SCIENCES SORBONNE UNIVERSITÉ

THALES

Implémentations d'Algorithmes de Cryptographie Post-Quantique Sécurisées contre les Attaques Physiques

CALLE VIERA Andersson

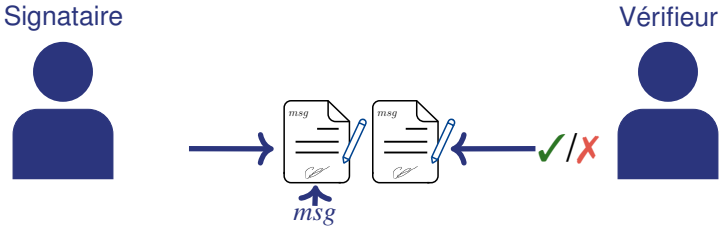
WRACH 2025

Le 22 avril 2025

Implémentations d'Algorithmes de **Cryptographie Post-Quantique** Sécurisées contre les Attaques Physiques

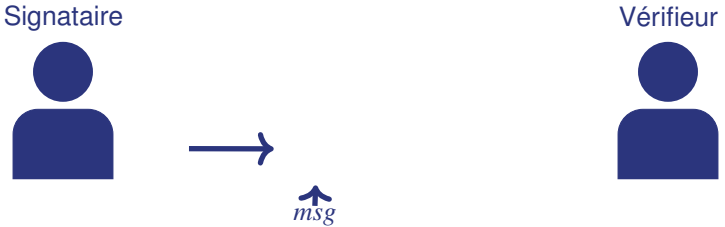
Signatures numériques

- **Reproduire les caractéristiques d'une signature manuscrite :**
  - Lier un document à son auteur
  - Garantir l'intégrité du document
  - Rendre la signature infalsifiable



Signatures numériques

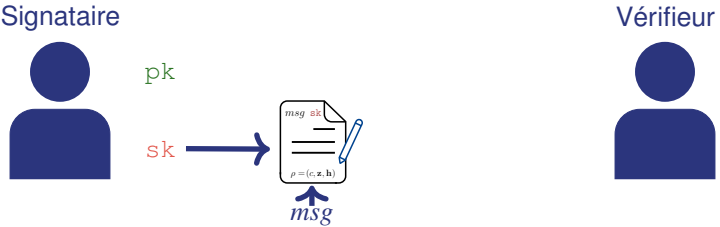
- **Reproduire les caractéristiques d'une signature manuscrite :**
  - Lier un document à son auteur
  - Garantir l'intégrité du document
  - Rendre la signature infalsifiable



- **Objectifs :**
  - Seulement le signataire avec sa clé secrète `sk` peut générer une signature d'un msg
  - Quiconque avec la clé publique `pk` peut vérifier l'authenticité de la signature d'un msg

Signatures numériques

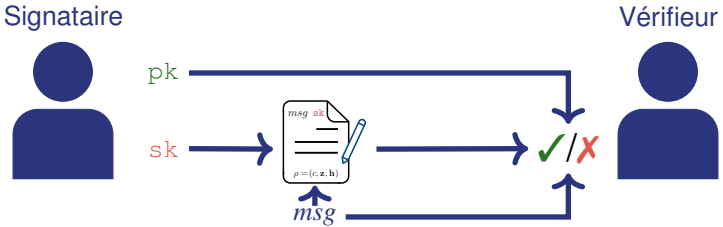
- **Reproduire les caractéristiques d'une signature manuscrite :**
  - Lier un document à son auteur
  - Garantir l'intégrité du document
  - Rendre la signature infalsifiable



- **Objectifs :**
  - Seulement le signataire avec sa clé secrète `sk` peut générer une signature d'un msg
  - Quiconque avec la clé publique `pk` peut vérifier l'authenticité de la signature d'un msg

Signatures numériques

- **Reproduire les caractéristiques d'une signature manuscrite :**
  - Lier un document à son auteur
  - Garantir l'intégrité du document
  - Rendre la signature infalsifiable



- **Objectifs :**
  - Seulement le signataire avec sa clé secrète `sk` peut générer une signature d'un msg
  - Quiconque avec la clé publique `pk` peut vérifier l'authenticité de la signature d'un msg

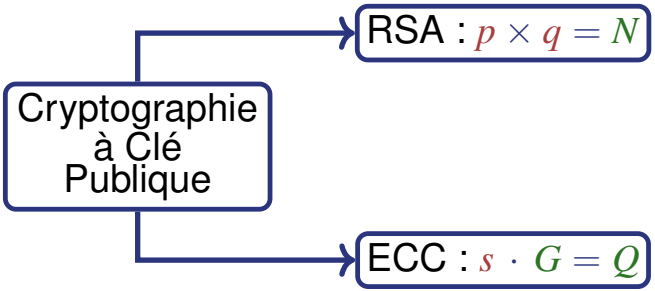
Cryptographie post-quantique

- Ordinateurs quantique (assez puissant) pourraient arriver d'ici 10 à 30 ans, ou jamais ...

Cryptographie
à Clé
Publique

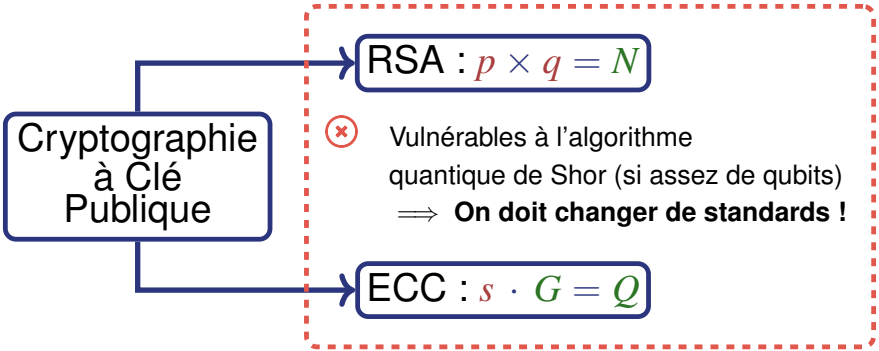
Cryptographie post-quantique

- Ordinateurs quantique (assez puissant) pourraient arriver d'ici 10 à 30 ans, ou jamais ...



Cryptographie post-quantique

- Ordinateurs quantique (assez puissant) pourraient arriver d'ici 10 à 30 ans, ou jamais ...



Quelles alternatives ?

NIST : National Institute of Standards and Technology

Quelles alternatives ?

NIST : National Institute of Standards and Technology

1997 : Compétition qui a standardisé l'**A**dvanced **E**ncryption **S**tandard

Quelles alternatives ?

NIST : National Institute of Standards and Technology

1997 : Compétition qui a standardisé l'**A**dvanced **E**ncryption **S**tandard

2007 : Compétition qui a standardisé **S**ecure **H**ash **A**lgorithm

Quelles alternatives ?

NIST : National Institute of Standards and Technology

1997 : Compétition qui a standardisé l'**A**dvanced **E**ncryption **S**tandard

2007 : Compétition qui a standardisé **S**ecure **H**ash **A**lgorithm

2016 : Compétition pour standardiser des algorithmes Post-Quantiques

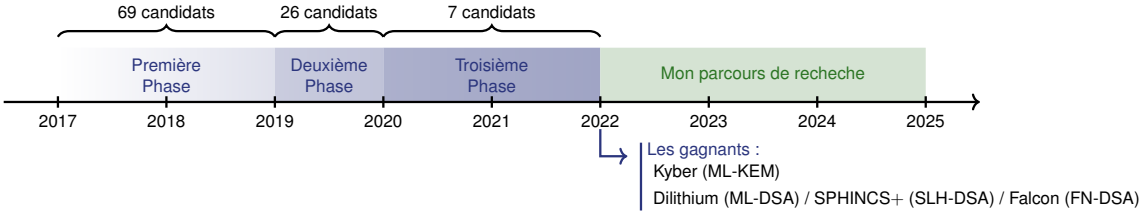
Quelles alternatives ?

NIST : National Institute of Standards and Technology

1997 : Compétition qui a standardisé l'**A**dvanced **E**ncryption **S**tandard

2007 : Compétition qui a standardisé **S**ecure **H**ash **A**lgorithm

2016 : Compétition pour standardiser des algorithmes Post-Quantiques



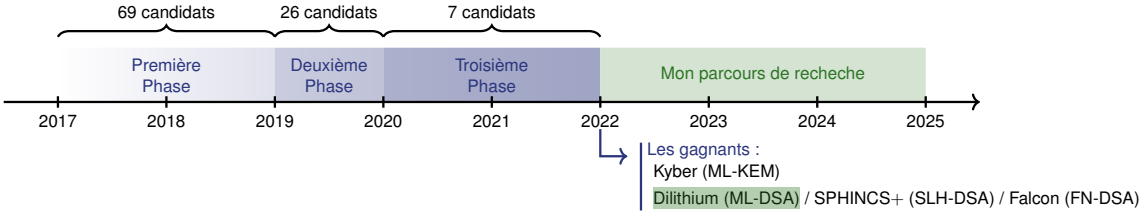
Quelles alternatives ?

NIST : National Institute of Standards and Technology

1997 : Compétition qui a standardisé l'**A**dvanced **E**ncryption **S**tandard

2007 : Compétition qui a standardisé **S**ecure **H**ash **A**lgorithm

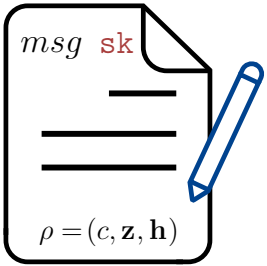
2016 : Compétition pour standardiser des algorithmes Post-Quantiques

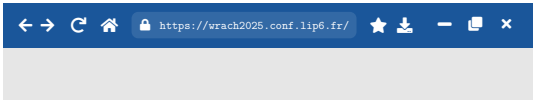


Implémentations d'Algorithmes de Cryptographie Post-Quantique Sécurisées contre les Attaques Physiques

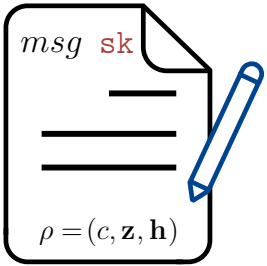
Implémentations

- **Communications**

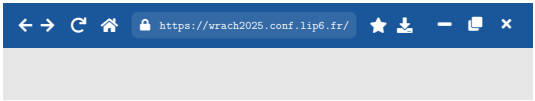




Implémentations

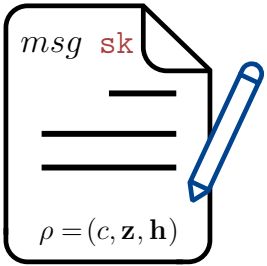


- **Communications**

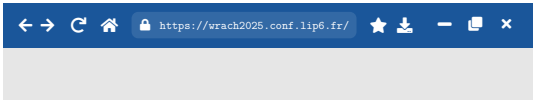


- **Vérification de logiciels**

Implémentations



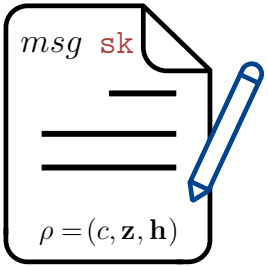
- **Communications**



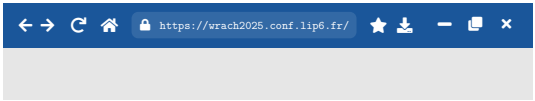
- **Vérification de logiciels**

- **Systèmes embarqués**

Implémentations



- **Communications**



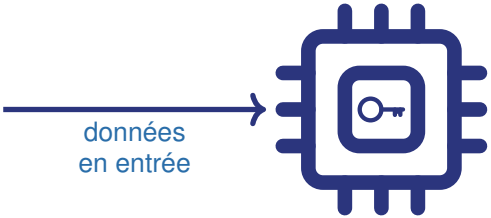
- **Vérification de logiciels**

- **Systèmes embarqués**

- **Contraintes :**
  - **Ressources limitées :** mémoire, puissance, . . .
  - **Sécurité :** l'utilisateur peut être malveillant

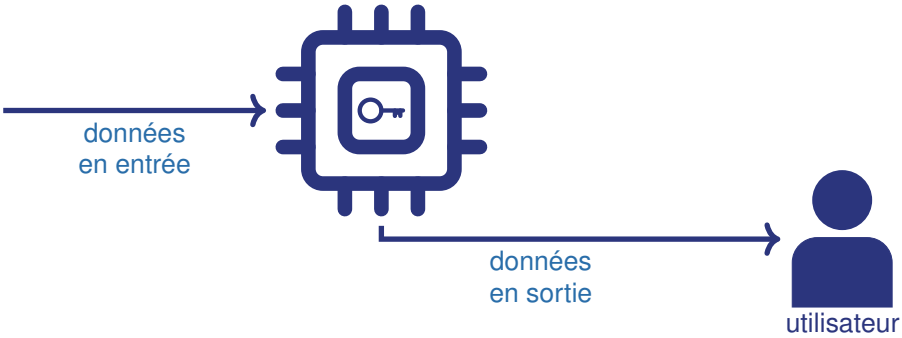
Implémentations d'Algorithmes de Cryptographie Post-Quantique Sécurisées contre les Attaques Physiques

Attaques par canaux auxiliaires (SCA)



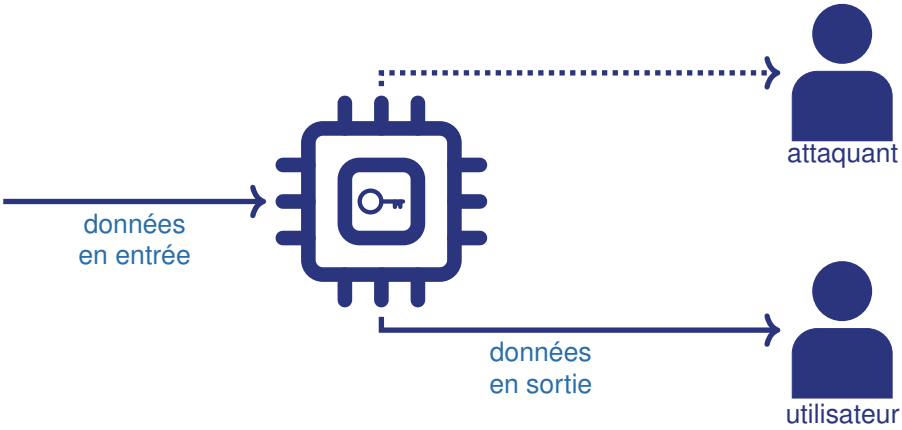
Attaques par canaux auxiliaires (SCA)

- Au lieu d'attaquer une information sensible ...



Attaques par canaux auxiliaires (SCA)

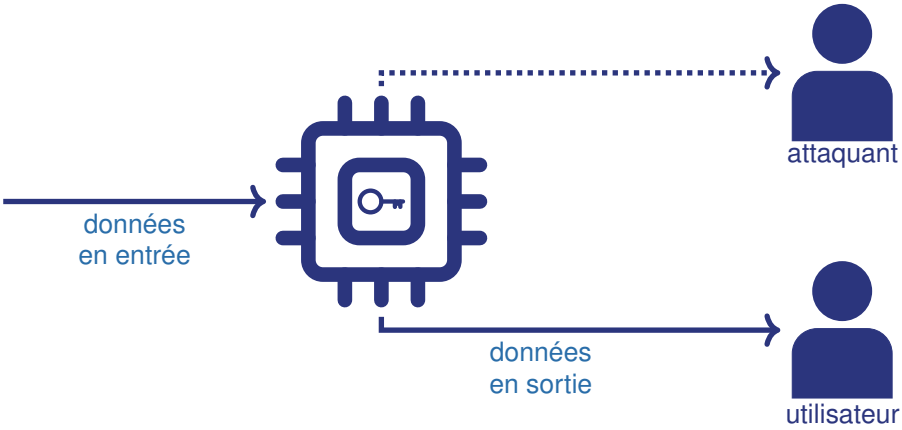
- Au lieu d'attaquer une information sensible ...
  ... on peut inférer dessus
  grâce à son implémentation



Attaques par canaux auxiliaires (SCA)

- Au lieu d'attaquer une information sensible ...
  ... on peut inférer dessus
  grâce à son implémentation

: temps d'exécution

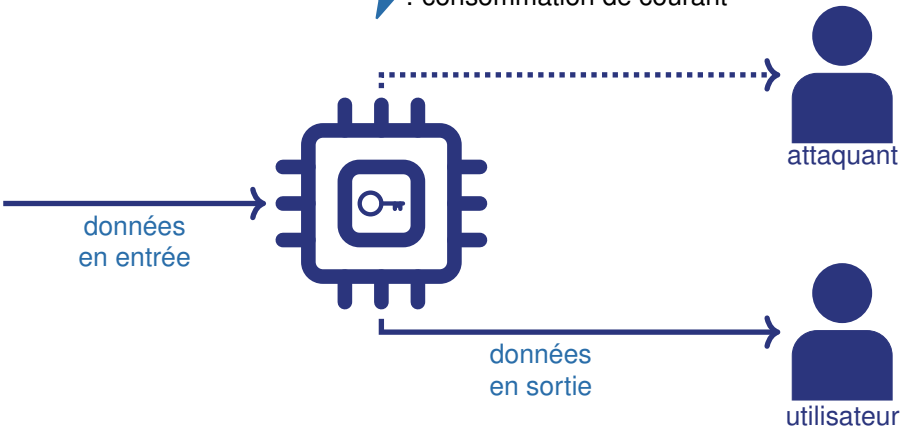


Attaques par canaux auxiliaires (SCA)

- Au lieu d'attaquer une information sensible ...
  ... on peut inférer dessus
  grâce à son implémentation

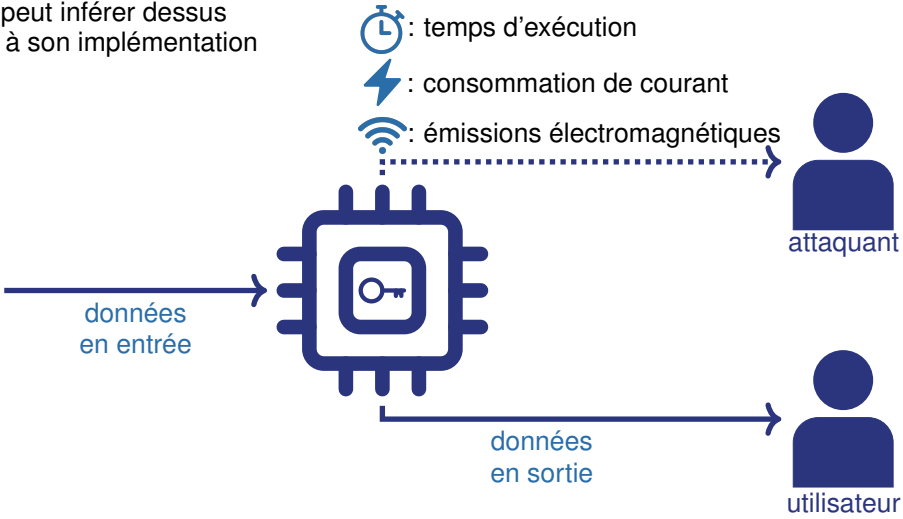
: temps d'exécution

: consommation de courant



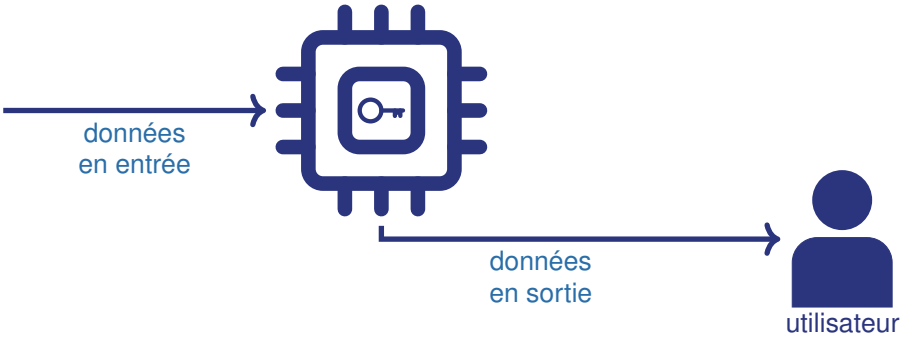
Attaques par canaux auxiliaires (SCA)

- Au lieu d'attaquer une information sensible ...
  ... on peut inférer dessus
  grâce à son implémentation



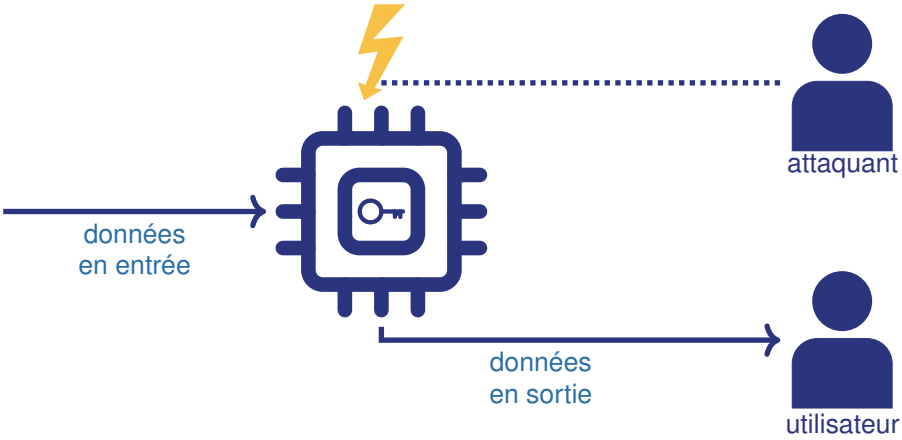
Attaques par fautes (FA)

- Au lieu d'observer une information ...



