Lattice-based NIST Candidates

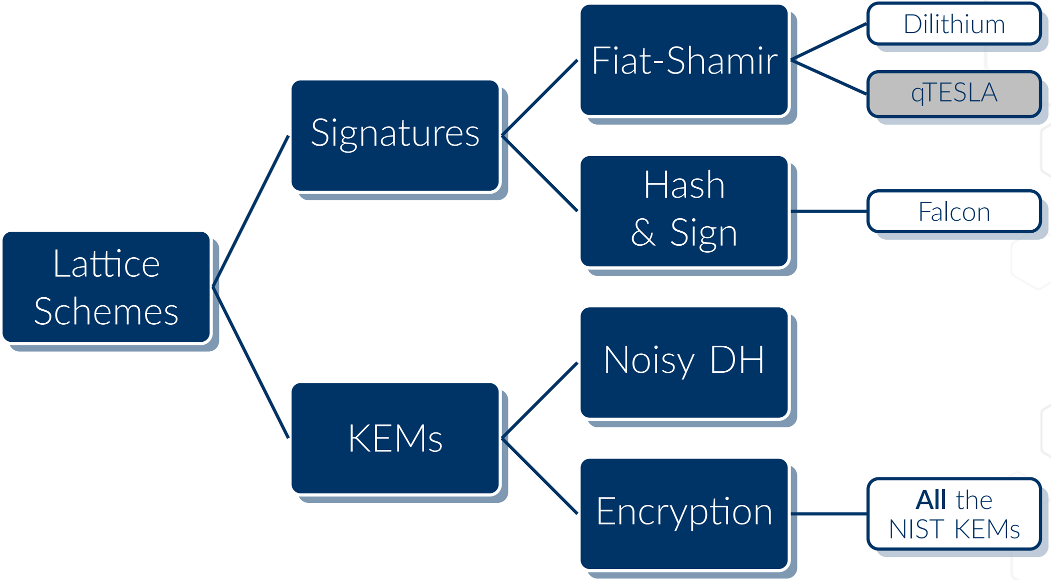
Abstractions and Ninja Tricks

Thomas Prest

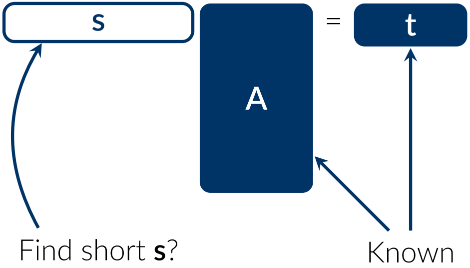
PQShield

- I Introduction
- II Key Encapsulation Mechanisms
- III Signatures
- IV Ninja Tricks

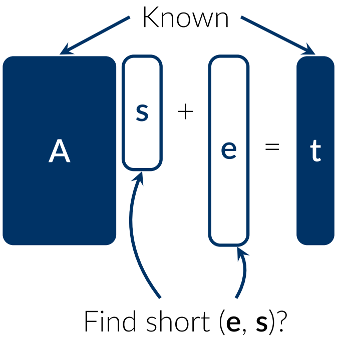
Introduction


Lattice Schemes
Signatures
KEMs
Fiat-Shamir
Hash & Sign
Noisy DH
Encryption
Dilithium
qTESLA
Falcon
All the NIST KEMs

Module-SIS



Module-LWE



NTRU

Find small f, g such that $g \cdot f^{-1} = h$ in $\mathcal{R}_q = \mathbb{Z}_q[x]/(\varphi)$

Key Encapsulation Mechanisms

Keygen($\mathbf{A} \in \mathcal{R}_q^{m\times m}$)

1. $\mathbf{S}, \mathbf{E} \leftarrow \chi_1 \times \chi_2$
2. $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
3. $\mathsf{sk} := (\mathbf{S}, \mathbf{E}), \mathsf{pk} := \mathbf{B}$

Enc($\mathtt{M}$, pk)

1. $\mathbf{R}, \mathbf{E}', \mathbf{E}'' \leftarrow \chi_3 \times \chi_4 \times \chi_5$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. $\mathbf{V} \leftarrow \mathbf{RB} + \mathbf{E}'' + \mathrm{Encode}(\mathtt{M})$
4. $\mathsf{ct} := (\mathbf{U}, \mathbf{V})$

Dec(ct, sk)

1. $\mathtt{M} \leftarrow \mathrm{Decode}(\mathbf{V} - \mathbf{US})$

Think of El Gamal, but with LWE/LWR.

Damien's excellent talk covers NTRU: https://youtu.be/yZhmKwmx48o

Keygen($\mathbf{A} \in \mathcal{R}_q^{m\times m}$)

1. $\mathbf{S}, \mathbf{E} \leftarrow \chi_1 \times \chi_2$
2. $\mathbf{B} \leftarrow \mathbf{A}\mathbf{S} + \mathbf{E}$
3. $\mathsf{sk} := (\mathbf{S}, \mathbf{E}), \mathsf{pk} := \mathbf{B}$

Enc($\mathtt{M}$, pk)

1. $\mathbf{R}, \mathbf{E}', \mathbf{E}'' \leftarrow \chi_3 \times \chi_4 \times \chi_5$
2. $\mathbf{U} \leftarrow \mathbf{R}\mathbf{A} + \mathbf{E}'$
3. $\mathbf{V} \leftarrow \mathbf{R}\mathbf{B} + \mathbf{E}'' + \mathrm{Encode}(\mathtt{M})$
4. $\mathtt{ct} := (\mathbf{U}, \mathbf{V})$

Dec(ct, sk)

1. $\mathtt{M} \leftarrow \mathrm{Decode}(\mathbf{V} - \mathbf{U}\mathbf{S})$

Underlying problem/lattice

- LWE
- Module-LWE
- Ring-LWE
- (Module-)LWR
- (Module-)Integer-LWE
- and also NTRU (not here)

Keygen$(\mathbf{A} \in \mathcal{R}_q^{m\times m})$

1. $\mathbf{S}, \mathbf{E} \leftarrow \chi_1 \times \chi_2$
2. $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
3. $\mathsf{sk} := (\mathbf{S}, \mathbf{E}), \mathsf{pk} := \mathbf{B}$

