    Compute isogeny $\phi_{i,j}$ of $deg(\phi_{i,j}) = p_i$ corresponding to the kernel $K_{i,j}^-$
    $E_{i,j} := \phi_{i,j}(E_{i,j-1})$, $K^- := \phi_{i,j}(K^-)$
    $P_{i,j}, Q_{i,j} := \phi_{i,j}(P_{i,j-1}), \phi_{i,j}(Q_{i,j-1})$
  end for
end for
$E_c := E_{n,e^n}$
$\phi(P), \phi(Q) := P_{n,e^n}, Q_{n,e^n}$
return $E_c, (\phi(P), \phi(Q))$

$$E_0 \xrightarrow{\psi_{com}} E_{com}$$

$$deg(\psi_{com}) = p_1^{e_1} \cdot p_2^{e_2} \cdots p_n^{e_n}$$

$$E_0 \xrightarrow{\psi_{com}^f} E_{com}^f$$

$$deg\left(\psi_{com}^f\right) = p_1^{e_1} \cdot p_2^{e_2} \cdots p_{if}^{e_{if}}$$

Randomization while generating the commitment ideal

Algorithm 3 Making primitive then generating an ideal I_{com}

Input: A p-extremal maximal order O_0
Input: A generator coordinates $[g]_B$ w.r.t. the standard basis
Input: The norm D_{com}
Output: A commitment ideal I_{com}

Let $B := \{1, i, j, ij\}$ be a set of the standard basis in quaternion algebra.
Let $B_{O_0} := \{\beta_1, \beta_2, \beta_3, \beta_4\}$ be a set of the basis of order O_0
Convert $[g]_B$ into $[g]_{O_0}$ using gaussian ellimination.
$content := gcd([g]_{O_0})$
$[g']_{O_0} := [g]_{O_0}/content$
$[g']_B := [g']_{O_0} * B_{O_0}$
$D'_{com} := D_{com}/gcd(content, D_{com})$
$I_{com} := O_0 < [g']_B, D'_{com} >$
return I_{com}

```
int quat_lattice_contains_without_alg(quat_alg_coord_t *coord,
    const quat_lattice_t *lat, const quat_alg_elem_t *x){
    // Test if rank 4 lattice under HNF ...
    // Convert the basis by using gaussian elimination
    // Final test
    // Copy result
    if(res && (coord != NULL)){
        for(int i = 0; i < 4;i++){
            ibz_copy(&((*coord)[i]),&(work_coord[i]));
        }
    }
    // Finalize
}
```

$$\mathcal{O}_0 \xrightarrow{I_{com}} \mathcal{O}_{com}$$

$$nrd(I_{com}) = deg(\psi_{com})$$

Randomization while generating the commitment ideal

Algorithm 3 Making primitive then generating an ideal I_{com}

Input: A p-extremal maximal order O_0
Input: A generator coordinates $[g]_B$ w.r.t. the standard basis
Input: The norm D_{com}
Output: A commitment ideal I_{com}

1: Let $B := \{1, i, j, ij\}$ be a set of the standard basis in quaternion algebra.
2: Let $B_{O_0} := \{\beta_1, \beta_2, \beta_3, \beta_4\}$ be a set of the basis of order O_0
3: Convert $[g]_B$ into $[g]_{O_0}$ using gaussian ellimination.
4: $content := gcd([g]_{O_0})$
5: $[g']_{O_0} := [g]_{O_0}/content$
6: $[g']_B := [g']_{O_0} * B_{O_0}$
7: $D'_{com} := D_{com}/gcd(content, D_{com})$
8: $I_{com} := O_0 < [g']_B, D'_{com} >$
9: **return** I_{com}

```
int quat_lattice_contains_without_alg(quat_alg_coord_t *coord,
    const quat_lattice_t *lat, const quat_alg_elem_t *x){
    // Test if rank 4 lattice under HNF ...
    // Convert the basis by using gaussian elimination
    // Final test
    // Copy result
    if(res && (coord != NULL)){
        for(int i = 0; i < 4;i++){
            ibz_copy(&((*coord)[i]),&(work_coord[i]));
        }
    }
    // Finalize
}
```

$$\mathcal{O}_0 \xrightarrow{I_{com}} \mathcal{O}_{com}$$

[Theorem 2]

$$\left(\frac{\varphi(D_{com})}{D_{com}}\right)^3 < P(I^f_{com} = \mathcal{O}_0) < \frac{\varphi(D_{com})}{D_{com}}$$

Heuristically,

$$P(I^f_{com} = \mathcal{O}_0) \approx \left(\frac{\varphi(D_{com})}{D_{com}}\right)^2$$

I^f_{com}: the faulty commitment ideal,
D_{com} : the degree of a commitment isogeny,
φ : Euler's totient function

Randomization while generating the commitment ideal

Algorithm 3 Making primitive then generating an ideal I_{com}

Input: A p-extremal maximal order $\mathcal{O}_0$
Input: A generator coordinates $[g]_B$ w.r.t. the standard basis
Input: The norm D_{com}
Output: A commitment ideal I_{com}

1: Let $B := \{1, i, j, ij\}$ be a set of the standard basis in quaternion algebra.
2: Let $B_{\mathcal{O}_0} := \{\beta_1, \beta_2, \beta_3, \beta_4\}$ be a set of the basis of order $\mathcal{O}_0$
3: Convert $[g]_B$ into $[g]_{\mathcal{O}_0}$ using gaussian ellimination.
4: $content := gcd([g]_{\mathcal{O}_0})$
5: $[g']_{\mathcal{O}_0} := [g]_{\mathcal{O}_0}/content$
6: $[g']_B := [g']_{\mathcal{O}_0} * B_{\mathcal{O}_0}$
7: $D'_{com} := D_{com}/gcd(content, D_{com})$
8: $I_{com} := \mathcal{O}_0 < [g']_B, D'_{com} >$
9: **return** I_{com}

```
int quat_lattice_contains_without_alg(quat_alg_coord_t *coord,
    const quat_lattice_t *lat, const quat_alg_elem_t *x){
    // Test if rank 4 lattice under HNF ...
    // Convert the basis by using gaussian elimination
    // Final test
    // Copy result
    if(res && (coord != NULL)){
        for(int i = 0; i < 4;i++){
            ibz_copy(&((*coord)[i]),&(work_coord[i]));
        }
    }
    // Finalize
}
```

$$\mathcal{O}_0 \xrightarrow{I_{com}} \mathcal{O}_{com}$$

NIST level	Upper bound $\frac{\varphi(D_{com})}{D_{com}}$	Lower bound $(\frac{\varphi(D_{com})}{D_{com}})^3$	Approximation $(\frac{\varphi(D_{com})}{D_{com}})^2$	Simulation $(\times 100{,}000)$
NIST-I	0.59843	0.21430	0.35811	0.35782
NIST-III	0.53314	0.15154	0.28424	0.27910
NIST-V	0.52081	0.14126	0.27124	0.26797

$\mathcal{O}_0$

$I^f_{com} = \mathcal{O}_0$

Attack scenarios

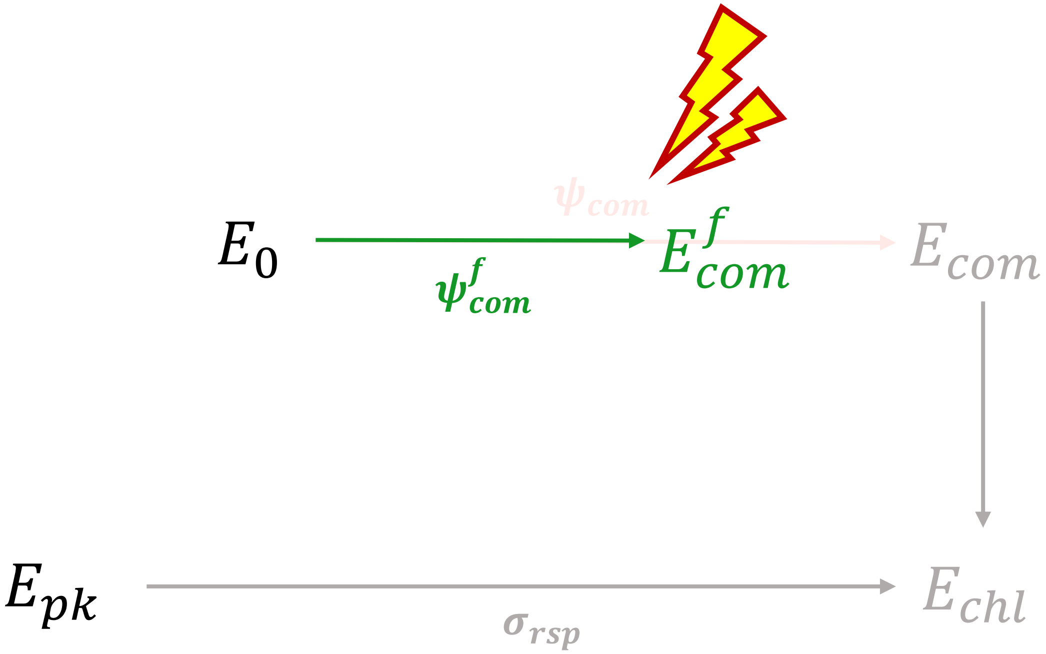
Intuition

- **Goal : Connect E_0 and E_{pk} via an isogeny → Equivalent secret key!**
- **Fault injection into commitment process to make it Brute-Forceable**

$$
\begin{tikzcd}
E_0 \arrow[r, "\psi_{com}"] & E_{com} \arrow[d] \\
E_{pk} \arrow[r, "\sigma_{rsp}"'] & E_{chl}
\end{tikzcd}
$$

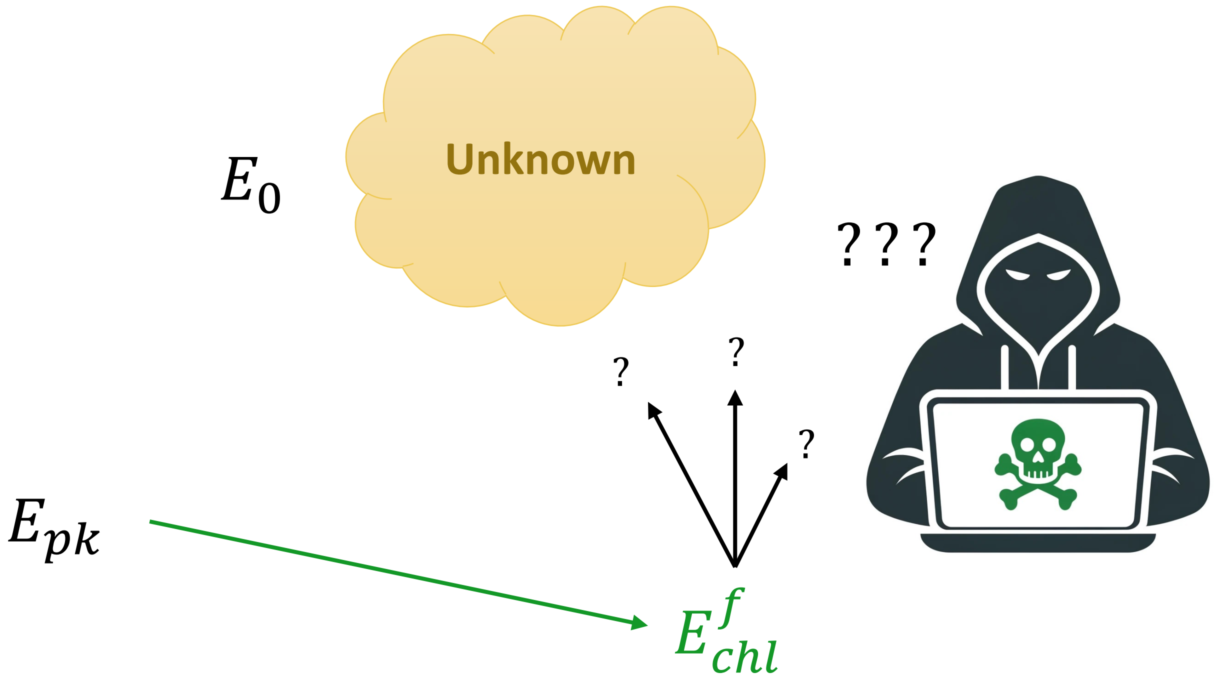
Intuition

- **Goal : Connect E_0 and E_{pk} via an isogeny → Equivalent secret key!**
- **Fault injection into commitment process to make it Brute-Forceable**



Intuition

- **Goal : Connect E_0 and E_{pk} via an isogeny → Equivalent secret key!**
- **Fault injection into commitment process to make it Brute-Forceable**
- **One problem…**
  - How to **recover the isogeny from E^f_{chl} to E^f_{com}** with the given information after the injection?



Fault attack on deterministic SQIsign

- **Attacker model**
  - The attacker is allowed to make multiple queries to a deterministic SQISign oracle with the same key

Msg → *A deterministic SQIsign signing oracle* ($\boldsymbol{sk}$) → *Sign*

Msg' → *A deterministic SQIsign signing oracle* ($\boldsymbol{sk}$) → *Sign'*

Msg'' → *A deterministic SQIsign signing oracle* ($\boldsymbol{sk}$) → *Sign''*

…

Fault attack on deterministic SQIsign

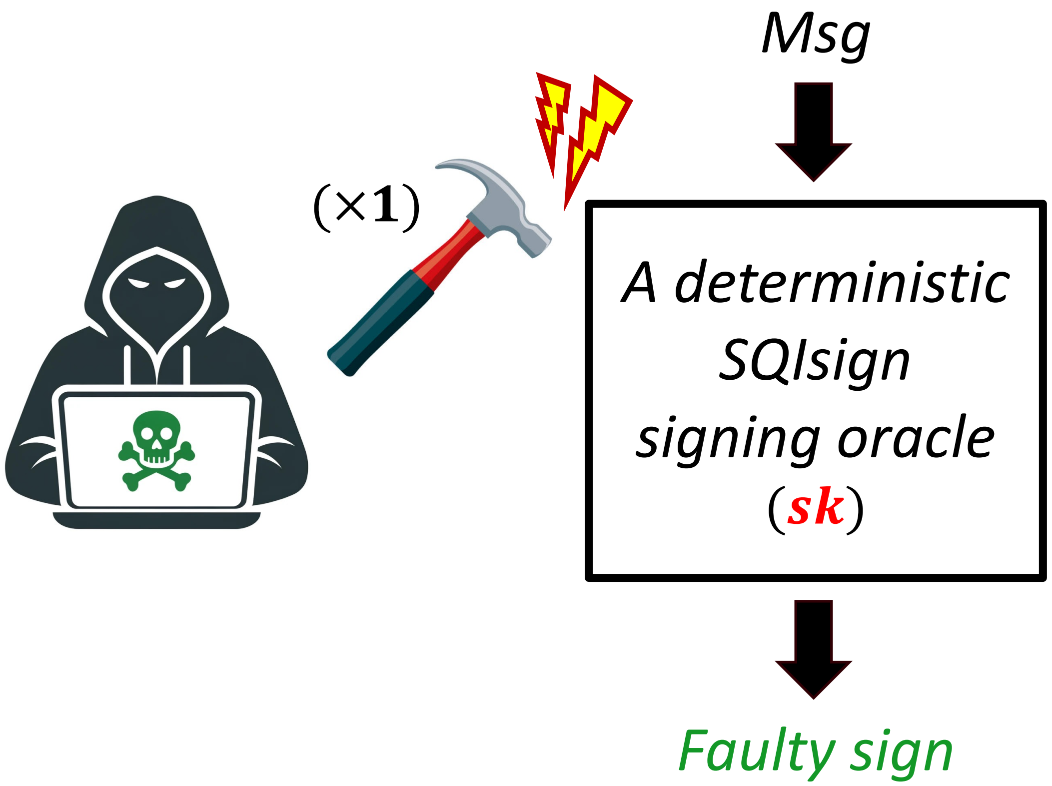
- **Attacker model**
  - The attacker is allowed to make multiple queries to a deterministic SQISign oracle with the same key
  - The oracle generates a signature for each query and the attacker receives it