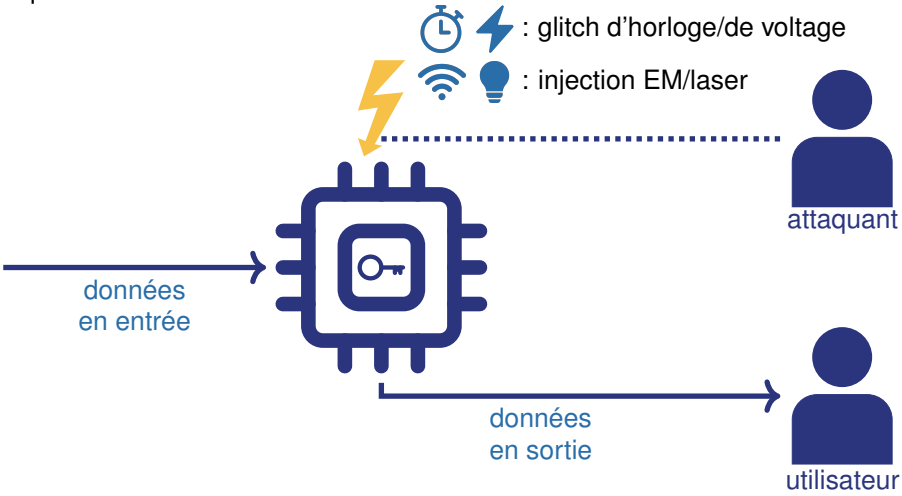
Attaques par fautes (FA)

- Au lieu d'observer une information ...
  ... l'attaquant est actif



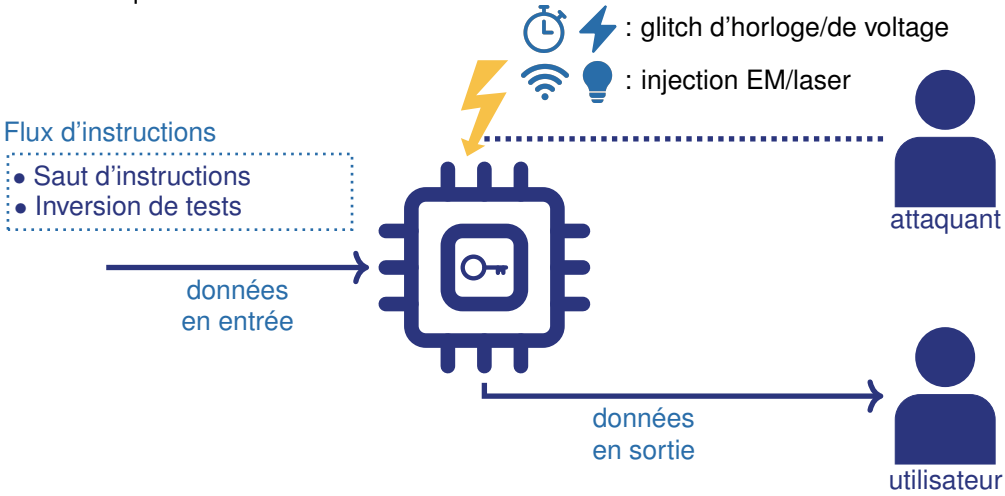
Attaques par fautes (FA)

- Au lieu d'observer une information ...
  ... l'attaquant est actif



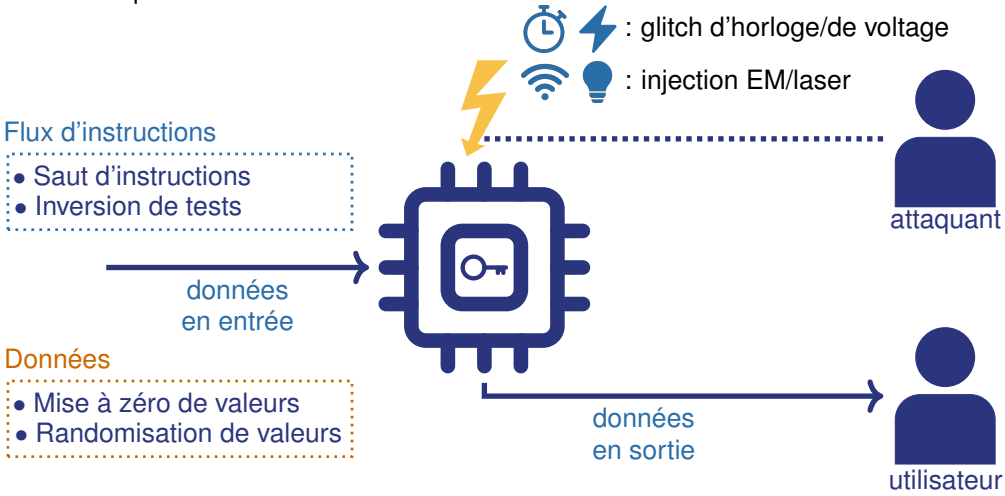
Attaques par fautes (FA)

- Au lieu d'observer une information ...
  ... l'attaquant est actif



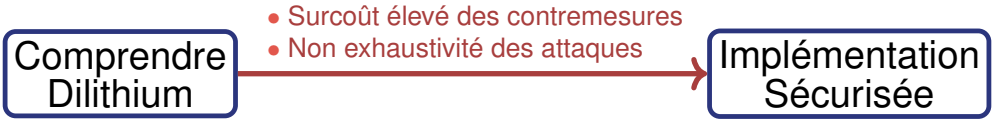
Attaques par fautes (FA)

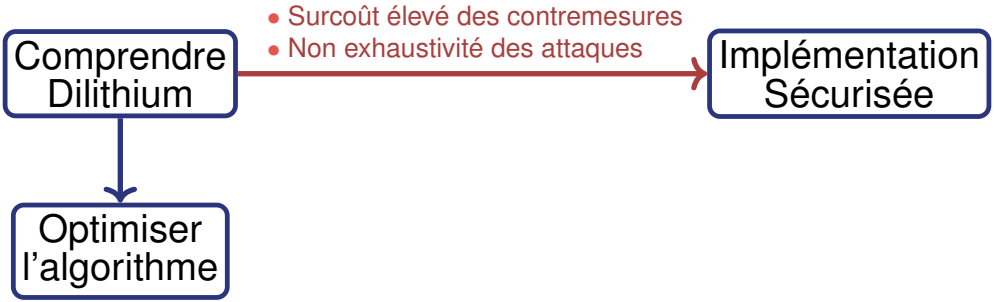
- Au lieu d'observer une information ...
  ... l'attaquant est actif

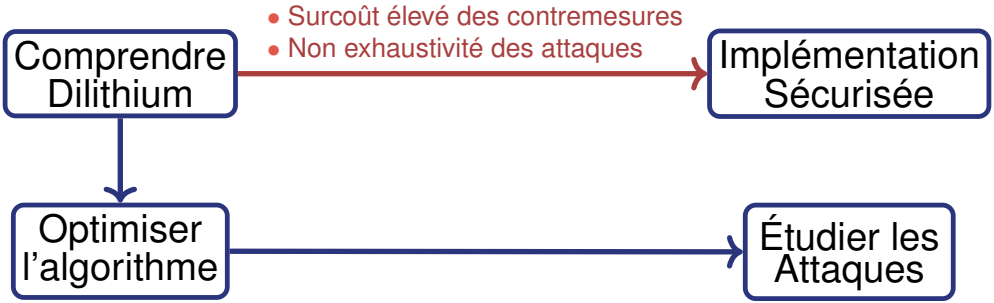


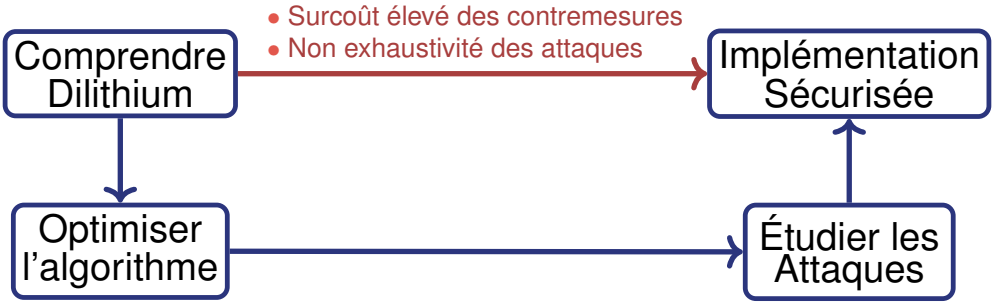
Comprendre Dilithium

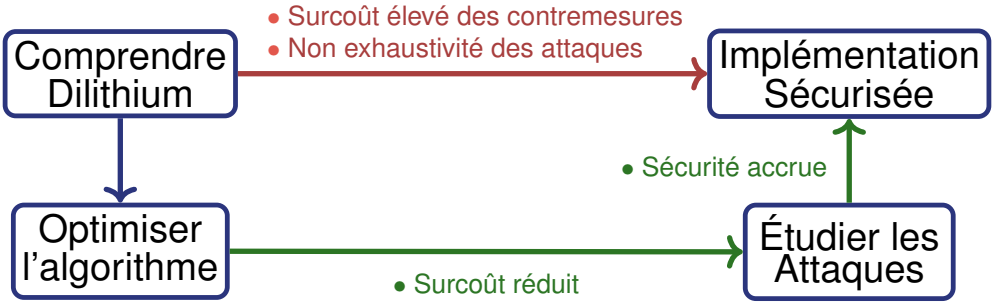

Comprendre Dilithium
Implémentation Sécurisée


Comprendre Dilithium
• Surcoût élevé des contremesures
• Non exhaustivité des attaques
Implémentation Sécurisée


Comprendre Dilithium
• Surcoût élevé des contremesures
• Non exhaustivité des attaques
Implémentation Sécurisée
Optimiser l'algorithme


Comprendre Dilithium
• Surcoût élevé des contremesures
• Non exhaustivité des attaques
Implémentation Sécurisée
Optimiser l'algorithme
Étudier les Attaques


• Surcoût élevé des contremesures
• Non exhaustivité des attaques
Comprendre Dilithium
Implémentation Sécurisée
Optimiser l'algorithme
Étudier les Attaques


Comprendre Dilithium
• Surcoût élevé des contremesures
• Non exhaustivité des attaques
Implémentation Sécurisée
• Sécurité accrue
Optimiser l'algorithme
• Surcoût réduit
Étudier les Attaques


Comprendre Dilithium
Implémentation Sécurisée
Optimiser l'algorithme
Étudier les Attaques

Dilithium

- Algorithme de signature numérique, sécurité basée sur les problèmes M-LWE et M-SIS
  (pas d'algorithme, classique ou quantique connu pour résoudre ces problèmes efficacement)

Dilithium

- Algorithme de signature numérique, sécurité basée sur les problèmes M-LWE et M-SIS
  (pas d'algorithme, classique ou quantique connu pour résoudre ces problèmes efficacement)
- Anneau $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ avec $n = 256 = 2^8$ et $q = 8\,380\,417 = 2^{23} - 2^{13} + 1$

Version	**Dilithium-2** (SHA-256)	**Dilithium-3** (AES-192)	**Dilithium-5** (AES-256)
η	2	4	2
(k, ℓ)	$(4, 4)$	$(6, 5)$	$(8, 7)$
γ_1	2^{17}	2^{19}	2^{19}
γ_2	$\frac{q-1}{88}$	$\frac{q-1}{32}$	$\frac{q-1}{32}$
$\beta = \eta \times \tau$	$2 \times 39 = 78$	$4 \times 49 = 196$	$2 \times 60 = 120$

Dilithium

- Algorithme de signature numérique, sécurité basée sur les problèmes M-LWE et M-SIS
  (pas d'algorithme, classique ou quantique connu pour résoudre ces problèmes efficacement)
- Anneau $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ avec $n = 256 = 2^8$ et $q = 8\,380\,417 = 2^{23} - 2^{13} + 1$

Version	Dilithium-2 (SHA-256)	Dilithium-3 (AES-192)	Dilithium-5 (AES-256)
η	2	4	2
(k, ℓ)	$(4, 4)$	$(6, 5)$	$(8, 7)$
γ_1	2^{17}	2^{19}	2^{19}
γ_2	$\frac{q-1}{88}$	$\frac{q-1}{32}$	$\frac{q-1}{32}$
$\beta = \eta \times \tau$	$2 \times 39 = 78$	$4 \times 49 = 196$	$2 \times 60 = 120$

- Schéma de signature recommandé :
  - Modularité des niveaux de sécurité : modifier k et ℓ
  - Taille de $\mathtt{pk}$ + taille de $sign$ réduite : stockage et partage allégés
  - Temps constant d'exécution, version "hedged" : sécurité contre les SCA et FA

KeyGen:

1 $\mathbf{A} \xleftarrow{} \mathcal{R}_q^{k \times \ell}$

2 $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{} S_\eta^\ell \times S_\eta^k$

3 $\mathbf{t} = A\,\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_q^k$

4 $(\mathbf{t}_1, \mathbf{t}_0) = \texttt{Power2Round}(\mathbf{t}, 13)$

5 return $\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, $\mathrm{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathrm{pk})$

KeyGen:

1 $\mathbf{A} \xleftarrow{} \mathcal{R}_q^{k\times\ell} = \begin{pmatrix} (\mathbf{A})_{0,0} & \cdots & (\mathbf{A})_{0,\ell-1} \\ (\mathbf{A})_{1,0} & \cdots & (\mathbf{A})_{1,\ell-1} \\ \vdots & \ddots & \vdots \\ (\mathbf{A})_{k-1,0} & \cdots & (\mathbf{A})_{k-1,\ell-1} \end{pmatrix}$

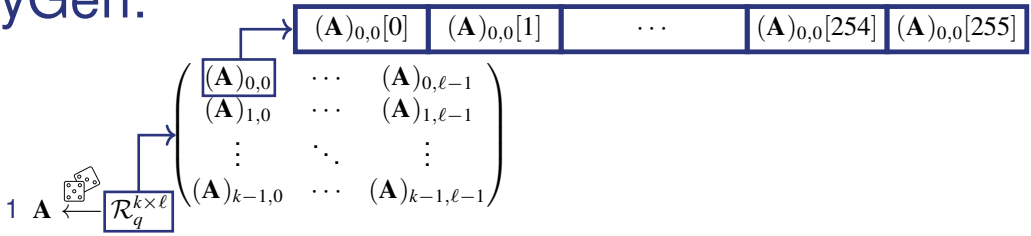
2 $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{} S_\eta^\ell \times S_\eta^k$

3 $\mathbf{t} = A\,\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_q^k$

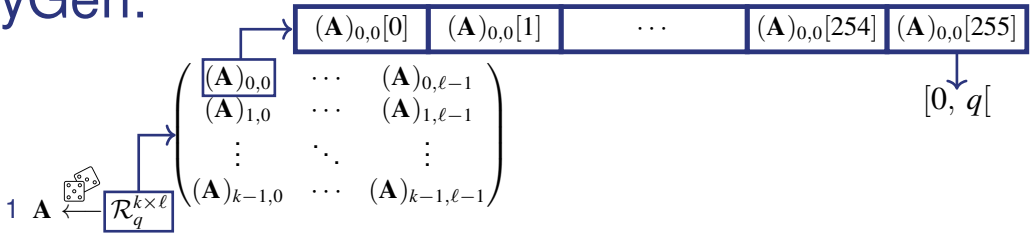
4 $(\mathbf{t}_1, \mathbf{t}_0) = \texttt{Power2Round}(\mathbf{t}, 13)$

5 return $\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, $\mathrm{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathrm{pk})$

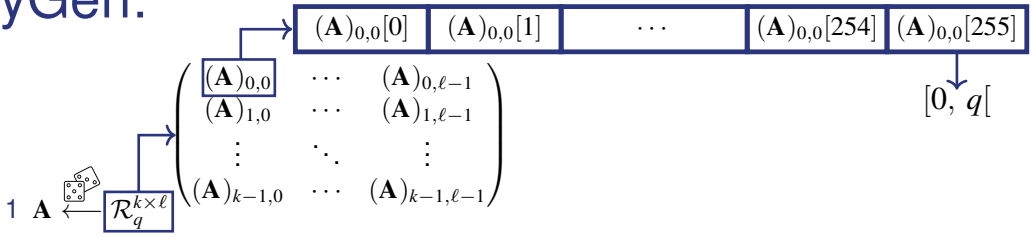
KeyGen:


$(\mathbf{A})_{0,0}[0]$ $(\mathbf{A})_{0,0}[1]$ $\cdots$ $(\mathbf{A})_{0,0}[254]$ $(\mathbf{A})_{0,0}[255]$
$(\mathbf{A})_{0,0}$ $\cdots$ $(\mathbf{A})_{0,\ell-1}$
$(\mathbf{A})_{1,0}$ $\cdots$ $(\mathbf{A})_{1,\ell-1}$
$(\mathbf{A})_{k-1,0}$ $\cdots$ $(\mathbf{A})_{k-1,\ell-1}$
1 $\mathbf{A} \xleftarrow{} \mathcal{R}_q^{k\times\ell}$
2 $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{} S_\eta^\ell \times S_\eta^k$
3 $\mathbf{t} = A\,\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_q^k$
4 $(\mathbf{t}_1, \mathbf{t}_0) = \texttt{Power2Round}(\mathbf{t}, 13)$
5 return $\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, $\mathrm{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathrm{pk})$

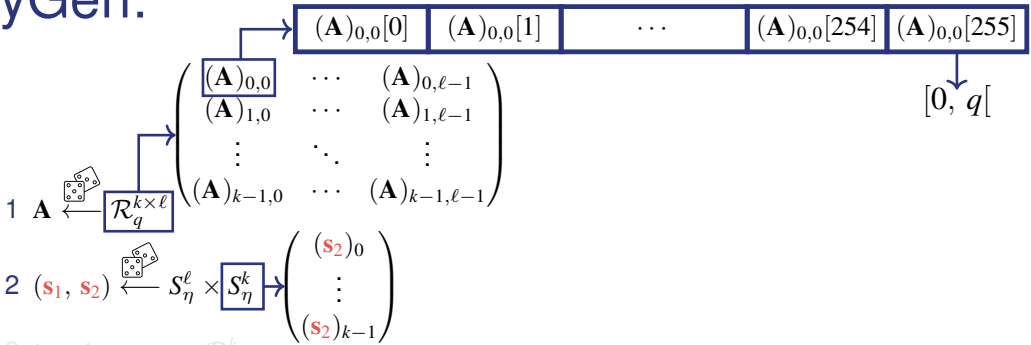
KeyGen:


