LWE with Side Information: Attacks and Concrete Security Estimation

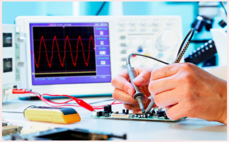
Dana Dachman-Soled[1], Léo Ducas[2]
Huijing Gong[1], Mélissa Rossi[3]

Workshop at the Simons Institute, Berkeley, CA, USA
Lattices: From Theory to Practice
April 2020

[1]University of Maryland, College Park, USA
[2]CWI, Amsterdam, The Netherlands
[3]ANSSI, Paris, France – ENS Paris, CNRS, PSL University, Paris, France

Algorithmic Cryptanalysis and Physical Cryptanalysis

Side Channel Cryptanalysis

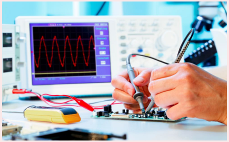


↓

↓

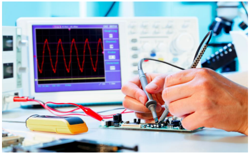
Algorithmic Cryptanalysis and Physical Cryptanalysis

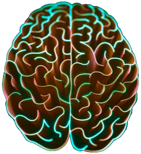
Side Channel Cryptanalysis



Algorithmic Cryptanalysis

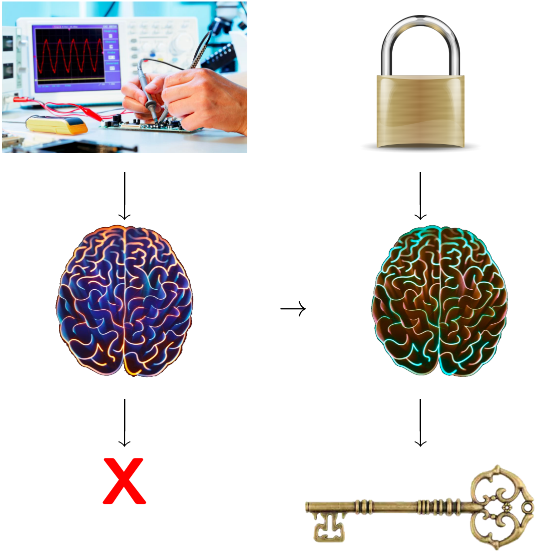
A Unified Cryptanalysis Framework for LWE-like Schemes



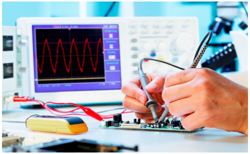


X

A Unified Cryptanalysis Framework for LWE-like Schemes



A Unified Cryptanalysis Framework for LWE-like Schemes



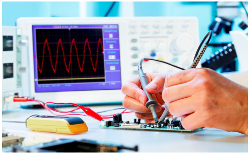
$\downarrow$ $\downarrow$

$\rightarrow$

$\downarrow$ $\downarrow$

X

A Unified Cryptanalysis Framework for LWE-like Schemes



Real-world Specs

NTRU, LAC, Round5, Lizard, NTRUPrime

Active Attacks

Decryption Failures

X

Our framework

Standard methodology (Primal Lattice Attack)

$$\text{Public Key} \xrightarrow{\begin{pmatrix}\text{ad-hoc}\\ \text{tweak}\end{pmatrix}} \mathrm{BDD}_{\Lambda,r} \xrightarrow{\text{Kannan}} \mathrm{uSVP}_{\Lambda',r'} \rightarrow \text{Lattice Reduction}$$

Our framework

Standard methodology (Primal Lattice Attack)

$$\text{Public Key} \xrightarrow{\begin{pmatrix}\text{ad-hoc}\\ \text{tweak}\end{pmatrix}} \text{BDD}_{\Lambda,r} \xrightarrow{\text{Kannan}} \text{uSVP}_{\Lambda',r'} \rightarrow \text{Lattice Reduction}$$

Our framework

$$\text{Public Key} \rightarrow \text{DBDD}_{\Lambda_0,\boldsymbol{\Sigma}_0,\boldsymbol{\mu}_0}$$

$$\downarrow \textbf{Hint}$$

$$\text{DBDD}_{\Lambda_1,\boldsymbol{\Sigma}_1,\boldsymbol{\mu}_1}$$

$$\vdots$$

$$\downarrow \textbf{Hint}$$

$$\text{DBDD}_{\Lambda_h,\boldsymbol{\Sigma}_h,\boldsymbol{\mu}_h} \xrightarrow[\substack{\text{Homogenize}\\ \text{\& Isotropize}}]{} \text{uSVP}_{\Lambda',1} \rightarrow \text{Lattice Reduction}$$