(Sign, Sign’, Sign’’, …)

Msg	*Msg’*	*Msg’’*	
A deterministic SQIsign signing oracle ($\boldsymbol{sk}$)	*A deterministic SQIsign signing oracle* ($\boldsymbol{sk}$)	*A deterministic SQIsign signing oracle* ($\boldsymbol{sk}$)	…
Sign	*Sign’*	*Sign’’*	

Fault attack on deterministic SQIsign

- **Attacker model**
  - The attacker is allowed to make multiple queries to a deterministic SQISign oracle with the same key
  - The oracle generates a signature for each query and the attacker receives it
  - The attacker can inject a fault once during the oracle's operation (1st order fault attack)



Fault attack on deterministic SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 1^{st}–order fault is injected while computing the commitment isogeny

$$\begin{tikzcd}
 & E_0 \arrow[r, "\boldsymbol{\psi_{com}}"] & E_{com} \arrow[d, "\boldsymbol{\varphi_{chl}}"] \\
E_{pk} \arrow[rr, "\boldsymbol{\sigma_{rsp}}"'] & & E_{chl}
\end{tikzcd}$$

Elliptic curves

$$\begin{tikzcd}
 & \mathcal{O}_0 \arrow[r, "\boldsymbol{I_{com}}"] \arrow[dl, "\boldsymbol{I_{sk}}"'] & \mathcal{O}_{com} \arrow[d, "\boldsymbol{I_{chl}}"] \\
\mathcal{O}_{pk} \arrow[rr, "\boldsymbol{J_{rsp}} \coloneqq SigningKLPT(\boldsymbol{\bar{I}_{sk} I_{com} I_{chl}})"'] & & \mathcal{O}_{chl}
\end{tikzcd}$$

Quaternion algebra

Fault attack on deterministic SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 1st–order fault is injected while computing the commitment isogeny

$$\begin{tikzcd}
 & E_0 \arrow[r, "\psi_{com}"] & E_{com} \arrow[d, "\varphi_{chl}"] \\
E_{pk} \arrow[rr, "\sigma_{rsp}"'] & & E_{chl}
\end{tikzcd}$$

Elliptic curves

$$\begin{tikzcd}
 & \mathcal{O}_0 \arrow[r, "I_{com}"] \arrow[dl, "I_{sk}"'] & \mathcal{O}_{com} \arrow[d, "I_{chl}"] \\
\mathcal{O}_{pk} \arrow[rr, "J_{rsp} := SigningKLPT(\bar{I}_{sk} I_{com} I_{chl})"'] & & \mathcal{O}_{chl}
\end{tikzcd}$$

Quaternion algebra

Fault attack on deterministic SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 1st–order fault is injected while computing the commitment isogeny

$$
\begin{tikzcd}
 & E_0 \arrow[r, "\psi_{com}^{f}"'] & E_{com}^{f} \arrow[r, "\psi_{com}"] & E_{com} \arrow[d, "\varphi_{chl}"] \\
E_{pk} \arrow[rrr, "\sigma_{rsp}"'] & & & E_{chl}
\end{tikzcd}
$$

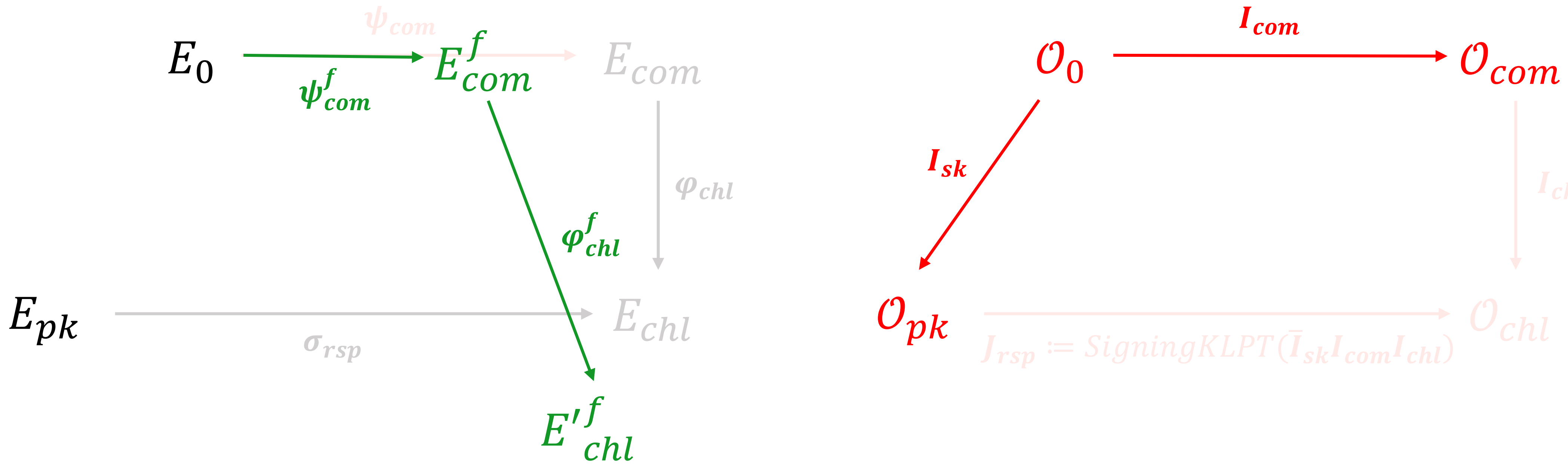
Elliptic curves

$$
\begin{tikzcd}
 & \mathcal{O}_0 \arrow[r, "I_{com}"] \arrow[dl, "I_{sk}"'] & \mathcal{O}_{com} \arrow[d, "I_{chl}"] \\
\mathcal{O}_{pk} \arrow[rr, "J_{rsp} \coloneqq SigningKLPT(\bar{I}_{sk} I_{com} I_{chl})"'] & & \mathcal{O}_{chl}
\end{tikzcd}
$$

Quaternion algebra

Fault attack on deterministic SQIsign

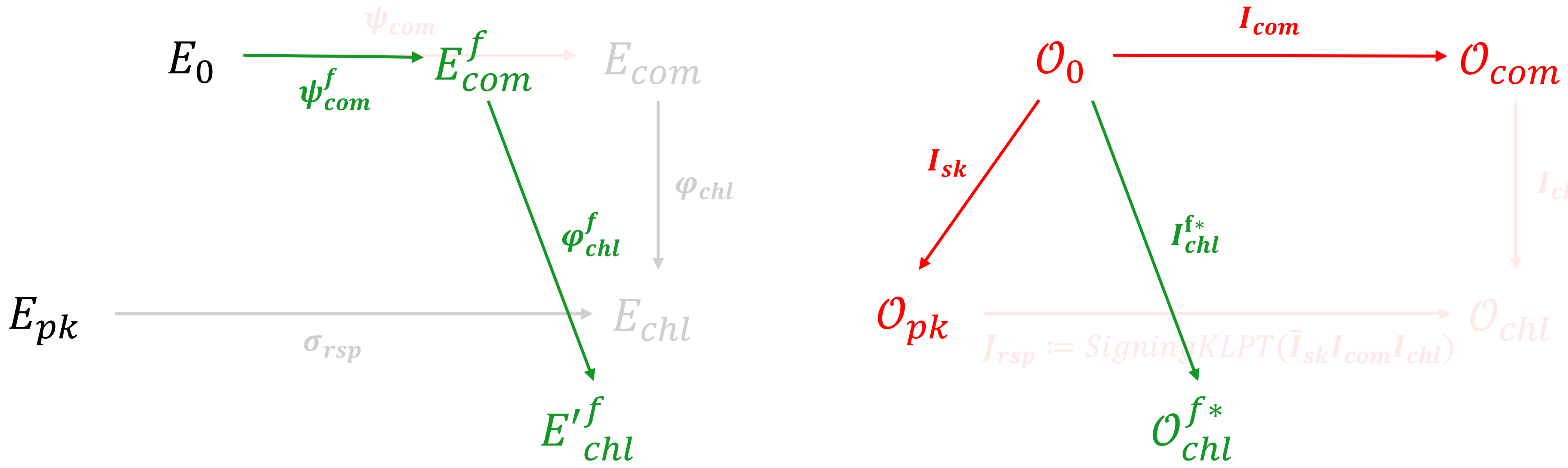
- **The data flow of the faulty SQIsign signing process**
  - 1st–order fault is injected while computing the commitment isogeny



Elliptic curves *Quaternion algebra*

Fault attack on deterministic SQIsign

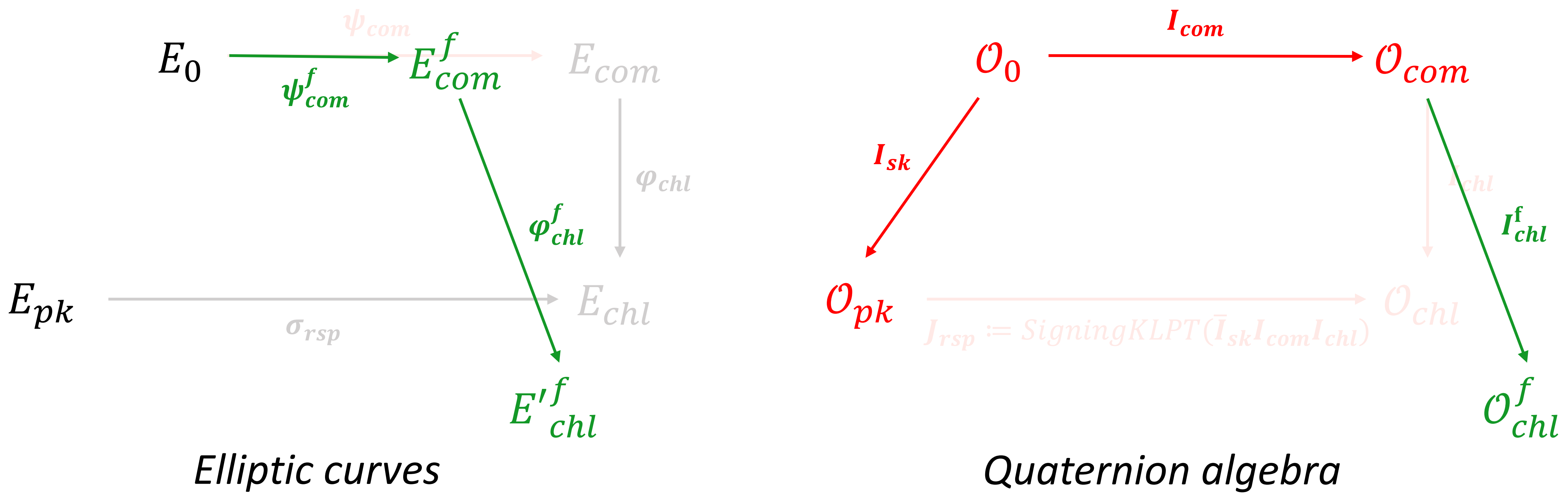
- **The data flow of the faulty SQIsign signing process**
  - 1^{st}–order fault is injected while computing the commitment isogeny



Elliptic curves *Quaternion algebra*

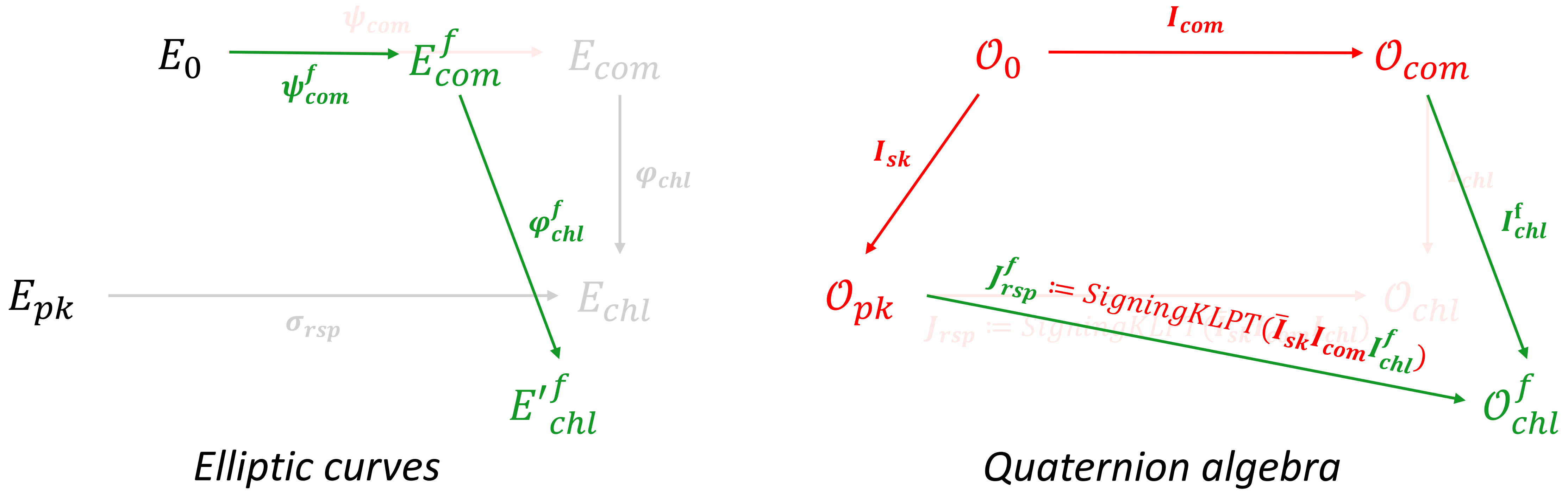
Fault attack on deterministic SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 1st–order fault is injected while computing the commitment isogeny



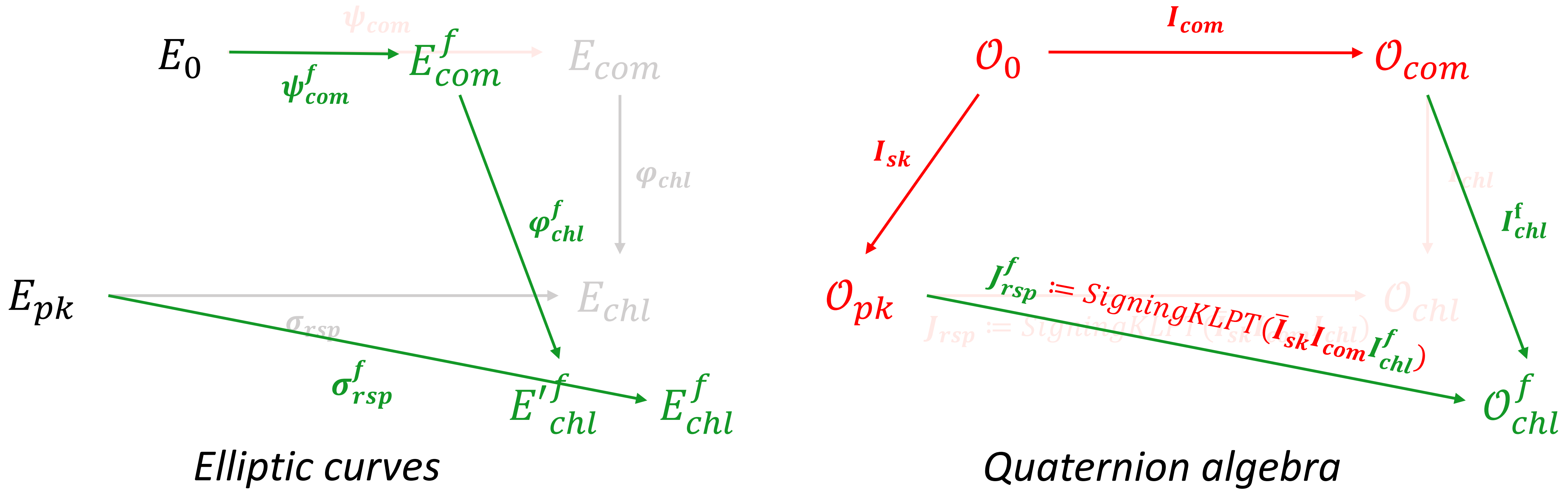
Fault attack on deterministic SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 1st–order fault is injected while computing the commitment isogeny