$(\mathbf{A})_{0,0}[0]$
$(\mathbf{A})_{0,0}[1]$
$\cdots$
$(\mathbf{A})_{0,0}[254]$
$(\mathbf{A})_{0,0}[255]$
$[0, q[$
$(\mathbf{A})_{0,0}$ $\cdots$ $(\mathbf{A})_{0,\ell-1}$
$(\mathbf{A})_{1,0}$ $\cdots$ $(\mathbf{A})_{1,\ell-1}$
$\vdots$ $\ddots$ $\vdots$
$(\mathbf{A})_{k-1,0}$ $\cdots$ $(\mathbf{A})_{k-1,\ell-1}$
1 $\mathbf{A} \xleftarrow{} \mathcal{R}_q^{k\times\ell}$
2 $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{} S_\eta^\ell \times S_\eta^k$
3 $\mathbf{t} = A\,\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_q^k$
4 $(\mathbf{t}_1, \mathbf{t}_0) = \texttt{Power2Round}(\mathbf{t}, 13)$
5 return $\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, $\mathrm{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathrm{pk})$

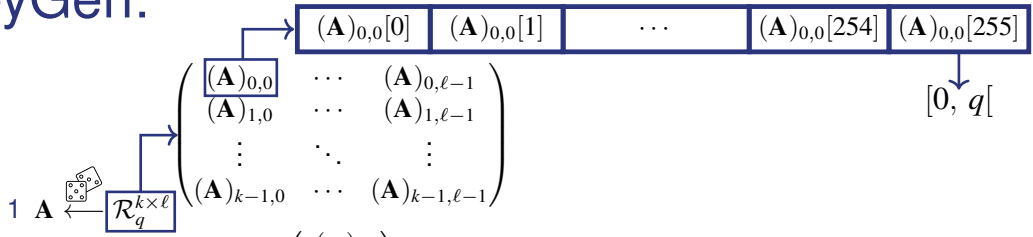
KeyGen:


$(\mathbf{A})_{0,0}[0]$
$(\mathbf{A})_{0,0}[1]$
$\cdots$
$(\mathbf{A})_{0,0}[254]$
$(\mathbf{A})_{0,0}[255]$
$[0, q[$
$(\mathbf{A})_{0,0}$ $\cdots$ $(\mathbf{A})_{0,\ell-1}$
$(\mathbf{A})_{1,0}$ $\cdots$ $(\mathbf{A})_{1,\ell-1}$
$(\mathbf{A})_{k-1,0}$ $\cdots$ $(\mathbf{A})_{k-1,\ell-1}$
1 $\mathbf{A} \xleftarrow{} \mathcal{R}_q^{k\times\ell}$
2 $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{} S_\eta^\ell \times S_\eta^k$
3 $\mathbf{t} = A\,\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_q^k$
4 $(\mathbf{t}_1, \mathbf{t}_0) = \texttt{Power2Round}(\mathbf{t}, 13)$
5 return $\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, $\mathrm{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathrm{pk})$

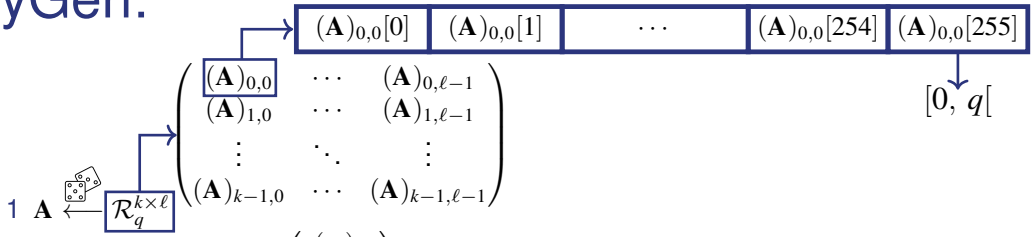
KeyGen:


$(\mathbf{A})_{0,0}[0]$
$(\mathbf{A})_{0,0}[1]$
$\cdots$
$(\mathbf{A})_{0,0}[254]$
$(\mathbf{A})_{0,0}[255]$
$[0, q[$
$(\mathbf{A})_{0,0}$ $\cdots$ $(\mathbf{A})_{0,\ell-1}$
$(\mathbf{A})_{1,0}$ $\cdots$ $(\mathbf{A})_{1,\ell-1}$
$(\mathbf{A})_{k-1,0}$ $\cdots$ $(\mathbf{A})_{k-1,\ell-1}$
1 $\mathbf{A} \xleftarrow{} \mathcal{R}_q^{k\times\ell}$
2 $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{} S_\eta^\ell \times S_\eta^k \to$
$(\mathbf{s}_2)_0$
$(\mathbf{s}_2)_{k-1}$
3 $\mathbf{t} = A\,\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_q^k$
4 $(\mathbf{t}_1, \mathbf{t}_0) = \texttt{Power2Round}(\mathbf{t}, 13)$
5 return $\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, $\mathrm{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathrm{pk})$

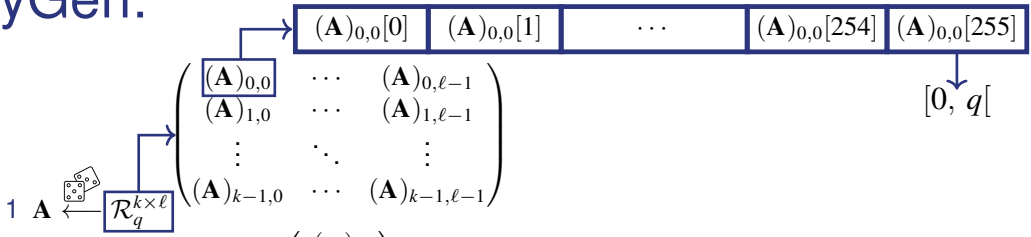
KeyGen:


$(\mathbf{A})_{0,0}[0]$
$(\mathbf{A})_{0,0}[1]$
$\cdots$
$(\mathbf{A})_{0,0}[254]$
$(\mathbf{A})_{0,0}[255]$
$[0, q[$
$(\mathbf{A})_{0,0}$ $\cdots$ $(\mathbf{A})_{0,\ell-1}$
$(\mathbf{A})_{1,0}$ $\cdots$ $(\mathbf{A})_{1,\ell-1}$
$(\mathbf{A})_{k-1,0}$ $\cdots$ $(\mathbf{A})_{k-1,\ell-1}$
1 $\mathbf{A} \xleftarrow{} \mathcal{R}_q^{k\times\ell}$
2 $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{} S_\eta^\ell \times S_\eta^k$
$(\mathbf{s}_2)_0$
$(\mathbf{s}_2)_{k-1}$
3 $\mathbf{t} = A\,\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_q^k$
4 $(\mathbf{t}_1, \mathbf{t}_0) = \texttt{Power2Round}(\mathbf{t}, 13)$
5 return $\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, $\mathrm{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathrm{pk})$

KeyGen:


$(\mathbf{A})_{0,0}[0]$
$(\mathbf{A})_{0,0}[1]$
$\cdots$
$(\mathbf{A})_{0,0}[254]$
$(\mathbf{A})_{0,0}[255]$
$[0, q[$
$(\mathbf{A})_{0,0}$ $\cdots$ $(\mathbf{A})_{0,\ell-1}$
$(\mathbf{A})_{1,0}$ $\cdots$ $(\mathbf{A})_{1,\ell-1}$
$(\mathbf{A})_{k-1,0}$ $\cdots$ $(\mathbf{A})_{k-1,\ell-1}$
1 $\mathbf{A} \xleftarrow{} \mathcal{R}_q^{k\times\ell}$
2 $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{} S_\eta^\ell \times S_\eta^k$
$(\mathbf{s}_2)_0$
$(\mathbf{s}_2)_{k-1}$
3 $\mathbf{t} = A\,\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_q^k$
4 $(\mathbf{t}_1, \mathbf{t}_0) = \texttt{Power2Round}(\mathbf{t}, 13) \longrightarrow \mathbf{t} = \mathbf{t}_1\, 2^{13} + \mathbf{t}_0$
5 return $\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, $\mathrm{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathrm{pk})$

KeyGen:


$(\mathbf{A})_{0,0}[0]$ $(\mathbf{A})_{0,0}[1]$ $\cdots$ $(\mathbf{A})_{0,0}[254]$ $(\mathbf{A})_{0,0}[255]$
$[0, q[$
$(\mathbf{A})_{0,0}$ $\cdots$ $(\mathbf{A})_{0,\ell-1}$
$(\mathbf{A})_{1,0}$ $\cdots$ $(\mathbf{A})_{1,\ell-1}$
$(\mathbf{A})_{k-1,0}$ $\cdots$ $(\mathbf{A})_{k-1,\ell-1}$
1 $\mathbf{A} \xleftarrow{} \mathcal{R}_q^{k\times\ell}$
2 $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{} S_\eta^\ell \times S_\eta^k \to$
$(\mathbf{s}_2)_0$
$(\mathbf{s}_2)_{k-1}$
3 $\mathbf{t} = A\,\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_q^k$
4 $(\mathbf{t}_1, \mathbf{t}_0) = \texttt{Power2Round}(\mathbf{t}, 13) \longrightarrow \mathbf{t} = \mathbf{t}_1\, 2^{13} + \mathbf{t}_0$
5 return $\texttt{pk} = (\mathbf{A}, \mathbf{t}_1)$, $\texttt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \texttt{pk})$

Dilithium Sign

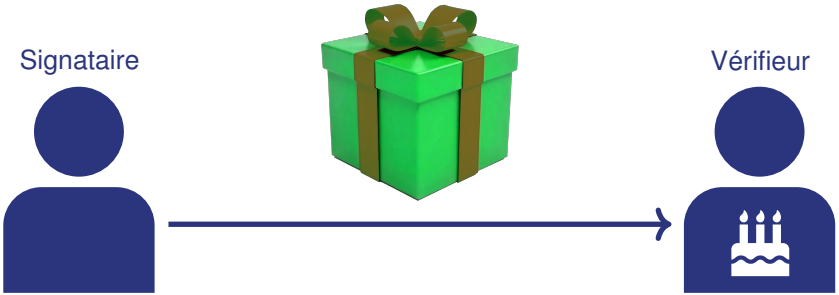
Signataire

Vérifieur

Offrir un cadeau en respectant certaines conditions :

- de validité
  - Aime les vélos
  - Aime le rouge
- de sécurité
  - Garder secrète la surprise

Dilithium Sign



Offrir un cadeau en respectant certaines conditions :

- de validité
  - Aime les vélos ✓
  - Aime le rouge ✗
- de sécurité
  - Garder secrète la surprise ✓

Dilithium Sign

Offrir un cadeau en respectant certaines conditions :

- de validité
  - Aime les vélos ✓
  - Aime le rouge ✓
- de sécurité
  - Garder secrète la surprise ✗

source : `https://blog.halfords.com/how-to-wrap-a-kids-bike-for-christmas/`

Dilithium Sign

Offrir un cadeau en respectant certaines conditions :

- de validité
  - Aime les vélos✓
  - Aime le rouge✓
- de sécurité
  - Garder secrète la surprise✓

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^{\ell}_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = \mathbf{y} + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

$(\mathbf{z}, \mathbf{h}) = \perp$

while $(\mathbf{z}, \mathbf{h}) = \perp$ do

$\mathbf{y} \xleftarrow{} \tilde{S}^{\ell}_{\gamma_1}$

$\mathbf{w} = \mathbf{A}\,\mathbf{y}$

$\mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w}) \longrightarrow \mathbf{w} = \mathbf{w}_1\, 2\gamma_2 + \mathbf{w}_0$

$c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$

$\mathbf{z} = \mathbf{y} + c\,\mathbf{s}_1$

$\mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$

if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$

else,

$\mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$

if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$

return $\sigma = (c, \mathbf{z}, \mathbf{h})$

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^{\ell}_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w}) \longrightarrow \mathbf{w} = \mathbf{w}_1\, 2\gamma_2 + \mathbf{w}_0$
6 $\quad c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = \mathbf{y} + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$

2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do

3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^{\ell}_{\gamma_1}$

4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$

5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w}) \longrightarrow \mathbf{w} = \mathbf{w}_1\, 2\gamma_2 + \mathbf{w}_0$

6 $\quad c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$

7 $\quad \mathbf{z} = \mathbf{y} + c\,\mathbf{s}_1$

8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$

9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$

10 $\quad$ else,

11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$

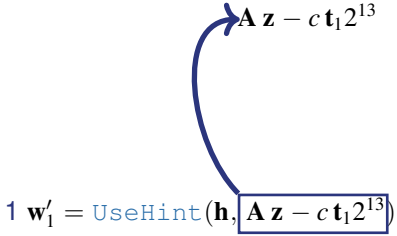
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$

13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

Verify($\mathtt{pk} = (\mathbf{A}, \mathbf{t}_1)$, msg, $\sigma = (c, \mathbf{z}, \mathbf{h})$):

1 $\mathbf{w}'_1 = \mathtt{UseHint}(\mathbf{h}, \mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13})$

Verify($\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, msg, $\sigma = (c, \mathbf{z}, \mathbf{h})$):



$\texttt{Verify}(\texttt{pk} = (\mathbf{A}, \mathbf{t}_1),\ msg,\ \sigma = (c, \mathbf{z}, \mathbf{h}))$:

$$\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13} = \mathbf{A}\,\overbrace{(\mathbf{y} + c\,\mathbf{s}_1)}^{\mathbf{z}} - c\,\overbrace{(\mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2 - \mathbf{t}_0)}^{\mathbf{t}_1 2^{13}}$$

1 $\mathbf{w}_1' = \texttt{UseHint}(\mathbf{h}, \boxed{\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13}})$

$\texttt{Verify}(\texttt{pk} = (\mathbf{A}, \mathbf{t}_1),\ msg,\ \sigma = (c, \mathbf{z}, \mathbf{h}))$:

$$\begin{aligned}
\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13} &= \mathbf{A}\,\overbrace{(\mathbf{y} + c\,\mathbf{s}_1)}^{\mathbf{z}} - c\,\overbrace{(\mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2 - \mathbf{t}_0)}^{\mathbf{t}_1 2^{13}} \\
&= \underbrace{\mathbf{A}\,\mathbf{y}}_{\mathbf{w}} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0 \\
&= \mathbf{w} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0
\end{aligned}$$

1 $\mathbf{w}'_1 = \texttt{UseHint}(\mathbf{h}, \boxed{\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13}})$

$\mathtt{Verify}(\mathtt{pk} = (\mathbf{A}, \mathbf{t}_1),\ msg,\ \sigma = (c, \mathbf{z}, \mathbf{h}))$:

$$\begin{aligned}\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13} &= \mathbf{A}\,\overbrace{(\mathbf{y} + c\,\mathbf{s}_1)}^{\mathbf{z}} - c\,\overbrace{(\mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2 - \mathbf{t}_0)}^{\mathbf{t}_1 2^{13}} \\ &= \underbrace{\mathbf{A}\,\mathbf{y}}_{\mathbf{w}} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0 \\ &= \mathbf{w} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0\end{aligned}$$

Lemme 1.1 [1] $\implies \mathtt{UseHint}(\mathbf{h},\ \mathbf{w} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0) = \mathtt{HighBits}(\mathbf{w} - \overbrace{c\,\mathbf{s}_2}^{\leq \beta})$

1 $\mathbf{w}'_1 = \mathtt{UseHint}(\mathbf{h}, \boxed{\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13}})$

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

Verify($\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, msg, $\sigma = (c, \mathbf{z}, \mathbf{h})$):

$$\begin{aligned}
\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13} &= \mathbf{A}\,\overbrace{(\mathbf{y} + c\,\mathbf{s}_1)}^{\mathbf{z}} - c\,\overbrace{(\mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2 - \mathbf{t}_0)}^{\mathbf{t}_1 2^{13}} \\
&= \underbrace{\mathbf{A}\,\mathbf{y}}_{\mathbf{w}} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0 \\
&= \mathbf{w} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0
\end{aligned}$$

Lemme 1.1 [1] $\implies$ $\texttt{UseHint}(\mathbf{h}, \mathbf{w} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0) = \texttt{HighBits}(\mathbf{w} - \overbrace{c\,\mathbf{s}_2}^{\leq \beta})$

Lemme 2 [1] $\implies$ $\texttt{HighBits}(\mathbf{w} - c\,\mathbf{s}_2) = \underbrace{\texttt{HighBits}(\mathbf{w})}_{\mathbf{w}_1}$

1 $\mathbf{w}_1' = \texttt{UseHint}(\mathbf{h}, \boxed{\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13}})$

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

Verify($\mathrm{pk} = (\mathbf{A}, \mathbf{t}_1)$, msg, $\sigma = (c, \mathbf{z}, \mathbf{h})$):

$$\begin{aligned}
\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13} &= \mathbf{A}\,\overbrace{(\mathbf{y} + c\,\mathbf{s}_1)}^{\mathbf{z}} - c\,\overbrace{(\mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2 - \mathbf{t}_0)}^{\mathbf{t}_1 2^{13}} \\
&= \underbrace{\mathbf{A}\,\mathbf{y}}_{\mathbf{w}} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0 \\
&= \mathbf{w} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0
\end{aligned}$$

Lemme 1.1 [1] $\implies$ $\mathtt{UseHint}(\mathbf{h}, \mathbf{w} - c\,\mathbf{s}_2 + c\,\mathbf{t}_0) = \mathtt{HighBits}(\mathbf{w} - \overbrace{c\,\mathbf{s}_2}^{\leq \beta})$

Lemme 2 [1] $\implies$ $\mathtt{HighBits}(\mathbf{w} - c\,\mathbf{s}_2) = \underbrace{\mathtt{HighBits}(\mathbf{w})}_{\mathbf{w}_1}$

1 $\mathbf{w}'_1 = \mathtt{UseHint}(\mathbf{h}, \boxed{\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13}})$
2 if $||\mathbf{z}||_\infty < \gamma_1 - \beta$ and $c = \mathtt{H}(\mathrm{pk}\,||\,msg\,||\,\mathbf{w}'_1)$ and $\|\mathbf{h}\|_1 \leq \omega$
3 return *True*
4 else
5 return *False*

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices


Comprendre Dilithium
Implémentation Sécurisée
Optimiser l'algorithme
Étudier les Attaques

- Dilithium Sign : Attaque par fautes sur la vérification de $\|\mathbf{r}_0\|_\infty$
- Dilithium Sign : Attaque par canaux auxilaires sur $\mathbf{w}_0$

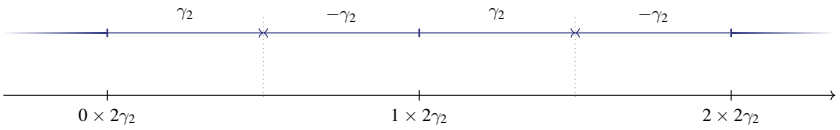
- Dilithium Sign : Attaque par fautes sur la vérification de $\|\mathbf{r}_0\|_\infty$
- Dilithium Sign : Attaque par canaux auxilaires sur $\mathbf{w}_0$

Decompose

- $\overbrace{\texttt{HighBits}(w)}^{w_1}, \overbrace{\texttt{LowBits}(w)}^{w_0} = \texttt{Decompose}(w)$ et tel que $w = w_1 2\gamma_2 + w_0$
- Division euclidienne avec le reste centré en 0 donc $-\gamma_2 < w_0 \leq \gamma_2$

$\texttt{Decompose}(w)$

1 $w := w \mod q$
2 $w_0 := w \mod^{\pm} 2\gamma_2$
3 if $w - w_0 = q - 1$
4 $\quad w_1 := 0$
5 $\quad w_0 := w_0 - 1$
6 else
7 $\quad w_1 := \dfrac{w - w_0}{2\gamma_2}$
8 return (w_1, w_0)