Enc$(\mathtt{M}, \mathbf{pk})$

1. $\mathbf{R}, \mathbf{E}', \mathbf{E}'' \leftarrow \chi_3 \times \chi_4 \times \chi_5$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. $\mathbf{V} \leftarrow \mathbf{RB} + \mathbf{E}'' + \mathrm{Encode}(\mathtt{M})$
4. $\mathsf{ct} := (\mathbf{U}, \mathbf{V})$

Dec$(\mathbf{ct}, \mathbf{sk})$

1. $\mathtt{M} \leftarrow \mathrm{Decode}(\mathbf{V} - \mathbf{US})$

Underlying ring $\mathcal{R}_q$

- → $\mathbb{Z}_q$, q small
- → $\mathbb{Z}_q[x]/(x^n + 1)$, q prime
- → $\mathbb{Z}_q[x]/(x^n + 1)$, q power-of-2
- → $\mathbb{Z}_q[x]/(P)$, $P = x^p - x - 1$ irreducible mod q, and q prime
- → $\mathbb{Z}_q$, q huge

Keygen($\mathbf{A} \in \mathcal{R}_q^{m\times m}$)

1. $\mathbf{S}, \mathbf{E} \leftarrow \chi_1 \times \chi_2$
2. $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
3. $\mathsf{sk} := (\mathbf{S}, \mathbf{E}), \mathsf{pk} := \mathbf{B}$

Enc($\mathtt{M}$, **pk**)

1. $\mathbf{R}, \mathbf{E}', \mathbf{E}'' \leftarrow \chi_3 \times \chi_4 \times \chi_5$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. $\mathbf{V} \leftarrow \mathbf{RB} + \mathbf{E}'' + \mathrm{Encode}(\mathtt{M})$
4. $\mathtt{ct} := (\mathbf{U}, \mathbf{V})$

Dec(**ct**, **sk**)

1. $\mathtt{M} \leftarrow \mathrm{Decode}(\mathbf{V} - \mathbf{US})$

Distributions χ_i

- → Binomial
- → Pseudo-Gaussian
- → Small uniform
- → Ternary (NTRU Prime, NTRU)
- → No error distribution (LWR)

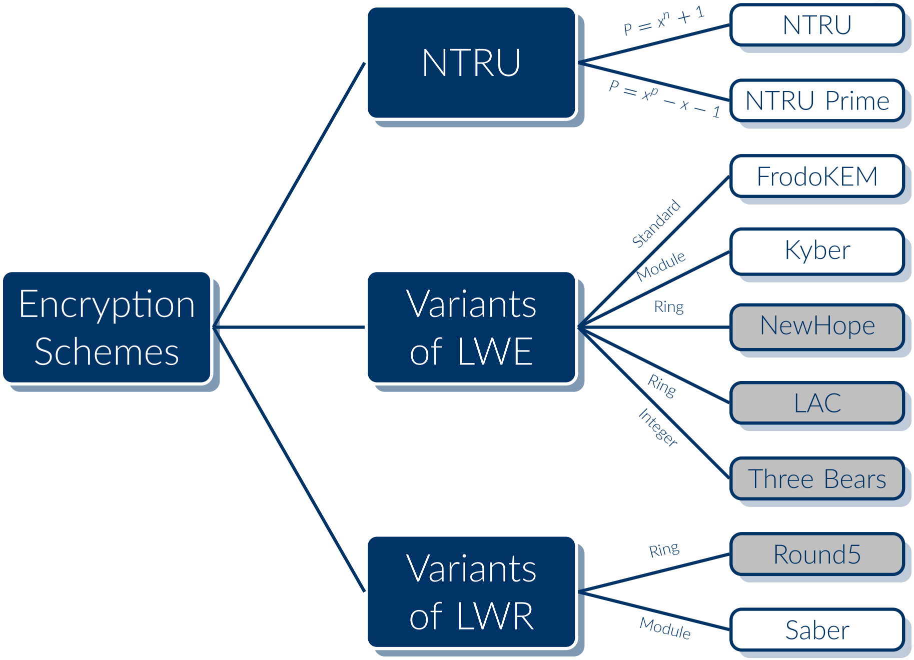
Obtaining security against active attackers

Most schemes rely on transforms in [HHK17]:

- Variants of Fujisaki-Okamoto, handle decryption failures [DRV20]
- Tight proofs in the ROM but not the QROM

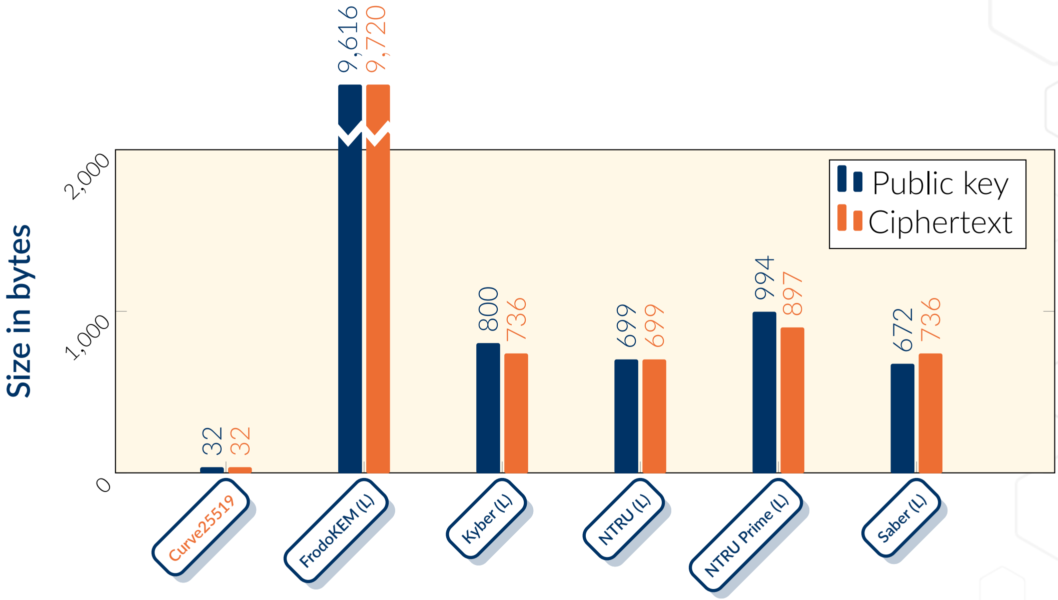
NTRU and NTRU Prime use [BP18] and [Den03] instead:

