Module 8: Concrete Security

Day 2

Hyeongmin Choe

Seoul National University

July 17, 2024

Module 8: Concrete Security, Day 2

- Quick Review
  - Day 1 tutorial
  - Smaug security estimation
- Falcon Security
  - Analysis in Falcon for NTRU attacks
  - Estimation with Lattice estimator
- Dilithium Security
  - Reductions lattice problems

Quick Review

- Updated Sagemath notebook!
  - Download it from github.com/hmchoe0528/PQC_training:

```
git clone https://github.com/hmchoe0528/PQC_training.git
```

  - Include:
    - Lattice-estimator
    - Security-estimates from Crystals-Kyber/Dilithium
    - ModifiedCBD-related stuffs
    - Try PQC_training/module8/Module8.ipynb
  - The repository is currently public, but may not be maintained after this PQC lectures.

Smaug Security

from Day 1

Note, we will take a look on Smaug v4.0, which is not yet public ☺

Smaug Security

$$LWE_{nk,\, nk,\, q,\, SparseTernary(nk,\, h/2,\, h/2),\, dGaussian(\sigma)}$$

&

$$LWE_{nk,\, n(k+1),\, q,\, mCBD(numCBD),Uniform Mod(q/p)}$$

$$\delta = \Pr\left[\|\boldsymbol{e}^t \cdot \boldsymbol{r} + e_2 - \boldsymbol{s}^t \cdot \boldsymbol{e}_1\|_\infty \geq \frac{q}{4}\right]$$

- MLWE error $\boldsymbol{e}$ ~ dGaussian(sigma)
- MLWR ModSwitch errors:
  - $\boldsymbol{e}_1$ ~ UniformMod(q/p)
  - e_2 ~ UniformMod(q/p')
- Secret vectors:
  - $\boldsymbol{s}$ ~ SparseTernary(nk, h/2, h/2)
  - $\boldsymbol{r}$ ~ modifiedCBD(numCBD)

Falcon Security

Falcon Security

- **Key Recovery** [Falcon R3]
  - For $\lambda = (2n - B)$-th GS norm of
    $$\mathcal{L} = \mathrm{span}\left(\begin{bmatrix} q & h \\ 0 & 1 \end{bmatrix}\right) = span\left(\begin{bmatrix} g & -f \\ G & -F \end{bmatrix}\right),$$
    key can be recovered when $\sqrt{B} \cdot \sigma_{f,g} \leq \sqrt{4/3} \cdot \lambda$.
  - Minimized GS norm: $\sigma_{f,g} = 1.17 \cdot \sqrt{q/2n}$
  - BKZ for NTRU lattice:
    $$\lambda = \left(\frac{B}{2\pi e}\right)^{1-n/B} \cdot \sqrt{q}.$$

⇨ $(B/2\pi e)^{1-n/B} \cdot \sqrt{q} \geq 1.17 \cdot \sqrt{3B/4 \cdot q/2n}$

Falcon Security

- **Key Recovery** [Falcon R3]
  - If $(B/2\pi e)^{1-n/B} \cdot \sqrt{q} \geq 1.17 \cdot \sqrt{3B/4 \cdot q/2n}$, key can be recovered with B-BKZ!
    - Estimated run-time: $2^{0.292B}$ in classical Core-SVP.

- Falcon512
  - n=512
  - q=12289
  - B=458

- Falcon1024
  - n=1024
  - q=12289
  - B=936

Falcon Security

- **Signature Forgery** [Falcon R3]
  - Kannan's embedding with $K \approx \sqrt{q}$, B-BKZ succeeds on

$$\mathcal{L} = \text{span}\left(\left[\begin{array}{cc|c} q & h & H(r||m) \\ 0 & 1 & 0 \\ \hline 0 & 0 & K \end{array}\right]\right)$$

  finding many short vectors ($\leq \beta$),

  possibly of form $(c, *, K)$, if $\left(\frac{B}{2\pi e}\right)^{n/B} \cdot \sqrt{q} \leq \beta$, where β: max norm of signatures.