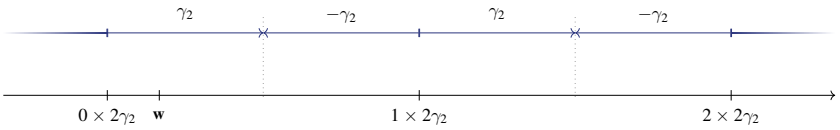


Decompose

- $\overbrace{\texttt{HighBits}(w)}^{w_1}$, $\overbrace{\texttt{LowBits}(w)}^{w_0}$ = $\texttt{Decompose}(w)$ et tel que $w = w_1 2\gamma_2 + w_0$
- Division euclidienne avec le reste centré en 0 donc $-\gamma_2 < w_0 \leq \gamma_2$

$\texttt{Decompose}(w)$

1 $w := w \mod q$
2 $w_0 := w \mod^{\pm} 2\gamma_2$
3 if $w - w_0 = q - 1$
4 $\quad w_1 := 0$
5 $\quad w_0 := w_0 - 1$
6 else
7 $\quad w_1 := \dfrac{w - w_0}{2\gamma_2}$
8 return (w_1, w_0)

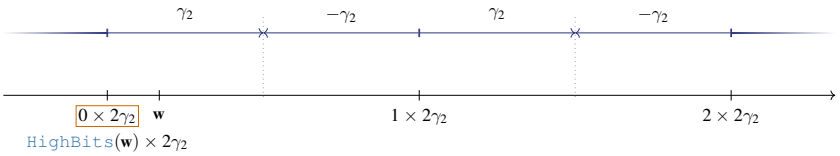


Decompose

- $\overbrace{\texttt{HighBits}(w)}^{w_1}, \overbrace{\texttt{LowBits}(w)}^{w_0} = \texttt{Decompose}(w)$ et tel que $w = w_1 2\gamma_2 + w_0$
- Division euclidienne avec le reste centré en 0 donc $-\gamma_2 < w_0 \leq \gamma_2$

$\texttt{Decompose}(w)$

1 $w := w \mod q$
2 $w_0 := w \mod^{\pm} 2\gamma_2$
3 if $w - w_0 = q - 1$
4 $\quad w_1 := 0$
5 $\quad w_0 := w_0 - 1$
6 else
7 $\quad w_1 := \dfrac{w - w_0}{2\gamma_2}$
8 return (w_1, w_0)

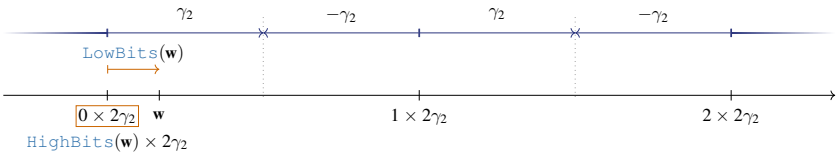


Decompose

- $\overbrace{\texttt{HighBits}(w)}^{w_1}$, $\overbrace{\texttt{LowBits}(w)}^{w_0}$ = $\texttt{Decompose}(w)$ et tel que $w = w_1 2\gamma_2 + w_0$
- Division euclidienne avec le reste centré en 0 donc $-\gamma_2 < w_0 \leq \gamma_2$

$\texttt{Decompose}(w)$

1 $w := w \mod q$
2 $w_0 := w \mod^{\pm} 2\gamma_2$
3 if $w - w_0 = q - 1$
4 $\quad w_1 := 0$
5 $\quad w_0 := w_0 - 1$
6 else
7 $\quad w_1 := \dfrac{w - w_0}{2\gamma_2}$
8 return (w_1, w_0)

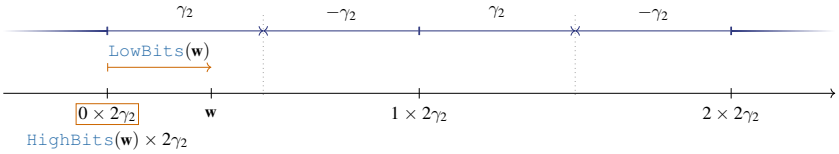


Decompose

- $\overbrace{\texttt{HighBits}(w)}^{w_1}, \overbrace{\texttt{LowBits}(w)}^{w_0} = \texttt{Decompose}(w)$ et tel que $w = w_1 2\gamma_2 + w_0$
- Division euclidienne avec le reste centré en 0 donc $-\gamma_2 < w_0 \leq \gamma_2$

$\texttt{Decompose}(w)$

1 $w := w \mod q$
2 $w_0 := w \mod^{\pm} 2\gamma_2$
3 if $w - w_0 = q - 1$
4 $\quad w_1 := 0$
5 $\quad w_0 := w_0 - 1$
6 else
7 $\quad w_1 := \dfrac{w - w_0}{2\gamma_2}$
8 return (w_1, w_0)

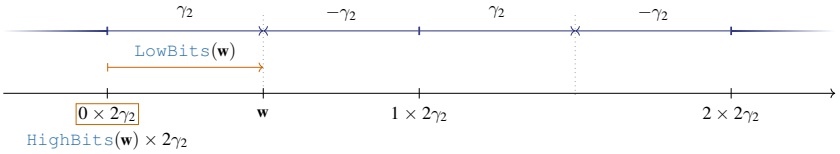


Decompose

- $\overbrace{\texttt{HighBits}(w)}^{w_1}$, $\overbrace{\texttt{LowBits}(w)}^{w_0}$ = $\texttt{Decompose}(w)$ et tel que $w = w_1 2\gamma_2 + w_0$
- Division euclidienne avec le reste centré en 0 donc $-\gamma_2 < w_0 \leq \gamma_2$

$\texttt{Decompose}(w)$

1 $w := w \mod q$
2 $w_0 := w \mod^{\pm} 2\gamma_2$
3 if $w - w_0 = q - 1$
4 $\quad w_1 := 0$
5 $\quad w_0 := w_0 - 1$
6 else
7 $\quad w_1 := \dfrac{w - w_0}{2\gamma_2}$
8 return (w_1, w_0)

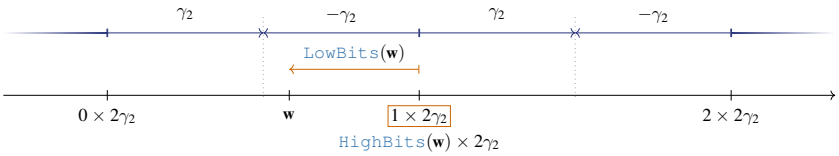


Decompose

- $\overbrace{\texttt{HighBits}(w)}^{w_1}, \overbrace{\texttt{LowBits}(w)}^{w_0} = \texttt{Decompose}(w)$ et tel que $w = w_1 2\gamma_2 + w_0$
- Division euclidienne avec le reste centré en 0 donc $-\gamma_2 < w_0 \leq \gamma_2$

$\texttt{Decompose}(w)$

1 $w := w \mod q$
2 $w_0 := w \mod^{\pm} 2\gamma_2$
3 if $w - w_0 = q - 1$
4 $\quad w_1 := 0$
5 $\quad w_0 := w_0 - 1$
6 else
7 $\quad w_1 := \dfrac{w - w_0}{2\gamma_2}$
8 return (w_1, w_0)

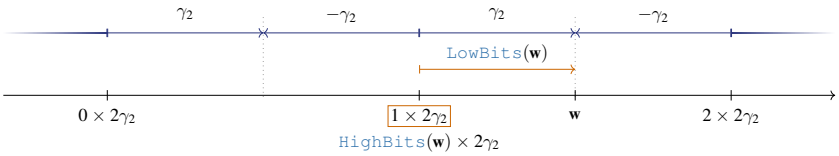


Decompose

- $\overbrace{\texttt{HighBits}(w)}^{w_1}, \overbrace{\texttt{LowBits}(w)}^{w_0} = \texttt{Decompose}(w)$ et tel que $w = w_1 2\gamma_2 + w_0$
- Division euclidienne avec le reste centré en 0 donc $-\gamma_2 < w_0 \leq \gamma_2$

$\texttt{Decompose}(w)$

1 $w := w \mod q$
2 $w_0 := w \mod^{\pm} 2\gamma_2$
3 if $w - w_0 = q - 1$
4 $\quad w_1 := 0$
5 $\quad w_0 := w_0 - 1$
6 else
7 $\quad w_1 := \dfrac{w - w_0}{2\gamma_2}$
8 return (w_1, w_0)



Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- Nécessaire pour la validité selon [1]

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \mathrm{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- Nécessaire pour la validité selon [1]
- On sait que $\mathbf{w}_0 - c\,\mathbf{s}_2 = \mathtt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)$
- donc on a $\overbrace{\mathtt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)}^{<\gamma_2-\beta} + \overbrace{c\,\mathbf{s}_2}^{\leq\beta} < \gamma_2$
- et alors $\mathtt{HighBits}(\mathbf{A}\,\mathbf{y} - \overbrace{c\,\mathbf{s}_2}^{\leq\beta}) = \mathtt{HighBits}(\mathbf{A}\,\mathbf{y})$

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \mathrm{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- Nécessaire pour la validité selon [1]
- On sait que $\mathbf{w}_0 - c\,\mathbf{s}_2 = \mathtt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)$
- donc on a $\overbrace{\mathtt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)}^{<\gamma_2-\beta} + \overbrace{c\,\mathbf{s}_2}^{\leq\beta} < \gamma_2$
- et alors $\mathtt{HighBits}(\mathbf{A}\,\mathbf{y} - \overbrace{c\,\mathbf{s}_2}^{\leq\beta}) = \mathtt{HighBits}(\mathbf{A}\,\mathbf{y})$
- On ne peut plus garantir la vérification
- Pour l'implémentation de Dilithium-2, sur $1\,000\,000\,000$ de signatures, toutes sont vérifiées

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

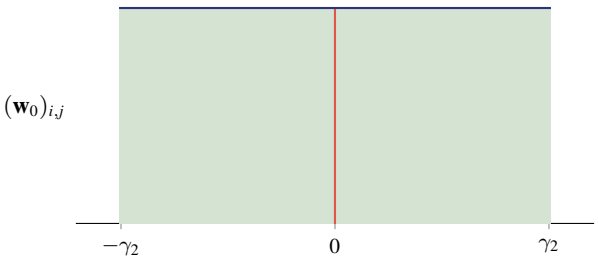
- Nécessaire pour la sécurité selon [1]

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

$(\mathbf{z}, \mathbf{h}) = \perp$
while $(\mathbf{z}, \mathbf{h}) = \perp$ do
  $\mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
  $\mathbf{w} = \mathbf{A}\,\mathbf{y}$
  $\mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
  $c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
  $\mathbf{z} = y + c\,\mathbf{s}_1$
  $\mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
  if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
  else,
    $\mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
    if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- Nécessaire pour la sécurité selon [1]

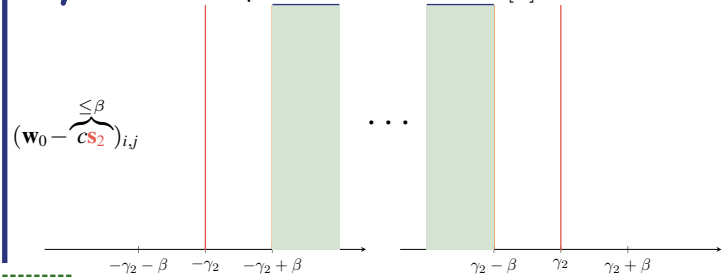


[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- Nécessaire pour la sécurité selon [1]

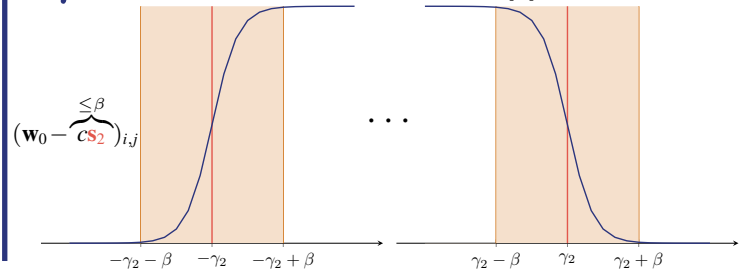


[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

Sign(msg, sk= ($\mathbf{A}$, $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{t}_0$, pk)):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \texttt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \texttt{H}(\texttt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \texttt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- Nécessaire pour la sécurité selon [1]



- **Complexité d'exploitation de ces signatures ?**

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Dilithium: Digital Signatures from Module Lattices

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

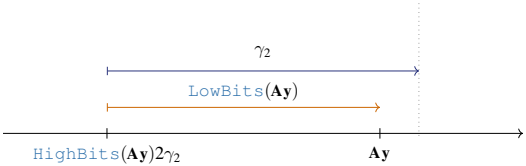
1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ qui aurait dû être rejetée

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

$(\mathbf{z}, \mathbf{h}) = \perp$
while $(\mathbf{z}, \mathbf{h}) = \perp$ do
  $\mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
  $\mathbf{w} = \mathbf{A}\,\mathbf{y}$
  $\mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
  $c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
  $\mathbf{z} = y + c\,\mathbf{s}_1$
  $\mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
  if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
  else,
    $\mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
    if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
return $\sigma = (c, \mathbf{z}, \mathbf{h})$

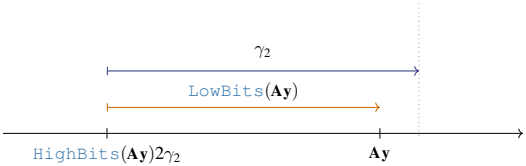
- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ qui aurait dû être rejetée



Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

$(\mathbf{z}, \mathbf{h}) = \perp$
while $(\mathbf{z}, \mathbf{h}) = \perp$ do
  $\mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
  $\mathbf{w} = \mathbf{A}\,\mathbf{y}$
  $\mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
  $c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
  $\mathbf{z} = y + c\,\mathbf{s}_1$
  $\mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
  if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
  else,
    $\mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
    if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ qui aurait dû être rejetée

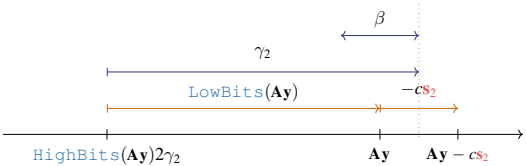


- Supposons que $-\beta \leq c\,\mathbf{s}_2 < 0$

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

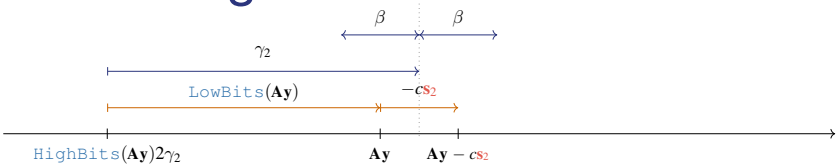
$(\mathbf{z}, \mathbf{h}) = \perp$
while $(\mathbf{z}, \mathbf{h}) = \perp$ do
  $\mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
  $\mathbf{w} = \mathbf{A}\,\mathbf{y}$
  $\mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
  $c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
  $\mathbf{z} = y + c\,\mathbf{s}_1$
  $\mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
  if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
  else,
    $\mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
    if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ qui aurait dû être rejetée

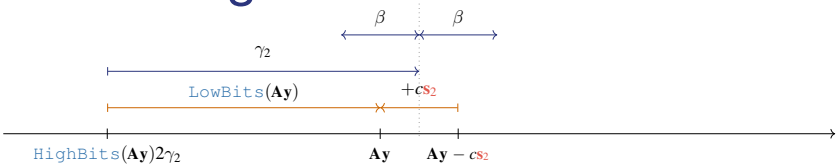


- Supposons que $-\beta \leq c\,\mathbf{s}_2 < 0$

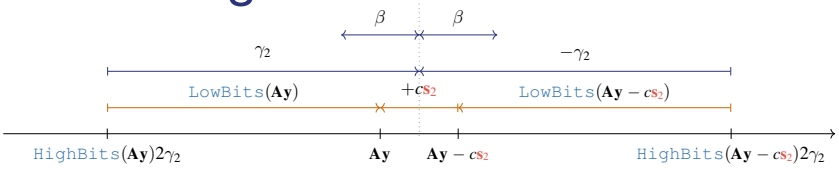
Former des inégalités



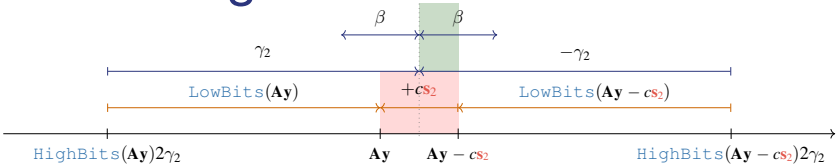
Former des inégalités



Former des inégalités



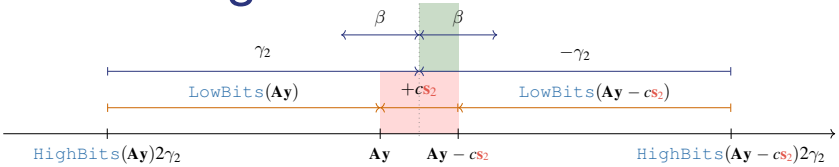
Former des inégalités



Proposition

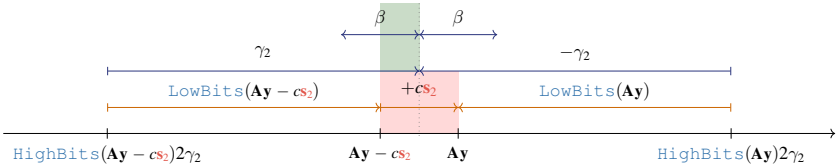
Si $\mathtt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j} < \mathtt{HighBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \leq -\gamma_2 - \mathtt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$

Former des inégalités



Proposition

Si $\mathtt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j} < \mathtt{HighBits}(\mathbf{A}\,\mathbf{y}-c\,\mathbf{s}_2)_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \leq -\gamma_2 - \mathtt{LowBits}(\mathbf{A}\,\mathbf{y}-c\,\mathbf{s}_2)_{i,j}$



Proposition

Si $\mathtt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j} > \mathtt{HighBits}(\mathbf{A}\,\mathbf{y}-c\,\mathbf{s}_2)_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \geq \gamma_2 - \mathtt{LowBits}(\mathbf{A}\,\mathbf{y}-c\,\mathbf{s}_2)_{i,j}$

Inégalités

Proposition

Si $\texttt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j} < \texttt{HighBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \leq -\gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$
Si $\texttt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j} > \texttt{HighBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \geq \gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ sans la vérification sur $\|\mathbf{r}_0\|_\infty$ mais vérifiée :

$$(\mathbf{w}_1')_{i,j} = \texttt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j}$$

Inégalités

Proposition

Si $(\mathbf{w}'_1)_{i,j} < \texttt{HighBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \leq -\gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$
Si $(\mathbf{w}'_1)_{i,j} > \texttt{HighBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \geq \gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ sans la vérification sur $\|\mathbf{r}_0\|_\infty$ mais vérifiée :

$$(\mathbf{w}'_1)_{i,j} = \texttt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j}$$

Inégalités

Proposition

Si $(\mathbf{w}'_1)_{i,j} < \texttt{HighBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \leq -\gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$
Si $(\mathbf{w}'_1)_{i,j} > \texttt{HighBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \geq \gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2)_{i,j}$

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ sans la vérification sur $\|\mathbf{r}_0\|_\infty$ mais vérifiée :
$$(\mathbf{w}'_1)_{i,j} = \texttt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j}$$
- On peut retrouver $\mathbf{t}_0$ avec $\approx 200\,000/500\,000$ signatures selon le niveau de sécurité [3] :
$$\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13} - c\,\mathbf{t}_0 = \mathbf{A}\,\mathbf{z} - c\,\mathbf{t} = \mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2$$

[3] P. Azevedo-Oliveira, A. Calle Viera, B. Cogliati, L. Goubin, Uncompressing Dilithium's public key

Inégalités

Proposition

Si $(\mathbf{w}'_1)_{i,j} < \texttt{HighBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \leq -\gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$
Si $(\mathbf{w}'_1)_{i,j} > \texttt{HighBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \geq \gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ sans la vérification sur $\|\mathbf{r}_0\|_\infty$ mais vérifiée :
$$(\mathbf{w}'_1)_{i,j} = \texttt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j}$$
- On peut retrouver $\mathbf{t}_0$ avec $\approx 200\,000/500\,000$ signatures selon le niveau de sécurité [3] :
$$\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13} - c\,\mathbf{t}_0 = \mathbf{A}\,\mathbf{z} - c\,\mathbf{t} = \mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2$$