- Do not require re-encryption
- Tight proofs in the QROM (under non-standard assumptions)

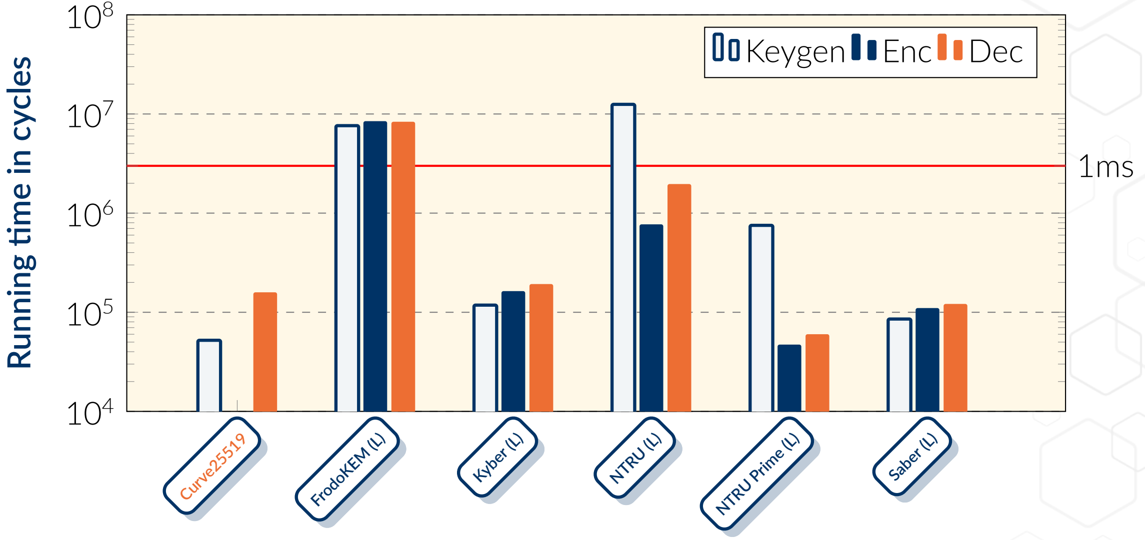


Encryption Schemes
NTRU
Variants of LWE
Variants of LWR
$P = x^n + 1$
$P = x^p - x - 1$
NTRU
NTRU Prime
Standard
Module
Ring
Ring
Integer
FrodoKEM
Kyber
NewHope
LAC
Three Bears
Ring
Module
Round5
Saber

Bandwidth cost of Level 1 KEMs

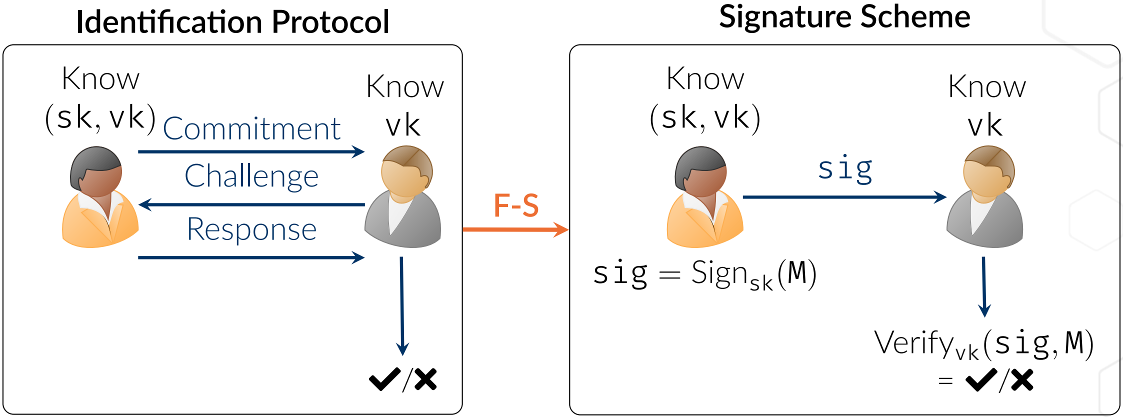


Computation Cost of Level 1 KEMs



Signatures

Fiat-Shamir Signatures



F-S refers to the Fiat-Shamir transform:

- → The challenge is now defined as $H(\text{Commitment}\|\mathtt{M})$.
- → The signature is (Commitment,Response).

Fiat-Shamir w/ discrete log

Keygen$(g \in G)$

1. $x \leftarrow \mathbb{Z}_q^\times$
2. $h \leftarrow g^x$
3. $\mathsf{sk} := x, \mathsf{pk} := h$

Sign$(\mathtt{M}, \mathbf{sk})$

1. $r \leftarrow \mathbb{Z}_q^\times$
2. $u \leftarrow g^r$
3. $c \leftarrow H(u \| \mathtt{M})$
4. $z \leftarrow r - cx$
5. $\mathtt{sig} := (c, z)$

Verify$(\mathtt{M}, \mathbf{pk})$

1. Accept if and only if $H(g^z \cdot h^c \| \mathtt{M}) = c$

3 crucial properties of ID protocol:

1. **Correctness:** *An honest prover can convince a verifier he knows* sk
2. **Soundness:** *A dishonest prover cannot convince a verifier he knows* sk
3. **(Honest Verifier) Zero-Knowledge:** *No information about* sk *is leaked*

Virtually all lattice-based Fiat-Shamir schemes transpose this blueprint to lattices, with 3 tricks:

- → Rejection sampling (a.k.a. *Fiat-Shamir with aborts*)
- → The Bai-Galbraith trick [BG14]
- → The Dilithium trick [LDK+17]

Keygen($\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$)

