One Bit Is All It Takes

A Devastating Attack Against BLISS Non-Constant Time Sign Flips

Mehdi Tibouchi, **Alexandre Wallet**

Summary

- Lattice signatures schemes have outputs distributed along sk
- BLISS.Sign hides this with **rejection sampling**
- Efficient rejection uses bimodal Gaussians via a **bitflip**
- Careless implementation **leaks** the bitflip

Results:

leakage + parametric inference	(maths) $\longrightarrow$	efficient statistical estimator for sk

+ Code to compute the estimator (`https://github.com/awallet/OneBitBliss`)

= **Recover** sk.†

†: with 1 bit of leakage by signature, for around 100.000 signatures

Summary

- Lattice signatures schemes have outputs distributed along sk
- BLISS.Sign hides this with **rejection sampling**
- Efficient rejection uses bimodal Gaussians via a **bitflip**
- Careless implementation **leaks** the bitflip

Results:

leakage + parametric inference	$\xrightarrow{\text{(maths)}}$	efficient statistical estimator for sk

$+$ Code to compute the estimator (`https://github.com/awallet/OneBitBliss`)

$=$ **Recover** sk.†

†: with 1 bit of leakage by signature, for around 100.000 signatures

What's BLISS?

Bimodal (gaussians) **L**att**I**ce-based **S**ignature **S**cheme [DDLL13]

Perks:

Efficient, compact

Secure†

Security:

Key-recovery $\sim$ NTRU

Forgery $\sim$ R-SIS

Notations:

$R = \mathbb{Z}[x]/(x^n + 1)$, $n = 512$
$\mathbf{c}^\star$: adjoint element
$\langle \mathbf{a}, \mathbf{bc} \rangle = \langle \mathbf{ac}^\star, \mathbf{b} \rangle$

$D_{\sigma,\mathbf{c}}$: discrete Gaussian over R
center $\mathbf{c}$, std.dev. σ

†: in a black-box model

Signing algorithm and parameters

BLISS.Sign(μ, pk $= (\mathbf{v}_1, q-2)$, sk $= (\mathbf{s}_1, \mathbf{s}_2)$)

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D_\sigma$;
2: $\mathbf{u} \leftarrow \zeta \cdot \mathbf{v}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$;
3: $\mathbf{c} \leftarrow \mathcal{H}(\mathbf{u}, \mu)$;
4: Choose a random bit b;
5: $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
$\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$;
6: **continue** with probability $\mathcal{P}_{rej}$
else **restart**;
7: $\mathbf{z}_2^\dagger \leftarrow$ Compress$(\mathbf{z}_2)$;
8: **return** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$;

$\|\mathbf{s}_1\|^2 = \lfloor \delta_1 n \rceil + 4 \lfloor \delta_2 n \rceil$
$\delta_1, \delta_2 \in (0,1)$ **known**

$\mathbf{c} \in \{0,1\}^n$, $\|\mathbf{c}\|_1 = \kappa$
ROM: $\mathbf{c} \sim$ uniform

rejection phase [L12]

Example (BLISS-I): $n = 512, \delta_1 = 0.3, \delta_2 = 0, \kappa = 23, \sigma = 215$

Signing algorithm and parameters

BLISS.Sign$(\mu, \mathsf{pk} = (\mathbf{v}_1, q-2), \mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2))$

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D_\sigma$;
2: $\mathbf{u} \leftarrow \zeta \cdot \mathbf{v}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$;
3: $\mathbf{c} \leftarrow \mathcal{H}(\mathbf{u}, \mu)$;
4: Choose a random bit b;
5: $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
$\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$;
6: **continue** with probability $\mathcal{P}_{rej}$
else **restart**;
7: $\mathbf{z}_2^\dagger \leftarrow$ Compress$(\mathbf{z}_2)$;
8: **return** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$;

$\|\mathbf{s}_1\|^2 = \lceil \delta_1 n \rceil + 4\lceil \delta_2 n \rceil$
$\delta_1, \delta_2 \in (0,1)$ **known**

$\mathbf{c} \in \{0,1\}^n$, $\|\mathbf{c}\|_1 = \kappa$
ROM: $\mathbf{c} \sim$ uniform

rejection phase [L12]

Example (BLISS-I): $n = 512, \delta_1 = 0.3, \delta_2 = 0, \kappa = 23, \sigma = 215$

Signing algorithm and parameters

BLISS.Sign$(\mu, \mathsf{pk} = (\mathbf{v}_1, q-2), \mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2))$