[3] P. Azevedo-Oliveira, A. Calle Viera, B. Cogliati, L. Goubin,
Uncompressing Dilithium's public key

Inégalités

Proposition

Si $(\mathbf{w}'_1)_{i,j} < \texttt{HighBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \leq -\gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$
Si $(\mathbf{w}'_1)_{i,j} > \texttt{HighBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \geq \gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ sans la vérification sur $\|\mathbf{r}_0\|_\infty$ mais vérifiée :

$$(\mathbf{w}'_1)_{i,j} = \texttt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j}$$

- On peut retrouver $\mathbf{t}_0$ avec $\approx 200\,000/500\,000$ signatures selon le niveau de sécurité [3] :

$$\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13} - c\,\mathbf{t}_0 = \mathbf{A}\,\mathbf{z} - c\,\mathbf{t} = \mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2$$

$$(c\,\mathbf{s}_2)_{i,j} = \underbrace{\begin{pmatrix}(cX^0)_j & (cX^1)_j & \cdots & (cX^{254} & (cX^{255})_j\end{pmatrix}}_{\mathbf{C}_i^+} \cdot \begin{pmatrix}(\mathbf{s}_2)_{i,0}\\(\mathbf{s}_2)_{i,1}\\\vdots\\(\mathbf{s}_2)_{i,255}\end{pmatrix} \geq \underbrace{\gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}}_{b_i^+}$$

[3] P. Azevedo-Oliveira, A. Calle Viera, B. Cogliati, L. Goubin, Uncompressing Dilithium's public key

Inégalités

Proposition

Si $(\mathbf{w}'_1)_{i,j} < \texttt{HighBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \leq -\gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$
Si $(\mathbf{w}'_1)_{i,j} > \texttt{HighBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$ alors $(c\,\mathbf{s}_2)_{i,j} \geq \gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}$

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$ sans la vérification sur $\|\mathbf{r}_0\|_\infty$ mais vérifiée :

$$(\mathbf{w}'_1)_{i,j} = \texttt{HighBits}(\mathbf{A}\,\mathbf{y})_{i,j}$$

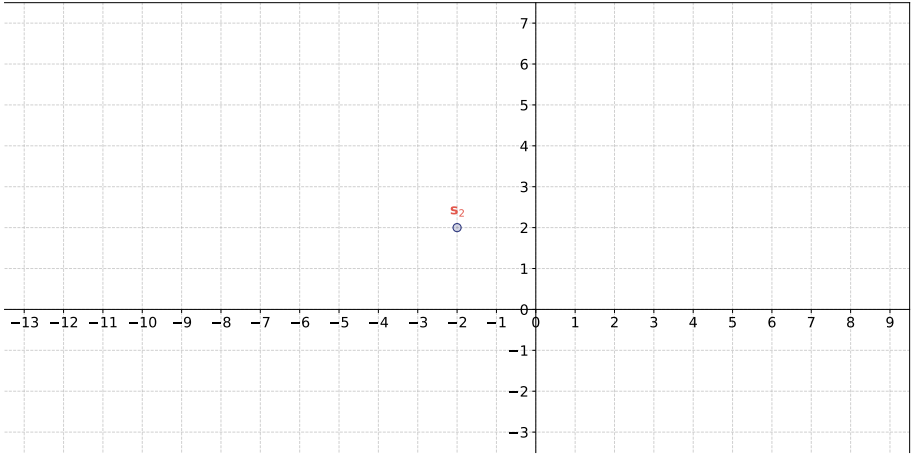
- On peut retrouver $\mathbf{t}_0$ avec $\approx 200\,000/500\,000$ signatures selon le niveau de sécurité [3] :

$$\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1 2^{13} - c\,\mathbf{t}_0 = \mathbf{A}\,\mathbf{z} - c\,\mathbf{t} = \mathbf{A}\,\mathbf{y} - c\,\mathbf{s}_2$$

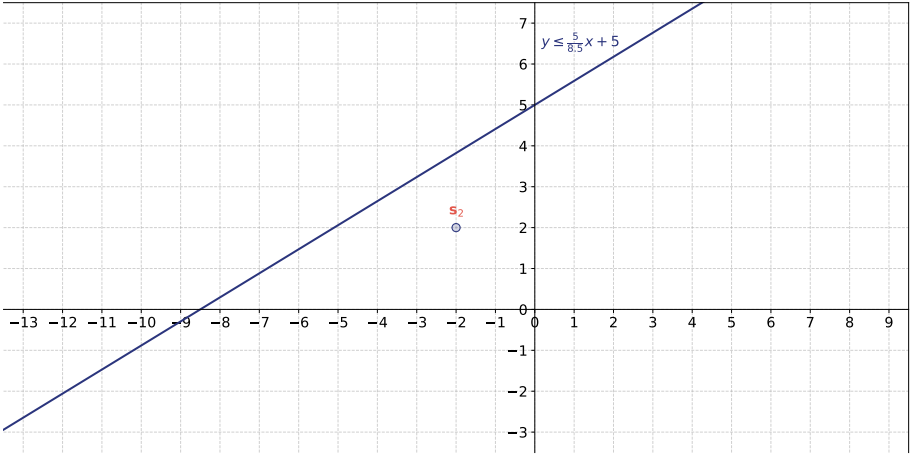
$$(c\,\mathbf{s}_2)_{i,j} = \underbrace{\begin{pmatrix}(cX^0)_j & (cX^1)_j & \cdots & (cX^{254} & (cX^{255})_j\end{pmatrix}}_{\mathbf{C}_i^-} \cdot \begin{pmatrix}(\mathbf{s}_2)_{i,0}\\(\mathbf{s}_2)_{i,1}\\\vdots\\(\mathbf{s}_2)_{i,255}\end{pmatrix} \leq \underbrace{-\gamma_2 - \texttt{LowBits}(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t})_{i,j}}_{b_i^-}$$

[3] P. Azevedo-Oliveira, A. Calle Viera, B. Cogliati, L. Goubin, Uncompressing Dilithium's public key

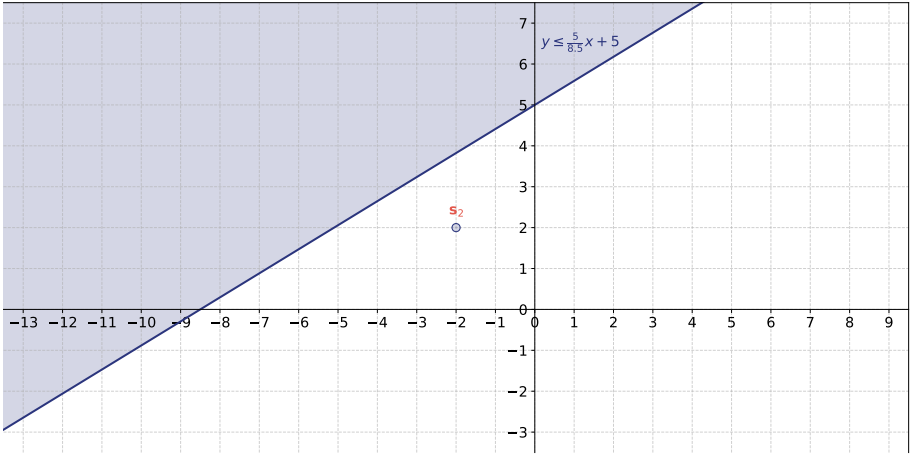
Inégalités



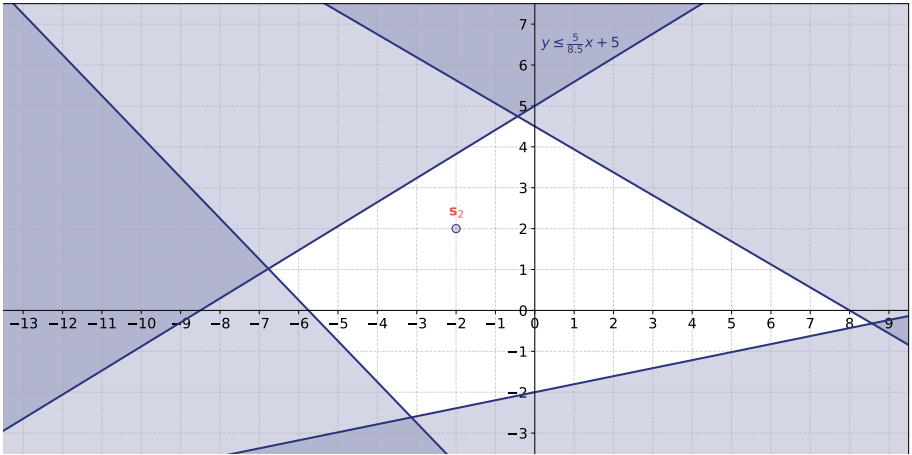
Inégalités



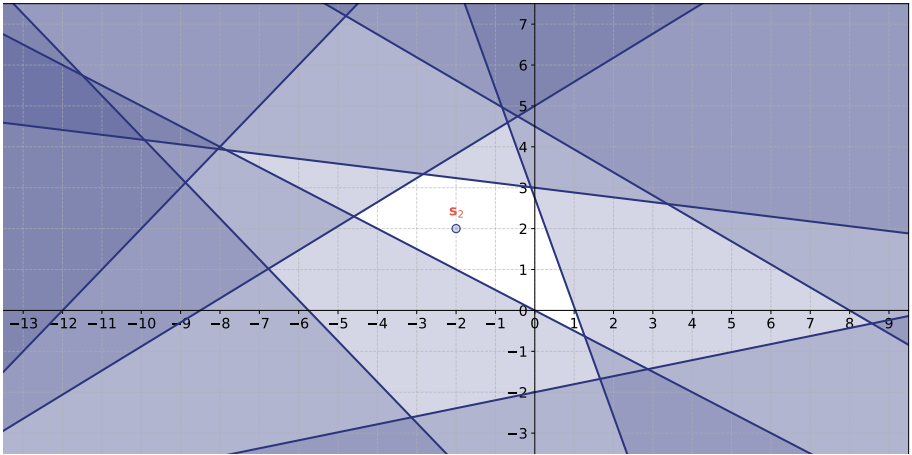
Inégalités



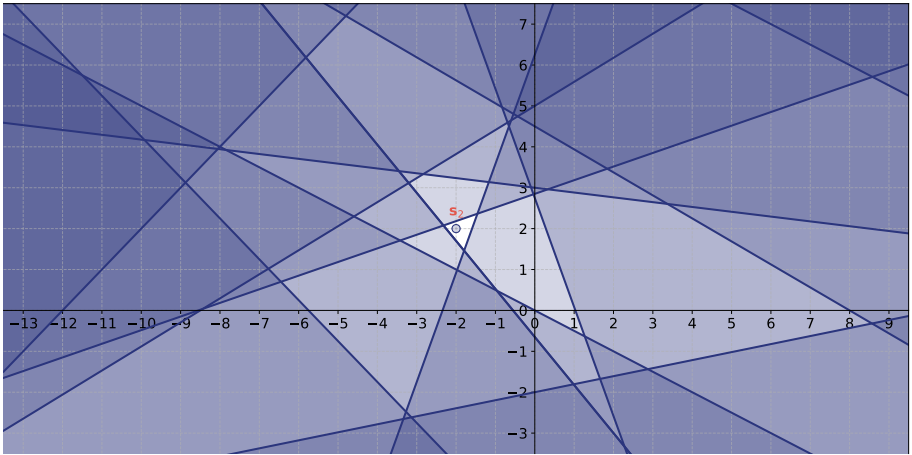
Inégalités



Inégalités



Inégalités



Résolution

- Pour chaque polynôme $i \in [0, k-1]$ on résout le programme linéaire :

$$\begin{array}{ll} \text{maximiser} & 0 \\ \text{sous contraintes} & \mathbf{C}_i^+(\mathbf{s}_2)_i \geq b_i^+ \\ & \mathbf{C}_i^-(\mathbf{s}_2)_i \leq b_i^- \\ & (\mathbf{s}_2)_i \in [-\eta, \eta]^{256} \end{array}$$

- Si on a assez d'inégalités, alors il y a une solution réelle
- Pour chaque solution $i \in [0, k-1]$, on arrondi les 256 coefficients à l'entier le plus proche
- On espère tomber sur la solution entière
- Comme on connait $\mathbf{t}_0$ (par hypothèse) et $\mathbf{s}_2$ (par résolution) alors :

$$\mathbf{s}_1 = ({}^t\mathbf{A}\,\mathbf{A})^{-1}\,{}^t\mathbf{A}\,(\mathbf{t}_1\,2^{13} + (\mathbf{t}_0 - \mathbf{s}_2)) \tag{1}$$

- Juste avec $\mathbf{s}_1$ on peut forger des signatures [3]

[3] L. Groot Bruinderink, P. Pessl,
Differential Fault Attacks on Deterministic Lattice Signatures

Évaluation

- Comme les signatures fautées (sans la condition sur $\|\mathbf{r}_0\|_\infty$) sont toujours vérifiées

Remarque

Aucun moyen de détecter des signatures produites sans la vérification de norme de $\mathbf{r}_0$

- À part faire l'attaque et construire les inégalités ...
- Attaque testée sur les 100 clés des fichiers de référence des KAT
- Résolution avec `lp_solve`
  - open-source
  - python
  - pas optimisé
- Attaque testée avec des fautes simulées
  - indépendant de l'implémentation
  - indépendant de la plateforme

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \in \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

Sign(msg, sk= ($\mathbf{A}$, $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{t}_0$, pk)):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \in \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \texttt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \texttt{H}(\texttt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \texttt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

```
168   polyveck_sub(&r0, &w0, &cs2);
169   polyveck_reduce(&r0);
170   if(polyveck_chknorm(&r0, GAMMA2 - BETA))
171     goto rej;
```

Sign(msg, sk= ($\mathbf{A}$, $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{t}_0$, pk)):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \in \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \texttt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \texttt{H}(\texttt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \texttt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

```
168   polyveck_sub(&r0, &w0, &cs2);
169   polyveck_reduce(&r0);
170   if(polyveck_chknorm(&r0, GAMMA2 - BETA))
171     goto rej;
```

Cortex M4 : -Os

```
1df6 mov     r1, fp  ;fp = GAMMA2 - BETA
1df8 adds    r0, #104
1dfa bl      23b4    ;appel polyveck_chknorm
1dfe cmp     r0, #0  ;polyveck_chknorm sortie
1e00 bne.w   1cc0    ;si pas 0 aller a rej
```

Sign(msg, sk$= (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0,$ pk$)$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \in \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \texttt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \texttt{H}(\texttt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \texttt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

```
168   polyveck_sub(&r0, &w0, &cs2);
169   polyveck_reduce(&r0);
170   if(polyveck_chknorm(&r0, GAMMA2 - BETA))
171     goto rej;
```

Cortex M4 : -Os

```
1df6 mov     r1, fp  ;fp = GAMMA2 - BETA
1df8 adds    r0, #104
1dfa bl      23b4    ;appel polyveck_chknorm
1dfe cmp     r0, #0  ;polyveck_chknorm sortie
1e00 bne.w   1cc0    ;si pas 0 aller a rej
```

Une seule instruction à sauter
- glitch sur l'horloge
- glitch de voltage

Résultats

- Dilithium-2

# signatures	# d'inégalités	Temps de résolution (moyen)	Proba. de succés
1 250 000	11 083	$\approx$ 5min	0.98*

- Dilithium-3

# signatures	# d'inégalités	Temps de résolution (moyen)	Proba. de succés
3 500 000	22 020	$\approx$ 21min	1

- Dilithium-5

# signatures	# d'inégalités	Temps de résolution (moyen)	Proba. de succés
4 000 000	15 348	$\approx$ 4min	1

Remarque

Le nombre de signatures fautées dépend du paramétre γ_2 (Dilithium-3 et 5 pareil)
Le temps de résolution dépend du paramétre η (Dilithium-2 et 5 pareil)

Conclusion

Finding a polytope:
A practical fault attack against Dilithium

Paco Azevedo-Oliveira, Andersson Calle Viera, Benoît Cogliati, Louis Goubin

accepté à **PKC25**

ia.cr/2025/195

- Autres résultats :
  - Estimation du nombre de signatures fautées nécessaires
  - Attaque sur la spécification de l'algorithme de signature Dilithium
    - Ne nécessite pas de connaître $\mathbf{t}_0$
    - Même nombre de signatures pour la résolution du programme linéaire
- Questions ouvertes :
  - Extension sur la vérification de la norme de $\mathbf{z}$
  - Contremesures efficaces pour protéger la vérification de la norme

- Dilithium Sign : Attaque par fautes sur le check de $\|\mathbf{r}_0\|_\infty$
- Dilithium Sign : Attaque par canaux auxilaires sur $\mathbf{w}_0$

Peut-on exploiter des valeurs intermédiaires ?

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$, si on connait $\mathbf{y}$ alors on peut retrouver $\mathbf{s}_1$:

$$\mathbf{s}_1 = c^{-1}(\mathbf{z} - \mathbf{y})$$

Qu'en est-il des autres valeurs intermédiaires ?

Peut-on exploiter des valeurs intermédiaires ?

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$, si on connait $\mathbf{y}$ alors on peut retrouver $\mathbf{s}_1$:

$$\mathbf{s}_1 = c^{-1}(\mathbf{z} - \mathbf{y})$$

Qu'en est-il des autres valeurs intermédiaires ?

- L'algorithme de vérification nous donne :

$$\mathbf{w}_1' = \texttt{UseHint}(\mathbf{h}, \mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^{13})$$

- Comme la signature est vérifiée $\mathbf{w}_1' = \mathbf{w}_1$ et on peut réécrire :

$$\begin{aligned}\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^{13} &= \mathbf{A}\,(\mathbf{y} + c\,\mathbf{s}_1) - c\,(\mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2 - \mathbf{t}_0)\\ &= \underbrace{\mathbf{A}\,\mathbf{y}}_{\mathbf{w}} - c\,\mathbf{s}_2 + c\mathbf{t}_0\\ &= \mathbf{w}_1\,2\gamma_2 + \mathbf{w}_0 + c\,(\mathbf{t}_0 - \mathbf{s}_2)\end{aligned}$$

- Si un attaquant peut distinguer quand un coefficient $(\mathbf{w}_0)_{i,j} = cst$ alors

$$(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^{13})_{i,j} = (\mathbf{w}_1)_{i,j}\,2\gamma_2 + cst + (c\,(\mathbf{t}_0 - \mathbf{s}_2))_{i,j} \qquad (2)$$

Peut-on exploiter des valeurs intermédiaires ?

- Pour une signature $\sigma = (c, \mathbf{z}, \mathbf{h})$, si on connait $\mathbf{y}$ alors on peut retrouver $\mathbf{s}_1$:

$$\mathbf{s}_1 = c^{-1}(\mathbf{z} - \mathbf{y})$$

Qu'en est-il des autres valeurs intermédiaires ?