Fault attack on deterministic SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 1st–order fault is injected while computing the commitment isogeny



Fault attack on deterministic SQIsign

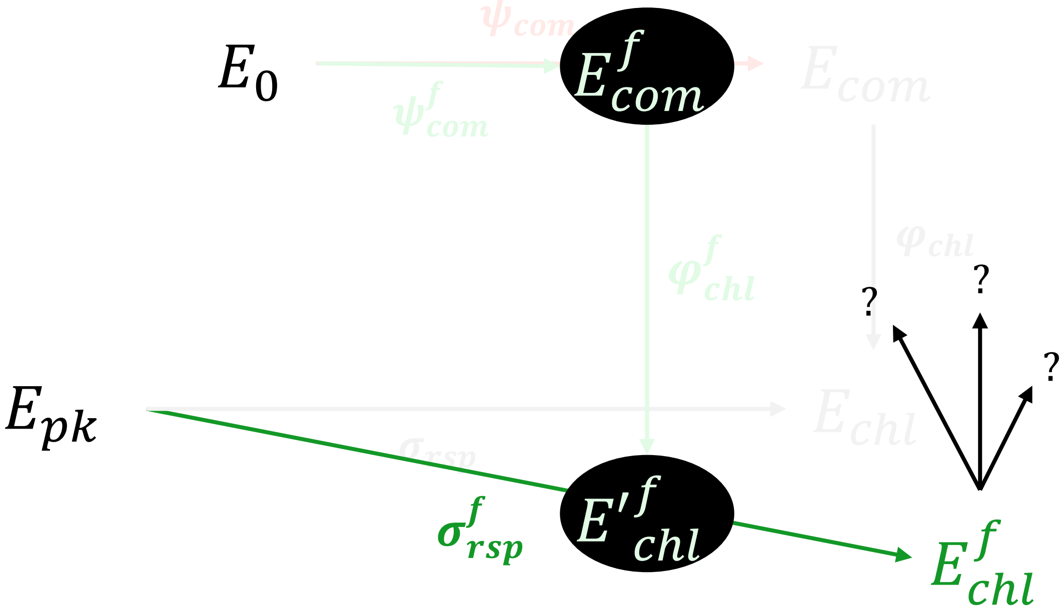
- **What the attacker receives…**

E_0

$$E_{pk} \xrightarrow{\sigma^f_{rsp}} E^f_{chl}$$

Fault attack on deterministic SQIsign

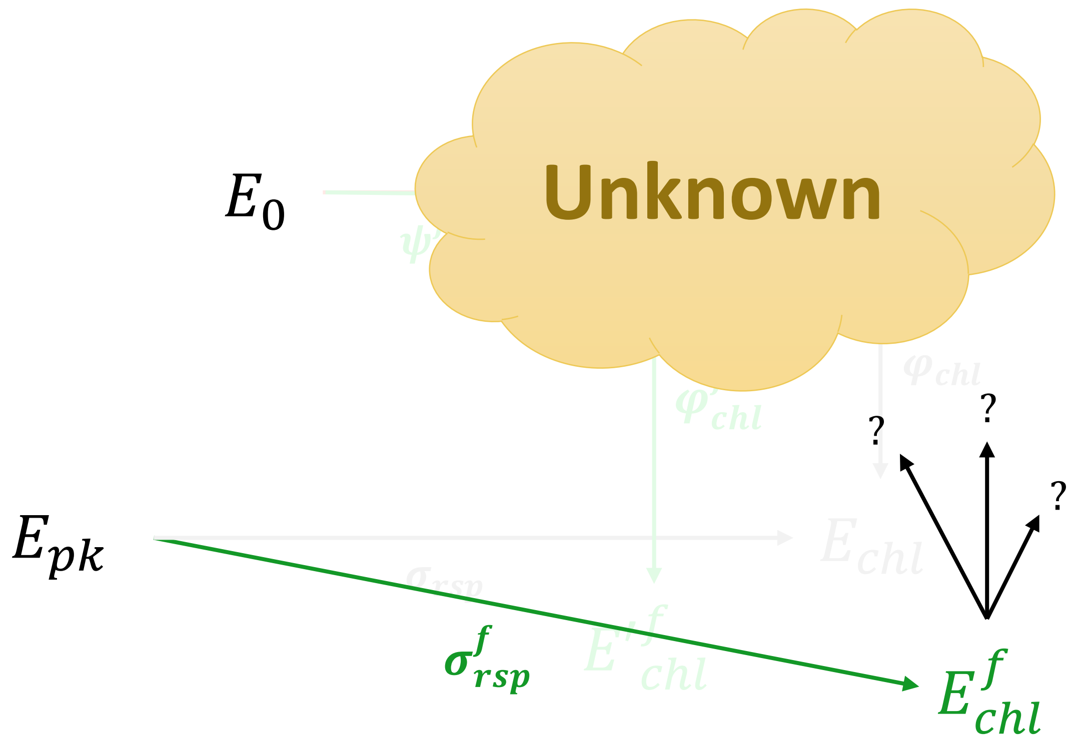
- **What the attacker receives…**



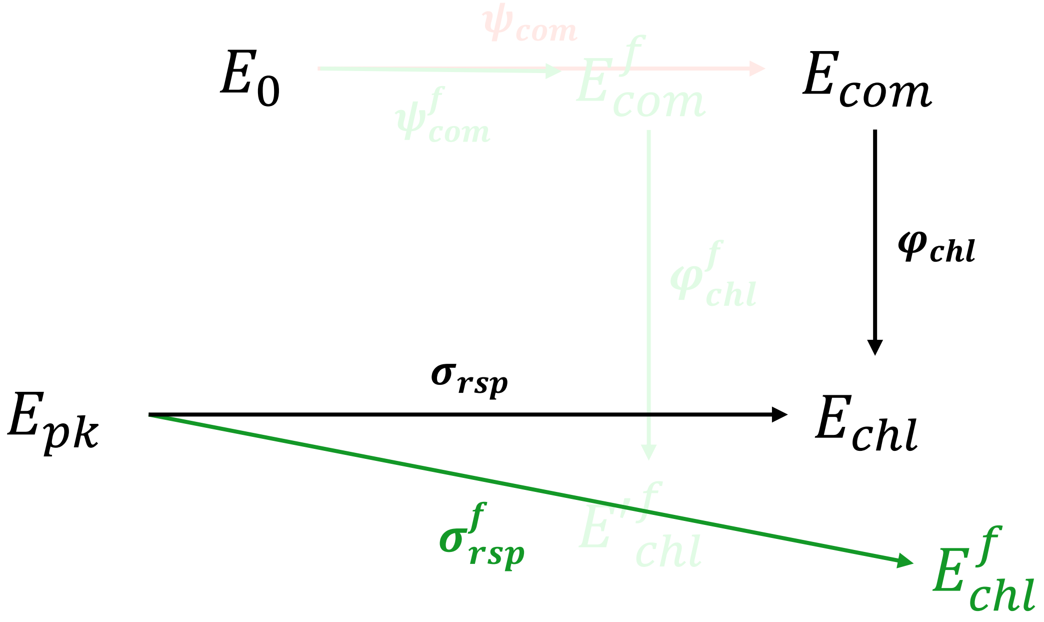
I can't recover E^f_{com} from E^f_{chl} and a faulty signature…

Fault attack on deterministic SQIsign

- What the attacker receives…



Fault attack on deterministic SQIsign



Let's use the ***normal*** *signature*
<u>with the same message</u>

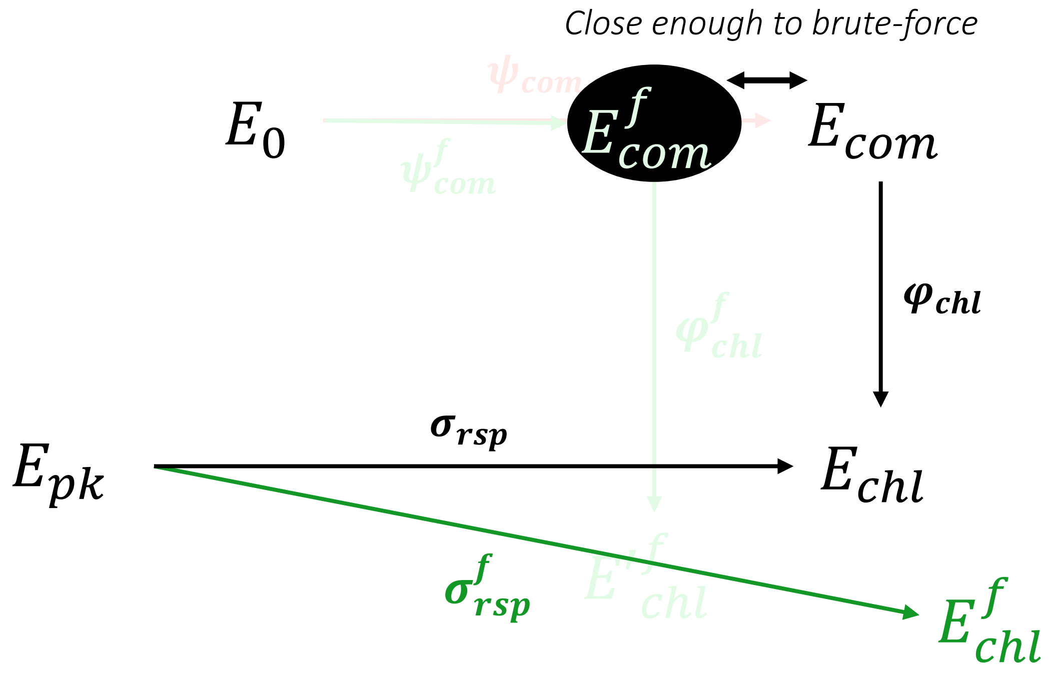
Fault attack on deterministic SQIsign

- **Key recovery attack scenario**
  - First, the attacker queries the signing oracle with normal execution

$$\begin{tikzcd} & E_{com} \arrow[d, "\boldsymbol{\varphi_{chl}}"] \\ E_{pk} \arrow[r, "\boldsymbol{\sigma_{rsp}}"] & E_{chl} \end{tikzcd}$$

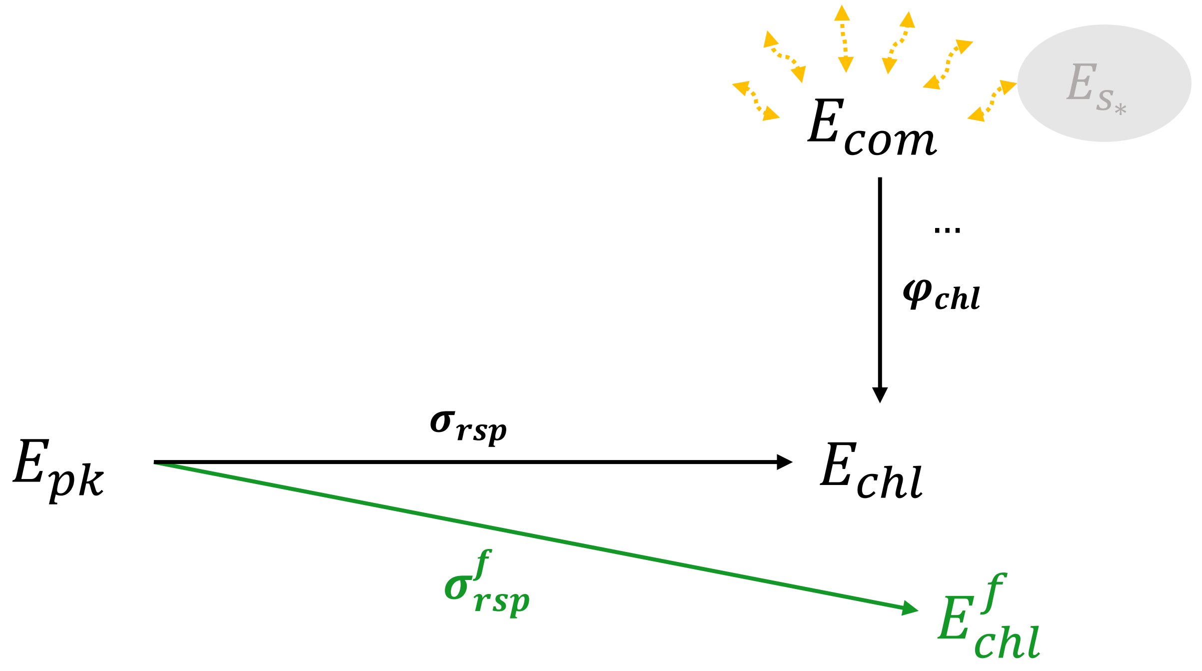
Fault attack on deterministic SQIsign

- **Key recovery attack scenario**
  - Second, the attacker queries the signing oracle with a fault injection **inducing E^f_{com} close to E_{com}**



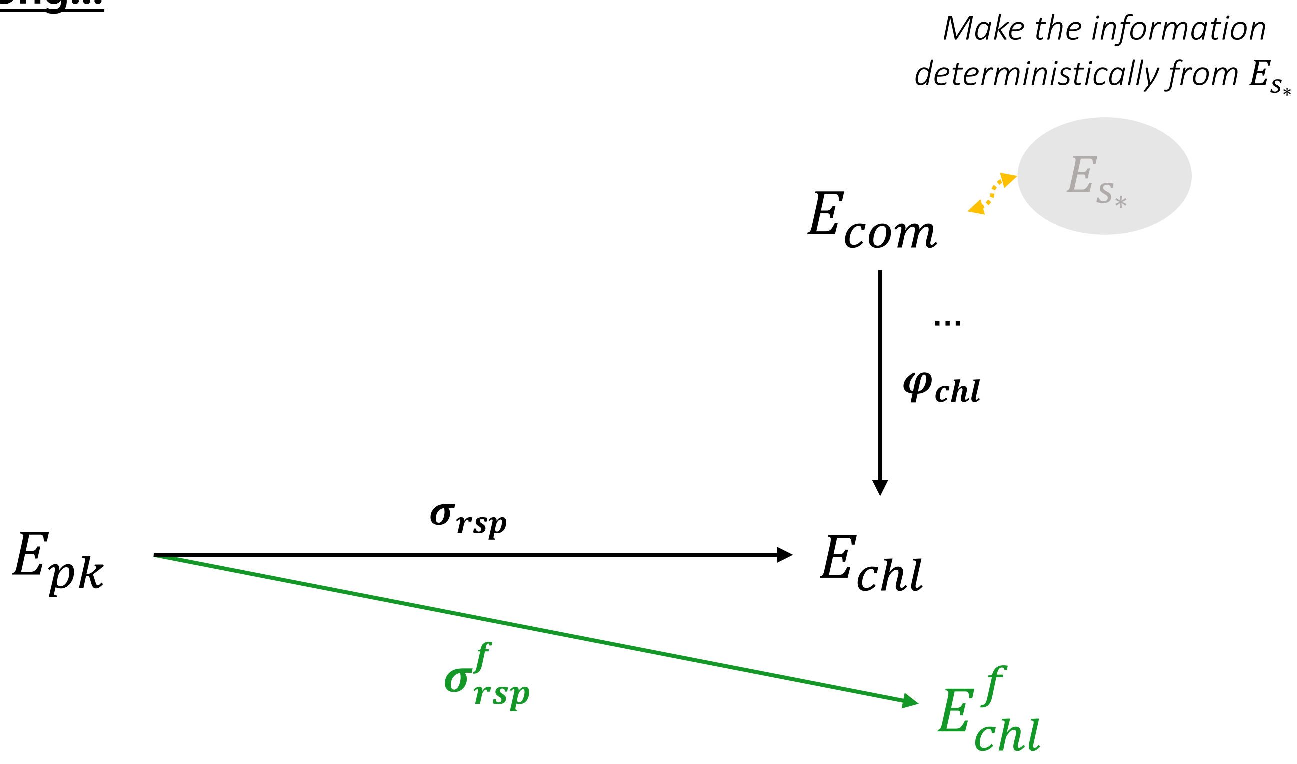
Fault attack on deterministic SQIsign

- **Key recovery attack scenario**
  - Third, the attacker brute-force to find the isogeny from E_{com} to E_{com}^{f}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **wrong…**



