Learning Lattice

A Guided Tour Through Lyubashevsky's Tutorial

Duc V. Le

AI-assisted study notes (written with Claude) — not peer-reviewed; may contain errors. The original paper is the authority.

Based on "Basic Lattice Cryptography: The Concepts Behind Kyber (ML-KEM) and Dilithium (ML-DSA)" by Vadim Lyubashevsky, 2024

Contents

1. What This Document Is

Lyubashevsky's tutorial is an excellent reference, but it buries intuition under notation and mixes high-level arguments with fine-grained details. This document is a companion guide that follows the tutorial section by section, in the paper's own order: LWE encryption (Section 2), the hardness of lattice problems (Section 3), encryption over polynomial rings and Kyber (Section 4), and digital signatures and Dilithium (Section 5). We present each part as:

- **Intuition first**, formulas second.
- **Proper reductions** written out as clean game hops.
- **Geometric pictures** for every key concept.
- **Explicit analogies** to ElGamal so you can map what you already know.

How we cite. "Section 2.4", "Eq. (13)", "Figure 2", "Lemma 1" always refer to *the paper's* numbering. Cross-references inside this guide are written with the section sign, e.g. §8.1, and our own numbered figures and equations are called "our Figure N" and "our Equation N". Anything that goes beyond the paper (our own derivations, toy numbers, proof sketches the paper omits) is marked "spelled out", "illustration", or "not in the paper".

Prerequisites: You should be comfortable with public-key encryption (RSA, ElGamal), CPA security definitions, and the hybrid argument. If you know what "the DDH assumption" means for ElGamal security, you have everything you need.

2. Notation

All arithmetic is modulo q unless stated otherwise. We work in $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$, but we think of it as centered: $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$ when measuring "smallness".

Notation Cheat Sheet

$[\beta]$	The set $\{-\beta, \ldots, -1, 0, 1, \ldots, \beta\}$
$[\beta]^m$	A vector of m entries, each in $[\beta]$
$\mathbf{s} \xleftarrow{\$} [\beta]^m$	Sample each entry of $\mathbf{s}$ uniformly from $[\beta]$
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$	Sample each entry of $\mathbf{A}$ uniformly from $\mathbb{Z}_q$
$\|\mathbf{v}\|_\infty$	$\max_i \|v_i\|$, the largest absolute entry
$\|\mathbf{v}\|$	$\sqrt{\sum v_i^2}$, the Euclidean length

Part I: Encryption from LWE

3. The Blueprint: ElGamal Over Matrices

Before introducing LWE, let us see the *structural skeleton* that all lattice encryption follows. This section corresponds to Lyubashevsky's Section 2.2.

3.1 ElGamal Recap

In ElGamal over a group $G = \langle g \rangle$ of order p:

	ElGamal	**Security relies on**
Secret key	$s \in \mathbb{Z}_p$	
Public key	$(g,\ h = g^s)$	DDH: $(g, g^s, g^r, g^{rs}) \approx (g, g^s, g^r, g^u)$
Encrypt μ	$(c_1 = g^r,\ c_2 = h^r \cdot \mu)$	
Decrypt	$\mu = c_2 / c_1^s$	Cancellation: $h^r / g^{rs} = g^{rs} / g^{rs} = 1$

The magic: the encryptor "entangles" the message with g^{rs} using the public key $h = g^s$, and the decryptor "un-entangles" it using s. Security comes from the DDH assumption: given g^r and g^s, no one can tell g^{rs} apart from a random group element. (Being unable to *compute* g^{rs} is the weaker CDH assumption, which is not enough for CPA security.)