- L'algorithme de vérification nous donne :

$$\mathbf{w}'_1 = \texttt{UseHint}(\mathbf{h}, \mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^{13})$$

- Comme la signature est vérifiée $\mathbf{w}'_1 = \mathbf{w}_1$ et on peut réécrire :

$$\begin{aligned}\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^{13} &= \mathbf{A}\,(\mathbf{y} + c\,\mathbf{s}_1) - c\,(\mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2 - \mathbf{t}_0)\\ &= \underbrace{\mathbf{A}\,\mathbf{y}}_{\mathbf{w}} - c\,\mathbf{s}_2 + c\mathbf{t}_0\\ &= \mathbf{w}_1\,2\gamma_2 + \mathbf{w}_0 + c\,(\mathbf{t}_0 - \mathbf{s}_2)\end{aligned}$$

- Si un attaquant peut distinguer quand un coefficient $(\mathbf{w}_0)_{0,0} = 0$ alors

$$(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^{13})_{0,0} = (\mathbf{w}_1)_{0,0}\,2\gamma_2 + \ 0\ + (c\,(\mathbf{t}_0 - \mathbf{s}_2))_{0,0} \qquad (2)$$

Résolution

- La multiplication polynomiale peut s'écrire :

$$(c\,(\mathbf{t}_0 - \mathbf{s}_2))_{i,j} = (\mathbf{t}_0 - \mathbf{s}_2)_{i,0}(cX^0)_j + (\mathbf{t}_0 - \mathbf{s}_2)_{i,1}(cX^1)_j + \ldots + (\mathbf{t}_0 - \mathbf{s}_2)_{i,255}(cX^{255})_j$$

$$= \begin{pmatrix}(cX^0)_j & (cX^1)_j & \cdots & (cX^{254})_j & (cX^{255})_j\end{pmatrix} \cdot \begin{pmatrix}(\mathbf{t}_0 - \mathbf{s}_2)_{i,0}\\(\mathbf{t}_0 - \mathbf{s}_2)_{i,1}\\\vdots\\(\mathbf{t}_0 - \mathbf{s}_2)_{i,255}\end{pmatrix}$$

- L'équation (2) peut donc se réécrire :

$$\underbrace{(\mathbf{A}\,\mathbf{z} - c\,\mathbf{t}_1\,2^{13} - \mathbf{w}_1\,2\gamma_2)_{i,j}}_{b_i} = \underbrace{\begin{pmatrix}(cX^0)_j & (cX^1)_j & \cdots & (cX^{254})_j & (cX^{255})_j\end{pmatrix}}_{\mathbf{C}_i} \cdot \begin{pmatrix}(\mathbf{t}_0 - \mathbf{s}_2)_{i,0}\\(\mathbf{t}_0 - \mathbf{s}_2)_{i,1}\\\vdots\\(\mathbf{t}_0 - \mathbf{s}_2)_{i,255}\end{pmatrix} + \underbrace{0}_{e_i}$$

- Répéter l'opération pour différentes signatures $\sigma = (c, \mathbf{z}, \mathbf{h})$
- On s'arrête quand on a assez de lignes pour chaque polynôme $i \in [0, k[$

Formulation d'un problème

- Pour un certain polynôme $i \in [0, k[$ et un nombre M de signatures on a :

$$\underbrace{\begin{pmatrix} b_{i,j_0} \\ b_{i,j_1} \\ \vdots \\ b_{i,j_{M-1}} \end{pmatrix}}_{b_i} = \underbrace{\begin{pmatrix} (cX^0)_{j_0} & (cX^1)_{j_0} & \cdots & (cX^{255})_{j_0} \\ (cX^0)_{j_1} & (cX^1)_{j_1} & \cdots & (cX^{255})_{j_1} \\ \vdots & \vdots & \cdots & \vdots \\ (cX^0)_{j_{M-1}} & (cX^1)_{j_{M-1}} & \cdots & (cX^{255})_{j_{M-1}} \end{pmatrix}}_{\mathbf{C}_i} \cdot \underbrace{\begin{pmatrix} (\mathbf{t}_0 - \mathbf{s}_2)_{i,0} \\ (\mathbf{t}_0 - \mathbf{s}_2)_{i,1} \\ \vdots \\ (\mathbf{t}_0 - \mathbf{s}_2)_{i,255} \end{pmatrix}}_{(\mathbf{t}_0 - \mathbf{s}_2)_i} + \underbrace{\begin{pmatrix} e_{i,j_0} \\ e_{i,j_1} \\ \vdots \\ e_{i,j_{M-1}} \end{pmatrix}}_{e_i}$$

Formulation d'un problème

- Pour un certain polynôme $i \in [0, k[$ et un nombre M de signatures on a :

$$\underbrace{\begin{pmatrix} b_{i,j_0} \\ b_{i,j_1} \\ \vdots \\ b_{i,j_{M-1}} \end{pmatrix}}_{b_i} = \underbrace{\begin{pmatrix} (cX^0)_{j_0} & (cX^1)_{j_0} & \cdots & (cX^{255})_{j_0} \\ (cX^0)_{j_1} & (cX^1)_{j_1} & \cdots & (cX^{255})_{j_1} \\ \vdots & \vdots & \cdots & \vdots \\ (cX^0)_{j_{M-1}} & (cX^1)_{j_{M-1}} & \cdots & (cX^{255})_{j_{M-1}} \end{pmatrix}}_{\mathbf{C}_i} \cdot \underbrace{\begin{pmatrix} (\mathbf{t}_0 - \mathbf{s}_2)_{i,0} \\ (\mathbf{t}_0 - \mathbf{s}_2)_{i,1} \\ \vdots \\ (\mathbf{t}_0 - \mathbf{s}_2)_{i,255} \end{pmatrix}}_{(\mathbf{t}_0 - \mathbf{s}_2)_i} + \underbrace{\begin{pmatrix} e_{i,j_0} \\ e_{i,j_1} \\ \vdots \\ e_{i,j_{M-1}} \end{pmatrix}}_{e_i}$$

- Alors on peut résoudre avec la méthode des moindres carrés :

$$(\mathbf{t}_0 \tilde{-} \mathbf{s}_2)_i = ({}^t\mathbf{C}_i\, \mathbf{C}_i)^{-1}\, {}^t\mathbf{C}_i\, b_i.$$

- Pour un nombre de signatures M assez grand [2], on peut garantir que :

$$\|(\mathbf{t}_0 - \mathbf{s}_2)_i - (\mathbf{t}_0 \tilde{-} \mathbf{s}_2)_i\|_\infty < \frac{1}{2}, \text{ et donc que } \lceil (\mathbf{t}_0 \tilde{-} \mathbf{s}_2)_i \rfloor = (\mathbf{t}_0 - \mathbf{s}_2)_i$$

[2] J. Bootle, C. Delaplace, T. Espitau, PA. Fouque, M. Tibouchi,
LWE Without Modular Reduction and Improved Side-Channel Attacks Against BLISS

Et après ?

- Le $\mathbf{t}_0 - \mathbf{s}_2$ correct permet de retrouver $\mathbf{s}_1$ avec des informations publiques

$$\mathbf{A}\,\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t}_1\,2^{13} + \mathbf{t}_0$$
$$\mathbf{A}\,\mathbf{s}_1 = \mathbf{t}_1\,2^{13} + (\mathbf{t}_0 - \mathbf{s}_2)$$

$\mathbf{A}$ n'est pas carrée, mais $({}^t\mathbf{A}\,\mathbf{A})$ carrée et inversible avec très grande probabilité

$$\mathbf{s}_1 = ({}^t\mathbf{A}\,\mathbf{A})^{-1}\ {}^t\mathbf{A}\,(\mathbf{t}_1\,2^{13} + (\mathbf{t}_0 - \mathbf{s}_2)) \tag{3}$$

- Juste avec $\mathbf{s}_1$ on peut forger des signatures [3]

En résumé

La résolution dépend du $\mathbf{t}_0 - \mathbf{s}_2$ trouvé et donc de la capacité à distinguer $(\mathbf{w}_0)_{i,j} = cst$

[3] L. Groot Bruinderink, P. Pessl,
Differential Fault Attacks on Deterministic Lattice Signatures

Sign(msg, $\mathtt{sk} = (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0, \mathtt{pk})$):

- **Où inférer de l'information sur $\mathbf{w}_0$?**

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 = \mathtt{Decompose}(\mathbf{w})$
6 $\quad c \in B_\tau = \mathtt{H}(\mathtt{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} = \mathtt{MakeHint}(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

Sign(msg, sk$=(\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0,$ pk$)$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 =$ Decompose$(\mathbf{w})$
6 $\quad c \in B_\tau = \mathrm{H}(\mathrm{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} =$ MakeHint$(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- **Où inférer de l'information sur $\mathbf{w}_0$?**
- Dans la fonction Decompose
  - entrée : $\mathbf{w}$ inconnu mais borné dans un intervalle connu
  - sortie : $\mathbf{w}_0$ ce qu'on cherche

Sign(msg, sk$= (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0,$ pk$)$):

1 $(\mathbf{z}, \mathbf{h}) = \perp$
2 while $(\mathbf{z}, \mathbf{h}) = \perp$ do
3 $\quad \mathbf{y} \xleftarrow{\$} \tilde{S}^l_{\gamma_1}$
4 $\quad \mathbf{w} = \mathbf{A}\,\mathbf{y}$
5 $\quad \mathbf{w}_1, \mathbf{w}_0 =$ Decompose$(\mathbf{w})$
6 $\quad c \in B_\tau = \mathrm{H}(\mathrm{pk} \,||\, msg \,||\, \mathbf{w}_1)$
7 $\quad \mathbf{z} = y + c\,\mathbf{s}_1$
8 $\quad \mathbf{r}_0 = \mathbf{w}_0 - c\,\mathbf{s}_2$
9 $\quad$ if $||\mathbf{z}||_\infty \geq \gamma_1 - \beta$ or $||\mathbf{r}_0||_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) = \perp$
10 $\quad$ else,
11 $\quad\quad \mathbf{h} =$ MakeHint$(\mathbf{w}_1, \mathbf{r}_0 + c\,\mathbf{t}_0)$
12 $\quad\quad$ if $\|c\,\mathbf{t}_0\|_\infty \geq \gamma_2$ or $\|\mathbf{h}\|_1 > \omega$, then $(\mathbf{z}, \mathbf{h}) = \perp$
13 return $\sigma = (c, \mathbf{z}, \mathbf{h})$

- **Où inférer de l'information sur $\mathbf{w}_0$?**

- Dans la fonction Decompose
  - entrée : $\mathbf{w}$ inconnu mais borné dans un intervalle connu
  - sortie : $\mathbf{w}_0$ ce qu'on cherche

- Durant la soustraction
  - entrée : deux opérandes

Étude de la fonction `Decompose`

- Pour Dilithium-2 on a $\gamma_2 = \dfrac{q-1}{88} = 95\,232$:

```
1  int32_t Decompose(int32_t *w0, int32_t w) {
2      int32_t w1;
3      w1  = (w + 127) >> 7;
4      w1  = (w1 * 11275 + (1 << 23)) >> 24;
5      w1 ^= ((43 - w1) >> 31) & w1;
6
7      *w0  = w - w1 * 2 * GAMMA2;
8      *w0 -= (((Q - 1) / 2 - *w0) >> 31) & Q;
9      return w1;
10 }
```

Figure: Extrait du code C de la fonction `Decompose`

Étude de la fonction `Decompose`

- Pour Dilithium-2 on a $\gamma_2 = \dfrac{q-1}{88} = 95\,232$:

```
1  int32_t Decompose(int32_t *w0, int32_t w) {
2      int32_t w1;
3      w1  = (w + 127) >> 7;
4      w1  = (w1 * 11275 + (1 << 23)) >> 24;
5      w1 ^= ((43 - w1) >> 31) & w1;
6
7      *w0  = w - w1 * 2 * GAMMA2;
8      *w0 -= (((Q - 1) / 2 - *w0) >> 31) & Q;
9      return w1;
10 }
```

Figure: Extrait du code C de la fonction `Decompose`

- **Quelles valeurs de w_0 cibler ?**

Quelles valeurs cibler ?

- Modèle de fuite pour les attaques par canaux auxilaires :
  - Hamming Weight (HW) : nombre de 1 dans la représentation binaire d'un nombre

HW	0	1	2	3	4	5	6	7	8
Nombre de valeurs	1	8	28	56	72	56	28	8	1

- Moins de représentants pour $\text{HW} = 0$ ou $\text{HW} = 8$ $\Longrightarrow$ moins d'erreurs potentielles

Quelles valeurs cibler ?

- Modèle de fuite pour les attaques par canaux auxilaires :
  - Hamming Weight (HW) : nombre de 1 dans la représentation binaire d'un nombre

HW	0	1	2	3	4	5	6	7	8
Nombre de valeurs	1	8	28	56	72	56	28	8	1

- Moins de représentants pour HW $= 0$ ou HW $= 8$ $\implies$ moins d'erreurs potentielles
- $(\mathbf{w}_0)_{i,j}$ est une valeur signée en complément à deux dans $[-95\,231, 95\,232]$ $[\texttt{FFFE8C01}, \texttt{00017400}]$ ↔ conversion hexadécimal
  - Si $(\mathbf{w}_0)_{i,j} \geq 0$ alors $\texttt{00000000} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{00017400}$
  - Si $(\mathbf{w}_0)_{i,j} < 0$ alors $\texttt{FFFE8C01} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{FFFFFFFF}$
- La valeur $(\mathbf{w}_0)_{i,j} = \texttt{00000000}$ devrait être la plus simple à distinguer

Quelles valeurs cibler ?

- Modèle de fuite pour les attaques par canaux auxilaires :
  - Hamming Weight (HW) : nombre de 1 dans la représentation binaire d'un nombre

HW	0	1	2	3	4	5	6	7	8
Nombre de valeurs	1	8	28	56	72	56	28	8	1

- Moins de représentants pour HW $= 0$ ou HW $= 8$ $\implies$ moins d'erreurs potentielles
- $(\mathbf{w}_0)_{i,j}$ est une valeur signée en complément à deux dans $[-95\,231, 95\,232]$ $[\texttt{FFFE8C01}, \texttt{00017400}]$ conversion hexadécimal

Byte 0

  - Si $(\mathbf{w}_0)_{i,j} \geq 0$ alors $\texttt{00}\,\texttt{000000} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{00}\,\texttt{017400}$
  - Si $(\mathbf{w}_0)_{i,j} < 0$ alors $\texttt{FF}\,\texttt{FE8C01} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{FF}\,\texttt{FFFFFF}$

- La valeur $(\mathbf{w}_0)_{i,j} = \texttt{00000000}$ devrait être la plus simple à distinguer

Quelles valeurs cibler ?

- Modèle de fuite pour les attaques par canaux auxilaires :
  - Hamming Weight (HW) : nombre de 1 dans la représentation binaire d'un nombre

HW	0	1	2	3	4	5	6	7	8
Nombre de valeurs	1	8	28	56	72	56	28	8	1

- Moins de représentants pour HW $= 0$ ou HW $= 8$ $\implies$ moins d'erreurs potentielles
- $(\mathbf{w}_0)_{i,j}$ est une valeur signée en complément à deux dans $[-95\,231, 95\,232]$ $[\texttt{FFFE8C01}, \texttt{00017400}]$ conversion hexadécimal

Byte 1

  - Si $(\mathbf{w}_0)_{i,j} \geq 0$ alors $\texttt{00}\boxed{\texttt{00}}\texttt{0000} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{00}\boxed{\texttt{01}}\texttt{7400}$
  - Si $(\mathbf{w}_0)_{i,j} < 0$ alors $\texttt{FF}\boxed{\texttt{FE}}\texttt{8C01} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{FF}\boxed{\texttt{FF}}\texttt{FFFF}$

- La valeur $(\mathbf{w}_0)_{i,j} = \texttt{00000000}$ devrait être la plus simple à distinguer

Quelles valeurs cibler ?

- Modèle de fuite pour les attaques par canaux auxilaires :
  - Hamming Weight (HW) : nombre de 1 dans la représentation binaire d'un nombre

HW	0	1	2	3	4	5	6	7	8
Nombre de valeurs	1	8	28	56	72	56	28	8	1

- Moins de représentants pour HW $= 0$ ou HW $= 8$ $\implies$ moins d'erreurs potentielles
- $(\mathbf{w}_0)_{i,j}$ est une valeur signée en complément à deux dans $[-95\,231, 95\,232]$ $[\texttt{FFFE8C01}, \texttt{00017400}]$ conversion hexadécimal

Byte 2

  - Si $(\mathbf{w}_0)_{i,j} \geq 0$ alors $\texttt{00000000} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{00017400}$
  - Si $(\mathbf{w}_0)_{i,j} < 0$ alors $\texttt{FFFE8C01} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{FFFFFFFF}$

- La valeur $(\mathbf{w}_0)_{i,j} = \texttt{00000000}$ devrait être la plus simple à distinguer

Quelles valeurs cibler ?

- Modèle de fuite pour les attaques par canaux auxilaires :
  - Hamming Weight (HW) : nombre de 1 dans la représentation binaire d'un nombre

HW	0	1	2	3	4	5	6	7	8
Nombre de valeurs	1	8	28	56	72	56	28	8	1

- Moins de représentants pour HW $= 0$ ou HW $= 8$ $\implies$ moins d'erreurs potentielles
- $(\mathbf{w}_0)_{i,j}$ est une valeur signée en complément à deux dans $[-95\,231, 95\,232]$ conversion hexadécimal $[\texttt{FFFE8C01}, \texttt{00017400}]$

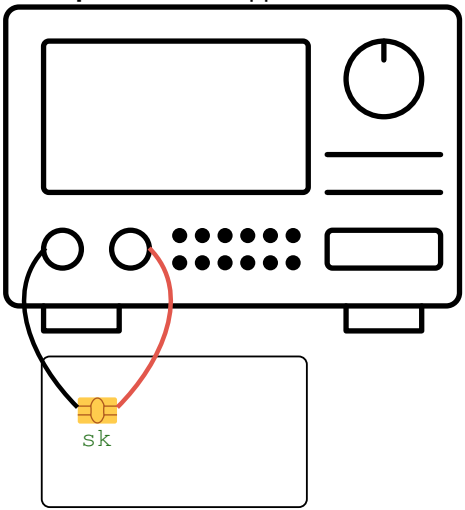
Byte 3

  - Si $(\mathbf{w}_0)_{i,j} \geq 0$ alors $\texttt{00000000} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{00017400}$
  - Si $(\mathbf{w}_0)_{i,j} < 0$ alors $\texttt{FFFE8C01} \leq (\mathbf{w}_0)_{i,j} \leq \texttt{FFFFFFFF}$

- La valeur $(\mathbf{w}_0)_{i,j} = \texttt{00000000}$ devrait être la plus simple à distinguer

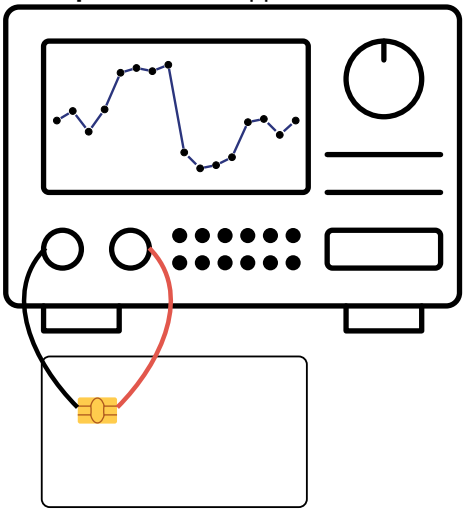
Attaque profilée

- **Étape 1** : sur un appareil clône, collecter des traces



Attaque profilée

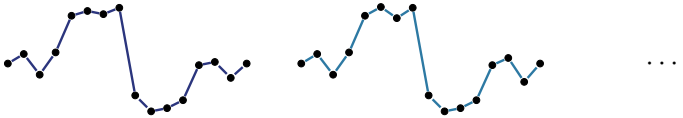
- **Étape 1** : sur un appareil clône, collecter des traces