- Falcon512
  - β^2= 34 034 726
- Falcon1024
  - β^2= 70 265 242

Dilithium Security

Dilithium Security

- **Key Recovery**
  - Public key: $(\mathrm{A}, t = As_1 + s_2)$ in $\mathcal{R}_q$, where $(s_1, s_2) \in S_\eta^l \times S_\eta^k$
    - MLWE instance with $n,\ k,\ l,\ q,\ D_s = D_e =$ Uniform$(-\eta, \eta)$

  ⇨ LWE instance with $nk,\ nl,\ q,\ D_s = D_e =$ Uniform $(-\eta, \eta)$

- Parameters: n=256, q=8380417
  - Level 2: $k = 4, l = 4, \eta = 2$
  - Level 3: $k = 6, l = 5, \eta = 2$
  - Level 5: $k = 8, l = 7, \eta = 2$

Dilithium Security

- **Key Recovery**
  - Public key: $(\mathrm{A}, t = As_1 + s_2)$ in $\mathcal{R}_q$, where $(s_1, s_2) \in S_\eta^l \times S_\eta^k$
    - MLWE instance with $n,\ k,\ l,\ q,\ D_s = D_e =$ Uniform in $[-\eta, \eta]$

  ⇨ LWE instance with $nk,\ nl,\ q,\ D_s = D_e =$ Uniform in $[-\eta, \eta]$

- Parameters: $n = 256, q = 8380417 = 2^{23} - 2^{13} + 1$
  - Level 2: $k = 4, l = 4, \eta = 2$
  - Level 3: $k = 6, l = 5, \eta = 2$
  - Level 5: $k = 8, l = 7, \eta = 2$

Dilithium Security

- **Signature Forgery**
  - Weak unforgeability (forgery with a new message):
    - Finding a short vector $(\boldsymbol{z}, c, \boldsymbol{v})$ with a message μ satisfying

$$H\left(\mu \parallel [\boldsymbol{A} \mid \boldsymbol{t} \mid \boldsymbol{Id}] \cdot \begin{bmatrix} \boldsymbol{z} \\ c \\ \boldsymbol{v} \end{bmatrix}\right) = c$$

⇨ SelfTargetMSIS $_{H,n,k,l+1,\, q,\, \zeta}$ where $\|(\boldsymbol{z}, c, \boldsymbol{v})\|_\infty \leq \zeta$

⇨ Security of H and MSIS $_{n,\, k,\, l+1,\, q,\, \zeta}$ where $\|(\boldsymbol{z}, c)\|_\infty \leq \zeta$

- Rejection condition: $\|\boldsymbol{z}\|_\infty \leq \gamma_1 - \beta$
- Compressed: $\|\boldsymbol{v} = \boldsymbol{u} + c\boldsymbol{t}_0\|_\infty \leq 2\gamma_2 + 1 + 2^{d-1} \cdot \tau$
  - *See page 25, Dilithium v3.1 for more detail..*

$$⇨ \zeta = \max\left(\gamma_1 - \beta, 2\gamma_2 + 1 + 2^{d-1} \cdot \tau\right)$$

Dilithium Security

- **Signature Forgery**
  - Strong unforgeability (forgery with one among the given messages):
    - **Forking Lemma**: with some *rewinding* success probability, forging can be *reprogrammed* as finding $(\boldsymbol{z}', c, \boldsymbol{v}')$ that satisfies

$$H\left(\mu \parallel [\boldsymbol{A} \mid \boldsymbol{t} \mid \boldsymbol{Id}] \cdot \begin{bmatrix} \boldsymbol{z} \\ c \\ \boldsymbol{v} \end{bmatrix}\right) = c = H\left(\mu \parallel [\boldsymbol{A} \mid \boldsymbol{t} \mid \boldsymbol{Id}] \cdot \begin{bmatrix} \boldsymbol{z}' \\ c \\ \boldsymbol{v}' \end{bmatrix}\right),$$

for given a set of valid message-signature pairs $(\mu, (\boldsymbol{z}, c, \boldsymbol{v}))$.

