One Bit to Rule Them All - Imperfect Randomness Harms Lattice Signatures

Simon Damm, Nicolai Kraus, Alexander May, Julian Nowakowski, Jonas Thietke

Ruhr University Bochum - PKC 2025

Fiat-Shamir with Aborts Signature Schemes (like ML-DSA)

Key Recovery from Public Key. Secret key $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}^n$. Public key gives equations

$$t_i = \langle \mathbf{a}, \mathbf{s}_1 \rangle + s_{2,i} \bmod q,$$

where $\mathbf{a} \in \mathbb{Z}_q^n$. Solve hard problem Learning With Errors (LWE) to recover secret key.

Key Recovery from Signatures. Challenge $\mathbf{c} \in \mathbb{Z}^n$, randomness $y \in \mathbb{Z}$. Signature

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

No reduction modulo q! In general, easy problem (Integer LWE) but rejection sampling ensures zero-knowledge.

Fiat-Shamir with Aborts Signature Schemes (like ML-DSA)

Key Recovery from Public Key. Secret key $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}^n$. Public key gives equations

$$t_i = \langle \mathbf{a}, \mathbf{s}_1 \rangle + s_{2,i} \bmod q,$$

where $\mathbf{a} \in \mathbb{Z}_q^n$. Solve hard problem Learning With Errors (LWE) to recover secret key.

Key Recovery from Signatures. Challenge $\mathbf{c} \in \mathbb{Z}^n$, randomness $y \in \mathbb{Z}$. Signature

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

No reduction modulo q! In general, easy problem (Integer LWE) but rejection sampling ensures zero-knowledge.

Fiat-Shamir with Aborts Signature Schemes (like ML-DSA)

Key Recovery from Public Key. Secret key $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}^n$. Public key gives equations

$$t_i = \langle \mathbf{a}, \mathbf{s}_1 \rangle + s_{2,i} \bmod q,$$

where $\mathbf{a} \in \mathbb{Z}_q^n$. Solve hard problem Learning With Errors (LWE) to recover secret key.

Key Recovery from Signatures. Challenge $\mathbf{c} \in \mathbb{Z}^n$, randomness $y \in \mathbb{Z}$. Signature

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

No reduction modulo q! In general, easy problem (Integer LWE) but rejection sampling ensures zero-knowledge.

What about side-channel attacks?

Breaking Zero-knowledge via Leak Bit: The Attack of [LZS$^+$20]

Attack setting:

- Assume an oracle that gives signatures

$$z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y, \text{ with } y \in [\pm 2^{17}],$$

with a leak bit y_j at index $j \geq 6$, where $y = (y_0, y_1, \ldots, y_{17}) \in \{0,1\}^{18}$.

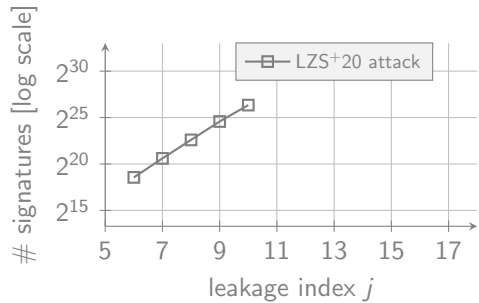
The attack:

1. Extract Integer LWE samples

$$\overline{z} = \langle \mathbf{c}, \mathbf{s}_1 \rangle + \overline{y} \text{ with } \overline{y} \in [\pm 2^j].$$

2. Key recovery via linear regression and rounding [BDE$^+$18].

Number of signatures for key recovery:



▷ Attack is infeasible for large j *due to space requirement*! How to improve?

Breaking Zero-knowledge via Leak Bit: The Attack of [LZS$^+$20]

Attack setting:

- Assume an oracle that gives signatures

$$z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y, \text{ with } y \in [\pm 2^{17}],$$

with a leak bit y_j at index $j \geq 6$, where $y = (y_0, y_1, \ldots, y_{17}) \in \{0,1\}^{18}$.

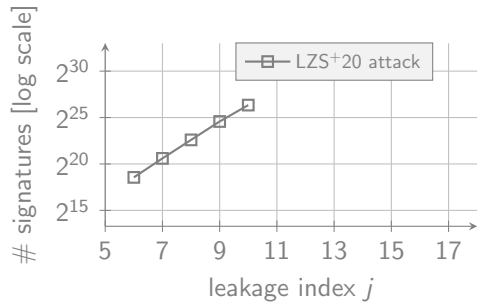
The attack:

1. Extract Integer LWE samples

$$\overline{z} = \langle \mathbf{c}, \mathbf{s}_1 \rangle + \overline{y} \text{ with } \overline{y} \in [\pm 2^j].$$

2. Key recovery via linear regression and rounding [BDE$^+$18].

Number of signatures for key recovery:



▷ Attack is infeasible for large j *due to space requirement*! How to improve?