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D_\sigma$;
2: $\mathbf{u} \leftarrow \zeta \cdot \mathbf{v}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$;
3: $\mathbf{c} \leftarrow \mathcal{H}(\mathbf{u}, \mu)$;
4: Choose a random bit b;
5: $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
$\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$;
6: **continue** with probability $\mathcal{P}_{rej}$
else **restart**;
7: $\mathbf{z}_2^\dagger \leftarrow \mathsf{Compress}(\mathbf{z}_2)$;
8: **return** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$;

$\|\mathbf{s}_1\|^2 = \lfloor \delta_1 n \rceil + 4 \lfloor \delta_2 n \rceil$
$\delta_1, \delta_2 \in (0,1)$ **known**

$\mathbf{c} \in \{0,1\}^n$, $\|\mathbf{c}\|_1 = \kappa$
ROM: $\mathbf{c} \sim$ uniform

rejection phase [L12]

Example (BLISS-I): $n = 512, \delta_1 = 0.3, \delta_2 = 0, \kappa = 23, \sigma = 215$

Signing algorithm and parameters

BLISS.Sign$(\mu, \mathsf{pk} = (\mathbf{v}_1, q-2), \mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2))$

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D_\sigma$;
2: $\mathbf{u} \leftarrow \zeta \cdot \mathbf{v}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$;
3: $\mathbf{c} \leftarrow \mathcal{H}(\mathbf{u}, \mu)$;
4: Choose a random bit b;
5: $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
$\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$;
6: **continue** with probability $\mathcal{P}_{rej}$
else **restart**;
7: $\mathbf{z}_2^\dagger \leftarrow$ Compress$(\mathbf{z}_2)$;
8: **return** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$;

$\|\mathbf{s}_1\|^2 = \lceil \delta_1 n \rceil + 4\lceil \delta_2 n \rceil$
$\delta_1, \delta_2 \in (0,1)$ **known**

$\mathbf{c} \in \{0,1\}^n,\ \|\mathbf{c}\|_1 = \kappa$
ROM: $\mathbf{c} \sim$ uniform

rejection phase [L12]

Example (BLISS-I): $n = 512, \delta_1 = 0.3, \delta_2 = 0, \kappa = 23, \sigma = 215$

BLISS vs. side-channel

Side-channel vulnerabilities

vs. Gaussian sampling
[BHLY16], [PBY17]

vs. rejection sampling
[EFGT17], [BDE+18], [BBE+19]

In this work: target the "bimodal" part (bitflip at step 4)

BLISS.Sign(μ, pk $= (\mathbf{v}_1, q-2)$, sk $= (\mathbf{s}_1, \mathbf{s}_2)$)

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D_\sigma$;
2: $\mathbf{u} \leftarrow \zeta \cdot \mathbf{v}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$;
3: $\mathbf{c} \leftarrow \mathcal{H}(\mathbf{u}, \mu)$;
4: **Choose a random bit b;**
5: $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
$\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$;
$\dots$

Roadmap of the attack

Conditional branching using b carelessly†

$\downarrow$

Get b (in timing-leakage model): "LEAK.Sign"

$\downarrow$

Explicitely compute a **maximum likelihood estimator** $\hat{\mathbf{s}}$ for $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$

$\downarrow$

With enough traces, $\hat{\mathbf{s}} = \mathbf{s}$.

†: original implem. [DDLL13], strongSwan VPN suite,...

Likelihoods

$\mathbf{s} \in \mathbb{R}^n$ a parameter and $\mathbf{X} \sim \mu_\mathbf{s}$ a random variable

(log)-likelihood function:

$$\ell_\mathbf{x}(\mathbf{s}) := \log \mu_\mathbf{s}(\mathbf{x}) = \log \mathbb{P}[\mathbf{X} = \mathbf{x}]$$

For samples $\mathbf{x}_1, \ldots, \mathbf{x}_m \hookleftarrow \mu_\mathbf{s}$:

$$\ell_{\mathbf{x}_1,\ldots,\mathbf{x}_m}(\mathbf{s}) = \sum_{i\in[m]} \log \mu_\mathbf{s}(\mathbf{x}_i)$$

Maximum likelihood estimator (MLE)

$\hat{\mathbf{s}}_m$ associated with $\mathbf{X}$:

$$\hat{\mathbf{s}}_m(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \mathrm{argmax}_\mathbf{s}\, \ell_{\mathbf{x}_1,\ldots,\mathbf{x}_m}(\mathbf{s})$$

$$(\forall\ \mathbf{x}_1, \ldots, \mathbf{x}_m \hookleftarrow \mu_\mathbf{s})$$

Theorem (under some technical conditions)

$(\hat{\mathbf{s}}_m)_m$ converges almost surely to $\mathbf{s}$.