3.2 The Same Skeleton, Over Matrices

Now replace the group G with matrices over $\mathbb{Z}_q$. We have $\mathbf{A} \in \mathbb{Z}_q^{m \times m}$ playing the role of g, and short vectors playing the role of exponents:

ElGamal	**Matrix Analogue**
$s \in \mathbb{Z}_p$	$\mathbf{s} \xleftarrow{\$} [\beta]^m$
$h = g^s$	$\mathbf{t} = \mathbf{A}\mathbf{s}$
$c_1 = g^r$	$\mathbf{u}^T = \mathbf{r}^T \mathbf{A}$
$c_2 = h^r \cdot \mu$	$v = \mathbf{r}^T \mathbf{t} + \mu$
$c_2 / c_1^s = \mu$	$v - \mathbf{u}^T \mathbf{s} = \mu$

Decryption works because:

$$v - \mathbf{u}^T\mathbf{s} = \mathbf{r}^T\mathbf{t} + \mu - \mathbf{r}^T\mathbf{A}\mathbf{s} = \mathbf{r}^T(\mathbf{A}\mathbf{s}) + \mu - \mathbf{r}^T(\mathbf{A}\mathbf{s}) = \mu.$$

This is exactly ElGamal's cancellation trick: $\mathbf{r}$ multiplies from the left, $\mathbf{s}$ from the right, and $\mathbf{A}$ sits in the middle.

Why $\mathbf{r}^T\mathbf{A}$ and not $\mathbf{A}\mathbf{r}$? — Associativity replaces commutativity

In ElGamal, the cancellation $(g^s)^r/(g^r)^s = g^{sr}/g^{rs} = 1$ works because integer multiplication is commutative: $sr = rs$.
Matrices are **not** commutative. If both $\mathbf{s}$ and $\mathbf{r}$ multiplied $\mathbf{A}$ from the *same* side (say the right: public key $\mathbf{t} = \mathbf{A}\mathbf{s}$, ciphertext $\mathbf{u} = \mathbf{A}\mathbf{r}$), then decryption would need:

$$\underbrace{\mathbf{t}^T\mathbf{r}}_{\mathbf{s}^T\mathbf{A}^T\mathbf{r}} - \underbrace{\mathbf{s}^T\mathbf{u}}_{\mathbf{s}^T\mathbf{A}\mathbf{r}} = \mathbf{s}^T(\mathbf{A}^T - \mathbf{A})\mathbf{r} \neq 0 \quad \text{(unless } \mathbf{A} \text{ is symmetric)}.$$

Since $\mathbf{A}$ is random, $\mathbf{A}^T \neq \mathbf{A}$, and this does *not* cancel.
The fix: put $\mathbf{r}$ on the **left** ($\mathbf{r}^T\mathbf{A}$) and $\mathbf{s}$ on the **right** ($\mathbf{A}\mathbf{s}$). Now:

$$\underbrace{\mathbf{r}^T(\mathbf{A}\mathbf{s})}_{\text{from } \mathbf{r}^T\mathbf{t}} - \underbrace{(\mathbf{r}^T\mathbf{A})\mathbf{s}}_{\text{from } \mathbf{u}^T\mathbf{s}} = \mathbf{r}^T\mathbf{A}\mathbf{s} - \mathbf{r}^T\mathbf{A}\mathbf{s} = 0$$

by **associativity** alone—no commutativity needed. This is the essential structural insight: matrix multiplication is associative ($\mathbf{r}^T(\mathbf{A}\mathbf{s}) = (\mathbf{r}^T\mathbf{A})\mathbf{s}$), and that is enough to build the cancellation trick.

This scheme is completely broken!

The "assumption" that $(\mathbf{A}, \mathbf{A}\mathbf{s})$ is indistinguishable from $(\mathbf{A}, \mathbf{u})$ is *false*. Just invert $\mathbf{A}$ (via Gaussian elimination) and check whether $\mathbf{A}^{-1}\mathbf{u}$ has small entries. This is the starting point—not the destination.

4. The LWE Problem

4.1 The Fix: Add Noise

The fix is beautifully simple. Instead of publishing $\mathbf{t} = \mathbf{A}\mathbf{s}$ exactly, publish $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ where $\mathbf{e}$ is a small error vector. Now Gaussian elimination gives you $\mathbf{A}^{-1}\mathbf{t} = \mathbf{s} + \mathbf{A}^{-1}\mathbf{e}$, and the term $\mathbf{A}^{-1}\mathbf{e}$ blows up (a random matrix times a small vector is a large vector), so you learn nothing about $\mathbf{s}$.