Other Contributions

Contributions

- An Sage implementation of our framework

Other Contributions

Contributions

- An Sage implementation of our framework
- A refined estimation method for the primal lattice attack

Other Contributions

Contributions

- An Sage implementation of our framework
- A refined estimation method for the primal lattice attack
- Systematize several ad-hoc massaging tricks
  - Rescaling/balancing of secret inputs
  - Optimal choice of Kannan's Embedding coefficient
  - Ignoring LWE samples

Other Contributions

Contributions

- An Sage implementation of our framework
- A refined estimation method for the primal lattice attack
- Systematize several ad-hoc massaging tricks
  - Rescaling/balancing of secret inputs
  - Optimal choice of Kannan's Embedding coefficient
  - Ignoring LWE samples

Application examples, exploiting data from:

- Uncompleted side-channel attack — 1st attack of [BFM+18]
- Decryption failures — revisiting [AVV18]
- Real-world specifications — NTRU, LAC, Round5

Overview

DBDD[1] and its concrete Hardness

Hints, and their integration into DBDD

Sage Implementation

Example Applications: from real-world to hints

[1] Distorted Bounded Distance Decoding

DBDD and its concrete Hardness

$$
\begin{array}{l}
\text{Public Key} \rightarrow \mathrm{DBDD}_{\Lambda_0,\boldsymbol{\Sigma}_0,\boldsymbol{\mu}_0} \\
\qquad\qquad\quad \big\downarrow \textbf{Hint} \\
\qquad\qquad\quad \mathrm{DBDD}_{\Lambda_1,\boldsymbol{\Sigma}_1,\boldsymbol{\mu}_1} \\
\qquad\qquad\quad \vdots \\
\qquad\qquad\quad \big\downarrow \textbf{Hint}
\end{array}
$$

$$\mathrm{DBDD}_{\Lambda_h,\boldsymbol{\Sigma}_h,\boldsymbol{\mu}_h} \xrightarrow[\text{\& Isotropize}]{\text{Homogenize}} \mathrm{uSVP}_{\Lambda',1} \rightarrow \text{Lattice Reduction}$$

DBDD: Distorted Bounded Distance Decoding

Bounded Distance Decoding

- Given a lattice $\Lambda \subset \mathbb{R}^d$, a target $\mathbf{t} \in \mathbb{R}^d$ and a radius $r > 0$
- Find the unique $\mathbf{s} \in \Lambda$ such that $\|\mathbf{s} - \mathbf{t}\| \leq r$

Distorted Bounded Distance Decoding $\quad \mathrm{DBDD}_{\Lambda,\boldsymbol{\Sigma},\boldsymbol{\mu}}$

- Given a lattice $\Lambda \subset \mathbb{R}^d$, a mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance $\boldsymbol{\Sigma} > \mathbf{0}$
- Find the unique $\mathbf{s} \in \Lambda$ such that $\|\mathbf{s} - \boldsymbol{\mu}\|_{\boldsymbol{\Sigma}} \leq d$ where

$$\|\mathbf{x}\|_{\boldsymbol{\Sigma}} := \mathbf{x}^t \cdot \boldsymbol{\Sigma}^{-1} \cdot \mathbf{x} \geq 0.$$

Intuition: **Balls** are replaced by general **Ellipsoids**.

DBDD: Distorted Bounded Distance Decoding

Bounded Distance Decoding

- Given a lattice $\Lambda \subset \mathbb{R}^d$, a target $\mathbf{t} \in \mathbb{R}^d$ and a radius $r > 0$
- Find the unique $\mathbf{s} \in \Lambda$ such that $\|\mathbf{s} - \mathbf{t}\| \leq r$

Distorted Bounded Distance Decoding $\quad \mathrm{DBDD}_{\Lambda,\boldsymbol{\Sigma},\boldsymbol{\mu}}$

- Given a lattice $\Lambda \subset \mathbb{R}^d$, a mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance $\boldsymbol{\Sigma} > \mathbf{0}$
- Find the unique $\mathbf{s} \in \Lambda$ such that $\|\mathbf{s} - \boldsymbol{\mu}\|_{\boldsymbol{\Sigma}} \leq d$ where

$$\|\mathbf{x}\|_{\boldsymbol{\Sigma}} := \mathbf{x}^t \cdot \boldsymbol{\Sigma}^{-1} \cdot \mathbf{x} \geq 0.$$

Intuition: **Balls** are replaced by general **Ellipsoids**.