Fault attack on deterministic SQIsign

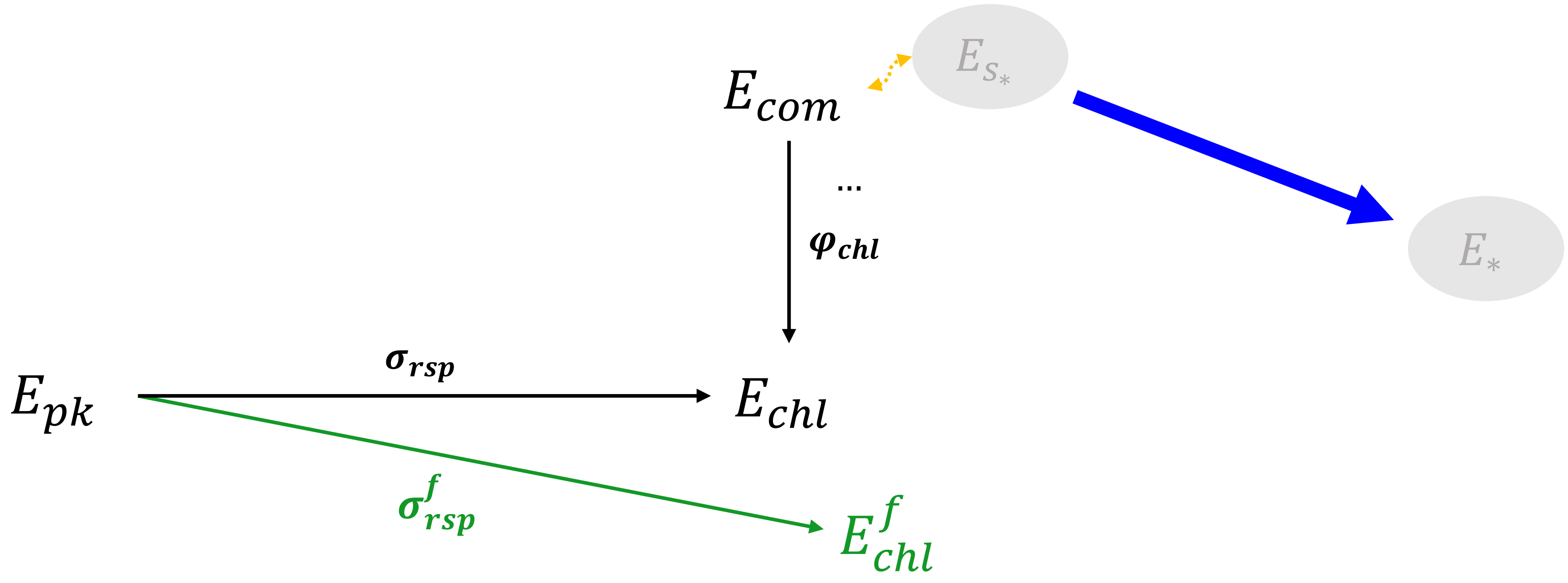
- **Key recovery attack scenario**
  - Third, the attacker brute-force to find the isogeny from E_{com} and E_{com}^{f}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **wrong…**



Fault attack on deterministic SQIsign

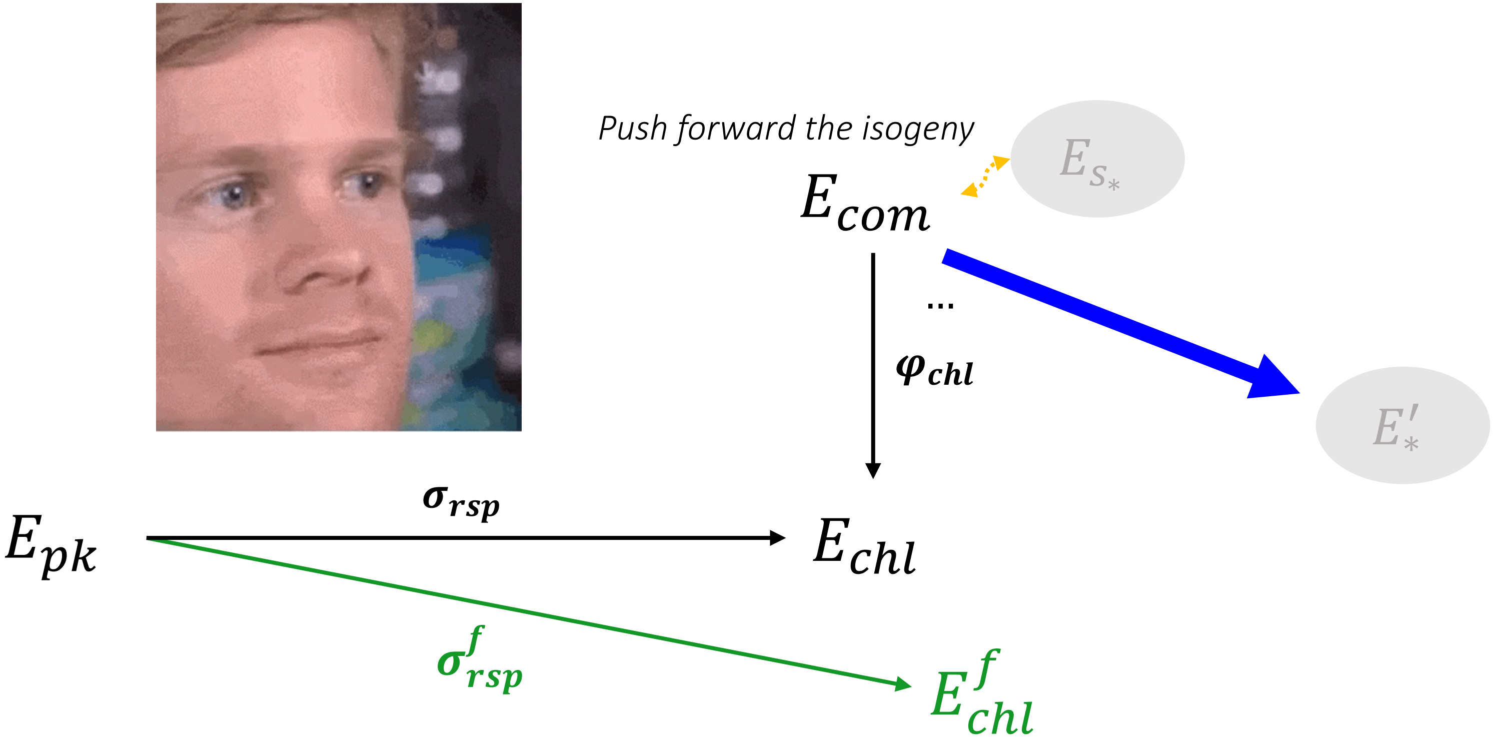
- **Key recovery attack scenario**
  - Third, the attacker brute-force to find the isogeny from E_{com} and E_{com}^{f}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **wrong…**

Make the isogeny with the information



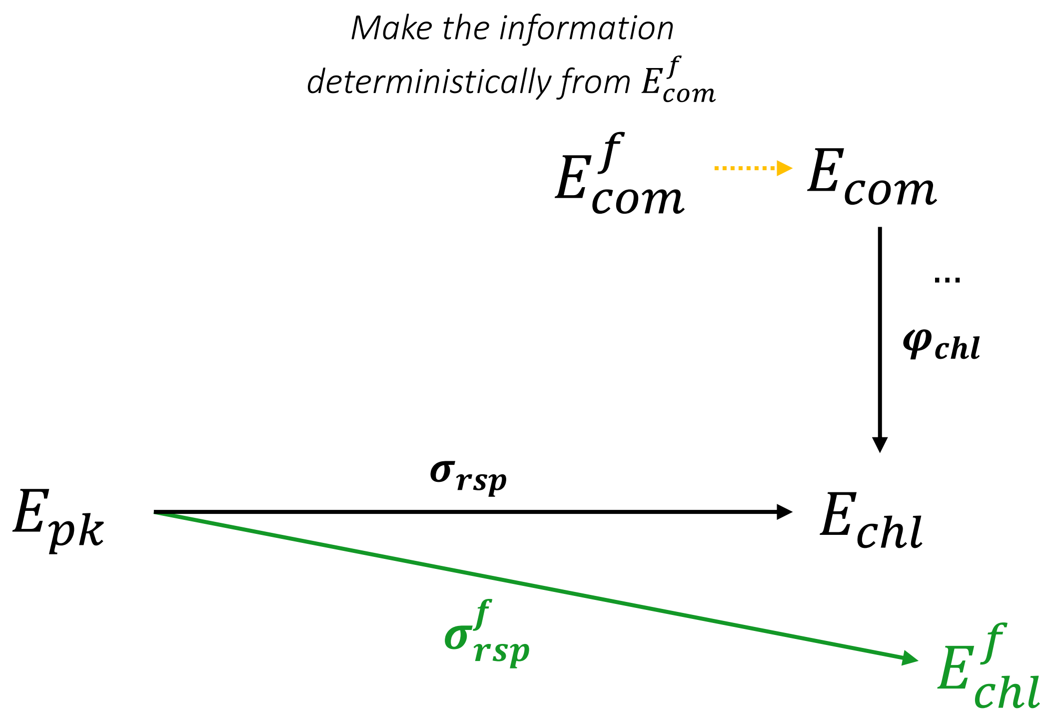
Fault attack on deterministic SQIsign

- **Key recovery attack scenario**
  - Third, the attacker brute-force to find the isogeny from E_{com} and E_{com}^{f}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **wrong...**



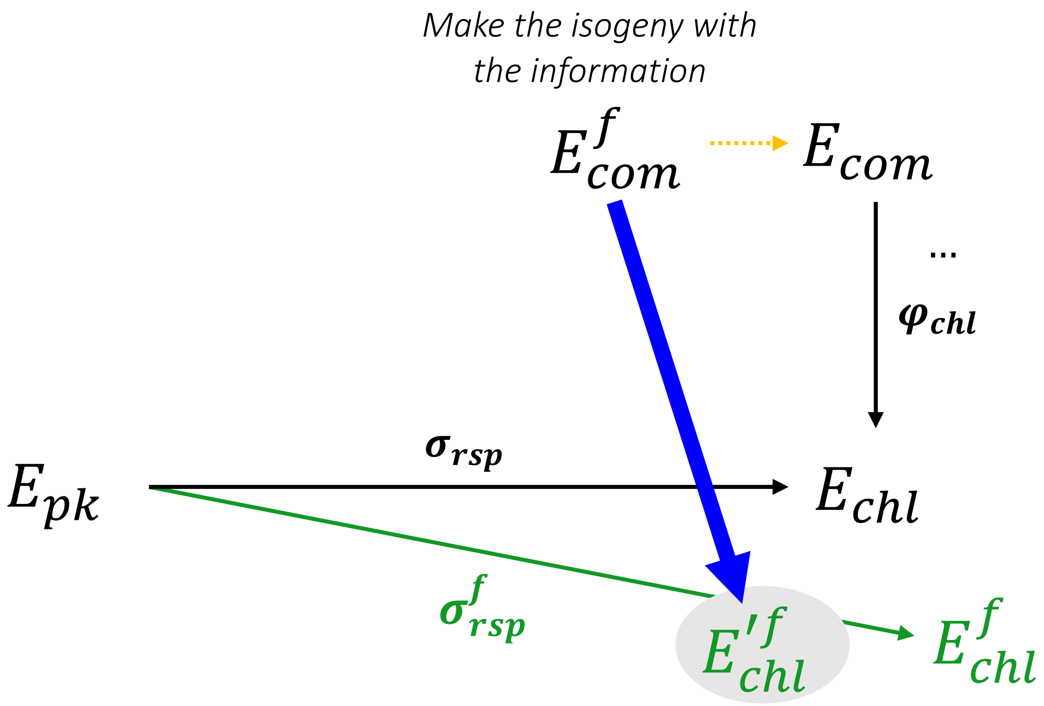
Fault attack on deterministic SQIsign

- **Key recovery attack scenario**
  - Third, the attacker brute-force to find the isogeny from E_{com} and E_{com}^f
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is <u>**right…**</u>



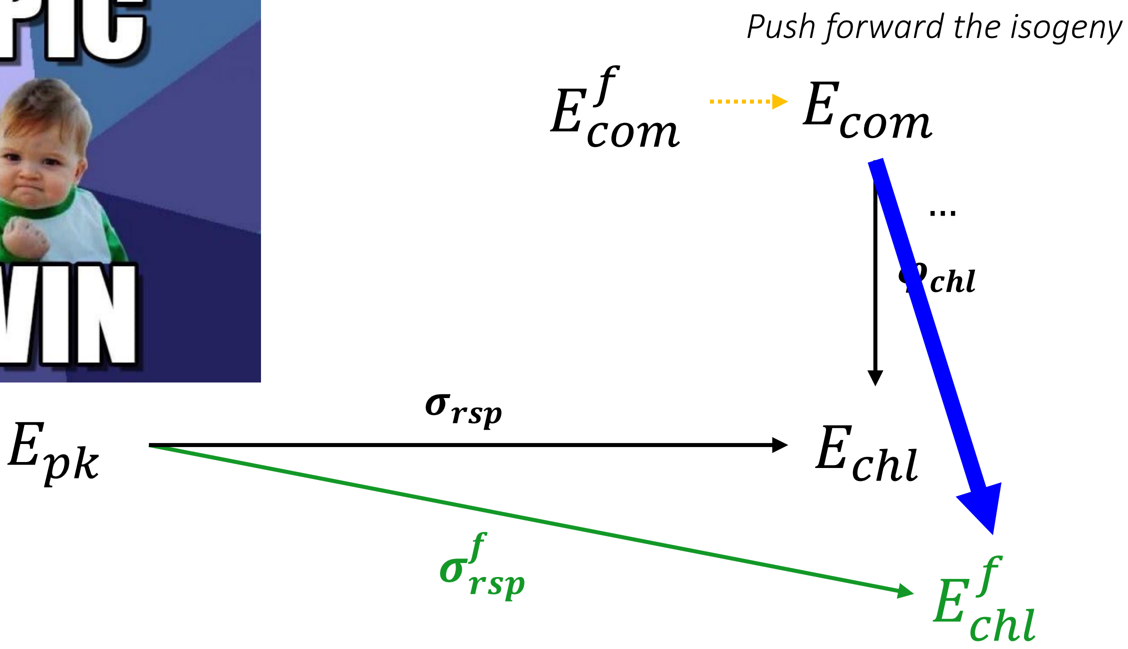
Fault attack on deterministic SQIsign

- **Key recovery attack scenario**
  - Third, the attacker brute-force to find the isogeny from E_{com} and E_{com}^{f}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **right…**