Breaking Zero-knowledge via Leak Bit: The Attack of [LZS+20]

Attack setting:

- Assume an oracle that gives signatures

$$z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y, \text{ with } y \in [\pm 2^{17}],$$

with a leak bit y_j at index $j \geq 6$, where $y = (y_0, y_1, \ldots, y_{17}) \in \{0,1\}^{18}$.

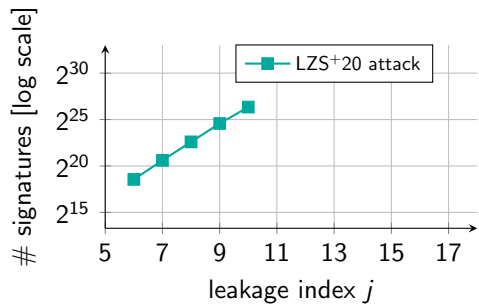
The attack:

1. Extract Integer LWE samples

$$\overline{z} = \langle \mathbf{c}, \mathbf{s}_1 \rangle + \overline{y} \text{ with } \overline{y} \in [\pm 2^j].$$

2. Key recovery via linear regression and rounding [BDE+18].

Number of signatures for key recovery:



- Attack is infeasible for large j *due to space requirement*! How to improve?

Breaking Zero-knowledge via Leak Bit: The Attack of [LZS+20]

Attack setting:

- Assume an oracle that gives signatures

$$z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y, \text{ with } y \in [\pm 2^{17}],$$

with a leak bit y_j at index $j \geq 6$, where $y = (y_0, y_1, \ldots, y_{17}) \in \{0,1\}^{18}$.

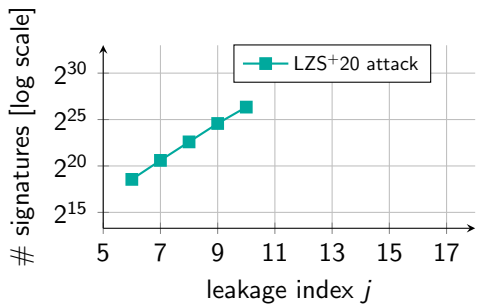
The attack:

1. Extract Integer LWE samples

$$\overline{z} = \langle \mathbf{c}, \mathbf{s}_1 \rangle + \overline{y} \text{ with } \overline{y} \in [\pm 2^j].$$

2. Key recovery via linear regression and rounding [BDE+18].

Number of signatures for key recovery:



- Attack is infeasible for large j *due to space requirement*! How to improve?

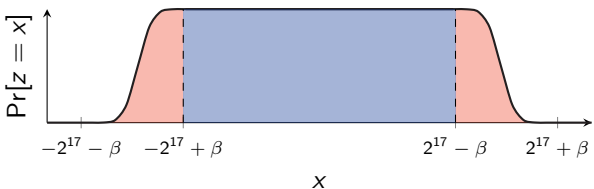
Leakage Model: Leak one bit y_j per signature. Leakage index j.

Our Work: Enable the Attack for Higher-Order Leakage Indices

1. **Improved sample extraction & analysis**
2. **Sample extraction independent of leakage index j**
3. **Accurate sample number prediction**

Gain an Intuition - The Necessity of Rejecting

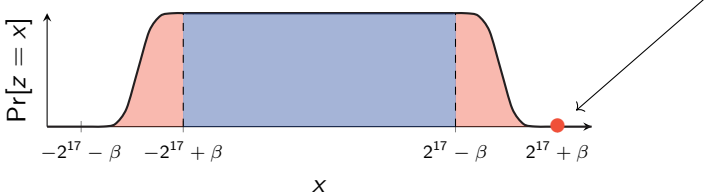
- Let $z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y$ with $|\langle \mathbf{c}, \mathbf{s}_1 \rangle| \leq \beta$ and $y \in [\pm 2^{17}]$.
- Notice that the equation holds over $\mathbb{Z}$.