Simplification: Gaussian $\simeq$ Uniform over an Ellipsoid.

Isotropization

Lattice Reduction algorithms are designed for balls.

$$\textbf{Isotropize} : (\Lambda, \boldsymbol{\Sigma}, \boldsymbol{\mu}) \mapsto (\sqrt{\boldsymbol{\Sigma}^{-1}} \cdot \Lambda, \mathbf{Id}, \sqrt{\boldsymbol{\Sigma}^{-1}} \cdot \boldsymbol{\mu})$$

After isotropization, the instance has the form

$$\mathrm{DBDD}_{\Lambda', \mathbf{Id}, \boldsymbol{\mu}'} = \mathrm{BDD}_{\Lambda', \sqrt{d}, \boldsymbol{\mu}'}.$$

Isotropization

Lattice Reduction algorithms are designed for balls.

$$\textbf{Isotropize} : (\Lambda, \mathbf{\Sigma}, \boldsymbol{\mu}) \mapsto (\sqrt{\mathbf{\Sigma}^{-1}} \cdot \Lambda, \mathbf{Id}, \sqrt{\mathbf{\Sigma}^{-1}} \cdot \boldsymbol{\mu})$$

After isotropization, the instance has the form

$$\mathrm{DBDD}_{\Lambda', \mathbf{Id}, \boldsymbol{\mu}'} = \mathrm{BDD}_{\Lambda', \sqrt{d}, \boldsymbol{\mu}'}.$$

Concrete hardness

- **grows** with the dimension d
- **shrink** with the volume $\mathrm{Vol}(\Lambda') = \mathrm{Vol}(\Lambda) \Big/ \sqrt{\det(\mathbf{\Sigma})}.$

Isotropization

Lattice Reduction algorithms are designed for balls.

$$\textbf{Isotropize} : (\Lambda, \boldsymbol{\Sigma}, \boldsymbol{\mu}) \mapsto (\sqrt{\boldsymbol{\Sigma}^{-1}} \cdot \Lambda, \mathbf{Id}, \sqrt{\boldsymbol{\Sigma}^{-1}} \cdot \boldsymbol{\mu})$$

After isotropization, the instance has the form

$$\mathrm{DBDD}_{\Lambda', \mathbf{Id}, \boldsymbol{\mu}'} = \mathrm{BDD}_{\Lambda', \sqrt{d}, \boldsymbol{\mu}'}.$$

Concrete hardness

- **grows** with the dimension d
- **shrink** with the volume $\mathrm{Vol}(\Lambda') = \mathrm{Vol}(\Lambda) \Big/ \sqrt{\det(\boldsymbol{\Sigma})}.$

Remark

- Generalizes ad-hoc “rescaling” NTRU, Lizard, NTRUPrime, . . .
- Trivialize the optimal choice of Kannan’s embedding coefficient

On Hardness Estimation

Our unit of security: the bikz

- Security expressed in β, the needed BKZ blocksize
- Roughly, 3 bikz $\approx$ 1 bit of security see [ACD+18]

On Hardness Estimation

Our unit of security: the **bikz**

- Security expressed in β, the needed BKZ blocksize
- Roughly, 3 bikz $\approx$ 1 bit of security see [ACD+18]

The 2015 estimates: GSA+Intersect [ADPS15, AGVW17]

$$\sqrt{\beta} \leq \delta_\beta^{2\beta-d-1} \cdot \mathrm{Vol}(\Lambda')^{1/d} \quad \text{where} \quad \delta_\beta = \ldots$$