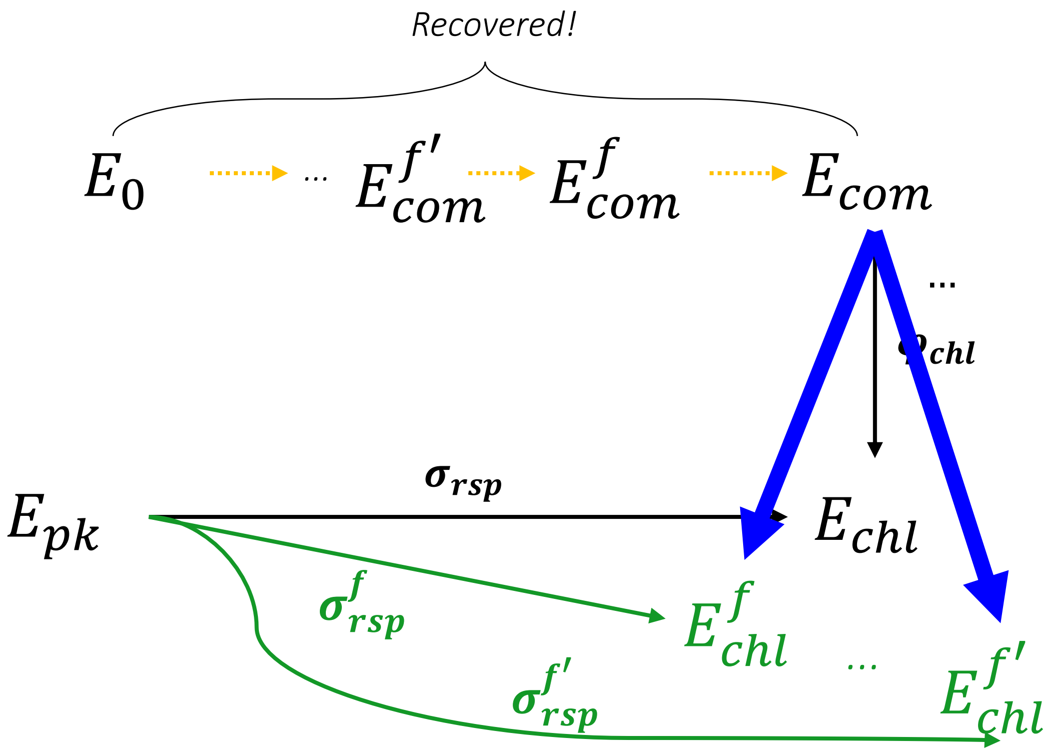
Fault attack on deterministic SQIsign

- **Key recovery attack scenario**
  - Third, the attacker brute-force to find the isogeny from E_{com} and E_{com}^{f}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **right…**



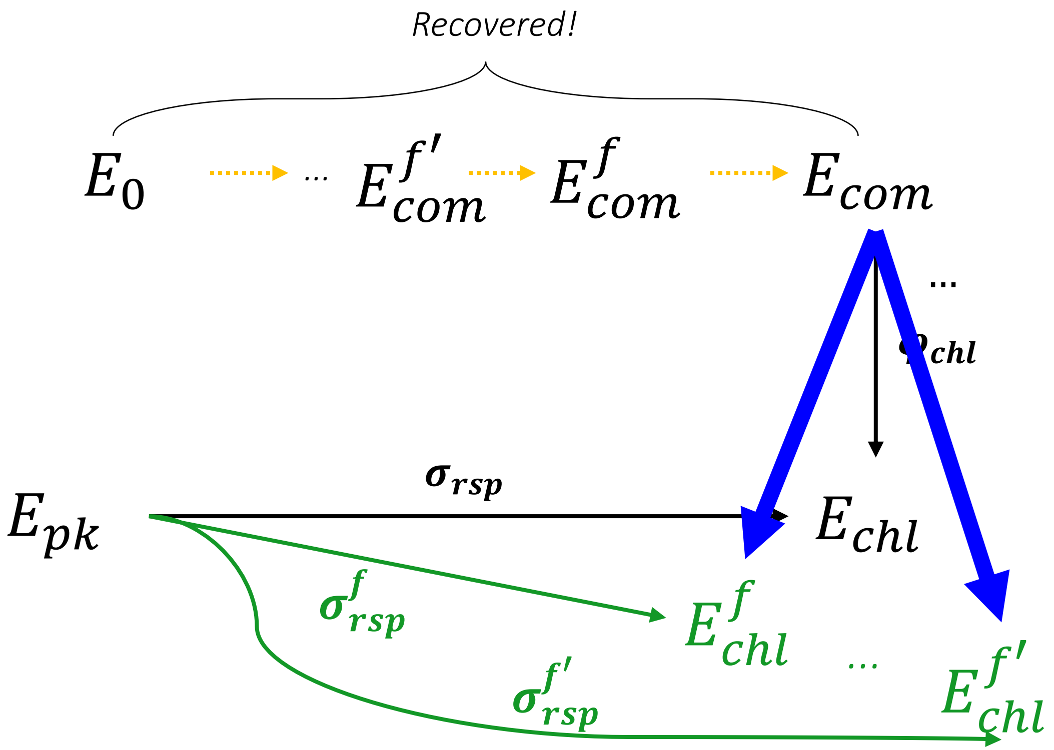
Fault attack on deterministic SQIsign

- **Key recovery attack scenario**
  - Fourth, the attacker repeats the previous steps to recover the entire commitment isogeny
  - Then, she gets the isogeny from E_0 to E_{pk} which is the equivalent secret key



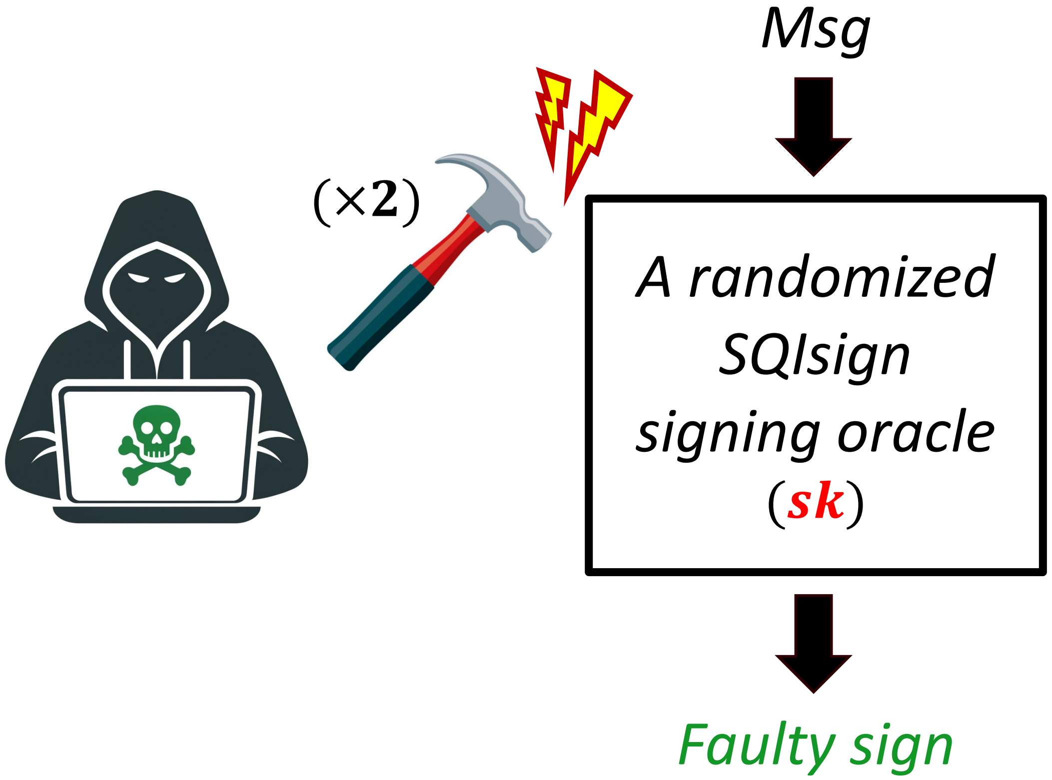
Fault attack on deterministic SQIsign

- **Key recovery attack scenario**
  - When the degree of a commitment isogeny, D_{com} is B-smooth, **the total number of queries** is $\boldsymbol{O(\log(D_{com}))}$ and **the time complexity** is $\boldsymbol{O(B \cdot \log(D_{com}))}$



Fault attack on randomized SQIsign

- **Attacker model**
  - The attacker is allowed to make multiple queries to a randomized SQISign oracle with the same key
  - The oracle generates a signature for each query and the attacker receives it
  - The attacker can inject faults twice during the oracle's operation (2nd order fault attack)



Fault attack on randomized SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 2nd–order fault is injected while computing the commitment isogeny and ideal, respectively

$$\begin{tikzcd}
& E_0 \arrow[r, "\psi_{com}"] & E_{com} \arrow[d, "\varphi_{chl}"] \\
E_{pk} \arrow[rr, "\sigma_{rsp}"'] & & E_{chl}
\end{tikzcd}$$

Elliptic curves

$$\begin{tikzcd}
& \mathcal{O}_0 \arrow[r, "I_{com}"] \arrow[dl, "I_{sk}"'] & \mathcal{O}_{com} \arrow[d, "I_{chl}"] \\
\mathcal{O}_{pk} \arrow[rr, "J_{rsp} \coloneqq SigningKLPT(\bar{I}_{sk} I_{com} I_{chl})"'] & & \mathcal{O}_{chl}
\end{tikzcd}$$

Quaternion algebra

Fault attack on randomized SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 2^{nd}–order fault is injected while computing the commitment isogeny and ideal, respectively

$$
\begin{tikzcd}
E_0 \arrow[r, "\psi^f_{com}"'] & E^f_{com} \arrow[r, "\psi_{com}"] & E_{com} \arrow[d, "\varphi_{chl}"] \\
E_{pk} \arrow[rr, "\sigma_{rsp}"'] & & E_{chl}
\end{tikzcd}
$$

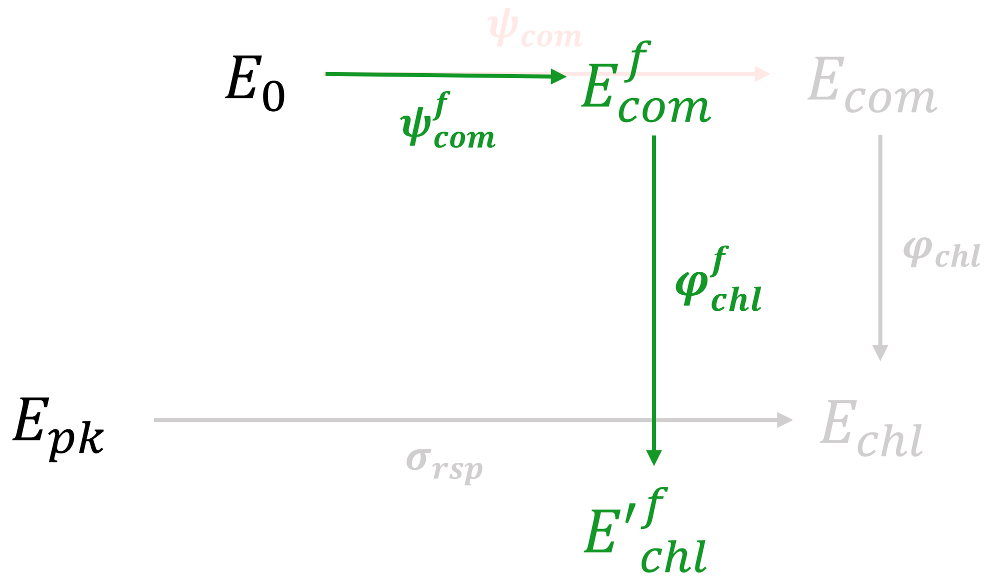
Elliptic curves

$$
\begin{tikzcd}
\mathcal{O}_0 \arrow[r, "I_{com}"] \arrow[d, "I_{sk}"'] \arrow[loop right, "I^f_{com} = \mathcal{O}_0"] & \mathcal{O}_{com} \arrow[d, "I_{chl}"] \\
\mathcal{O}_{pk} \arrow[r, "J_{rsp} \coloneqq SigningKLPT(\bar{I}_{sk} I_{com} I_{chl})"'] & \mathcal{O}_{chl}
\end{tikzcd}
$$

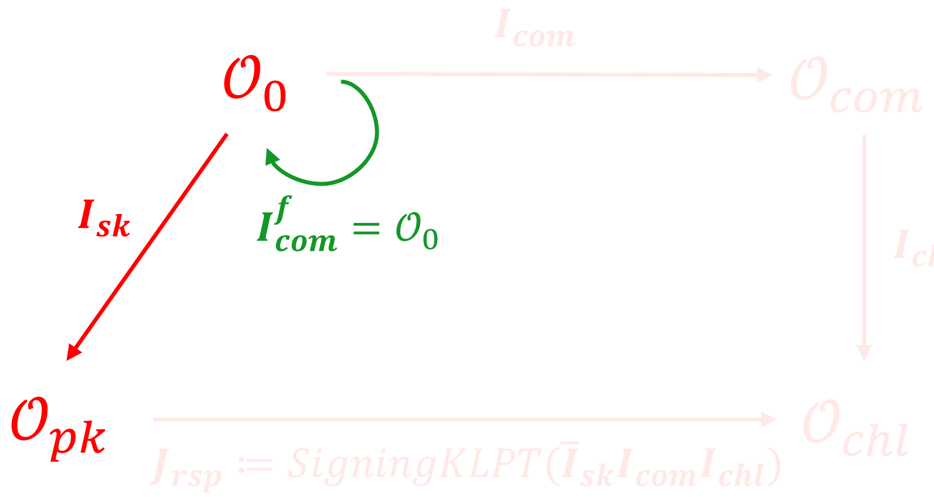
Quaternion algebra

Fault attack on randomized SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 2nd–order fault is injected while computing the commitment isogeny and ideal, respectively