Gain an Intuition - The Necessity of Rejecting

- Let $z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y$ with $|\langle \mathbf{c}, \mathbf{s}_1 \rangle| \leq \beta$ and $y \in [\pm 2^{17}]$.
- Notice that the equation holds over $\mathbb{Z}$.

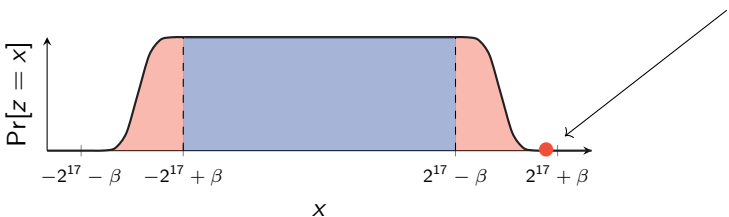
$$z = 2^{17} + \beta \implies \langle \mathbf{c}, \mathbf{s}_1 \rangle = \beta$$



Gain an Intuition - The Necessity of Rejecting

- Let $z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y$ with $|\langle \mathbf{c}, \mathbf{s}_1 \rangle| \leq \beta$ and $y \in [\pm 2^{17}]$.
- Notice that the equation holds over $\mathbb{Z}$.

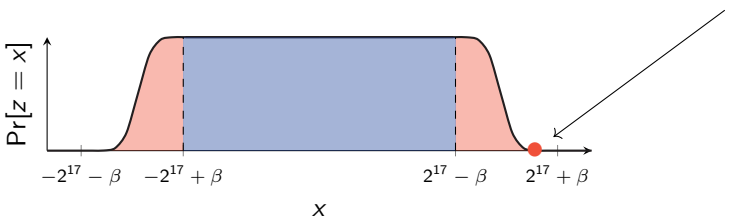
$$z = 2^{17} + \beta - 1 \implies \langle \mathbf{c}, \mathbf{s}_1 \rangle \geq \beta - 1$$



Gain an Intuition - The Necessity of Rejecting

- Let $z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y$ with $|\langle \mathbf{c}, \mathbf{s}_1 \rangle| \leq \beta$ and $y \in [\pm 2^{17}]$.
- Notice that the equation holds over $\mathbb{Z}$.

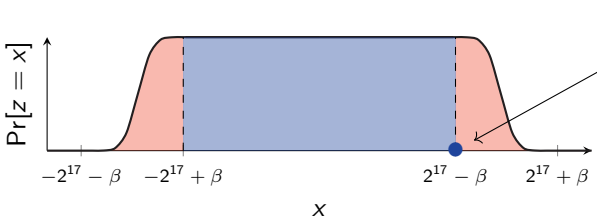
$$z = 2^{17} + \beta - 2 \implies \langle \mathbf{c}, \mathbf{s}_1 \rangle \geq \beta - 2$$



Gain an Intuition - The Necessity of Rejecting

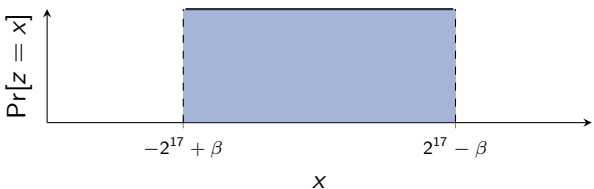
- Let $z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y$ with $|\langle \mathbf{c}, \mathbf{s}_1 \rangle| \leq \beta$ and $y \in [\pm 2^{17}]$.
- Notice that the equation holds over $\mathbb{Z}$.

$$z = 2^{17} - \beta \implies \langle \mathbf{c}, \mathbf{s}_1 \rangle \geq -\beta$$



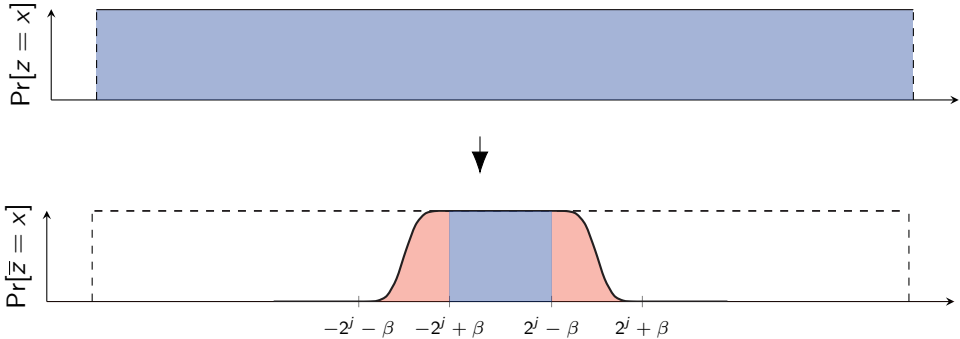
Gain an Intuition - The Necessity of Rejecting

- Let $z = \langle \mathbf{c}, \mathbf{s}_1 \rangle + y$ with $|\langle \mathbf{c}, \mathbf{s}_1 \rangle| \leq \beta$ and $y \in [\pm 2^{17}]$.
- Notice that the equation holds over $\mathbb{Z}$.