On Hardness Estimation

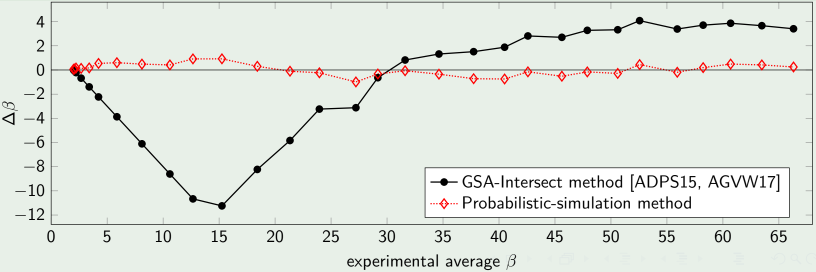
Our unit of security: the **bikz**

- Security expressed in β, the needed BKZ blocksize
- Roughly, 3 bikz $\approx$ 1 bit of security see [ACD+18]

The 2015 estimates: GSA+Intersect [ADPS15, AGVW17]

$$\sqrt{\beta} \leq \delta_\beta^{2\beta-d-1} \cdot \mathrm{Vol}(\Lambda')^{1/d} \quad \text{where} \quad \delta_\beta = \dots$$

A refined estimation: Simulation+Probabilities (somewhat new)



Hints, and their integration in DBDD

Public Key $\rightarrow$ $\mathrm{DBDD}_{\Lambda_0,\boldsymbol{\Sigma}_0,\boldsymbol{\mu}_0}$

$\Big\downarrow$ **Hint**

$\mathrm{DBDD}_{\Lambda_1,\boldsymbol{\Sigma}_1,\boldsymbol{\mu}_1}$

$\vdots$

$\Big\downarrow$ **Hint**

$\mathrm{DBDD}_{\Lambda_h,\boldsymbol{\Sigma}_h,\boldsymbol{\mu}_h} \xrightarrow[\text{\& Isotropize}]{\text{Homogenize}} \mathrm{uSVP}_{\Lambda',1} \rightarrow$ Lattice Reduction

Hints !

Four types of hints:

- Perfect hints: $\langle \mathbf{s}, \ \mathbf{v} \rangle = \ell$
- Modular hints: $\langle \mathbf{s}, \ \mathbf{v} \rangle = \ell \bmod k$
- Approximate hints: $\langle \mathbf{s}, \ \mathbf{v} \rangle \approx \ell$
- Short vector hints: $\mathbf{v} \in \Lambda$

Each hint may affect the dimension of Λ, its volume, and the covariance of $\mathbf{\Sigma}$ in predictible ways.

Hints !

Four types of hints:

- Perfect hints: $\langle \mathbf{s},\ \mathbf{v} \rangle = \ell$
- Modular hints: $\langle \mathbf{s},\ \mathbf{v} \rangle = \ell \bmod k$
- Approximate hints: $\langle \mathbf{s},\ \mathbf{v} \rangle \approx \ell$
- Short vector hints: $\mathbf{v} \in \Lambda$

Each hint may affect the dimension of Λ, its volume, and the covariance of $\mathbf{\Sigma}$ in predictible ways.

Simplification for this talk: Hints are homogeneous $\ell = 0$.

Perfect Hint $\langle \mathbf{s}, \mathbf{v} \rangle = 0$

Effect on a DBDD instance — Slice the lattice, condition the Gaussian

$$\Lambda \mapsto \Lambda \cap \mathbf{v}^{\perp}$$

$$\mathbf{\Sigma} \mapsto \mathbf{\Sigma} - \frac{(\mathbf{v}\mathbf{\Sigma})^T \mathbf{v}\mathbf{\Sigma}}{\mathbf{v}\mathbf{\Sigma}\mathbf{v}^T}$$

Effect on the hardness — **Easier**

- The lattice dimension d **decreases** by 1
- The lattice volume **increases** by a factor $\|\mathbf{v}\|$

Modular Hint $\langle \mathbf{s}, \ \mathbf{v} \rangle = 0 \bmod k$

Effect on a DBDD instance — Sparsify the lattice

$$\Lambda \mapsto \Lambda \cap \{\mathbf{x} \in \mathbb{Z}^d \mid \langle \mathbf{x}, \ \mathbf{v} \rangle = 0 \bmod k\}$$
$$\mathbf{\Sigma} \mapsto \mathbf{\Sigma} + \epsilon$$