1. $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \chi_1 \times \chi_2$ (short)
2. $\mathbf{t} \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
3. $\mathsf{sk} := (\mathbf{s}_1, \mathbf{s}_2), \mathsf{pk} := \mathbf{t}$

Sign($\mathtt{M}$, sk)

1. $\mathbf{r}_1, \mathbf{r}_2 \leftarrow \chi_3 \times \chi_4$ (short)
2. $\mathbf{u} \leftarrow \mathbf{A}\mathbf{r}_1 + \mathbf{r}_2$
3. $\mathbf{c} \leftarrow H(\mathbf{u} \| \mathtt{M})$ (short)
4. $\mathbf{z}_1 \leftarrow \mathbf{r}_1 - \mathbf{c}\mathbf{s}_1$
5. $\mathbf{z}_2 \leftarrow \mathbf{r}_2 - \mathbf{c}\mathbf{s}_2$
6. Rej. sampling (for HVZK)
7. $\mathtt{sig} := (\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$

Verify($\mathtt{M}$, pk)

1. Accept iff $(\mathbf{z}_1, \mathbf{z}_2)$ is short and $H(\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c} \| \mathtt{M}) = \mathbf{c}$

$\mathsf{Keygen}(\mathbf{A} \in \mathcal{R}_q^{k\times\ell})$

1. $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \chi_1 \times \chi_2$ (short)
2. $\mathbf{t} \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
3. $\mathsf{sk} := (\mathbf{s}_1, \mathbf{s}_2), \mathsf{pk} := \mathbf{t}$

$\mathsf{Sign}(\mathtt{M}, \mathbf{sk})$

1. $\mathbf{r}_1, \mathbf{r}_2 \leftarrow \chi_3 \times \chi_4$ (short)
2. $\mathbf{u} \leftarrow \mathbf{A}\mathbf{r}_1 + \mathbf{r}_2$
3. $\mathbf{c} \leftarrow H(\mathbf{u} \| \mathtt{M})$ (short)
4. $\mathbf{z}_1 \leftarrow \mathbf{r}_1 - \mathbf{c}\mathbf{s}_1$
5. $\mathbf{z}_2 \leftarrow \mathbf{r}_2 - \mathbf{c}\mathbf{s}_2$
6. Rej. sampling
7. $\mathtt{sig} := (\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$

$\mathsf{Verify}(\mathtt{M}, \mathbf{pk})$

1. Accept iff $(\mathbf{z}_1, \mathbf{z}_2)$ is short and $H(\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c} \| \mathtt{M}) = \mathbf{c}$

Bai-Galbraith and Dilithium tricks:
Only care about most significant bits (MSB).

Bai-Galbraith trick: Discard commitment's LSBs.
$\Rightarrow$ Shorter signatures.

Keygen($\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$)

1. $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \chi_1 \times \chi_2$ (short)
2. $\mathbf{t} \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
3. $\mathtt{sk} := (\mathbf{s}_1, \mathbf{s}_2), \mathtt{pk} := \mathbf{t}$

Sign($\mathtt{M}$, sk)

1. $\mathbf{r} \leftarrow \chi_3$ (short)
2. $\mathbf{u} \leftarrow \mathbf{A}\mathbf{r}$
3. $\mathbf{c} \leftarrow H(\text{MSB}(\mathbf{u}) \| \mathtt{M})$ (short)
4. $\mathbf{z} \leftarrow \mathbf{r} - \mathbf{c}\mathbf{s}_1$
5. Rej. sampling
6. $\mathtt{sig} := (\mathbf{c}, \mathbf{z})$

Verify($\mathtt{M}$, pk)

1. Accept iff $\mathbf{z}$ is short and $H(\text{MSB}(\mathbf{A}\mathbf{z} - \mathbf{t}\mathbf{c}) \| \mathtt{M}) = \mathbf{c}$

Bai-Galbraith and Dilithium tricks:
Only care about most significant bits (MSB).

Bai-Galbraith trick: Discard commitment's LSBs.
$\Rightarrow$ Shorter signatures.

Keygen$(\mathbf{A} \in \mathcal{R}_q^{k\times\ell})$

1. $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \chi_1 \times \chi_2$ (short)
2. $\mathbf{t} \leftarrow \mathrm{MSB}(\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$
3. $\mathsf{sk} := \mathbf{s}$, $\mathsf{pk} := \mathbf{t}$

Sign$(\mathtt{M}, \mathsf{sk})$

1. $\mathbf{r} \leftarrow \chi_3$ (short)
2. $\mathbf{u} \leftarrow \mathbf{A}\mathbf{r}$
3. $\mathbf{c} \leftarrow H(\mathrm{MSB}(\mathbf{u}) \| \mathtt{M})$ (short)
4. $\mathbf{z} \leftarrow \mathbf{r} - \mathbf{c}\mathbf{s}_1$
5. $\mathbf{h} \leftarrow \mathrm{MSB}(\mathbf{A}\mathbf{z} - \mathbf{t}\mathbf{c}) \oplus \mathrm{MSB}(\mathbf{u})$
6. Rej. sampling + check $\mathbf{h}$ short
7. $\mathtt{sig} := (\mathbf{c}, \mathbf{z}, \mathbf{h})$

Verify$(\mathtt{M}, \mathbf{pk})$

1. Accept iff $\mathbf{z}$ short, $\mathbf{h}$ short and $H(\mathrm{MSB}(\mathbf{A}\mathbf{z} - \mathbf{t}\mathbf{c}) \oplus \mathbf{h} \| \mathtt{M}) = \mathbf{c}$

Bai-Galbraith and Dilithium tricks:
Only care about most significant bits (MSB).

Bai-Galbraith trick: Discard commitment's LSBs.
$\Rightarrow$ Shorter signatures.

Dilithium trick: Discard public key's LSBs, send hint in signature to compensate error.
$\Rightarrow$ Shorter public key, slightly larger signatures.