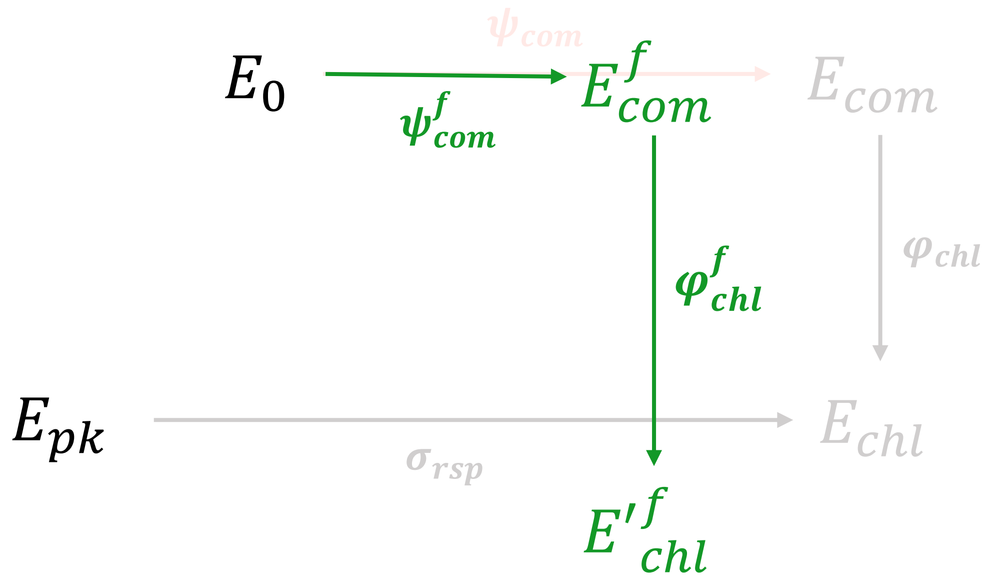
Elliptic curves



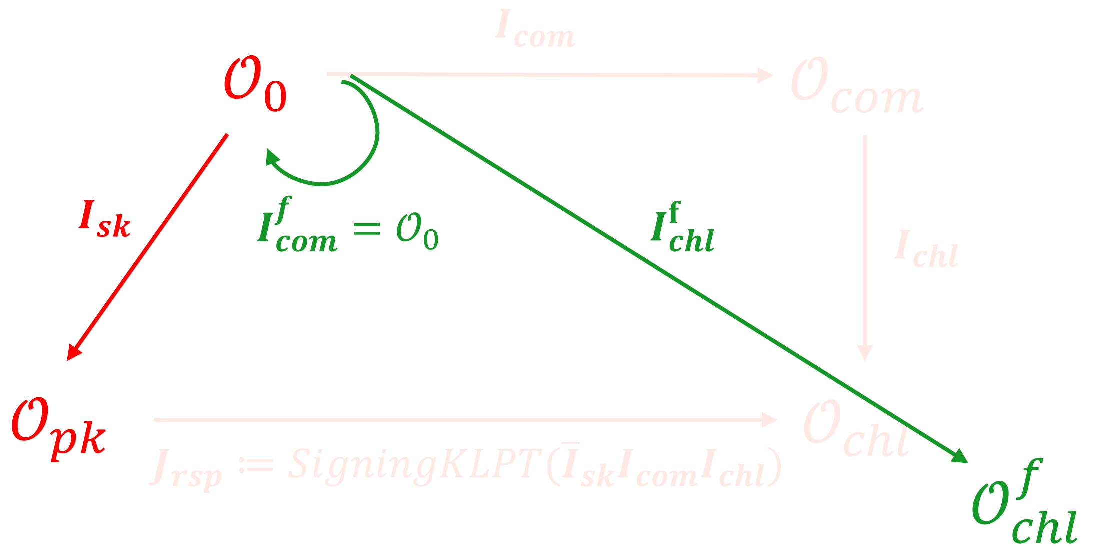
Quaternion algebra

Fault attack on randomized SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 2^{nd}–order fault is injected while computing the commitment isogeny and ideal, respectively



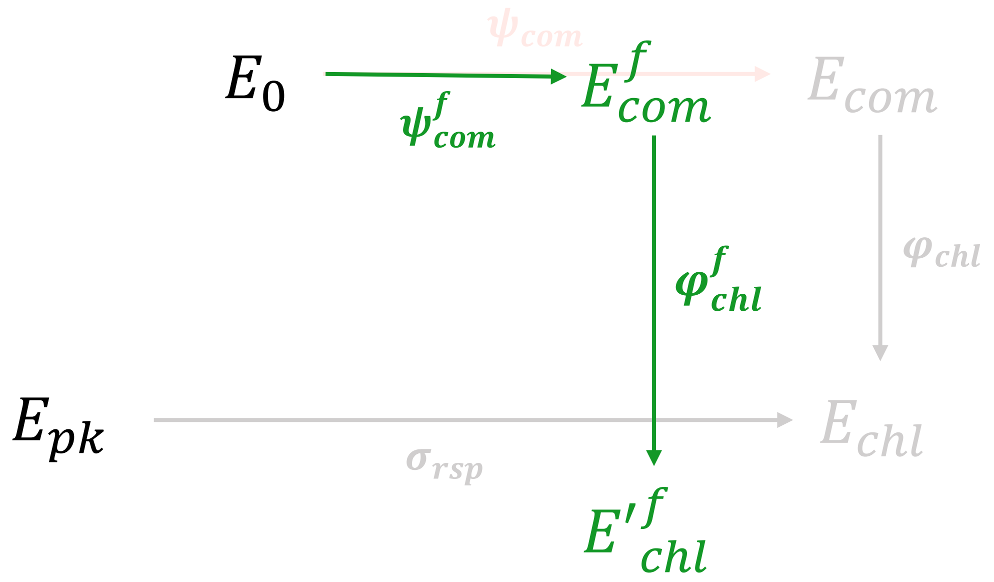
Elliptic curves



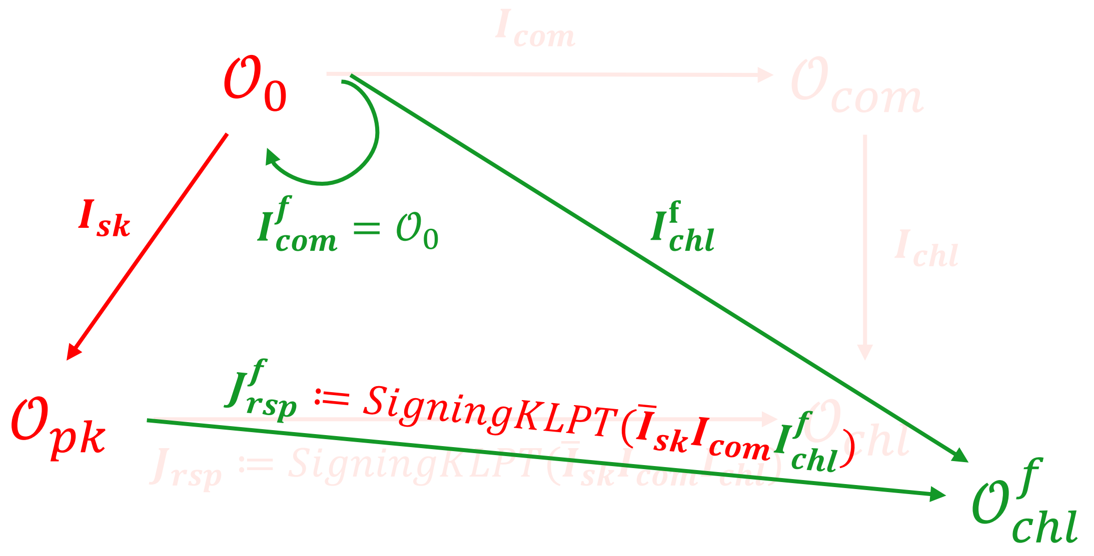
Quaternion algebra

Fault attack on randomized SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 2nd–order fault is injected while computing the commitment isogeny and ideal, respectively



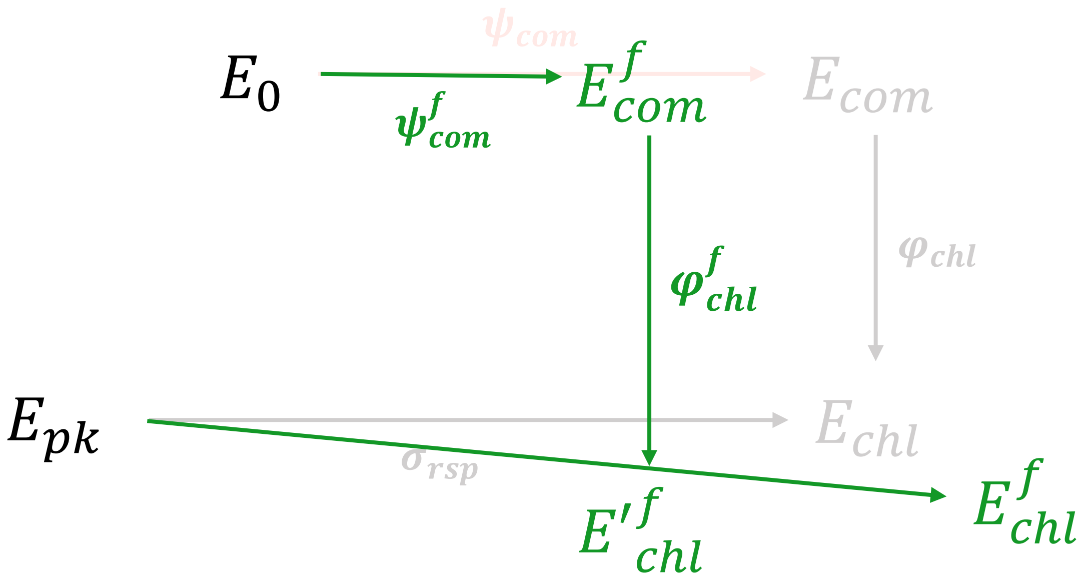
Elliptic curves



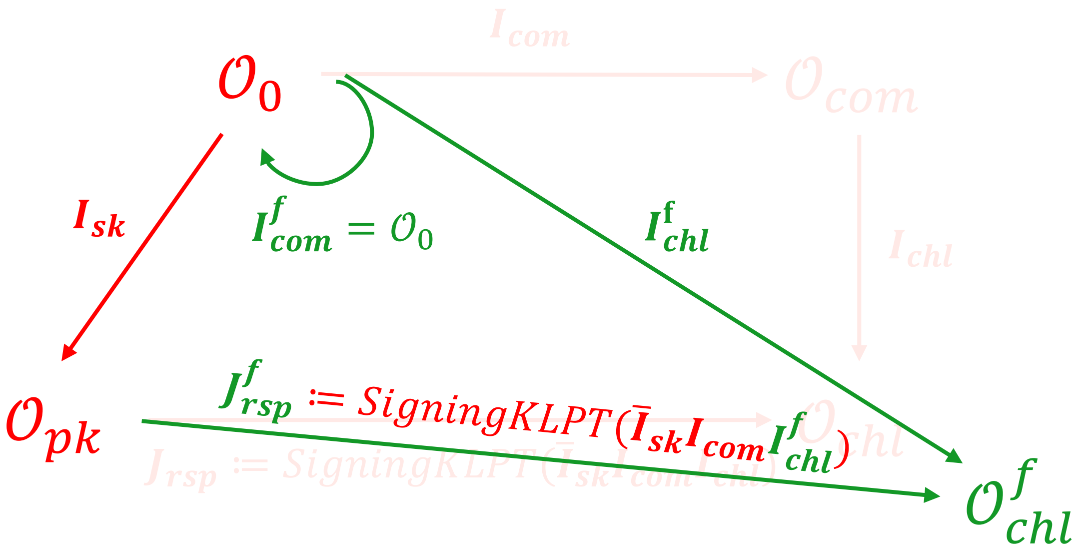
Quaternion algebra

Fault attack on randomized SQIsign

- **The data flow of the faulty SQIsign signing process**
  - 2nd–order fault is injected while computing the commitment isogeny and ideal, respectively