Effect on the hardness — **Easier**

- The lattice volume **increases** by a factor k

Approximate Hint $\langle \mathbf{s}, \mathbf{v} \rangle \approx 0$

More precisely: $\langle \mathbf{s}, \mathbf{v} \rangle = e$, for e a Gaussian error of variance σ

Effect on a DBDD instance — Condition the Gaussian

$$\Lambda \mapsto \Lambda$$

$$\mathbf{\Sigma} \mapsto \mathbf{\Sigma} - \frac{(\mathbf{v}\mathbf{\Sigma})^T \mathbf{v}\mathbf{\Sigma}}{\mathbf{v}\mathbf{\Sigma}\mathbf{v}^T + \sigma^2}$$

Effect on the hardness — **Easier**

- The covariance **decreases**

Effect on a DBDD instance — Project the lattice

$$\Lambda \mapsto \Pi_{\mathbf{v}}^{\perp} \cdot \Lambda$$
$$\mathbf{\Sigma} \mapsto \Pi_{\mathbf{v}}^{\perp} \cdot \mathbf{\Sigma} \cdot (\Pi_{\mathbf{v}}^{\perp})^{T}$$

Effect on the hardness — **Trade-off**

- The dimension **decreases** by 1
- The volume **decreases** by a factor $\|\mathbf{v}\|$

Short vector hints $\mathbf{v} \in \Lambda$

Effect on a DBDD instance — Project the lattice

$$\Lambda \mapsto \Pi_{\mathbf{v}}^{\perp} \cdot \Lambda$$
$$\mathbf{\Sigma} \mapsto \Pi_{\mathbf{v}}^{\perp} \cdot \mathbf{\Sigma} \cdot (\Pi_{\mathbf{v}}^{\perp})^T$$

Effect on the hardness — **Trade-off**

- The dimension **decreases** by 1
- The volume **decreases** by a factor $\|\mathbf{v}\|$

Remark

- Typical example: q-vectors $(q, 0, 0, \ldots, 0)$, $(0, q, 0, \ldots, 0)$, ...
- Integrating a q-vectors $\Leftrightarrow$ Ignoring one LWE sample
- This generalize the usual 'dimension-volume' **trade-off**

Sage Implementation

Implementation details

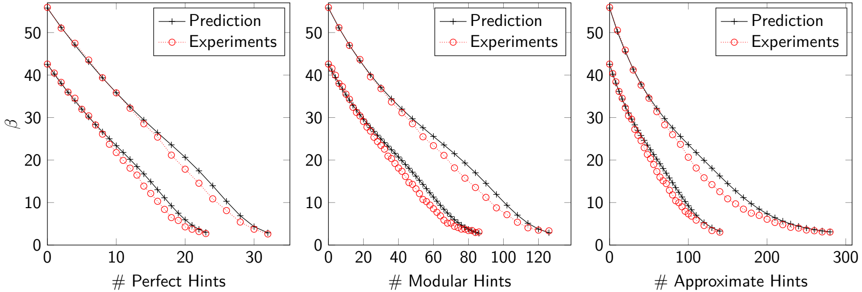
One interactive Python Class, 3 implementations

- A full-fledge version — Attack, Prediction
- A fast version — Prediction[1]
- A faster version, with restrictions — Prediction[1]

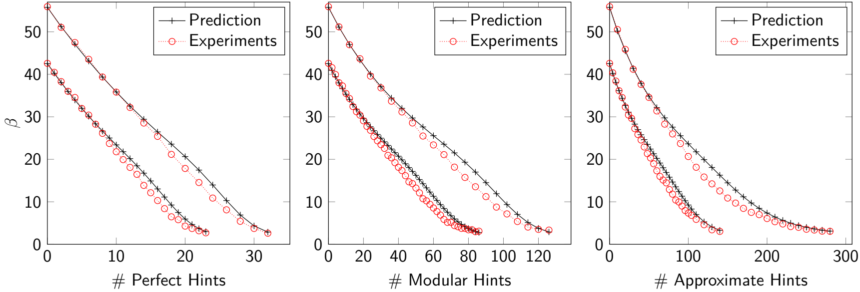
```
load("../framework/instance_gen.sage")
n = m = 70
q = 3301
D_e = D_s = build_centered_binomial_law(40)
_, _, dbdd = initialize_from_LWE_instance(DBDD, n, q, m, D_e, D_s)
dbdd.estimate_attack()

 >   Build DBDD from LWE
 > n= 70   m= 70   q=3301
 >   Attack Estimation
 > dim=141       δ=1.012362      β=45.40


v = vec([randint(0, 1) for i in range(m + n)])
dbdd.leak(v)

 > 27

dbdd.integrate_perfect_hint(v0, 27)

 > integrate perfect hint   u0 + u1 + u7 + u8 + u9 + ... = 27
 >          Worthy hint !
 > dim=140       δ=1.01252643       β=41.93
```