Likelihoods

$\mathbf{s} \in \mathbb{R}^n$ a parameter and $\mathbf{X} \sim \mu_{\mathbf{s}}$ a random variable

(log)-likelihood function:

$$\ell_{\mathbf{x}}(\mathbf{s}) := \log \mu_{\mathbf{s}}(\mathbf{x}) = \log \mathbb{P}[\mathbf{X} = \mathbf{x}]$$

For samples $\mathbf{x}_1, \ldots, \mathbf{x}_m \hookleftarrow \mu_{\mathbf{s}}$:

$$\ell_{\mathbf{x}_1,\ldots,\mathbf{x}_m}(\mathbf{s}) = \sum_{i\in[m]} \log \mu_{\mathbf{s}}(\mathbf{x}_i)$$

Maximum likelihood estimator (MLE)

$\hat{\mathbf{s}}_m$ associated with $\mathbf{X}$:

$$\hat{\mathbf{s}}_m(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \mathrm{argmax}_{\mathbf{s}}\, \ell_{\mathbf{x}_1,\ldots,\mathbf{x}_m}(\mathbf{s})$$

$$(\forall\ \mathbf{x}_1, \ldots, \mathbf{x}_m \hookleftarrow \mu_{\mathbf{s}})$$

Theorem (under some technical conditions)

$(\hat{\mathbf{s}}_m)_m$ converges almost surely to $\mathbf{s}$.

MLE for BLISS

Before rejection: $\mathbf{z} \hookleftarrow D_{\sigma,(-1)^b \mathbf{sc}}$

Keep it with proba $\mathcal{P}_{rej}$

$\Rightarrow$

LEAK.Sign outputs $(b, \mathbf{z}, \mathbf{c}) \hookleftarrow \mu_{\mathbf{s}}$

$\mu_{\mathbf{s}}(b, \mathbf{z}, \mathbf{c}) = D_{\sigma,(-1)^b \mathbf{sc}}(\mathbf{z}) \cdot \mathcal{P}_{rej}$

Likelihood function: $\ell_{(b,\mathbf{z},\mathbf{c})}(\mathbf{s}) = -\varphi(\langle (-1)^b \mathbf{z}\mathbf{c}^\star, \mathbf{s}\rangle)$,

where $\varphi(t) = -\log\left(1 + \exp(-\frac{2t}{\sigma^2})\right)$
(analytic, strictly concave)

There is a unique MLE $\hat{\mathrm{s}}_m$ on the sphere of radius $\|\mathbf{s}\|$.

How good is the estimator $\hat{\mathrm{s}}_m$?

The Fisher information

For $\mathbf{X} \sim \mu_{\mathbf{s}}$ and the likelihood function $\ell_{\mathbf{x}}(\mathbf{s})$:

Fisher Information Matrix

$$I(\mathbf{s}) = -\mathbb{E}_{\mathbf{X}}\left[\frac{\partial^2}{\partial \mathbf{s}_i \partial \mathbf{s}_j} \ell_{\mathbf{X}}(\mathbf{s})\right]_{i,j}$$

Theorem: Convergence in law

$$\sqrt{m}(\hat{\mathbf{s}}_m - \mathbf{s}) \longrightarrow \mathcal{N}(\mathbf{0}, I(\mathbf{s})^{-1})$$

Expression of the Fisher information

Let $\mathbf{w} := (-1)^b \mathbf{z}\mathbf{c}^\star$ and $\overline{\mathbf{w}} := \mathbb{E}_{b,\mathbf{z},\mathbf{c}}[\mathbf{w}]$.

We show (with heuristics†):

$$I(\mathbf{s}) \approx \mathbb{E}_{\mathbf{w}}\left[\cosh\left(\frac{\langle \mathbf{w}, \mathbf{s} \rangle}{\sigma^2}\right)^{-2}\right] \cdot \frac{\kappa}{\sigma^2}\left(\mathbf{I}_n + \frac{\overline{\mathbf{w}} \cdot \overline{\mathbf{w}}^\top}{\kappa\sigma^2}\right).$$

Behaviour of $I(\mathrm{s})^{-1}$?