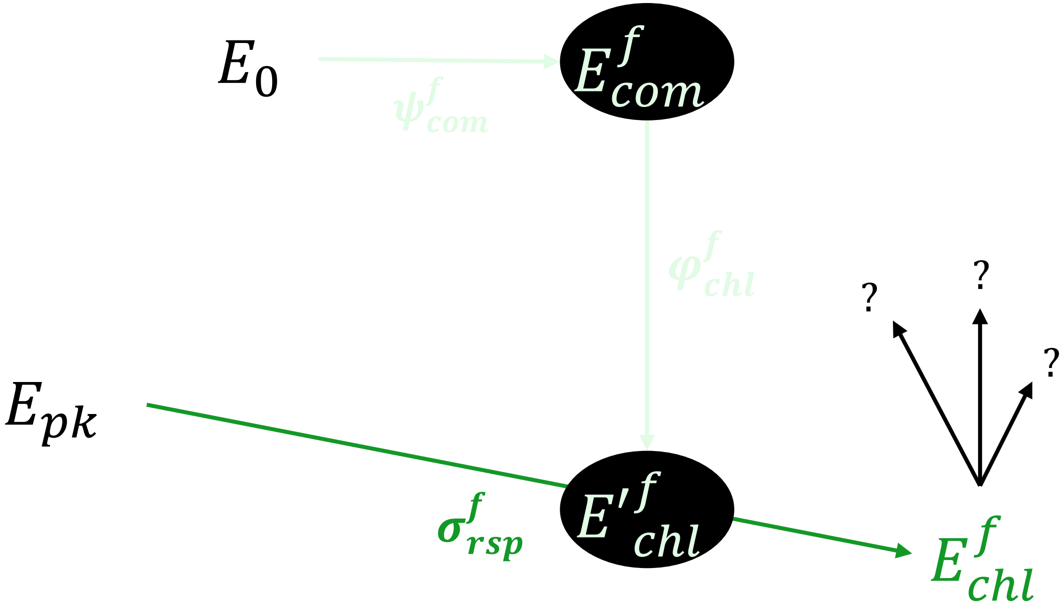
Elliptic curves

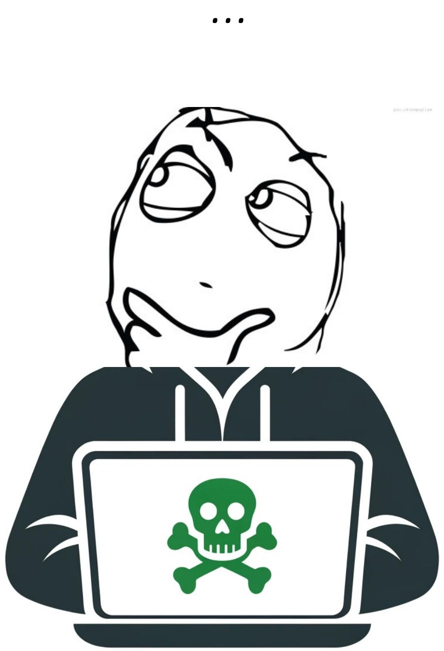


Quaternion algebra

Fault attack on randomized SQIsign

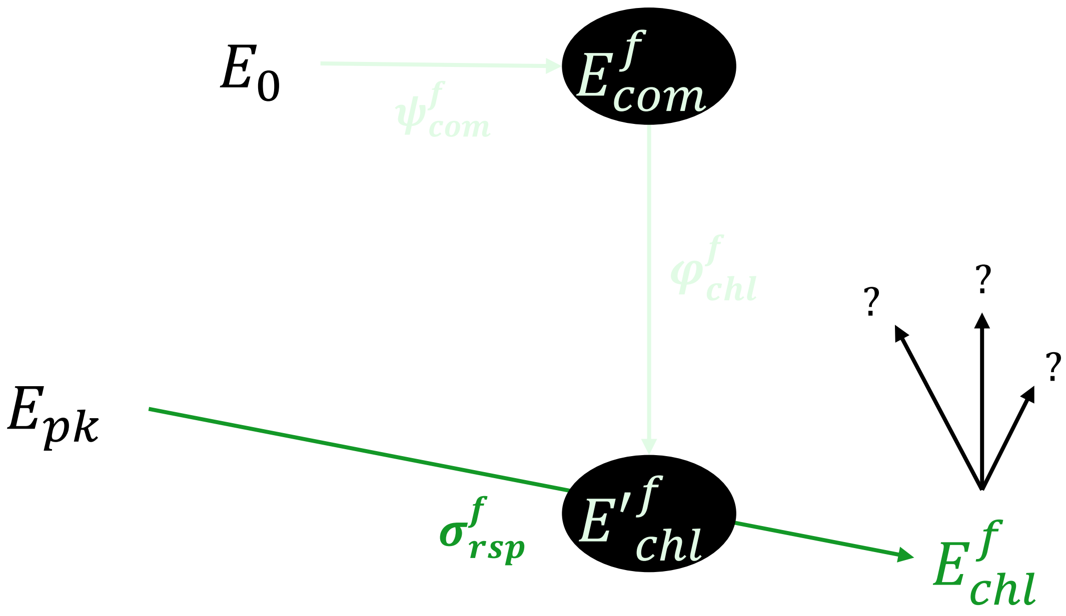
- **What the attacker receives…**

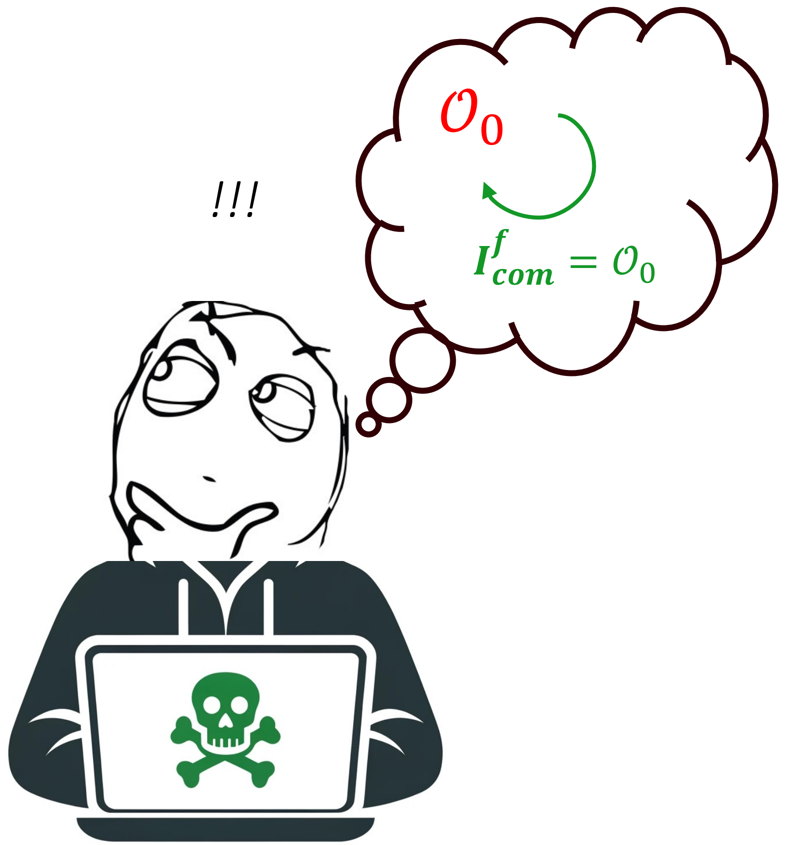




Fault attack on randomized SQIsign

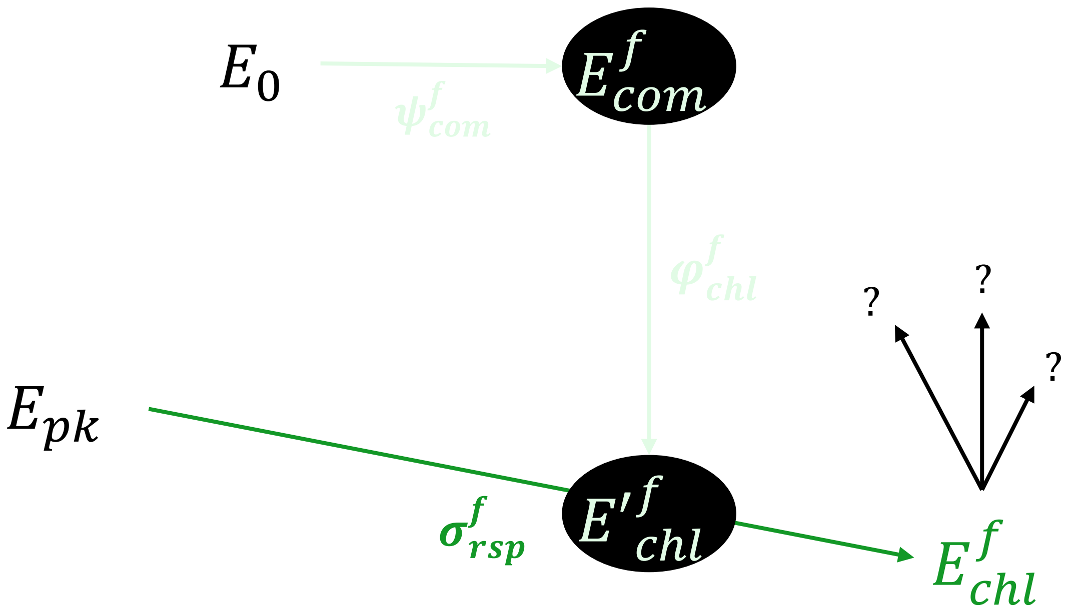
- **What the attacker receives…**

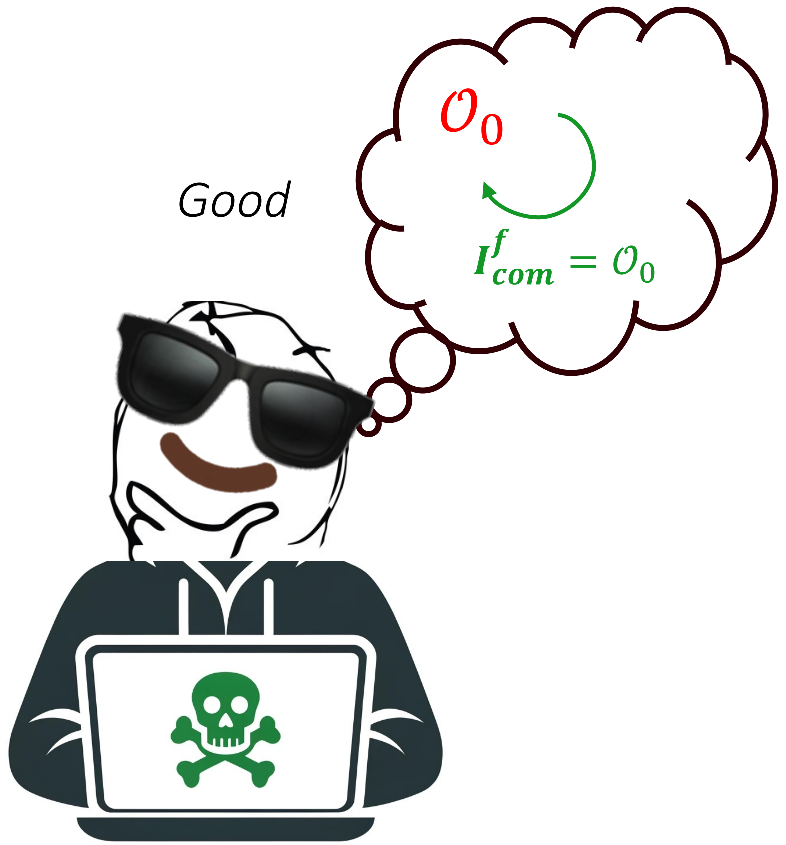




Fault attack on randomized SQIsign

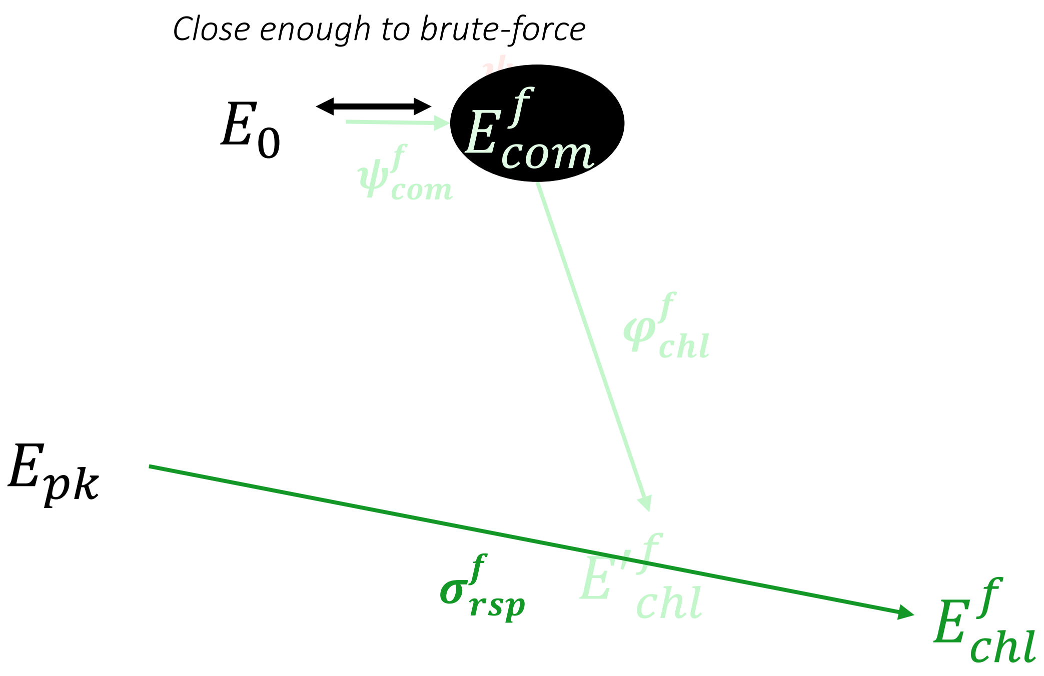
- **What the attacker receives…**





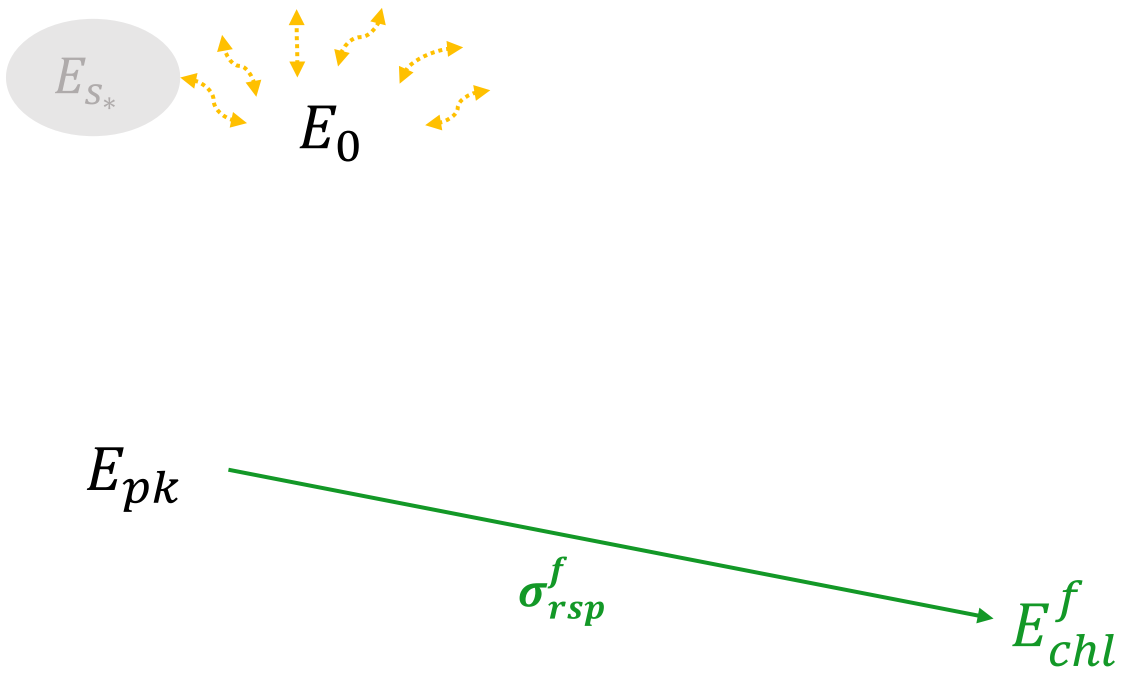
Fault attack on randomized SQIsign

- **Key recovery attack scenario**
  - First, the attacker queries the signing oracle with 2^{nd} order fault injection causing $\boldsymbol{E^f_{com}}$ **to be close to** $\boldsymbol{E_0}$ and $\boldsymbol{I_{com}}$ **to become** $\boldsymbol{\mathcal{O}_0}$



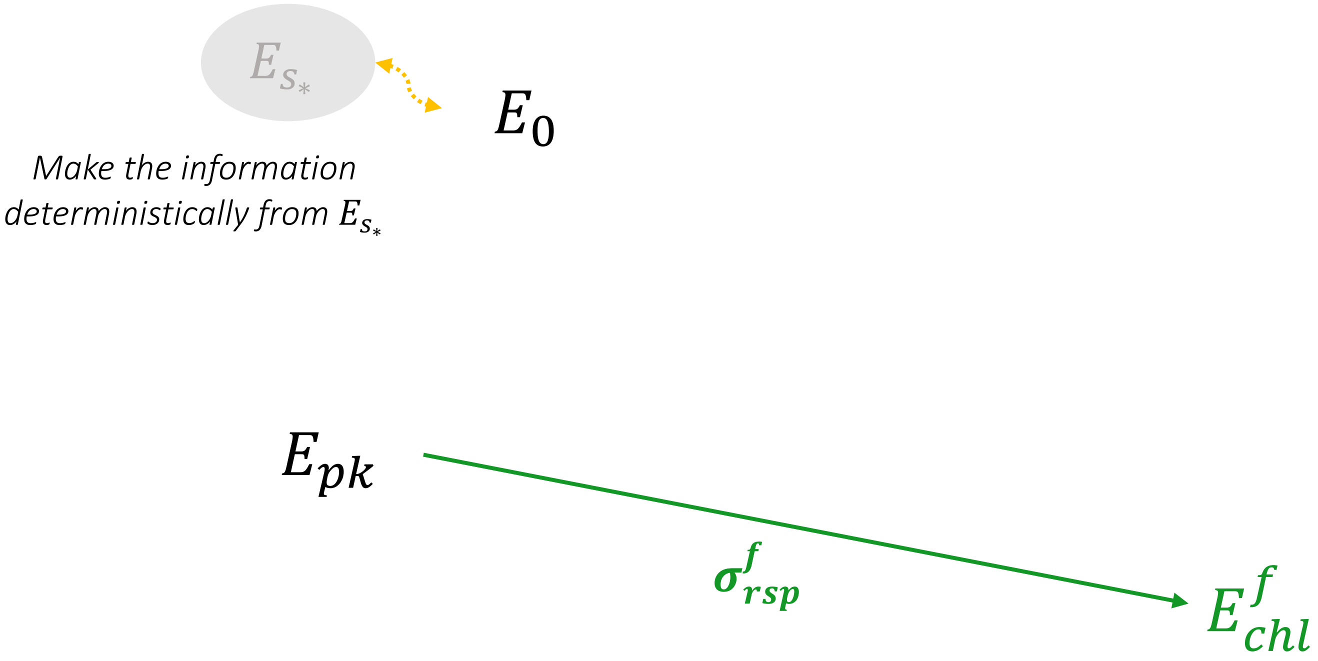
Fault attack on randomized SQIsign

- **Key recovery attack scenario**
  - Second, the attacker brute-force to find the isogeny from E_0 and E_{com}^f
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **wrong...**