Attaque profilée

- **Étape 1** : sur un appareil clône, collecter des traces

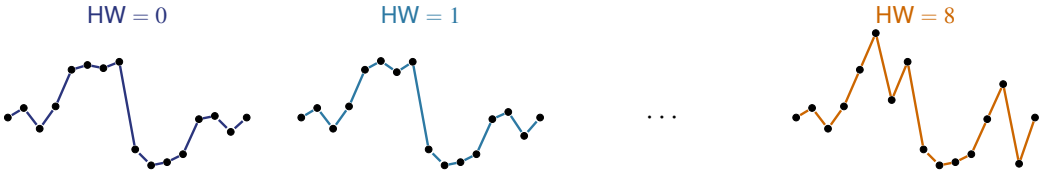
HW $= 0$ HW $= 1$



$\ldots$

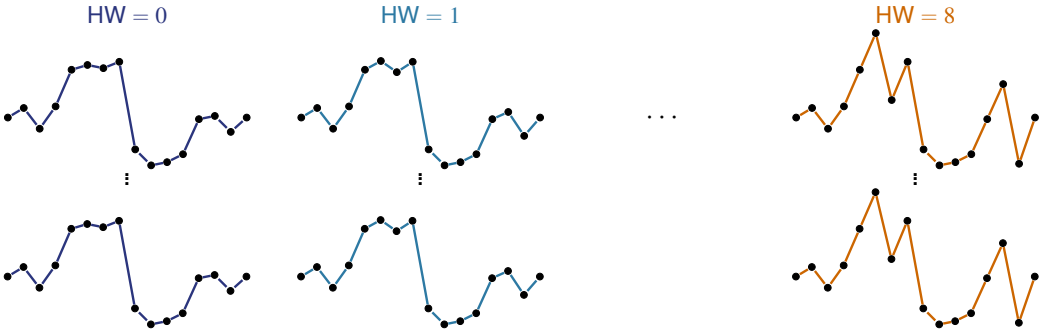
Attaque profilée

- **Étape 1** : sur un appareil clône, collecter des traces



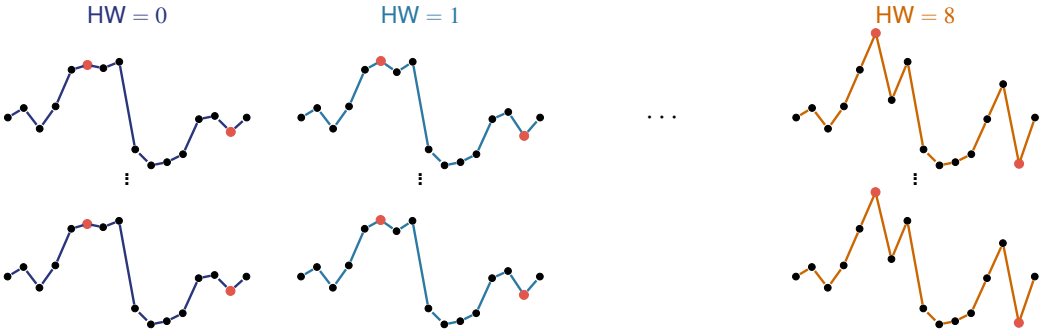
Attaque profilée

- **Étape 1** : sur un appareil clône, collecter des traces



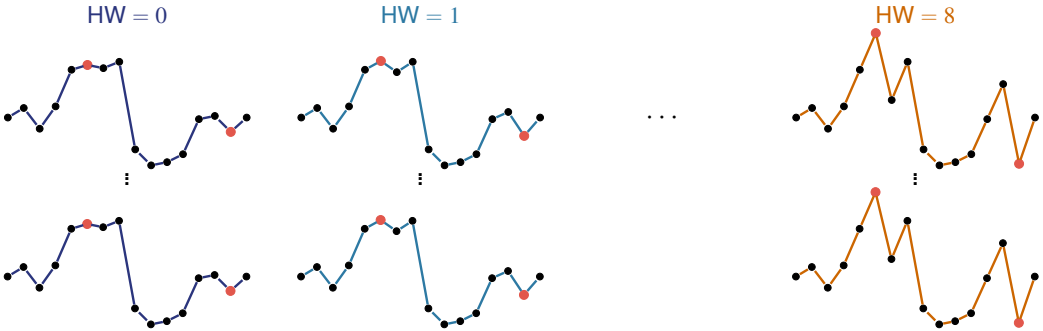
Attaque profilée

- **Étape 1** : sur un appareil clône, collecter des traces



Attaque profilée

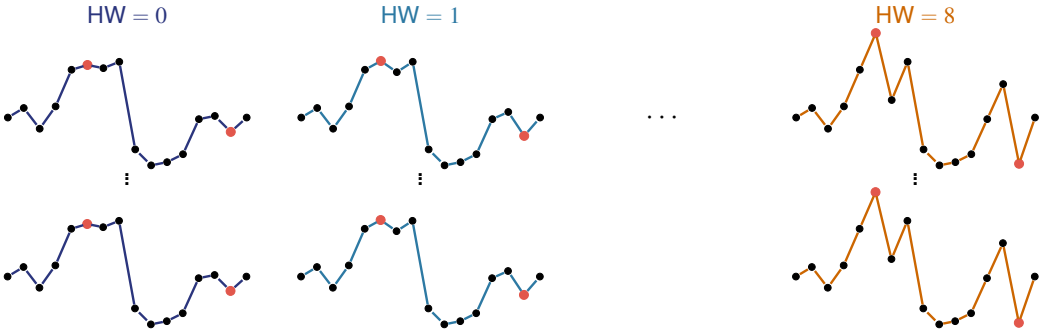
- **Étape 1** : sur un appareil clône, collecter des traces



- **Étape 2** : calculer les profils
  - Le vecteur des moyennes aux POIs pour chaque HW
  - La matrice de covariance entre chaque POIs

Attaque profilée

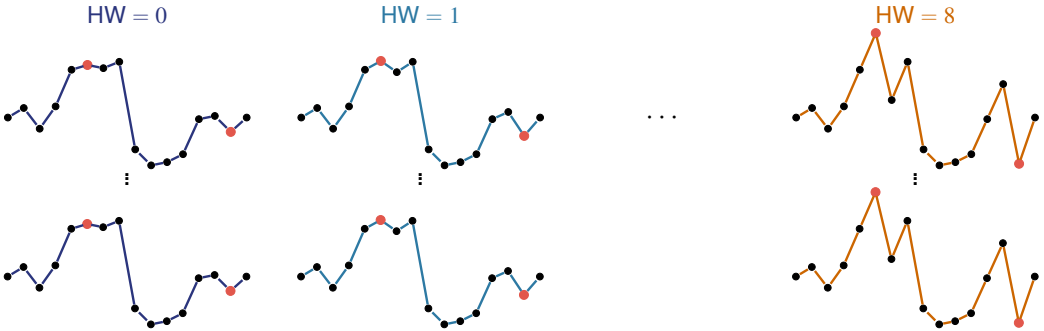
- **Étape 1** : sur un appareil clône, collecter des traces



- **Étape 2** : calculer les profils
  - Le vecteur des moyennes aux POIs pour chaque HW
  - La matrice de covariance entre chaque POIs
- **Étape 3** : collecter quelques traces sur l'appareil ciblé avec la `sk` inconnue

Attaque profilée

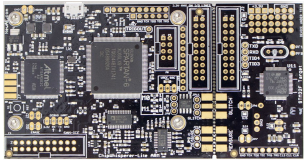
- **Étape 1** : sur un appareil clône, collecter des traces



- **Étape 2** : calculer les profils
  - Le vecteur des moyennes aux POIs pour chaque HW
  - La matrice de covariance entre chaque POIs
- **Étape 3** : collecter quelques traces sur l'appareil ciblé avec la `sk` inconnue
- **Étape 4** : appliquer nos profils et détection du candidat

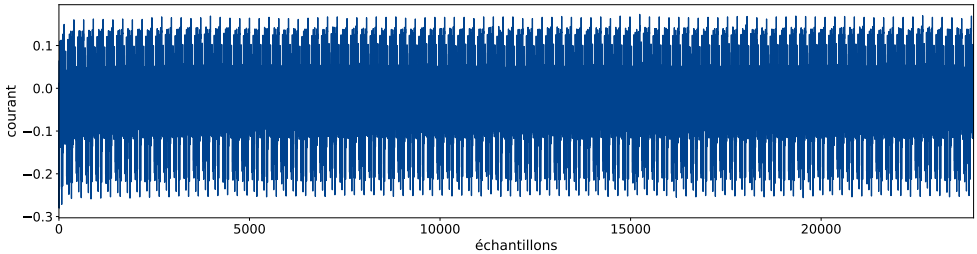
Étape 1 : collecte de traces

On envoie un msg



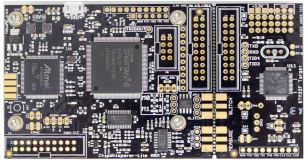
- ARM Cortex M4
- CPU: 32 bits
- RAM: 48kB
- 4 échantillons/cycle
- Max 24 400 échantillons

trace associée à un `Decompose`



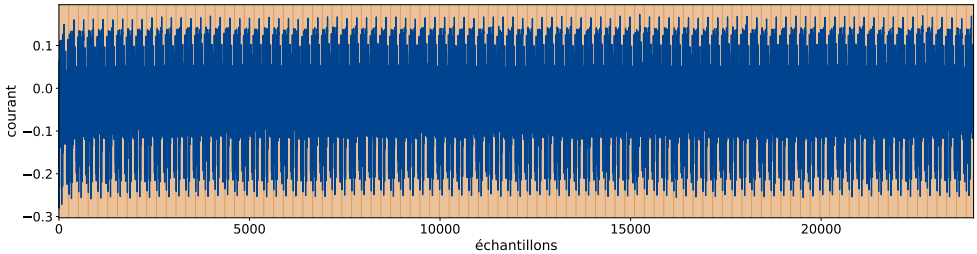
Étape 1 : collecte de traces

On envoie un msg



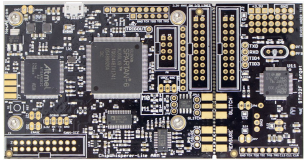
- ARM Cortex M4
- CPU: 32 bits
- RAM: 48kB
- 4 échantillons/cycle
- Max 24 400 échantillons

trace associée à un `Decompose`



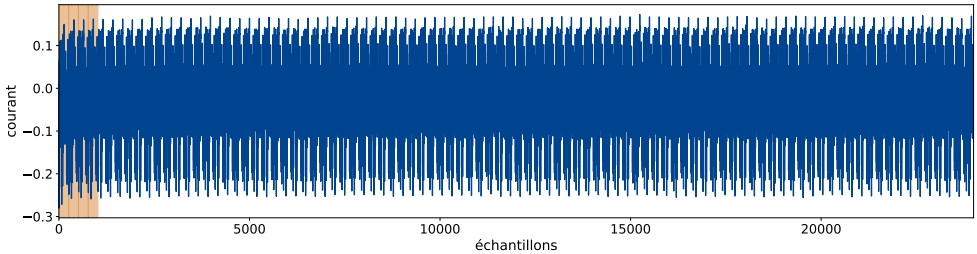
Étape 1 : collecte de traces

On envoie un msg



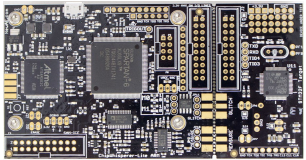
- ARM Cortex M4
- CPU: 32 bits
- RAM: 48kB
- 4 échantillons/cycle
- Max 24 400 échantillons

trace associée à un `Decompose`



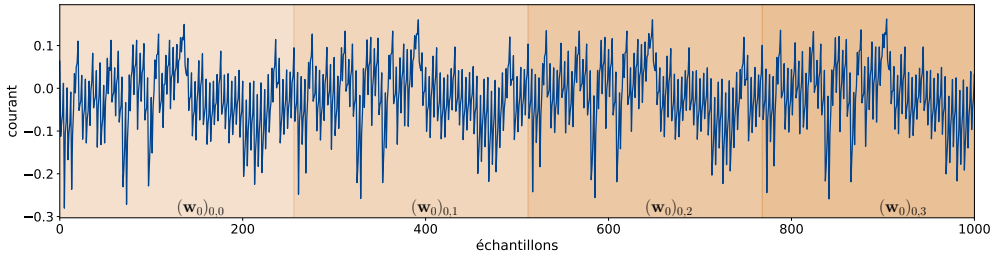
Étape 1 : collecte de traces

On envoie un msg



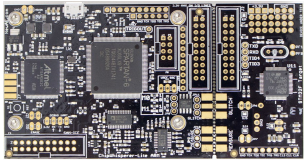
- ARM Cortex M4
- CPU: 32 bits
- RAM: 48kB
- 4 échantillons/cycle
- Max 24 400 échantillons

trace associée à un `Decompose`



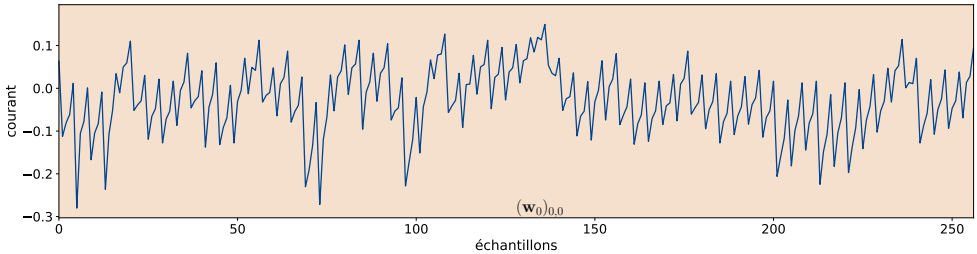
Étape 1 : collecte de traces

On envoie un msg



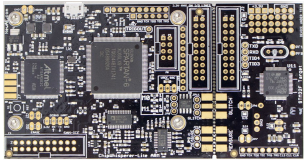
- ARM Cortex M4
- CPU: 32 bits
- RAM: 48kB
- 4 échantillons/cycle
- Max 24 400 échantillons

trace associée à un `Decompose`



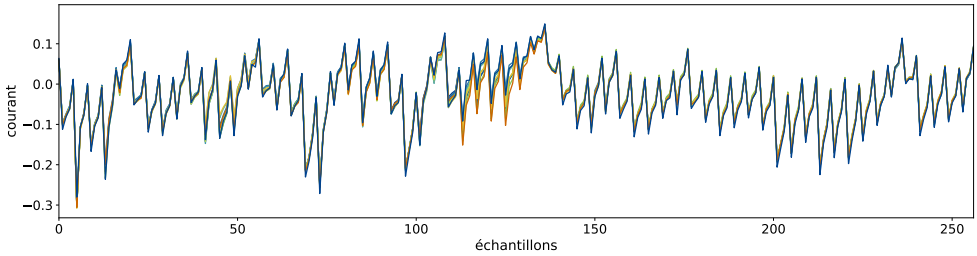
Étape 1 : collecte de traces

On envoie des msg différents



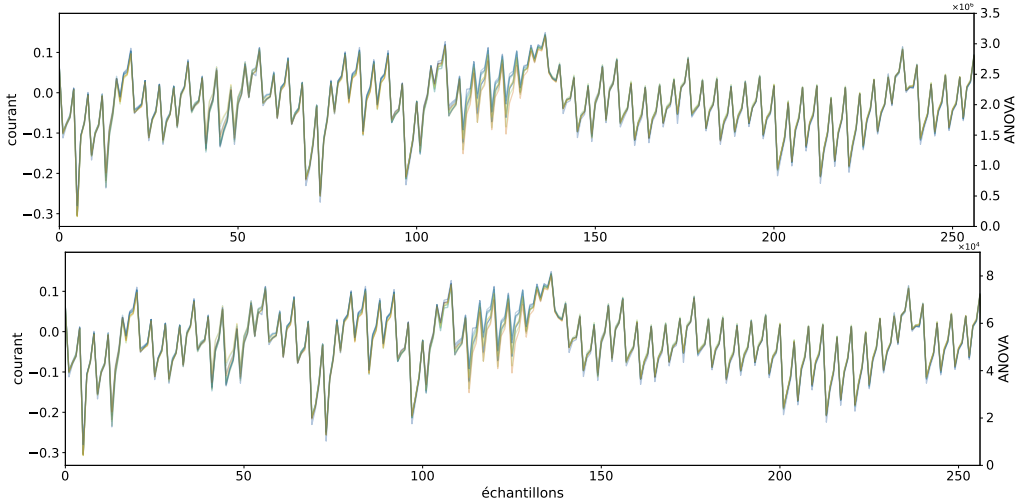
- ARM Cortex M4
- CPU: 32 bits
- RAM: 48kB
- 4 échantillons/cycle
- Max 24 400 échantillons

traces associées à des `Decompose` différents



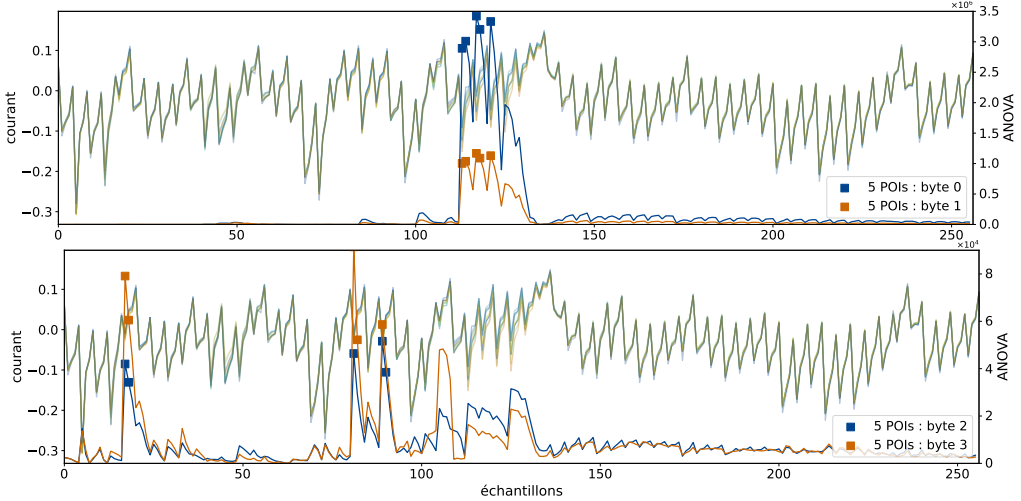
Étape 1 : analyse de fuite d'information

- Test statistique utilisé : ANOVA

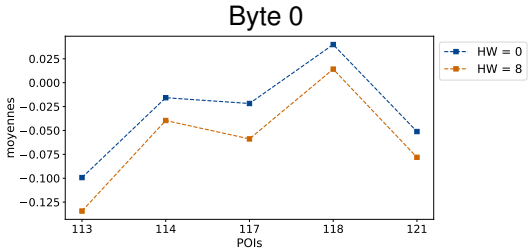


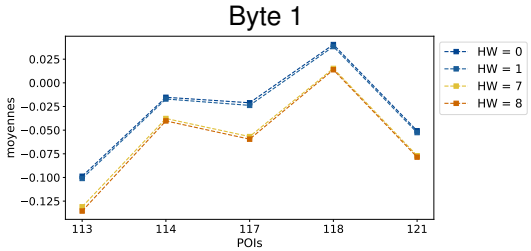
Étape 1 : analyse de fuite d'information

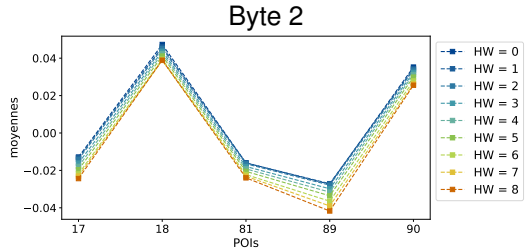
- Test statistique utilisé : ANOVA

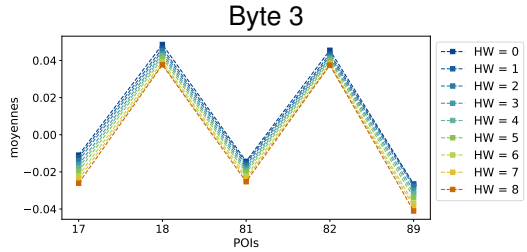


Étape 2 : construction des références



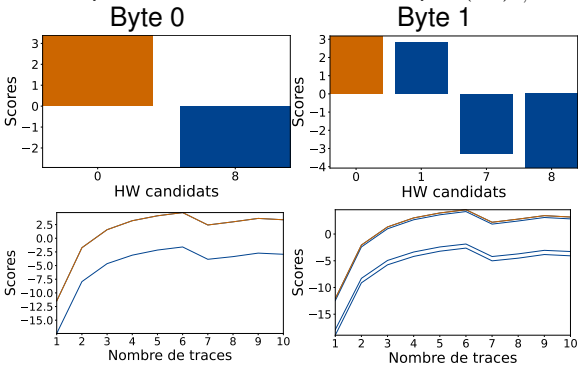






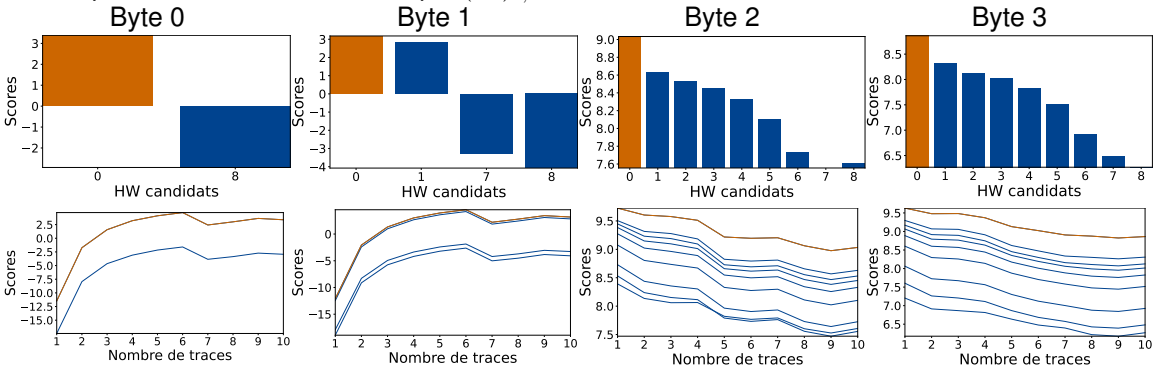
Étape 3 et 4 : identification du candidat