Keygen($\mathbf{A} \in \mathcal{R}_q^{k \times \ell}$)

1. $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \chi_1 \times \chi_2$ (short)
2. $\mathbf{t} \leftarrow \mathsf{MSB}(\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$
3. $\mathsf{sk} := \mathbf{s}, \mathsf{pk} := \mathbf{t}$

Sign($\mathtt{M}$, sk)

1. $\mathbf{r} \leftarrow \chi_3$ (short)
2. $\mathbf{u} \leftarrow \mathbf{A}\mathbf{r}$
3. $\mathbf{c} \leftarrow H(\mathsf{MSB}(\mathbf{u}) \| \mathtt{M})$ (short)
4. $\mathbf{z} \leftarrow \mathbf{r} - \mathbf{c}\mathbf{s}_1$
5. $\mathbf{h} \leftarrow \mathsf{MSB}(\mathbf{A}\mathbf{z} - \mathbf{t}\mathbf{c}) \oplus \mathsf{MSB}(\mathbf{u})$
6. Rej. sampling + check $\mathbf{h}$ short
7. $\mathtt{sig} := (\mathbf{c}, \mathbf{z}, \mathbf{h})$

Verify($\mathtt{M}$, pk)

1. Accept iff $\mathbf{z}$ short, $\mathbf{h}$ short and $H(\mathsf{MSB}(\mathbf{A}\mathbf{z} - \mathbf{t}\mathbf{c}) \oplus \mathbf{h} \| \mathtt{M}) = \mathbf{c}$

Bai-Galbraith and Dilithium tricks:
Only care about most significant bits (MSB).

Bai-Galbraith trick: Discard commitment's LSBs.
$\Rightarrow$ Shorter signatures.

Dilithium trick: Discard public key's LSBs, send hint in signature to compensate error.
$\Rightarrow$ Shorter public key, slightly larger signatures.

Keygen($\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$)

1. $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \chi_1 \times \chi_2$ (short)
2. $\mathbf{t} \leftarrow \mathsf{MSB}(\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$
3. $\mathsf{sk} := \mathbf{s}, \mathsf{pk} := \mathbf{t}$

Sign($\mathtt{M}$, sk)

1. $\mathbf{r} \leftarrow \chi_3$ (short)
2. $\mathbf{u} \leftarrow \mathbf{A}\mathbf{r}$
3. $\mathbf{c} \leftarrow H(\mathsf{MSB}(\mathbf{u})\|\mathtt{M})$ (short)
4. $\mathbf{z} \leftarrow \mathbf{r} - \mathbf{c}\mathbf{s}_1$
5. $\mathbf{h} \leftarrow \mathsf{MSB}(\mathbf{A}\mathbf{z} - \mathbf{t}\mathbf{c}) \oplus \mathsf{MSB}(\mathbf{u})$
6. Rej. sampling + check $\mathbf{h}$ short
7. $\mathtt{sig} := (\mathbf{c}, \mathbf{z}, \mathbf{h})$

Verify($\mathtt{M}$, pk)

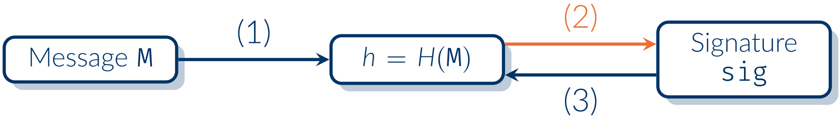
1. Accept iff $\mathbf{z}$ short, $\mathbf{h}$ short and $H(\mathsf{MSB}(\mathbf{A}\mathbf{z} - \mathbf{t}\mathbf{c}) \oplus \mathbf{h} \|\mathtt{M}) = \mathbf{c}$

In the Dilithium trick, it is **vital** to control the norm and Hamming weight of $\mathbf{h}$. Otherwise forgery is simple:

1. Sample random $\mathbf{z}$ and $\mathbf{u}$
2. $\mathbf{c} \leftarrow H(\mathsf{MSB}(\mathbf{u})\|\mathtt{M})$
3. $\mathbf{h} \leftarrow \mathsf{MSB}(\mathbf{A}\mathbf{z} - \mathbf{t}\mathbf{c}) \oplus \mathsf{MSB}(\mathbf{u})$

See qTESLA-*-s [BAA+19].

Signatures based on *Hash-then-Sign*



- **The signer** computes (1), then (2) using the signing key sk.
- **The verifier** computes (1), then (3) using the verification key pk, and checks that the results match.

In RSA signatures, (2) + (3) define a trapdoor permutation, but lattices rely on weaker notions: TPSF and average TPSF.

Trapdoor permutation $\Rightarrow$ TPSF $\Rightarrow$ Average TPSF

GPV signatures (2008)

Falcon instantiates this blueprint.

Keygen(1^λ)

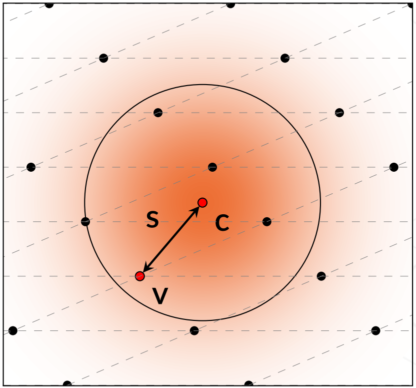
1. Gen. matrices $\mathbf{A}, \mathbf{B}$ s.t.:
   - $\mathbf{B} \cdot \mathbf{A} = 0$
   - $\mathbf{B}$ has small coefficients
2. $\mathsf{pk} := \mathbf{A}, \mathsf{sk} := \mathbf{B}$

Sign$(\mathtt{M}, \mathbf{sk} = \mathrm{B})$

1. Compute $\mathbf{c}$ such that $\mathbf{c} \cdot \mathbf{A} = H(\mathtt{M})$
2. $\mathbf{v} \leftarrow$ vector in $\mathcal{L}(\mathbf{B})$, close to $\mathbf{c}$
3. $\mathtt{sig} := \mathbf{s} = (\mathbf{c} - \mathbf{v})$

Verify$(\mathtt{M}, \mathbf{pk} = \mathrm{A}, \mathbf{\mathtt{sig}} = \mathrm{s})$

Check ($\mathbf{s}$ short) & ($\mathbf{s} \cdot \mathbf{A} = H(\mathtt{M})$)



How to compute efficiently a close vector (the second algorithm assumes we precomputed the Gram-Schmidt orthogonalization $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$).