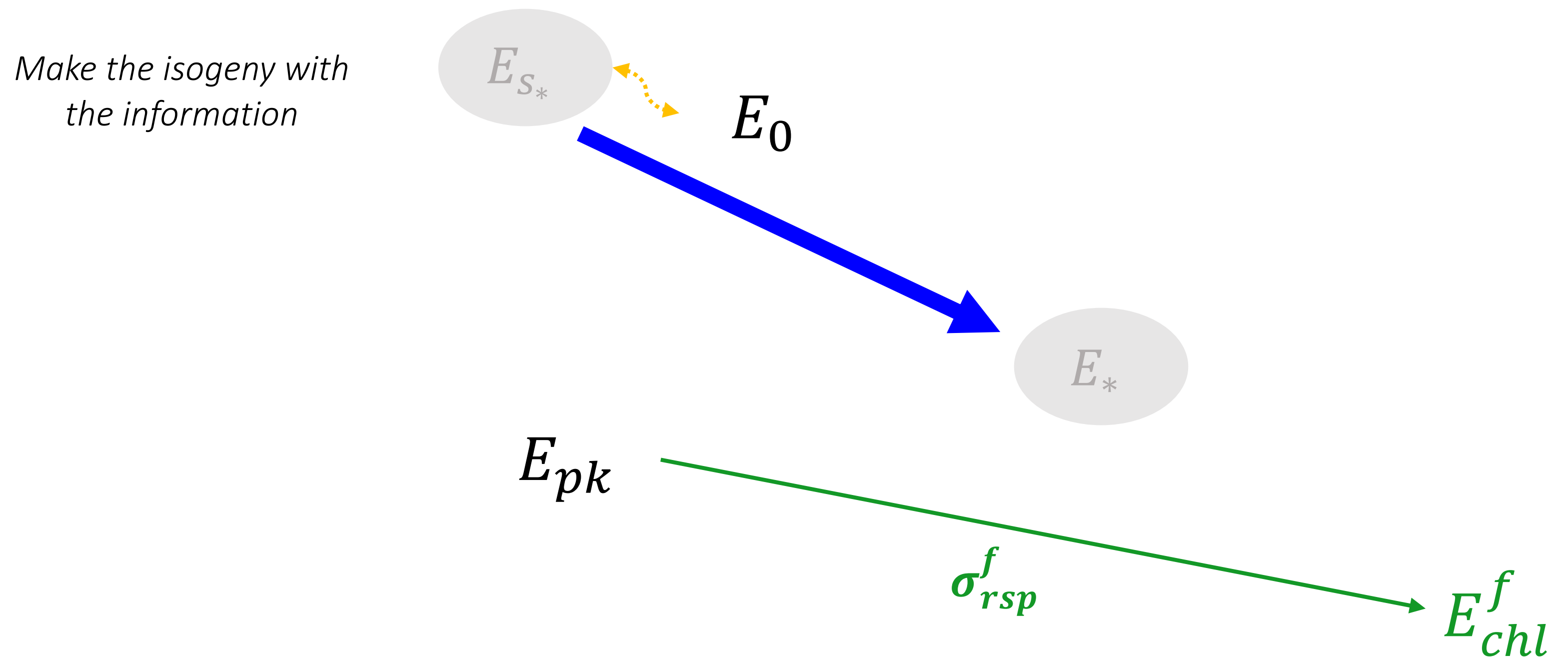
Fault attack on randomized SQIsign

- **Key recovery attack scenario**
  - Second, the attacker brute-force to find the isogeny from E_0 and E_{com}^f
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **wrong…**



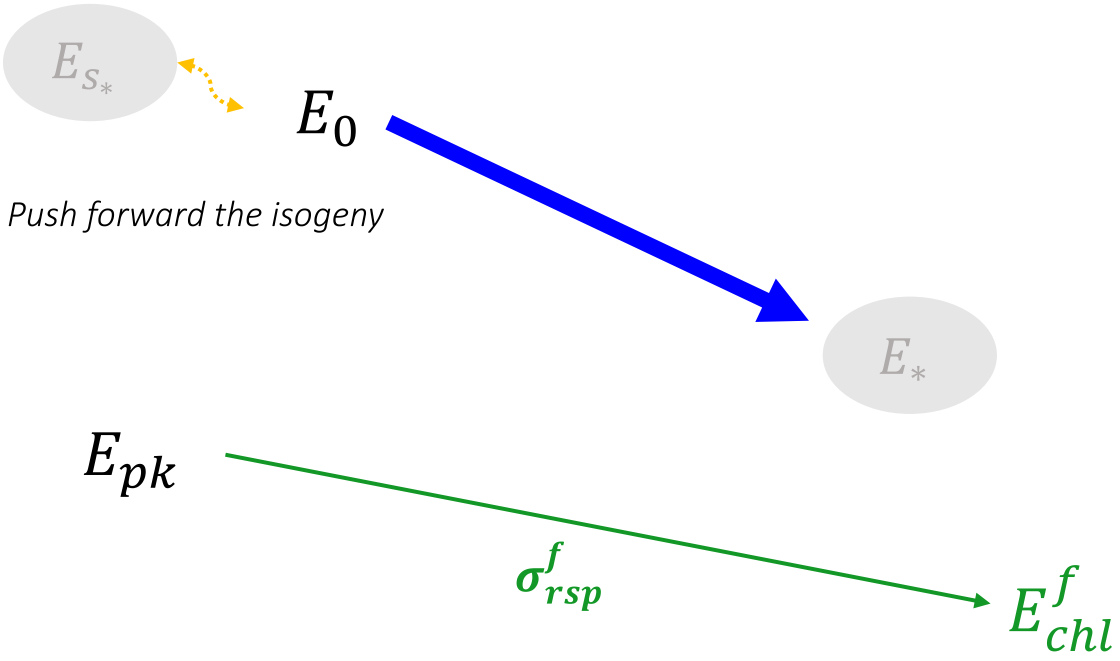
Fault attack on randomized SQIsign

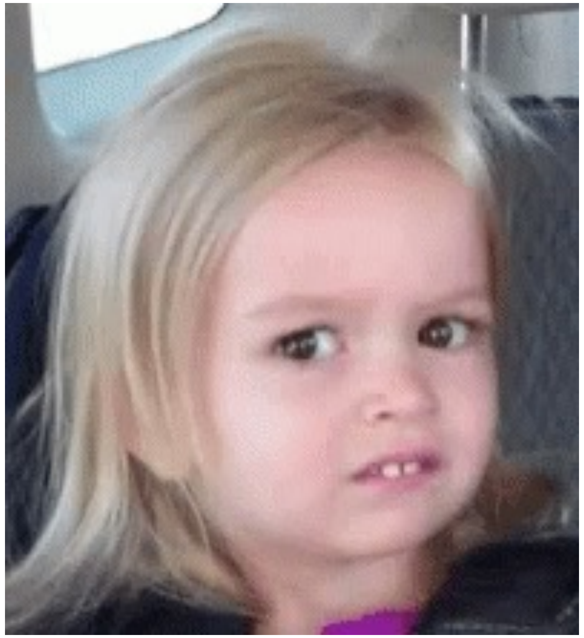
- **Key recovery attack scenario**
  - Second, the attacker brute-force to find the isogeny from E_0 and E_{com}^{f}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **wrong…**



Fault attack on randomized SQIsign

- **Key recovery attack scenario**
  - Second, the attacker brute-force to find the isogeny from E_0 and E^f_{com}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **wrong...**





Fault attack on randomized SQIsign

- **Key recovery attack scenario**
  - Second, the attacker brute-force to find the isogeny from E_0 and E_{com}^{f}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **right…**

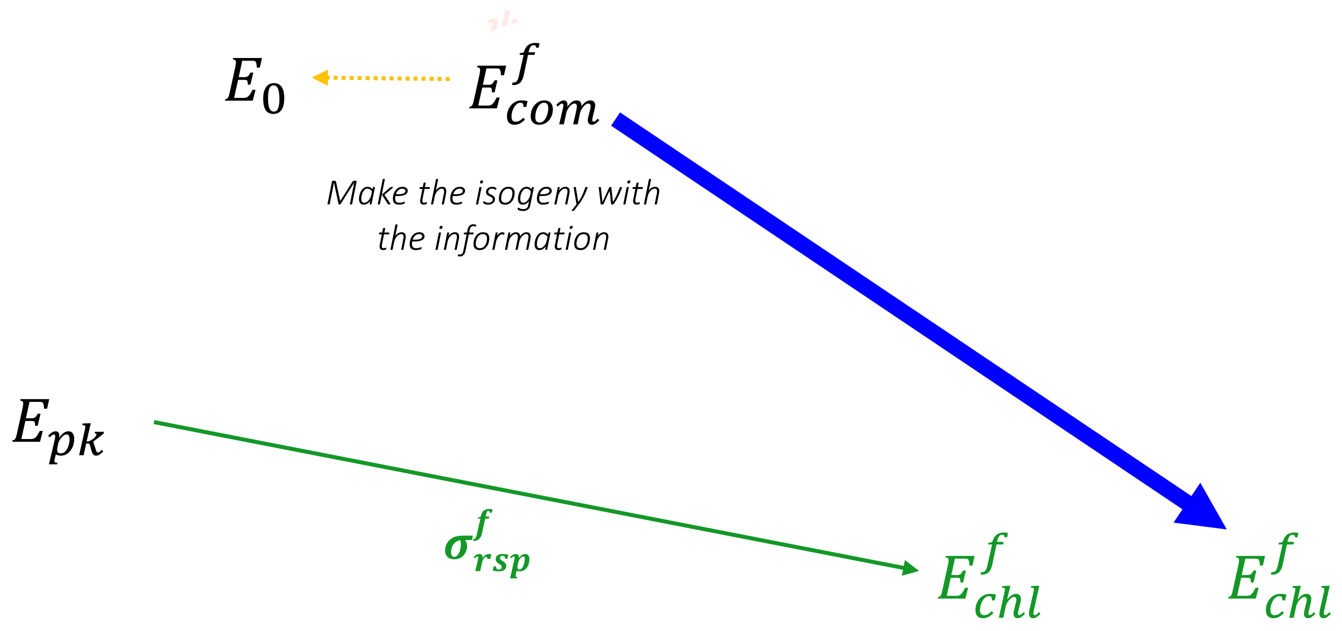
Make the information deterministically from E_{com}^{f}

$$E_0 \longleftarrow E_{com}^{f}$$

$$E_{pk} \xrightarrow{\sigma_{rsp}^{f}} E_{chl}^{f}$$

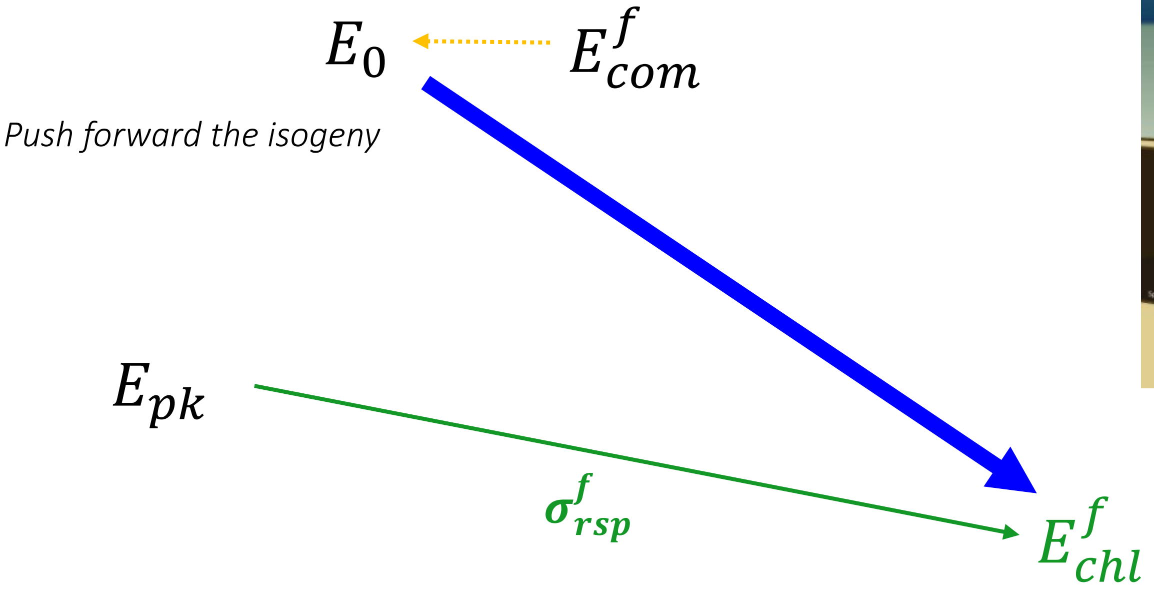
Fault attack on randomized SQIsign

- **Key recovery attack scenario**
  - Second, the attacker brute-force to find the isogeny from E_0 and E^f_{com}
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **right…**



Fault attack on randomized SQIsign

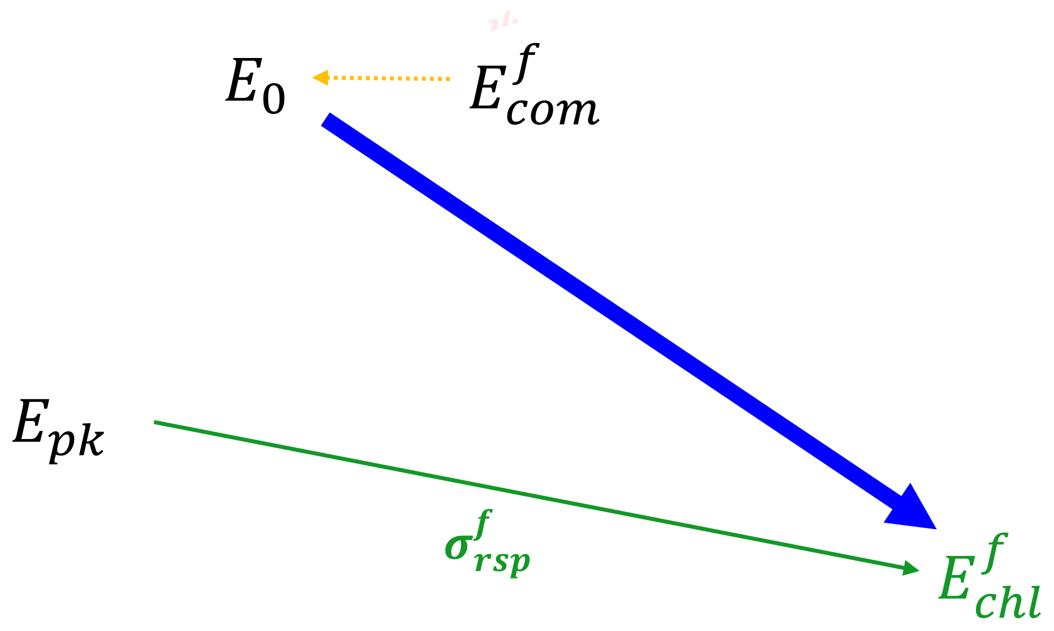
- **Key recovery attack scenario**
  - Second, the attacker brute-force to find the isogeny from E_0 and E_{com}^f
  - How can she check whether the guessed curve is right or wrong?
    - If the guessed curve is **right…**
  - She gets the isogeny from E_0 to E_{pk} which is the equivalent secret key



Fault attack on randomized SQIsign

- **Key recovery attack scenario**
  - When the degree of a commitment isogeny D_{com} is B-smooth,

    the total number of queries is $\boldsymbol{O}\left(\left(\frac{\boldsymbol{D_{com}}}{\boldsymbol{\varphi(D_{com})}}\right)^{\boldsymbol{3}}\right)$ by Theorem 2 and **the time complexity** is $\boldsymbol{O(B)}$



Conclusion and countermeasures

1. **We found two fault vulnerabilities in SQIsign**

2. **We showed the key recovery attack scenarios on both deterministic SQIsign and randomized SQIsign using these vulnerabilities**

3. **Both vulnerabilities can be countered using intuitive methods**

   - Verify the iterator and norm!
   - Multiple checkers for high-order fault attacks!

Q&A

Thanks

https://ia.cr/2024/581

Email : hwani0814@korea.ac.kr