Rejecting $|z| > 2^{17} - \beta$ ensures zero-knowledge. Notice that the threshold for zero-knowledge depends on the range of y.

Analysis - What Causes the Space Requirement?

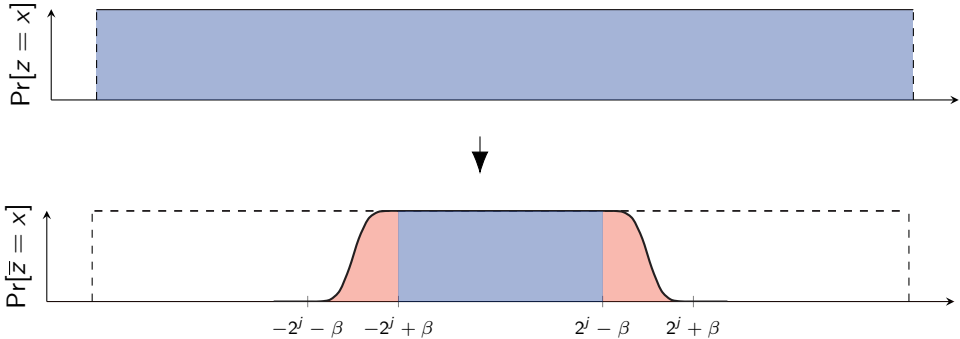


The sample extraction yields $\overline{z} = \langle \mathbf{c}, \mathbf{s_1} \rangle + \overline{y}$ with $\overline{y} \in [\pm 2^j]$.

Essentially, back to situation before rejection sampling. Back to the easy problem.

But for every increment of j, the number of zero-knowledge samples doubles.

Analysis - What Causes the Space Requirement?

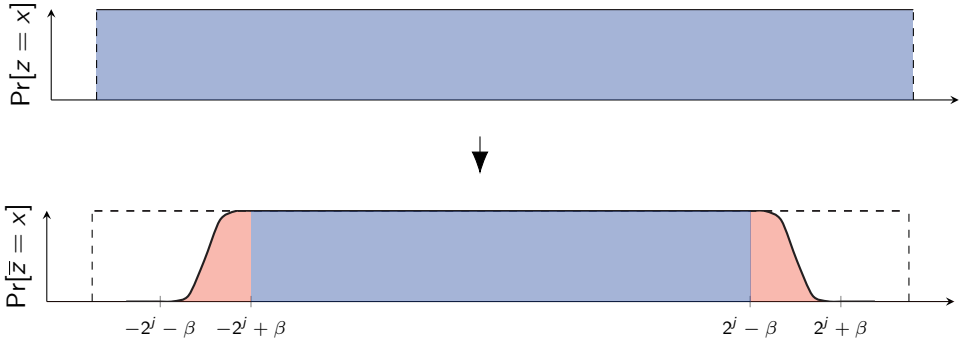


The sample extraction yields $\overline{z} = \langle \mathbf{c}, \mathbf{s_1} \rangle + \overline{y}$ with $\overline{y} \in [\pm 2^j]$.

Essentially, back to situation before rejection sampling. Back to the easy problem.

But for every increment of j, the number of zero-knowledge samples doubles.

Analysis - What Causes the Space Requirement?