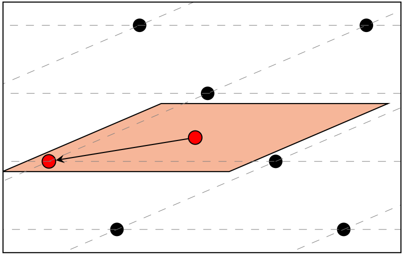
RoundOff(B, c)

1. $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
2. For $j \in \{n, \ldots, 1\}$:
   1. $z_j \leftarrow \lfloor t_j \rceil$
3. Return $\mathbf{v} := \mathbf{z} \cdot \mathbf{B}$

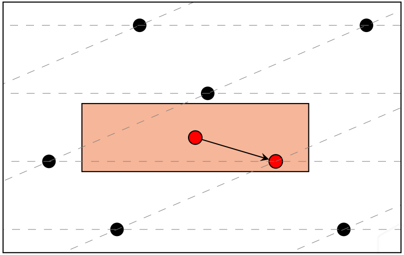
NearestPlane(B, L, c)

1. $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
2. For $j \in \{n, \ldots, 1\}$:
   1. $z_j \leftarrow \lfloor t_j + \sum_{i>j} (t_1 - z_i) L_{i,j} \rceil$
3. Return $\mathbf{v} := \mathbf{z} \cdot \mathbf{B}$

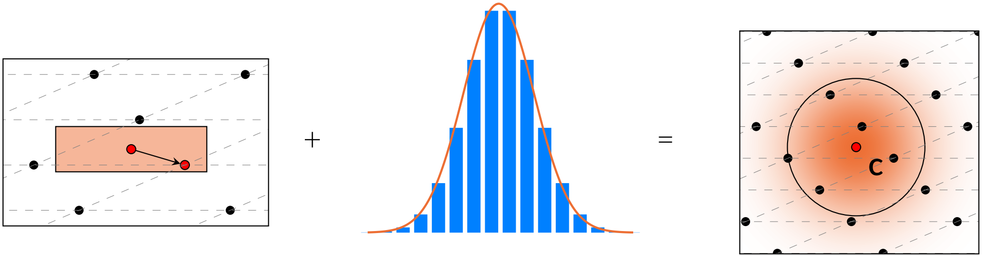
$\Downarrow$

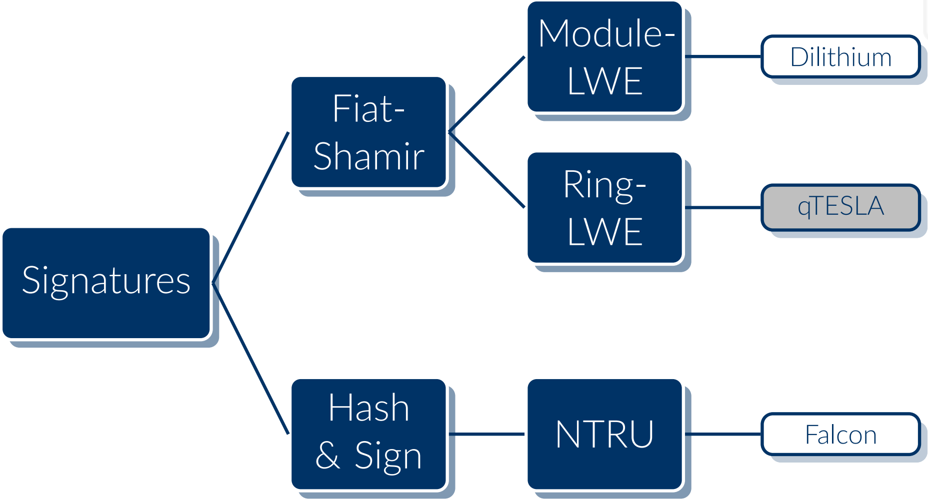


$\Downarrow$

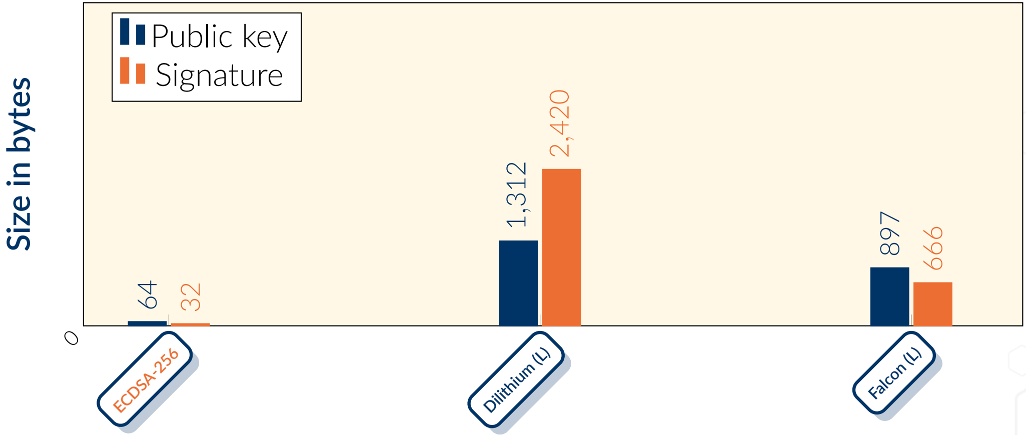


- **Problem:** When used for signing, the algorithms RoundOff and NearestPlane leak the shape of the private key **B**, leading to attacks.
- **Solution:** Replace rounding with (Gaussian) randomized rounding.