The LWE Problem (Definition 1 in the paper)

For parameters n, m, q, β with $\beta < q$, the $\mathrm{LWE}_{n,m,q,\beta}$ problem asks to distinguish:

1. $(\mathbf{A},\ \mathbf{A}\mathbf{s} + \mathbf{e})$ where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \xleftarrow{\$} [\beta]^m$, $\mathbf{e} \xleftarrow{\$} [\beta]^n$
2. $(\mathbf{A},\ \mathbf{u})$ where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$

No efficient algorithm should be able to tell which distribution a given sample came from.

4.2 What the Adversary Sees

This is a **decision problem**. A challenger flips a bit b. If $b = 0$, they produce $(\mathbf{A}, \mathbf{As} + \mathbf{e})$. If $b = 1$, they produce $(\mathbf{A}, \mathbf{u})$. The adversary sees one sample and guesses b.

Think of each *row* of the equation $\mathbf{t} = \mathbf{As} + \mathbf{e}$ as one "LWE sample":

$$t_i = \mathbf{a}_i^T \mathbf{s} + e_i \qquad \text{for } i = 1, \ldots, n$$

where $\mathbf{a}_i^T$ is the i-th row of $\mathbf{A}$. So having n rows means having n samples, all sharing the *same* secret $\mathbf{s}$.

Multiple-samples view vs. matrix view

Regev's original formulation gives the adversary many pairs $(\mathbf{a}_i, b_i)$ where $b_i = \mathbf{a}_i^T \mathbf{s} + e_i$. Stacking them into a matrix gives $(\mathbf{A}, \mathbf{t})$ with $\mathbf{t} = \mathbf{As}+\mathbf{e}$. These are the same problem—just different notation.

4.3 What Makes It Hard: The Parameters

- **Dimension** m (number of columns of $\mathbf{A}$, dimension of $\mathbf{s}$): This is the main security parameter. Larger m means harder problem.
- **Noise ratio** β/q: Larger ratio means more noise relative to the modulus, making the problem harder. But too much noise breaks decryption.
- **Number of samples** n (rows of $\mathbf{A}$): Surprisingly, this has little effect on hardness, except in extreme cases where $n \approx m^{2\beta+1}$.

4.4 Why Uniform Noise Instead of Gaussian?

If you have seen LWE before, you may recall the error was sampled from a **discrete Gaussian**. Lyubashevsky uses **uniform** over $[\beta]$ for simplicity. Here is why both are fine:

1. **For worst-case reductions:** Regev's original proof that LWE is as hard as worst-case lattice problems specifically requires Gaussian errors. Later work [DM13, MP13] showed this is not strictly necessary (Section 2.3 of the paper).
2. **For concrete security:** The best known attacks depend on the *norm* of the error vector $\|(\mathbf{s}, \mathbf{e})\|$, not its distribution shape. A uniform vector in $[\beta]^m$ and a Gaussian vector with matching variance produce vectors of similar Euclidean length $\approx \sigma\sqrt{m}$, and are equally hard to attack.
3. **In practice:** Kyber uses a binomial distribution (fast to sample). Dilithium uses uniform (simple to implement). The paper uses uniform (clean to present).

The bottom line: the distribution is a second-order concern. What drives security is the norm.

5. The LWE Encryption Scheme

This is **Section 2.3.1** of the paper—the core construction.

5.1 The Scheme

LWE Encryption – Single Bit

Key Generation:

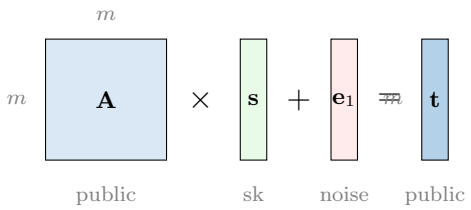
$$\mathbf{s} \xleftarrow{\$} [\beta]^m, \quad \mathbf{e}_1 \xleftarrow{\$} [\beta]^m$$
$$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times m}$$
$$\text{sk} = \mathbf{s}, \quad \text{pk} = (\mathbf{A},\ \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}_1)$$

Encryption of $\mu \in \{0,1\}$: sample $\mathbf{r}, \mathbf{e}_2 \xleftarrow{\$} [\beta]^m$ and $e_3 \xleftarrow{\$} [\beta]$, compute:

$$\mathbf{u}^T = \mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T$$
$$v = \mathbf{r}^T\mathbf{t} + e_3 + \lfloor \tfrac{q}{2} \rceil \cdot \mu$$

Output ciphertext $(\mathbf{u}, v)$.
Decryption: Compute $v - \mathbf{u}^T\mathbf{s}$ and output 0 if the result is closer to 0, or 1 if closer to $\lfloor q/2 \rceil$.