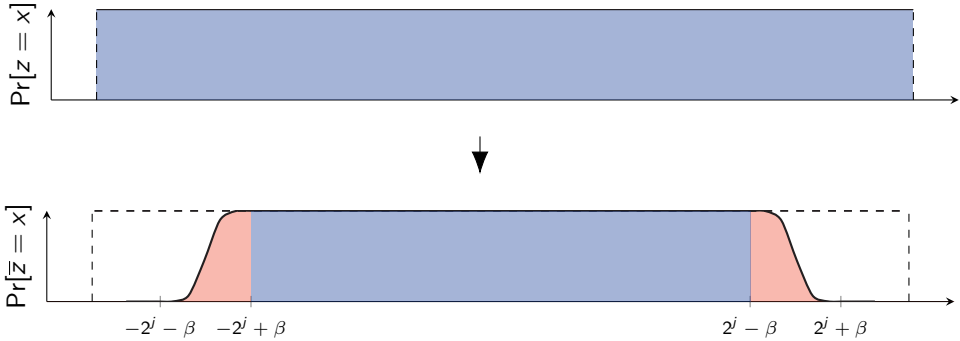


The sample extraction yields $\overline{z} = \langle \mathbf{c}, \mathbf{s_1} \rangle + \overline{y}$ with $\overline{y} \in [\pm 2^j]$.

Essentially, back to situation before rejection sampling. Back to the easy problem.

But for every increment of j, the number of zero-knowledge samples doubles.

Analysis - What Causes the Space Requirement?



The sample extraction yields $\overline{z} = \langle \mathbf{c}, \mathbf{s_1} \rangle + \overline{y}$ with $\overline{y} \in [\pm 2^j]$.

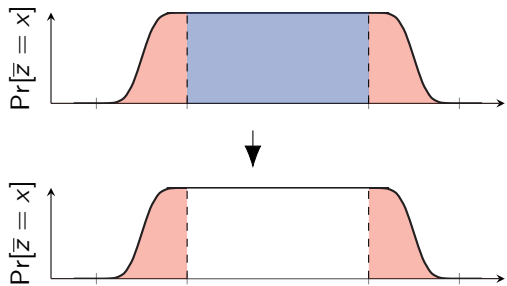
Essentially, back to situation before rejection sampling. Back to the easy problem.

But for every increment of j, the number of zero-knowledge samples doubles.

To fix the space issue, get rid of the zero-knowledge samples!

How to Achieve Independence of the Leakage Index j & Reduce the Error

1. We use only *informative* samples from the tails. We discard zero-knowledge samples.



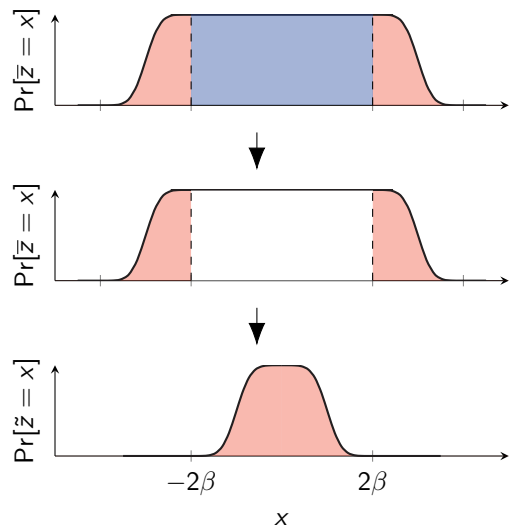
How to Achieve Independence of the Leakage Index j & Reduce the Error

1. We use only *informative* samples from the tails. We discard zero-knowledge samples.

2. We transform *informative* samples resulting in

$$\tilde{z} = \langle \mathbf{c}, \mathbf{x} \rangle + \tilde{y} \text{ with } \tilde{y} \in [\pm\beta],$$

where β depends on the ML-DSA parameter set.



Putting Things Together: Our Attack

Algorithm Secret Key Recovery

Input: Oracle for signatures $(z, \mathbf{c}, y_j)$ with $z = \langle \mathbf{c}, \mathbf{x} \rangle + y$ and leak bit y_j

repeat
  repeat
    Obtain $(z, \mathbf{c}, y_j)$.
    Compute [LZS+20] extraction.
  until sample is *informative*
  Apply j-independence transformation.
until sufficiently many samples collected.
Compute $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ via linear regression.

Output: Secret key $\mathbf{s}_1 = \lfloor \hat{\mathbf{s}}_1 \rceil$.

Putting Things Together: Our Attack

Algorithm Secret Key Recovery

Input: Oracle for signatures $(z, \mathbf{c}, y_j)$ with $z = \langle \mathbf{c}, \mathbf{x} \rangle + y$ and leak bit y_j

1: **repeat**
2: **repeat**
3: Obtain $(z, \mathbf{c}, y_j)$.
4: Compute [LZS+20] extraction.
5: **until** sample is *informative*
6: Apply j-independence transformation.
7: **until** sufficiently many samples collected.
8: Compute $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ via linear regression.

Output: Secret key $\mathbf{s}_1 = \lfloor \hat{\mathbf{s}}_1 \rceil$.