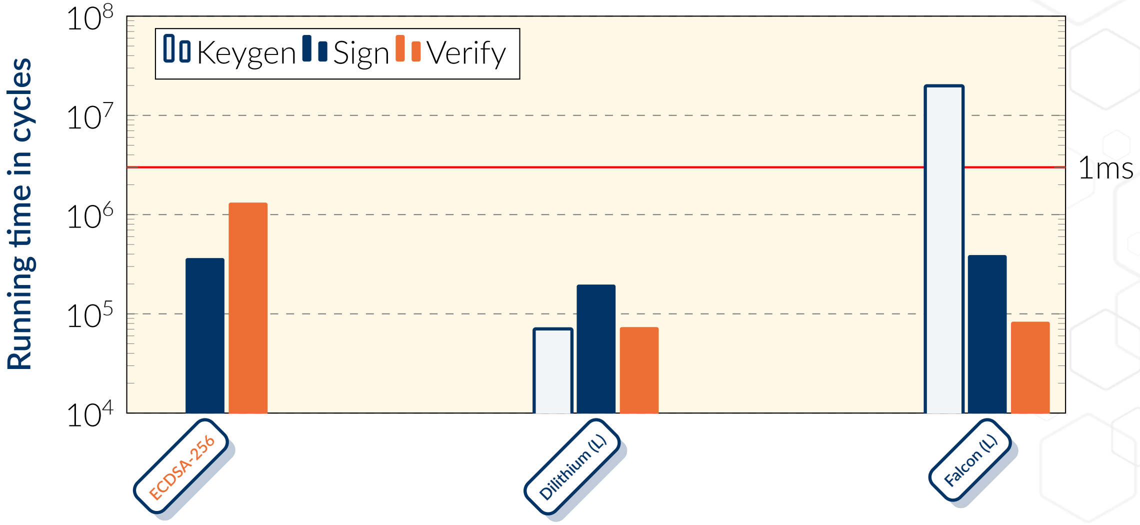



Signatures
Fiat-Shamir
Hash & Sign
Module-LWE
Ring-LWE
NTRU
Dilithium
qTESLA
Falcon

Bandwidth cost of Level 1 Signatures



Computation Cost of Level 1 Signatures



Ninja Tricks

Figure 1: Broadcast

Figure 2: Group Messaging

In LWE/LWR proposals, **U** does almost not depend on the public key.

- → Use the same **A** for all public keys.
- → Use the same **U** when encrypting **the same M** to several recipients.

$\mathsf{Enc}(\mathtt{M}, \mathbf{pk} = (\mathbf{A}, \mathbf{B}))$

1. $\mathbf{R}, \mathbf{E}', \mathbf{E}'' \leftarrow \chi_3 \times \chi_4 \times \chi_5$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. $\mathbf{V} \leftarrow \mathbf{RB} + \mathbf{E}'' + \mathsf{Encode}(\mathtt{M})$
4. $\mathtt{ct} := (\mathbf{U}, \mathbf{V})$

In LWE/LWR proposals, **U** does almost not depend on the public key.

- → Use the same **A** for all public keys.
- → Use the same **U** when encrypting **the same** $\mathtt{M}$ to several recipients.

Enc($\mathtt{M}$, **pk** = (**A**, **B**))

1. $\mathbf{R}, \mathbf{E}', \mathbf{E}'' \leftarrow \chi_3 \times \chi_4 \times \chi_5$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. $\mathbf{V} \leftarrow \mathbf{RB} + \mathbf{E}'' + \mathrm{Encode}(\mathtt{M})$
4. $\mathtt{ct} := (\mathbf{U}, \mathbf{V})$

$\Longrightarrow$

MultiEnc($\mathtt{M}$, $\mathbf{pk}_1, \ldots, \mathbf{pk}_k$)

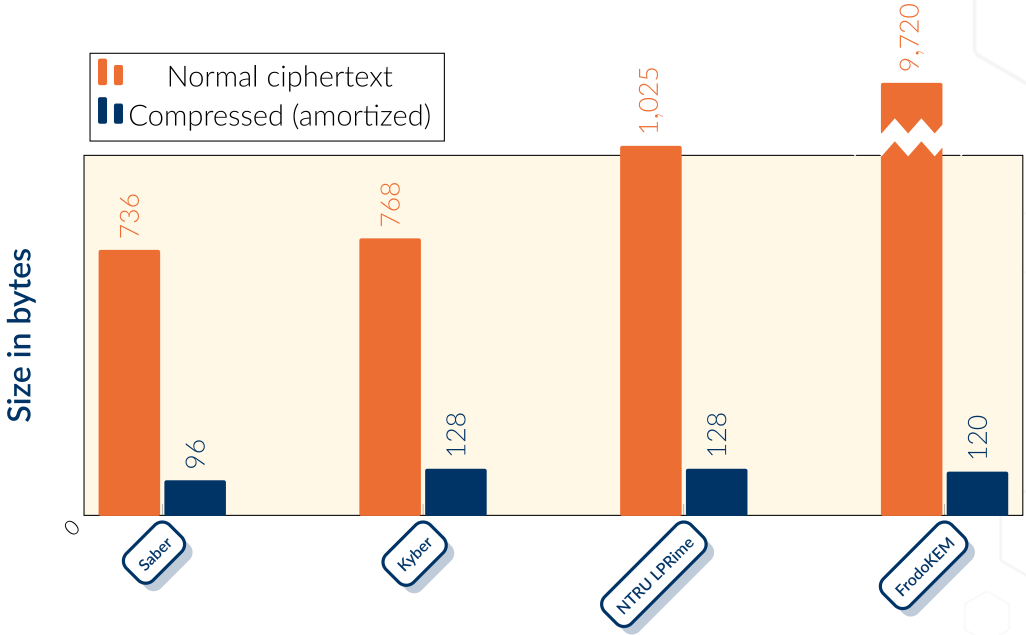
1. $\mathbf{R}, \mathbf{E}' \leftarrow \chi_3 \times \chi_4$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. For $i = 1, \ldots, k$:
   1. $\mathbf{E}''_i \leftarrow \chi_5$
   2. $\mathbf{V}_i \leftarrow \mathbf{RB}_i + \mathbf{E}''_i + \mathrm{Encode}(\mathtt{M})$
4. $\mathtt{ct} := (\mathbf{U}, \mathbf{V}_1, \ldots, \mathbf{V}_k)$

This improves amortized costs by **factors up to 169**.

- → Faster encryption
- → Smaller ciphertexts

See [KKPP20].