- Acquisition de 10 traces telles que $(\mathbf{w}_0)_{0,0} = 0$



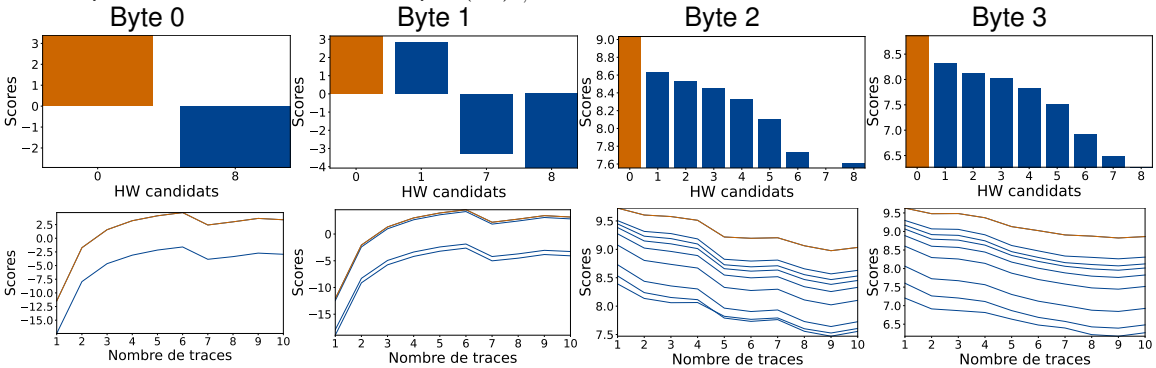
Étape 3 et 4 : identification du candidat

- Acquisition de 10 traces telles que $(\mathbf{w}_0)_{0,0} = 0$



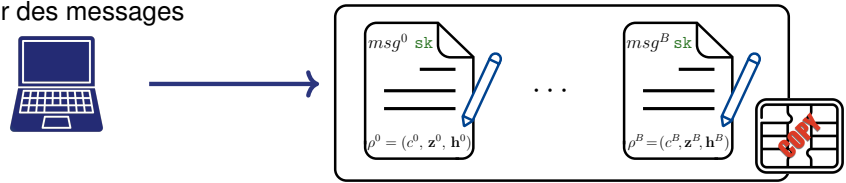
Étape 3 et 4 : identification du candidat

- Acquisition de 10 traces telles que $(\mathbf{w}_0)_{0,0} = 0$

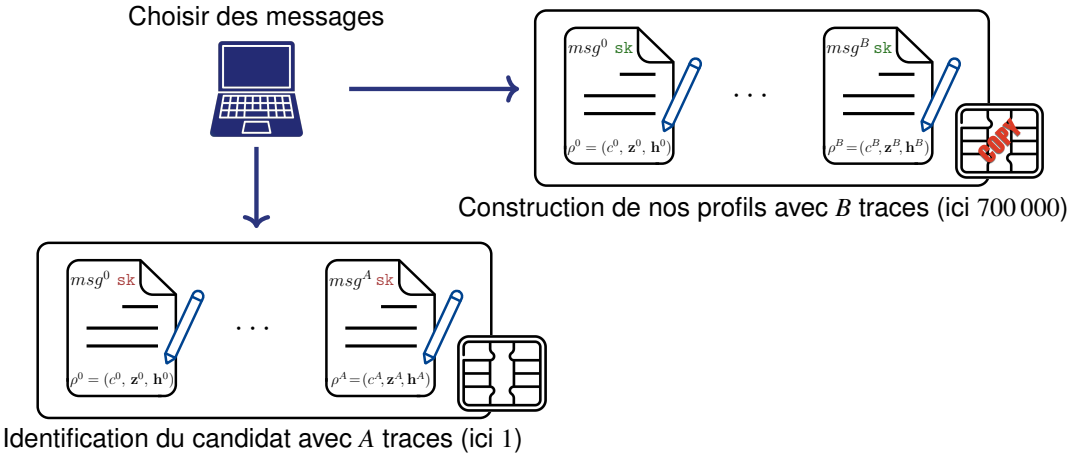


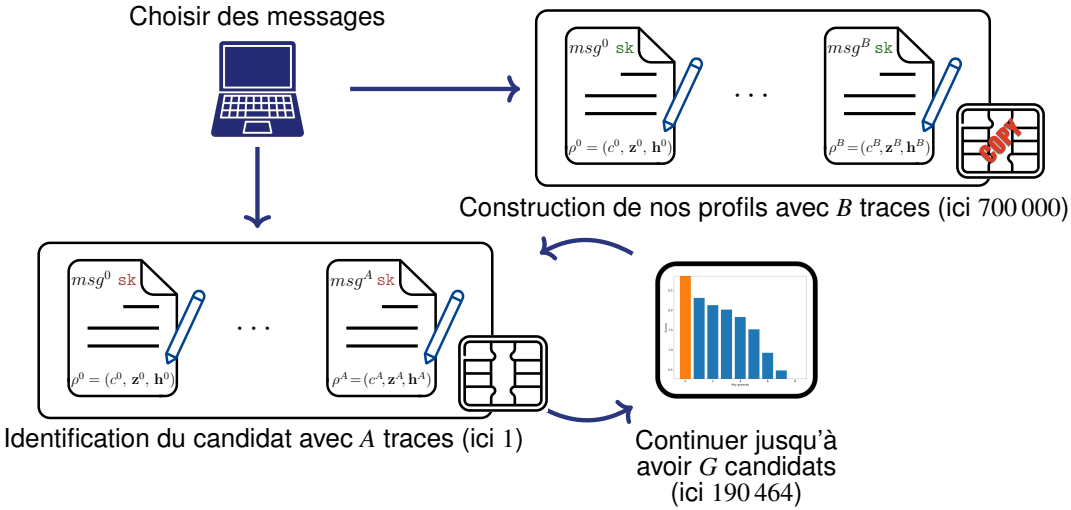
- Faux positifs : prédire $(\mathbf{w}_0)_{i,j} = 0$ alors que ce n'est pas le cas
  $0.067\% \Rightarrow \leq 1$ coefficient parmis les $k \times n$ au total
- Faux négatifs : prédire $(\mathbf{w}_0)_{i,j} \neq 0$ alors que ce n'est pas le cas
  $0.174\% \Rightarrow$ un (petit) peu plus de signatures à collecter

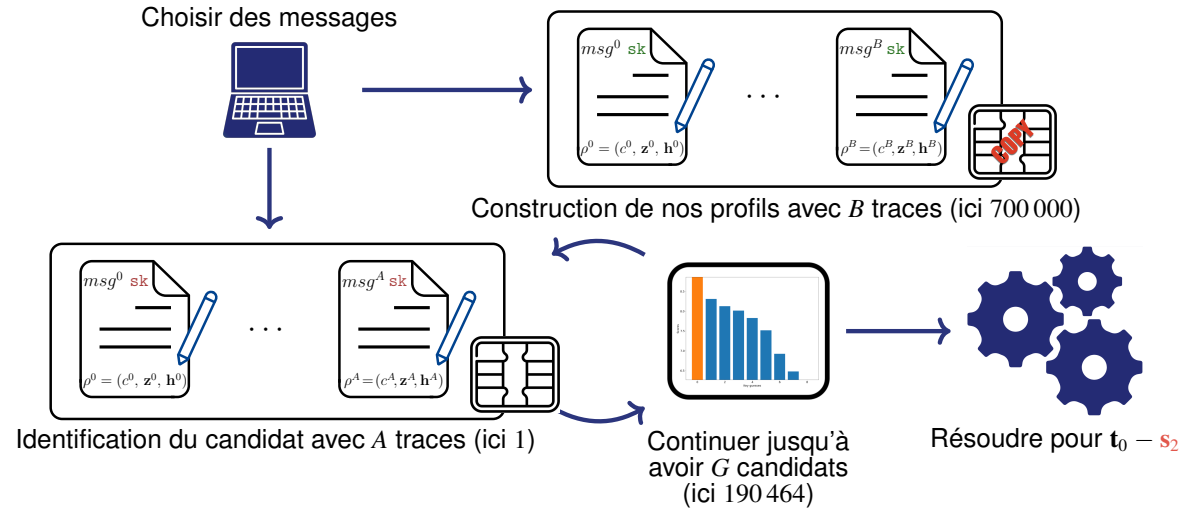
Choisir des messages

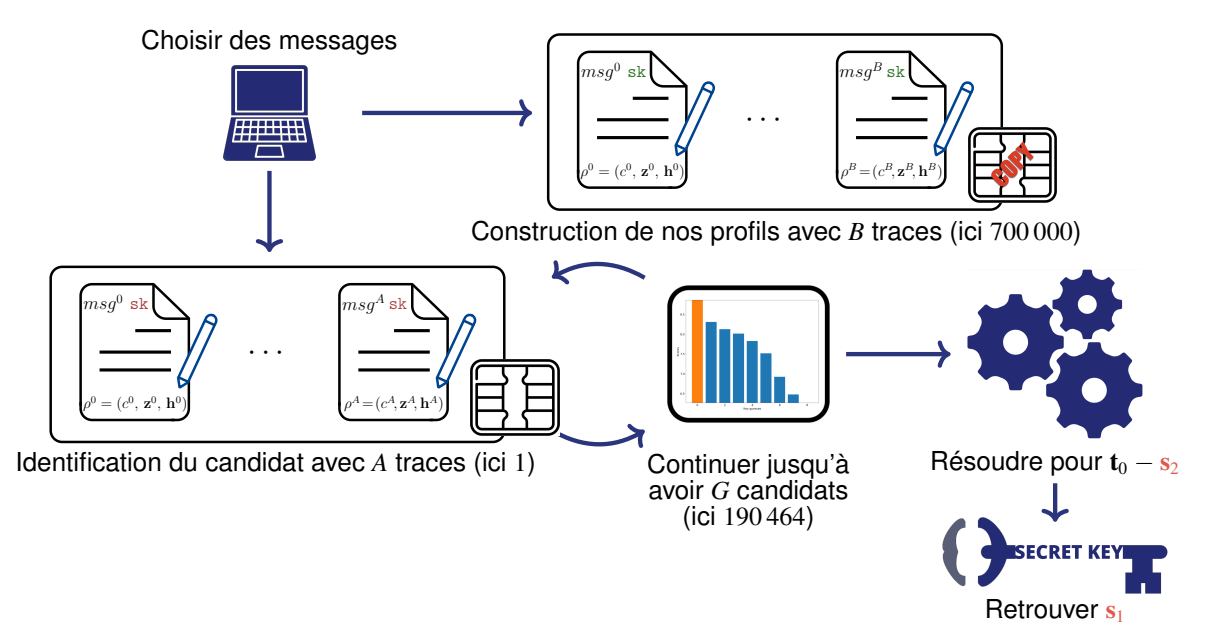


Construction de nos profils avec B traces (ici $700\,000$)


Choisir des messages
msg^0 sk
$\rho^0 = (c^0, \mathbf{z}^0, \mathbf{h}^0)$
msg^B sk
$\rho^B = (c^B, \mathbf{z}^B, \mathbf{h}^B)$
COPY
Construction de nos profils avec B traces (ici $700\,000$)
msg^0 sk
$\rho^0 = (c^0, \mathbf{z}^0, \mathbf{h}^0)$
msg^A sk
$\rho^A = (c^A, \mathbf{z}^A, \mathbf{h}^A)$
Identification du candidat avec A traces (ici 1)


Choisir des messages
msg^0 sk
$\rho^0 = (c^0, \mathbf{z}^0, \mathbf{h}^0)$
msg^B sk
$\rho^B = (c^B, \mathbf{z}^B, \mathbf{h}^B)$
COPY
Construction de nos profils avec B traces (ici $700\,000$)
msg^0 sk
$\rho^0 = (c^0, \mathbf{z}^0, \mathbf{h}^0)$
msg^A sk
$\rho^A = (c^A, \mathbf{z}^A, \mathbf{h}^A)$
Identification du candidat avec A traces (ici 1)
Continuer jusqu'à avoir G candidats (ici $190\,464$)


Choisir des messages
msg^0 sk
$\rho^0 = (c^0, \mathbf{z}^0, \mathbf{h}^0)$
msg^B sk
$\rho^B = (c^B, \mathbf{z}^B, \mathbf{h}^B)$
COPY
Construction de nos profils avec B traces (ici $700\,000$)
msg^0 sk
$\rho^0 = (c^0, \mathbf{z}^0, \mathbf{h}^0)$
msg^A sk
$\rho^A = (c^A, \mathbf{z}^A, \mathbf{h}^A)$
Identification du candidat avec A traces (ici 1)
Continuer jusqu'à avoir G candidats (ici $190\,464$)
Résoudre pour $\mathbf{t}_0 - \mathbf{s}_2$


Choisir des messages
msg^0 sk
$\rho^0 = (c^0, \mathbf{z}^0, \mathbf{h}^0)$
msg^B sk
$\rho^B = (c^B, \mathbf{z}^B, \mathbf{h}^B)$
COPY
Construction de nos profils avec B traces (ici 700 000)
msg^0 sk
$\rho^0 = (c^0, \mathbf{z}^0, \mathbf{h}^0)$
msg^A sk
$\rho^A = (c^A, \mathbf{z}^A, \mathbf{h}^A)$
Identification du candidat avec A traces (ici 1)
Continuer jusqu'à avoir G candidats (ici 190 464)
Résoudre pour $\mathbf{t}_0 - \mathbf{s}_2$
SECRET KEY
Retrouver $\mathbf{s}_1$

Conclusion

Exploiting Intermediate Value Leakage in Dilithium: A Template-Based Approach

Alexandre Berzati, Andersson Calle Viera, Maya Chartouny, Steven Madec, Damien Vergnaud and David Vigilant

présenté à **TCHES23**

ia.cr/2023/050

- Autres résultats :
  - Méthode pour filtrer et réduire les faux positifs
  - Méthode alternative de résolution pour exploiter des inégalités (LP)
  - Méthode pour gérer des (petites) erreurs de $(\mathbf{w}_0)_{i,j}$
- Questions ouvertes :
  - Utilisation d'autres valeurs
  - Exploitation de la fuite autrement
  - Attaque sur implémentation masquée

Conclusion

- Pour résumer :
  - Meilleure compréhension des chemins d'attaques sur les valeurs intermédiaires
  - Meilleure compréhension des méthodes de résolution
- Ce qu'il reste à faire :
  - Nouveaux vecteurs d'attaques ?
  - Nouvelles méthodes de résolution ?
  - Nouvelles attaques sur les implémentations sécurisées ?
  - Nouvelles contremesures combinant sécurité contre les SCA et les FA ?

Merci !

Bibliographie

[1] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé,
CRYSTALS - Dilithium: Digital Signatures from Module Lattices

[2] P. Azevedo-Oliveira, A. Calle Viera, B. Cogliati, L. Goubin,
Uncompressing Dilithium's public key

[3] L. Groot Bruinderink, P. Pessl,
Differential Fault Attacks on Deterministic Lattice Signatures

[4] J. Bootle, C. Delaplace, T. Espitau, PA. Fouque, M. Tibouchi,
LWE Without Modular Reduction and Improved Side-Channel Attacks Against BLISS

Contenu Additionnel

- Dilithium Sign : Implémentation optimisée de l'algorithme de signature
- Dilithium Sign : Attaque par fautes sur la vérification de la norme
- Dilithium Sign : Sensibilité aux attaques par fautes de la vérification

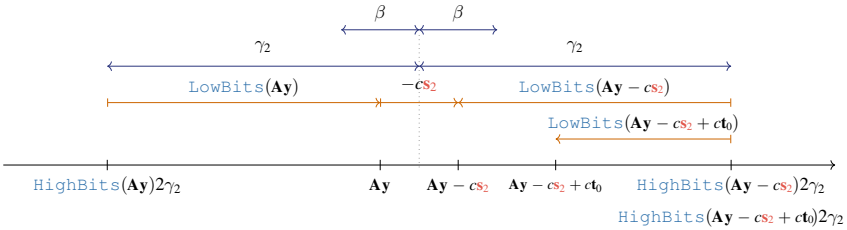
Optimisations de l'algorithme de signature

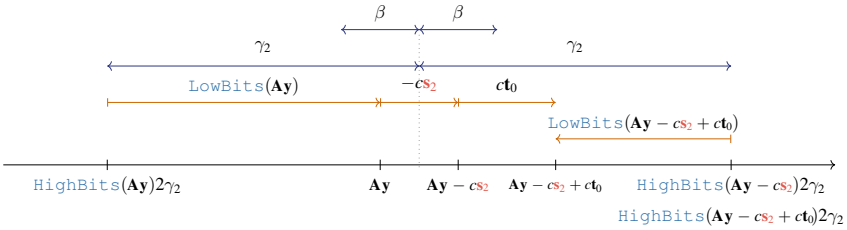
- **Problème 1:** Taille des clés plus grande que algos "pré-quantiques" ($\approx 7\times$ plus)
- **Problème 2:** Taille des signatures plus grande que algos "pré-quantiques" ($\approx 12\times$ plus)
- **Problème 3:** Taille des éléments plus grand que algos "pré-quantiques"
  - Chaque polynôme a 256 coefficients qui tiennent sur 4 bytes chacun
    $\implies 256 \times 4 = 1024$ bytes/polynôme
  - Les vecteurs ont au minimum $k = 4$ ou $l = 4$ polynômes (Dilithium-2)
    $\implies 4 \times 1024 = 4096$ bytes/vecteur
  - La matrice A a au minimum $k \times l = 4 \times 4$ polynômes (Dilithium-2)
    $\implies 4 \times 4 \times 1024 = 16\,384$ bytes/matrice

Niveau de sécurité	Dilithium-2	Dilithium-3	Dilithium-5
Taille (en bytes)	46 080	72 704	113 664

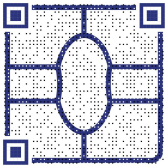
Table: Tailles mémoire des éléments arithmétiques (approximation).

Différences spécification vs. implémentation





Attaque par faute sur la vérification



10.1007/978-3-031-54409-5_4

Titre : Fault Attacks sensitivity of Dilithium Verify (CARDIS2023)

- Analyse de la sensibilité aux attaques par fautes de Verify
- Focus sur les opérations habituellement non protégées
- Idée principale : rendre $c\mathbf{t}_1 2^{13}$ plus petit que normalement

Sauter la soustraction

$$\mathbf{w}_1' = \texttt{UseHint}(\mathbf{h}, \mathbf{A}\,\mathbf{z} - c\mathbf{t}_1 2^{13})$$

Mettre à zéro c

Changer l'exposant d

- Permet de faire accepter des fausses signatures avec peu de simples fautes
- Contremesures simples et efficaces introduites