†: analyzed more rigorously in the article

Analysis of the Fisher information

$$I(\mathbf{s}) \approx \mathbb{E}_{\mathbf{w}}\left[\cosh\left(\frac{\langle \mathbf{w}, \mathbf{s}\rangle}{\sigma^2}\right)^{-2}\right] \cdot \frac{\kappa}{\sigma^2}\left(\mathbf{I}_n + \frac{\overline{\mathbf{w}} \cdot \overline{\mathbf{w}}^\top}{\kappa\sigma^2}\right).$$

Heuristic†: $\mathbf{w} \sim \mathcal{N}(\kappa \cdot \mathbf{s}, \sigma\sqrt{\kappa})$

s sparse, "centered"
expect $|\langle \mathbf{w}, \mathbf{s}\rangle| = \kappa\|\mathbf{s}\|^2 + \alpha \cdot \sigma\sqrt{\kappa}$
(for small α)

$\Rightarrow \cosh(\langle \mathbf{w}, \mathbf{s}\rangle/\sigma^2) \approx 1$

$\overline{\mathbf{w}} \cdot \overline{\mathbf{w}}^\top$ rank 1, eigenvalue $\kappa^2\|\mathbf{s}\|^2$
for BLISS-*, $\kappa^2\|\mathbf{s}\|^2 \ll \kappa\sigma^2$

$\Rightarrow I(\mathbf{s})$ invertible

The Fisher Information is essentially scalar:

$$I(\mathbf{s})^{-1} \approx \frac{\sigma^2}{\kappa}\mathbf{I}_n.$$

†: analyzed more rigorously in the article.

Analysis of the Fisher information

$$I(\mathbf{s}) \approx \mathbb{E}_{\mathbf{w}}\left[\cosh\left(\frac{\langle \mathbf{w}, \mathbf{s}\rangle}{\sigma^2}\right)^{-2}\right] \cdot \frac{\kappa}{\sigma^2}\left(\mathbf{I}_n + \frac{\overline{\mathbf{w}} \cdot \overline{\mathbf{w}}^\top}{\kappa\sigma^2}\right).$$

Heuristic†: $\mathbf{w} \sim \mathcal{N}(\kappa \cdot \mathbf{s}, \sigma\sqrt{\kappa})$

$\mathbf{s}$ sparse, "centered"
expect $|\langle \mathbf{w}, \mathbf{s}\rangle| = \kappa\|\mathbf{s}\|^2 + \alpha \cdot \sigma\sqrt{\kappa}$
(for small α)

$\Rightarrow \cosh(\langle \mathbf{w}, \mathbf{s}\rangle/\sigma^2) \approx 1$

$\overline{\mathbf{w}} \cdot \overline{\mathbf{w}}^\top$ rank 1, eigenvalue $\kappa^2\|\mathbf{s}\|^2$
for BLISS-*, $\kappa^2\|\mathbf{s}\|^2 \ll \kappa\sigma^2$

$\Rightarrow I(\mathbf{s})$ invertible

The Fisher Information is essentially scalar:

$$I(\mathbf{s})^{-1} \approx \frac{\sigma^2}{\kappa}\mathbf{I}_n.$$

†: analyzed more rigorously in the article.

How many traces are needed?

The MLE satisfies $\sqrt{m}(\hat{\mathbf{s}}_m - \mathbf{s}) \sim \mathcal{N}(\mathbf{0}, \frac{\sigma^2}{\kappa})$

$\mathbf{s} \in \mathbb{Z}^n$: we want $\|\hat{\mathbf{s}}_m - \mathbf{s}\|_\infty \leq \frac{1}{2}$

$\Rightarrow$ Use the Gaussian tail bound.

Conclusion: When $m \geq 16 \log(2n) \frac{\sigma^2}{\kappa}$, except with proba. $\leq \frac{1}{2n}$, we have $\lceil \hat{\mathbf{s}}_m \rfloor = \mathbf{s}$.