Impact on Potential NIST Standards (Level I)

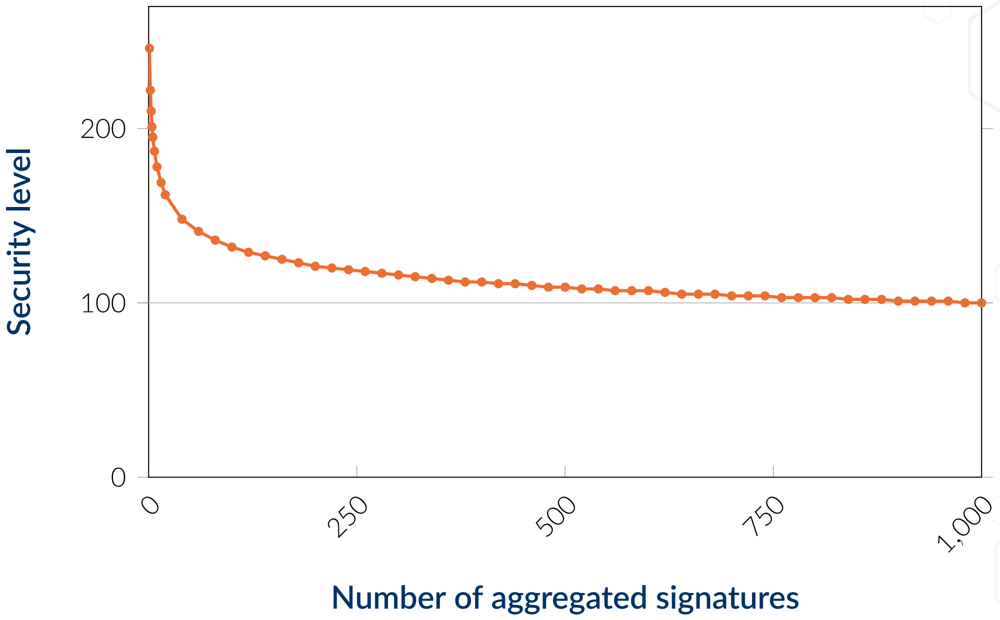


A cute property of GPV signatures

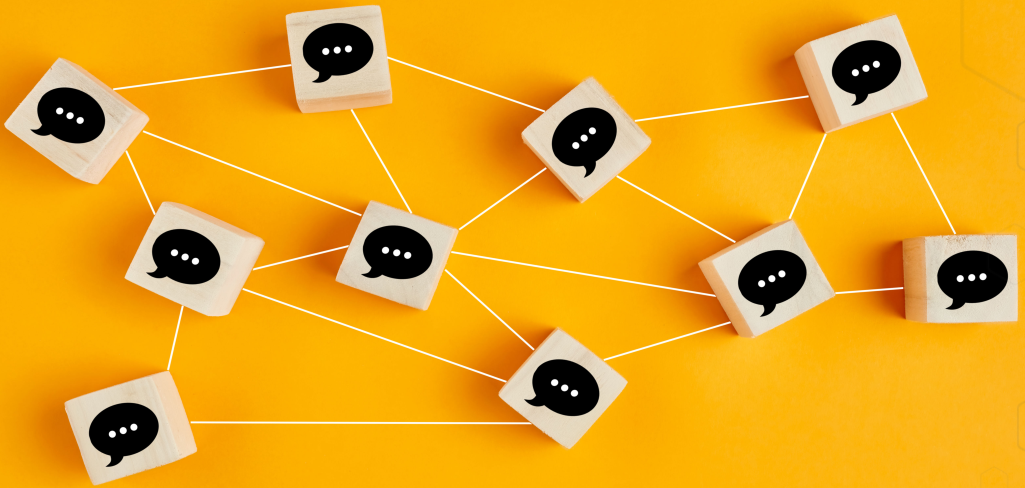
Like all GPV signatures, Falcon supports *single-signer* signature aggregation.

$$\left.\begin{aligned} \mathbf{s}_1 \cdot \mathbf{A} &= H(\mathtt{M}_1) \\ &\vdots \\ \mathbf{s}_k \cdot \mathbf{A} &= H(\mathtt{M}_k) \end{aligned}\right\} \Longrightarrow \left(\sum_i \mathbf{s}_i\right) \cdot \mathbf{A} = \sum_i H(\mathtt{M}_i)$$

For up to **1000 signatures**, aggregate signature size **< 3kB**.



Security level
200
100
0
0
250
500
750
1,000
Number of aggregated signatures



Questions?

Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon.
qTESLA.
Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

Shi Bai and Steven D. Galbraith.
An improved compression technique for signatures based on learning with errors.
In Josh Benaloh, editor, CT-RSA 2014, volume 8366 of LNCS, pages 28–47. Springer, Heidelberg, February 2014.

Daniel J. Bernstein and Edoardo Persichetti.
Towards KEM unification.
Cryptology ePrint Archive, Report 2018/526, 2018.
https://eprint.iacr.org/2018/526.

Alexander W. Dent.

A designer's guide to KEMs.
In Kenneth G. Paterson, editor, 9th IMA International Conference on Cryptography and Coding, volume 2898 of LNCS, pages 133–151. Springer, Heidelberg, December 2003.

Jan-Pieter D'Anvers, Mélissa Rossi, and Fernando Virdia.
(One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes.
In Anne Canteaut and Yuval Ishai, editors, EUROCRYPT 2020, Part III, volume 12107 of LNCS, pages 3–33. Springer, Heidelberg, May 2020.

Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz.
A modular analysis of the Fujisaki-Okamoto transformation.
In Yael Kalai and Leonid Reyzin, editors, TCC 2017, Part I, volume 10677 of LNCS, pages 341–371. Springer, Heidelberg, November 2017.

Shuichi Katsumata, Kris Kwiatkowski, Federico Pintore, and Thomas Prest.

Scalable ciphertext compression techniques for post-quantum KEMs and their applications.

In Shiho Moriai and Huaxiong Wang, editors, ASIACRYPT 2020, Part I, volume 12491 of LNCS, pages 289–320. Springer, Heidelberg, December 2020.

Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehlé.
CRYSTALS-DILITHIUM.
Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.