Putting Things Together: Our Attack

Algorithm Secret Key Recovery

Input: Oracle for signatures $(z, \mathbf{c}, y_j)$ with $z = \langle \mathbf{c}, \mathbf{x} \rangle + y$ and leak bit y_j

repeat
  repeat
    Obtain $(z, \mathbf{c}, y_j)$.
    Compute [LZS+20] extraction.
  until sample is *informative*
  Apply j-independence transformation.
until sufficiently many samples collected.
Compute $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ via linear regression.

Output: Secret key $\mathbf{s}_1 = \lfloor \hat{\mathbf{s}}_1 \rceil$.

Putting Things Together: Our Attack

Algorithm Secret Key Recovery

Input: Oracle for signatures $(z, \mathbf{c}, y_j)$ with $z = \langle \mathbf{c}, \mathbf{x} \rangle + y$ and leak bit y_j

1: **repeat**
2: **repeat**
3: Obtain $(z, \mathbf{c}, y_j)$.
4: Compute [LZS+20] extraction.
5: **until** sample is *informative*
6: Apply j-independence transformation.
7: **until** sufficiently many samples collected.
8: Compute $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ via linear regression.

Output: Secret key $\mathbf{s}_1 = \lfloor \hat{\mathbf{s}}_1 \rceil$.

Putting Things Together: Our Attack

Algorithm Secret Key Recovery

Input: Oracle for signatures $(z, \mathbf{c}, y_j)$ with $z = \langle \mathbf{c}, \mathbf{x} \rangle + y$ and leak bit y_j

1: **repeat**
2: **repeat**
3: Obtain $(z, \mathbf{c}, y_j)$.
4: Compute [LZS+20] extraction.
5: **until** sample is *informative*
6: Apply j-independence transformation.
7: **until** sufficiently many samples collected.
8: Compute $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ via linear regression.

Output: Secret key $\mathbf{s}_1 = \lfloor \hat{\mathbf{s}}_1 \rceil$.

Putting Things Together: Our Attack

Algorithm Secret Key Recovery

Input: Oracle for signatures $(z, \mathbf{c}, y_j)$ with $z = \langle \mathbf{c}, \mathbf{x} \rangle + y$ and leak bit y_j

1: **repeat**
2: **repeat**
3: Obtain $(z, \mathbf{c}, y_j)$.
4: Compute [LZS+20] extraction.
5: **until** sample is *informative*
6: Apply j-independence transformation.
7: **until** sufficiently many samples collected.
8: Compute $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ via linear regression.

Output: Secret key $\mathbf{s}_1 = \lfloor \hat{\mathbf{s}}_1 \rceil$.

Putting Things Together: Our Attack

Algorithm Secret Key Recovery

Input: Oracle for signatures $(z, \mathbf{c}, y_j)$ with $z = \langle \mathbf{c}, \mathbf{x} \rangle + y$ and leak bit y_j

1: **repeat**
2: **repeat**
3: Obtain $(z, \mathbf{c}, y_j)$.
4: Compute [LZS+20] extraction.
5: **until** sample is *informative*
6: Apply j-independence transformation.
7: **until** sufficiently many samples collected.
8: Compute $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ via linear regression.

Output: Secret key $\mathbf{s}_1 = \lfloor \hat{\mathbf{s}}_1 \rceil$.

Putting Things Together: Our Attack

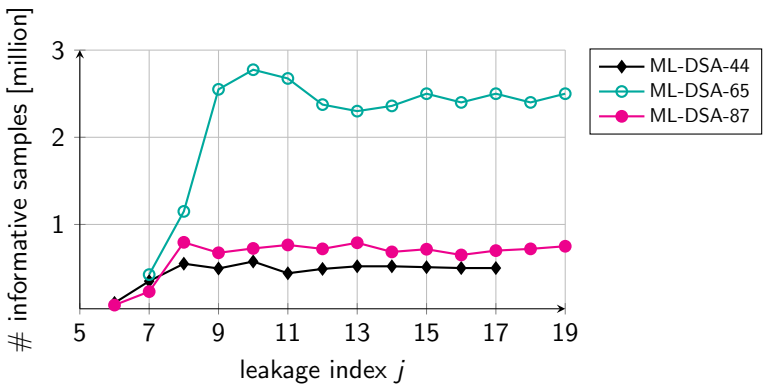
Algorithm Secret Key Recovery

Input: Oracle for signatures $(z, \mathbf{c}, y_j)$ with $z = \langle \mathbf{c}, \mathbf{x} \rangle + y$ and leak bit y_j