Algorithmic aspect of the attack

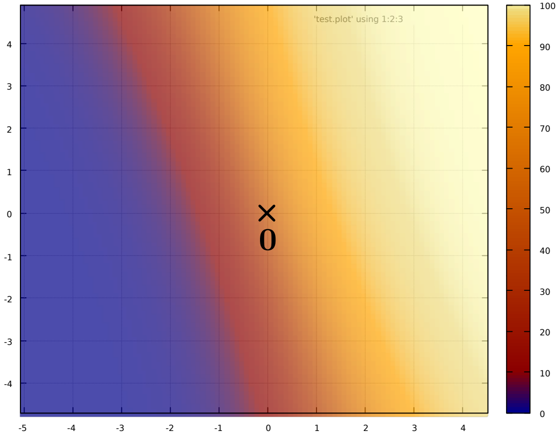
LEAK.Sign gives $(b_i, \mathbf{z}_i, \mathbf{c}_i)_{i\in[m]}$. Let $\mathbf{w}_i := (-1)^{b_i}\mathbf{z}_i\mathbf{c}_i^\star$

Goal: maximize

$$\ell(\mathbf{s}) = -\sum_{i\in[m]} \log\left(1 + \exp(-\frac{2\langle \mathbf{w}_i, \mathbf{s}\rangle}{\sigma^2})\right)$$

Technique:

Gradient descent



Algorithmic aspect of the attack

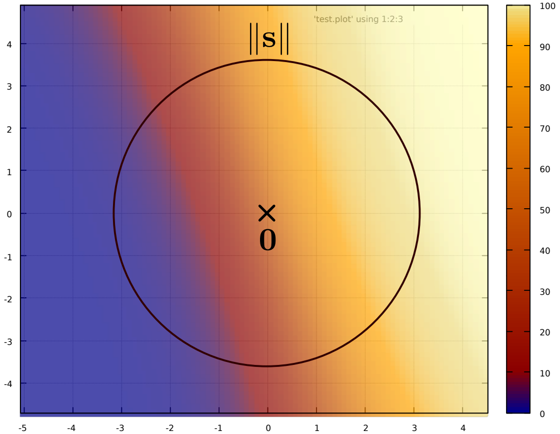
LEAK.Sign gives $(b_i, \mathbf{z}_i, \mathbf{c}_i)_{i\in[m]}$. Let $\mathbf{w}_i := (-1)^{b_i}\mathbf{z}_i\mathbf{c}_i^\star$

Goal: maximize

$$\ell(\mathbf{s}) = -\sum_{i\in[m]} \log\left(1 + \exp(-\tfrac{2\langle \mathbf{w}_i, \mathbf{s}\rangle}{\sigma^2})\right)$$

Technique:

Gradient descent



Algorithmic aspect of the attack

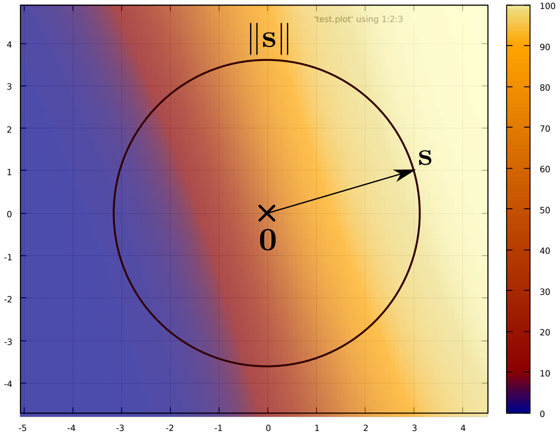
LEAK.Sign gives $(b_i, \mathbf{z}_i, \mathbf{c}_i)_{i\in[m]}$. Let $\mathbf{w}_i := (-1)^{b_i}\mathbf{z}_i\mathbf{c}_i^\star$

Goal: maximize

$$\ell(\mathbf{s}) = -\sum_{i\in[m]} \log\left(1 + \exp(-\tfrac{2\langle \mathbf{w}_i,\mathbf{s}\rangle}{\sigma^2})\right)$$

Technique:

Gradient descent



Practical results

Table: Results of our experiments.

BLISS–	I	II	III	IV
Theoretical m for success: $16\log(2n)\sigma^2/\kappa$	223,000	55,000	231,000	209,000
Experimental m for full recovery (LQ)	120,000	60,000	160,000	170,000
Experimental m for full recovery (median)	130,000	70,000	180,000	200,000
Experimental m for full recovery (UQ)	150,000	80,000	200,000	230,000
Experimental m for n'/n recovery (LQ)	70,000	40,000	90,000	110,000
Experimental m for n'/n recovery (median)	70,000	40,000	100,000	110,000
Experimental m for n'/n recovery (UQ)	80,000	40,000	110,000	120,000

Code: `https://github.com/awallet/OneBitBliss`

Conclusion

MAKE THINGS CONSTANT-TIME.

(thanks!)