Key Generation: $\text{pk} = (\mathbf{A},\ \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}_1)$



Encryption: $\text{ct} = (\mathbf{u},\ v)$

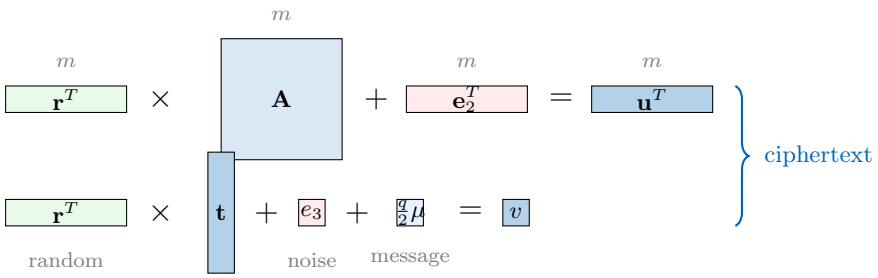


Figure 1: Single-bit LWE encryption: matrix dimensions and data flow.

5.2 Comparing with the Broken Scheme

Three changes from the noiseless version:

1. **Noise in the public key:** $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}_1$ instead of $\mathbf{t} = \mathbf{A}\mathbf{s}$.
2. **Noise in the ciphertext:** $\mathbf{u}^T = \mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T$ and v includes e_3.
3. **Message encoding:** $\mu \in \{0,1\}$ is scaled by $\lfloor q/2 \rceil$ to maximize distance from the noise.

5.3 Why Decryption Works (Almost Always)

Expand $v - \mathbf{u}^T\mathbf{s}$:

$$\begin{aligned} v - \mathbf{u}^T\mathbf{s} &= \left(\mathbf{r}^T\mathbf{t} + e_3 + \tfrac{q}{2}\mu\right) - \left(\mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T\right)\mathbf{s} \\ &= \mathbf{r}^T(\mathbf{A}\mathbf{s} + \mathbf{e}_1) + e_3 + \tfrac{q}{2}\mu - \mathbf{r}^T\mathbf{A}\mathbf{s} - \mathbf{e}_2^T\mathbf{s} \\ &= \underbrace{\mathbf{r}^T\mathbf{e}_1 + e_3 - \mathbf{e}_2^T\mathbf{s}}_{\text{noise}} + \frac{q}{2}\mu \end{aligned} \tag{1}$$

The $\mathbf{r}^T\mathbf{A}\mathbf{s}$ terms cancel (the ElGamal trick), leaving the message scaled by $q/2$ plus accumulated noise.

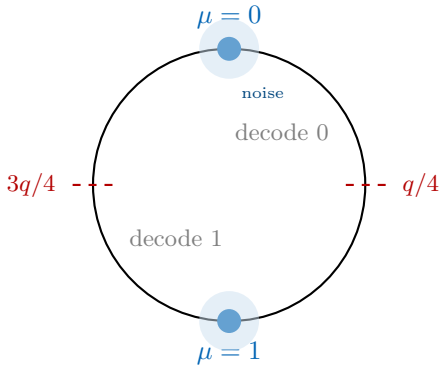
Decryption Condition

Decryption is correct if and only if the noise term satisfies:

$$\left|\mathbf{r}^T\mathbf{e}_1 + e_3 - \mathbf{e}_2^T\mathbf{s}\right| < \frac{q}{4}.$$

This ensures the result is unambiguously closer to 0 (if $\mu = 0$) or to $q/2$ (if $\mu = 1$).

Geometric picture: Think of $\mathbb{Z}_q$ as a clock. The two "message points" sit at 12 o'clock ($\mu = 0$, value 0) and 6 o'clock ($\mu = 1$, value $q/2$). The noise jiggles the result around the correct message point. As long as the jiggle stays within a quarter-turn ($q/4$), we land in the correct half of the clock.