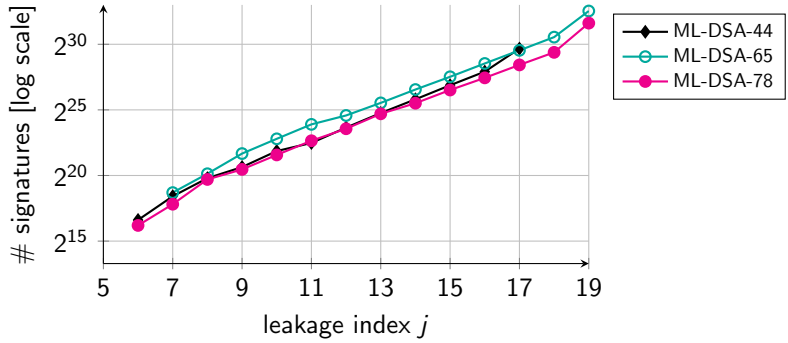
1: **repeat**
2: **repeat**
3: Obtain $(z, \mathbf{c}, y_j)$.
4: Compute [LZS$^+$20] extraction.
5: **until** sample is *informative*
6: Apply j-independence transformation.
7: **until** sufficiently many samples collected.
8: Compute $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ via linear regression.

Output: Secret key $\mathbf{s}_1 = \lfloor \hat{\mathbf{s}}_1 \rceil$.

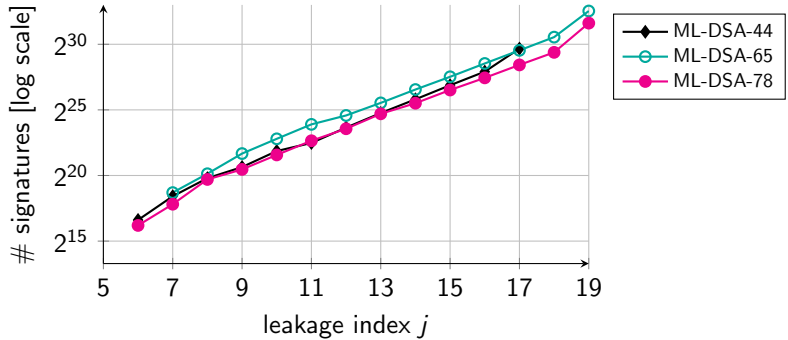
A Constant Number of Informative Samples for Key Recovery



But the Number of Signatures Doubles for Every Increment of j ...

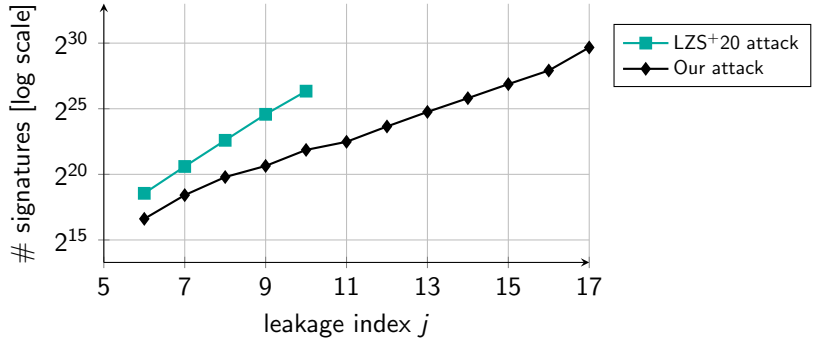


But the Number of Signatures Doubles for Every Increment of j . . .



. . . because the number of zero-knowledge samples doubles with every increment of j! And the best we can do is discard them.

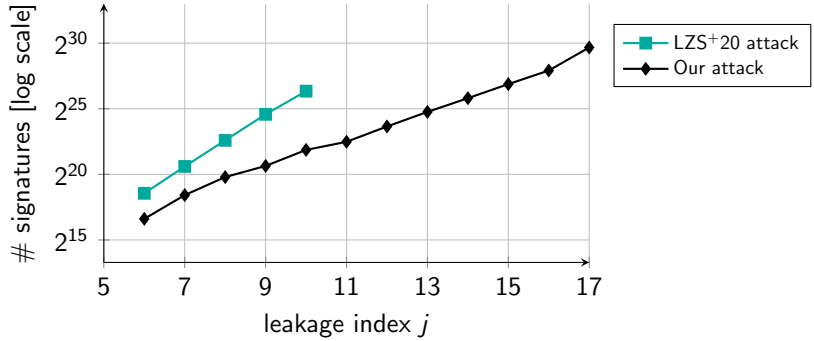
Comparison with [LZS$^+$20]: We Achieve More With Less Work



We require less signatures as we discard zero-knowledge samples and reduce the error.