[1] Assuming hints are never redundant

Predictions vs. Experiments



Predictions vs. Experiments



Looks pretty[1] good !

[1] *i.e.* not perfect.

Example Applications: from real-world to hints

$$\text{Public Key} \rightarrow \mathrm{DBDD}_{\Lambda_0,\boldsymbol{\Sigma}_0,\boldsymbol{\mu}_0}$$

$$\Big\downarrow \boxed{\textbf{Hint}}$$

$$\mathrm{DBDD}_{\Lambda_1,\boldsymbol{\Sigma}_1,\boldsymbol{\mu}_1}$$

$$\vdots$$

$$\Big\downarrow \boxed{\textbf{Hint}}$$

$$\mathrm{DBDD}_{\Lambda_h,\boldsymbol{\Sigma}_h,\boldsymbol{\mu}_h} \xrightarrow[\text{\& Isotropize}]{\text{Homogenize}} \mathrm{uSVP}_{\Lambda',1} \rightarrow \text{Lattice Reduction}$$

Example 0: Hamming Weight

Typically, one does **not** get **linear** hints.
One needs a bit of creativity to extract some linear hint.

Power-analysis 101: Hamming Weight

Example 0: Hamming Weight

Typically, one does **not** get **linear** hints.
One needs a bit of creativity to extract some linear hint.

Power-analysis 101: Hamming Weight

- From the scheme design we know $\mathbf{s}_i \in \{-5, \ldots, 5\}$

Example 0: Hamming Weight

Typically, one does **not** get **linear** hints.
One needs a bit of creativity to extract some linear hint.

Power-analysis 101: Hamming Weight

- From the scheme design we know $\mathbf{s}_i \in \{-5, \ldots, 5\}$
- From **power analysis**, we learn $\mathrm{HW}(\mathbf{s}_0) = 2$

Example 0: Hamming Weight

Typically, one does **not** get **linear** hints.
One needs a bit of creativity to extract some linear hint.

Power-analysis 101: Hamming Weight

- From the scheme design we know $\mathbf{s}_i \in \{-5, \ldots, 5\}$
- From **power analysis**, we learn $\mathrm{HW}(\mathbf{s}_0) = 2$
- We deduce $\mathbf{s}_0 \in \{3, 5\}$

Example 0: Hamming Weight

Typically, one does **not** get **linear** hints.
One needs a bit of creativity to extract some linear hint.

Power-analysis 101: Hamming Weight

- From the scheme design we know $\mathbf{s}_i \in \{-5, \ldots, 5\}$
- From **power analysis**, we learn $\mathrm{HW}(\mathbf{s}_0) = 2$
- We deduce $\mathbf{s}_0 \in \{3, 5\}$
- We encode this knowledge with two hints
  - A modular hint: $\langle \mathbf{s}, (1, 0, \ldots, 0) \rangle = 1 \bmod 2$
  - A approximate hint: $\langle \mathbf{s}, (1, 0, \ldots, 0) \rangle \approx 4$, with error variance 1

Example 1: Profiling

Two single-trace attacks on FrodoKEM [BFM+18]

For the 1^{st}, data leaked, but was too weak for a key-recovery.

Exactly our intended **use-case**.

	NIST1	NIST2	CCS1	CCS2
Attack without hints (bikz)	487	708	239	448
Attack with hints (bikz)	**337**	**471**	**190**	**297**
Attack with hints & guesses (bikz)	**298**	**403**	**126**	**110**

Table: Cost of the attacks without/with hints & without/with guesses.

Example 1: Profiling

Two single-trace attacks on FrodoKEM [BFM+18]

For the 1^{st}, data leaked, but was too weak for a key-recovery.

Exactly our intended **use-case**.