5.3.1 Geometric Picture: Ciphertext Space in 2D

To build intuition, set $n = 1$ and $m = 1$ (one scalar LWE sample) and draw the ciphertext (u, v) in the plane $\mathbb{Z}_q \times \mathbb{Z}_q$. This is a *cartoon*: with $m = 1$ the LWE problem is actually easy (the paper notes in Section 2.3 that it is "not difficult" to tell whether $\mathbf{A}\mathbf{s} + \mathbf{e}$ is close to a multiple of $\mathbf{A}$). The picture shows the geometry; the hardness only appears in high dimension.

- The public key gives a slope: $t = as + e_1 \approx as$, so the "secret direction" has slope s.
- **Encrypt $\mu = 0$:** $(u, v) = (ra + e_2,\ r \cdot t + e_3)$. Without noise this lies on the line $L_0 : v = s\,u$ (through the origin). Noise spreads it into a **band** around L_0.
- **Encrypt $\mu = 1$:** $(u, v) = (ra + e_2,\ r \cdot t + e_3 + q/2)$. This lies in a band around the **parallel line** $L_1 : v = s\,u + q/2$.
- **Decryption** computes $v - s\,u$, which *projects* perpendicular to the secret direction, collapsing each band to a cluster near 0 or $q/2$ on the clock.

- **Without s:** an adversary cannot determine the slope, so the two bands are computationally indistinguishable from random points (by the LWE assumption).

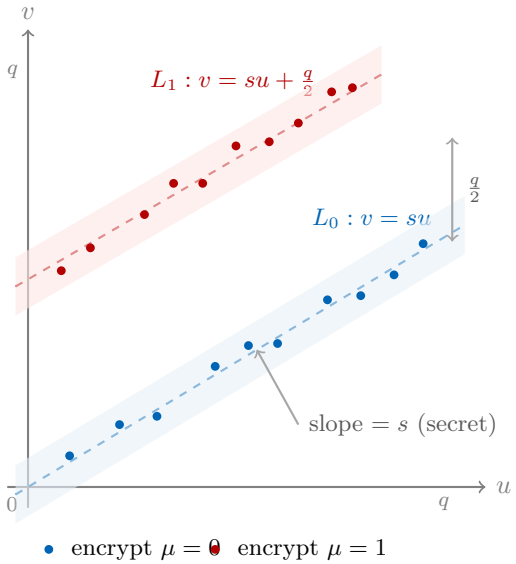


Figure 2: Ciphertext space for 1-bit LWE encryption (scalar case). The secret key determines the slope; knowing s lets you separate the two bands via the projection $v - su$.

Why LWE ⇒ Security

Without the secret slope s, an eavesdropper sees random-looking points in $\mathbb{Z}_q^2$. In high dimension (large m) the LWE assumption says exactly this: the points are computationally indistinguishable from uniform, so the adversary cannot tell which band a ciphertext belongs to. (In the 2D cartoon itself the bands are easy to spot—that is why real parameters use m in the hundreds.)

5.4 Bounding the Total Error, Exactly (Section 2.3.2)

5.4.1 The Worst Case Is Far Too Pessimistic

Every coefficient of $\mathbf{r}, \mathbf{s}, \mathbf{e}_1, \mathbf{e}_2$ lies in $[\beta]$, so each of the inner products $\mathbf{r}^T\mathbf{e}_1$ and $\mathbf{e}_2^T\mathbf{s}$ is a sum of m terms of magnitude at most β^2. Adding $|e_3| \le \beta$, the decryption noise always satisfies

$$\left|\mathbf{r}^T\mathbf{e}_1 + e_3 - \mathbf{e}_2^T\mathbf{s}\right| \;\le\; 2m\beta^2 + \beta,$$

and setting $2m\beta^2 + \beta < q/4$ guarantees that decryption *never* fails. But the coefficients are random and centred around 0, so the terms mostly cancel: the paper notes that the noise is in fact closer to $O(\sqrt{m}\,\beta^2)$. Reaching the worst case needs all $2m$ products to have the same sign and maximal size at once, which essentially never happens.