By processing signatures *on the fly*, we enable the attack for higher-order leakage indices.

Comparison with [LZS$^+$20]: We Achieve More With Less Work



We require less signatures as we discard zero-knowledge samples and reduce the error.

By processing signatures *on the fly*, we enable the attack for higher-order leakage indices.

Our Attack Runtime:

ML-DSA-44	ML-DSA-65	ML-DSA-87
4s	40s	7s

Summary

- ML-DSA is susceptible to randomness leakage attacks.
- Leaking a single bit y_j at index $j \geq 6$ is sufficient for key recovery.
- The attack requires 500.000, 800.000, or 2.500.000 samples, independent of j.
- But the number of required signatures doubles for every increment of j.
- The attack is applicable to noisy side-channels as a bias is already sufficient.
- We did not cover module lattices in this talk, but the attack translates directly.

ePrint: 2025/820

Summary

- ML-DSA is susceptible to randomness leakage attacks.
- Leaking a single bit y_j at index $j \geq 6$ is sufficient for key recovery.
- The attack requires 500.000, 800.000, or 2.500.000 samples, independent of j.
- But the number of required signatures doubles for every increment of j.
- The attack is applicable to noisy side-channels as a bias is already sufficient.
- We did not cover module lattices in this talk, but the attack translates directly.

ePrint: 2025/820

Summary

- ML-DSA is susceptible to randomness leakage attacks.
- Leaking a single bit y_j at index $j \geq 6$ is sufficient for key recovery.
- The attack requires 500.000, 800.000, or 2.500.000 samples, independent of j.
- But the number of required signatures doubles for every increment of j.
- The attack is applicable to noisy side-channels as a bias is already sufficient.
- We did not cover module lattices in this talk, but the attack translates directly.

ePrint: 2025/820

Summary

- ML-DSA is susceptible to randomness leakage attacks.
- Leaking a single bit y_j at index $j \geq 6$ is sufficient for key recovery.
- The attack requires 500.000, 800.000, or 2.500.000 samples, independent of j.
- But the number of required signatures doubles for every increment of j.
- The attack is applicable to noisy side-channels as a bias is already sufficient.
- We did not cover module lattices in this talk, but the attack translates directly.

ePrint: 2025/820

Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi.
LWE without modular reduction and improved side-channel attacks against BLISS.
pages 494–524, 2018.

Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, Rui Zhang, and Jingdian Ming.
On the security of lattice-based fiat-shamir signatures in the presence of randomness leakage.
IEEE Transactions on Information Forensics and Security, 16:1868–1879, 2020.

Transferring the Attack to Module Lattices (ML-DSA)

Let $R = \mathbb{Z}[X]/(X^n + 1)$. Secret key $\mathbf{s}_1 \in R^\ell$, $\mathbf{s}_2 \in R^k$. Signing produces for a challenge c and a random mask $\mathbf{y} \in R^\ell$ a signature

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

The attack is not applicable to $\mathbf{z}$, but it does apply to the signature coefficients

$$z = \langle \mathbf{c}, \mathbf{x} \rangle + y,$$

where $\mathbf{x} \in R$ is a partial key of $\mathbf{s}_1 = (\mathbf{x}, \dots)$.

Now, only a $\frac{1}{\ell}$-fraction of the secret key can be recovered from a single bit leak. To recover the entire secret key, one must leak ℓ bits, one for each of the ℓ rings in the module.

Noisy Side-Channel - What if the Leak Bit is Incorrect?

Always incorrect leakage. Assume the oracle gives $(z, \mathbf{c}, y'_j)$ with an incorrect leak bit $y'_j = y_j \oplus 1$. Then, the attack returns $-\mathbf{s}_1$, the negation of the secret key!

Partly incorrect leakage. For probabilities $p \in (0.5, 1]$ the attack returns

$$p \cdot \mathbf{s}_1 + (1 - p) \cdot (-\mathbf{s}_1) = (2p - 1) \cdot \mathbf{s}_1.$$

Scale by $\frac{1}{2p-1}$ to recover the secret key.