    - Equivalently, it is finding $\mathbf{x}' = (\boldsymbol{z}', \boldsymbol{v}') \neq \boldsymbol{x} = (\boldsymbol{z}, \boldsymbol{v})$ that satisfies

$$[\boldsymbol{A} \mid \boldsymbol{Id}] \cdot (\boldsymbol{x} - \boldsymbol{x}') = \mathbf{0}$$

⇨ MSIS $_{n,k,l,\, q,\, \zeta'}$ where $\|(\boldsymbol{z}, \boldsymbol{v})\|_\infty \leq \zeta'$

Dilithium Security

- **Signature Forgery**
  - Strong unforgeability (forgery with one among the given messages):
    - Equivalently, it is finding $\mathbf{x}' = (\boldsymbol{z}', \boldsymbol{v}') \neq \boldsymbol{x} = (\boldsymbol{z}, \boldsymbol{v})$ that satisfies

$$[\boldsymbol{A} \mid \boldsymbol{Id}] \cdot (\boldsymbol{x} - \boldsymbol{x}') = \mathbf{0}$$

⇨ MSIS $_{n,k,l,\,q,\,\zeta'}$ where $\|(\boldsymbol{x} - \boldsymbol{x}') = (\boldsymbol{z}, \boldsymbol{v}) - (\boldsymbol{z}', \boldsymbol{v}')\|_\infty \leq \zeta'$

- $\|\boldsymbol{z} - \boldsymbol{z}'\|_\infty \leq \|\boldsymbol{z}\|_\infty + \|\boldsymbol{z}'\|_\infty \leq 2(\gamma_1 - \beta)$
- $\|\boldsymbol{v} - \boldsymbol{v}' = \boldsymbol{u} - \boldsymbol{u}'\|_\infty \leq \|\boldsymbol{u}\|_\infty + \|\boldsymbol{u}'\|_\infty \leq 2(2\gamma_2 + 1)$

$$\Rightarrow \zeta' = \max(2(\gamma_1 - \beta), 4\gamma_2 + 2)$$

Dilithium Security

- **Signature Forgery**
  - Weak unforgeability (forgery with a new message):
    - SelfTargetMSIS $_{H,n,k,l+1,\,q,\,\zeta}$
  - Strong unforgeability (forgery with one among the given messages):
    - MSIS $_{n,k,l,\,q,\,\zeta'}$

$$\mathrm{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{Dilithium}}(\mathsf{A}) \le \mathrm{Adv}^{\mathsf{MLWE}}_{k,\ell,D}(\mathsf{B}) + \mathrm{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathrm{H},k,\ell+1,\zeta}(\mathsf{C}) + \mathrm{Adv}^{\mathsf{MSIS}}_{k,\ell,\zeta'}(\mathsf{D}) + 2^{-254} \quad ,$$

$$\zeta = \max\{\gamma_1 - \beta, 2\gamma_2 + 1 + 2^{d-1} \cdot \tau\},$$

$$\zeta' = \max\{2(\gamma_1 - \beta), 4\gamma_2 + 2\}.$$

Dilithium v3.1

- $\beta = \tau \cdot \eta$: ∞-norm bound for $c\boldsymbol{s}_2$
- γ_1, γ_2 : coefficient ranges for $\boldsymbol{y}$ and LowBits
- d : compression bit for public key

Dilithium Security

- **Etc. Challenge Entropy**

$\binom{n}{\tau} \cdot 2$^τ

NIST Security Level	2	3	5
Parameters			
q [modulus]	8380417	8380417	8380417
d [dropped bits from $\mathbf{t}$]	13	13	13
τ [# of ± 1's in c]	39	49	60
challenge entropy [$\log \binom{256}{\tau} + \tau$]	192	225	257
γ_1 [$\mathbf{y}$ coefficient range]	2^{17}	2^{19}	2^{19}
γ_2 [low-order rounding range]	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k, ℓ) [dimensions of $\mathbf{A}$]	$(4,4)$	$(6,5)$	$(8,7)$
η [secret key range]	2	4	2
β [$\tau \cdot \eta$]	78	196	120
ω [max. # of 1's in the hint $\mathbf{h}$]	80	55	75
Repetitions (from Eq. (5))	4.25	5.1	3.85

Dilithium v3.1

Thank You!