	NIST1	NIST2	CCS1	CCS2
Attack without hints (bikz)	487	708	239	448
Attack with hints (bikz)	**337**	**471**	**190**	**297**
Attack with hints & guesses (bikz)	**298**	**403**	**126**	**110**

Table: Cost of the attacks without/with hints & without/with guesses.

Remark

- Hints are approximate, given in **a-posteriori** form
- Profiling data currently not used optimally (update in progress)

Example 2: Decryption Failures

A **decryption failure** occurs when the ciphertext's noise $\mathbf{w}$ is s.t.

$$\langle \mathbf{s}, \mathbf{w} \rangle \geq t := q/4.$$

Brute-forces decryption request with random $\mathbf{w}$ until it triggers.

Example 2: Decryption Failures

A **decryption failure** occurs when the ciphertext's noise $\mathbf{w}$ is s.t.

$$\langle \mathbf{s}, \mathbf{w} \rangle \geq t := q/4.$$

Brute-forces decryption request with random $\mathbf{w}$ until it triggers.

$\mathbf{w}$ is not independent of $\mathbf{s}$

- ✗ 1-dim. approx. hint

$$\langle \mathbf{s}, \mathbf{w} \rangle \approx q/4$$

Example 2: Decryption Failures

A **decryption failure** occurs when the ciphertext's noise $\mathbf{w}$ is s.t.

$$\langle \mathbf{s}, \mathbf{w} \rangle \geq t := q/4.$$

Brute-forces decryption request with random $\mathbf{w}$ until it triggers.

$\mathbf{w}$ is not independent of $\mathbf{s}$

✗ 1-dim. approx. hint

$$\langle \mathbf{s}, \mathbf{w} \rangle \approx q/4$$

✓ full-dim. approx. hint

$$\mathbf{s} \approx \frac{d\sigma^2}{t} \cdot \mathbf{w}$$

with variance $(\|\mathbf{w}\| d\sigma^2 / t)^2$

Example 2: Decryption Failures

A **decryption failure** occurs when the ciphertext's noise $\mathbf{w}$ is s.t.

$$\langle \mathbf{s}, \mathbf{w} \rangle \geq t := q/4.$$

Brute-forces decryption request with random $\mathbf{w}$ until it triggers.

$\mathbf{w}$ is not independent of $\mathbf{s}$

✗ 1-dim. approx. hint

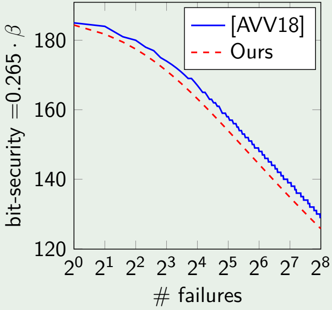
$$\langle \mathbf{s}, \mathbf{w} \rangle \approx q/4$$

✓ full-dim. approx. hint

$$\mathbf{s} \approx \frac{d\sigma^2}{t} \cdot \mathbf{w}$$

with variance $(\|\mathbf{w}\| d\sigma^2/t)^2$

Reproduction of [AVV18]



Example 3: Real-World Specifications

Several real-world schemes (NTRU, LAC, Round5) use ternary secrets, with a fixed amount of 1 and -1 $\Rightarrow$ **perfect hint**:

$$\langle \mathbf{s}, (1, 1, 1, \dots 1) \rangle = \ell$$

	LAC-128	LAC-192	LAC-256
without hints	509.03	985.64	1104.83
with 2 hints	505.94	982.74	1101.61

	R5ND_{1}KEM_0d	R5ND_{3}KEM_0d	R5ND_{5}KEM_0d
without hints	494.39	658.67	877.71
with 1 hint	492.94	657.23	876.24

	ntruhps2048509	ntruhps2048677	ntruhps4096821
without hint	372.58	515.36	617.71
with 1 hint	371.23	513.95	616.39

Remark

- A few more interesting to be said on NTRU, w.r.t. to the attack of [MS01] exploiting symmetries (update in progress)

That's it

Thanks for code-sharing, pertinent comments, and valuable feedback

- Martin Albrecht
- Jan-Pieter D'Anvers
- Thibauld Feneuil
- Henri Gilbert
- Marco Martinoli
- Ange Martinelli
- Thomas Prest
- John Schanck

Questions ?