What we actually want is a bound that holds with very high probability, say $1 - 2^{-150}$. Then decryption fails with probability at most 2^{-150}, which the paper says is tolerable both in applications and when the scheme is used inside other constructions (e.g. CCA-secure encryption, Section 4.8). So we want the *exact* value of (the paper's Eq. (10))

$$\Pr_{\mathbf{s},\mathbf{r},\mathbf{e}_1,\mathbf{e}_2 \leftarrow [\beta]^m,\; e_3 \leftarrow [\beta]} \left[\mathbf{r}^T\mathbf{e}_1 + e_3 - \mathbf{e}_2^T\mathbf{s} \in [\alpha]\right]. \tag{2}$$

5.4.2 The Tool: Distributions as Polynomials

Sums of random variables = products of polynomials

Let A and B be independent random variables over $[\gamma]$, with $A_i = \Pr[A = i]$ and $B_i = \Pr[B = i]$. Encode each distribution as a polynomial with negative powers allowed:

$$A(X) = \sum_{i=-\gamma}^{\gamma} A_i X^i, \qquad B(X) = \sum_{i=-\gamma}^{\gamma} B_i X^i, \qquad C(X) = A(X) \cdot B(X) = \sum_{i=-2\gamma}^{2\gamma} C_i X^i.$$

Then (the paper's Eqs. (11)–(12))

$$\Pr[A + B = i] = C_i, \qquad \Pr\big[A + B \in [\alpha]\big] = \sum_{i=-\alpha}^{\alpha} C_i.$$

Why this works (spelled out). When you multiply the two polynomials, the coefficient of X^i collects every pair of terms whose exponents add up to i:

$$C_i = \sum_j A_j \, B_{i-j} = \sum_j \Pr[A = j] \Pr[B = i - j] = \Pr[A + B = i],$$

where the last step uses independence: the events "$A = j$ and $B = i - j$" for different j are disjoint, and together they make up the event "$A + B = i$". Multiplying polynomials *is* convolving distributions. The same holds for any number of independent variables: the distribution of a sum of k of them is the product of their k polynomials.

A typo in the paper's Eq. (12)

The paper writes the summand in Eq. (12) as C_α; it should be C_i (we sum the coefficients from $-\alpha$ to α), as the paper itself writes a few lines later.

Rolling dice

This is the classic generating-function trick for dice. One die is $\frac{1}{6}(X + X^2 + \cdots + X^6)$. The coefficient of X^7 in its square is $\frac{6}{36}$, because there are 6 ways to roll a total of 7 with two dice. Here the "dice" are the products xy and the error e_3.

5.4.3 Applying It to the Decryption Noise

The noise $\mathbf{r}^T\mathbf{e}_1 - \mathbf{e}_2^T\mathbf{s} = \sum_{i=1}^m r_i e_{1,i} - \sum_{i=1}^m e_{2,i} s_i$ is a sum of $2m$ *independent* products, since every product uses its own two fresh coordinates. Each product is distributed as

$$A_i = \Pr[A = i] = \Pr_{x,y \leftarrow [\beta]}[xy = i].$$

(The minus sign in front of $\mathbf{e}_2^T\mathbf{s}$ does not matter: $[\beta]$ is symmetric, so $-xy$ has the same distribution as xy.) The last term e_3 is uniform over $[\beta]$.

Worked example: $\beta = 2$ (from the paper). There are $5 \times 5 = 25$ equally likely pairs $(x, y) \in [2]^2$. Counting them:

xy	pairs (x, y)	probability
0	$x = 0$ (5 pairs) or $y = 0$ (5 pairs), minus $(0, 0)$ counted twice	9/25
± 1	$(1, 1), (-1, -1)$ give $+1$; $(1, -1), (-1, 1)$ give -1	2/25 each
± 2	$(1, 2), (2, 1), (-1, -2), (-2, -1)$ give $+2$; similarly for -2	4/25 each
± 4	$(2, 2), (-2, -2)$ give $+4$; $(2, -2), (-2, 2)$ give -4	2/25 each

The values ± 3 never occur. This matches the paper: $A_{\pm 4} = \frac{2}{25}$, $A_{\pm 2} = \frac{4}{25}$, $A_{\pm 1} = \frac{2}{25}$, $A_0 = \frac{9}{25}$. The full noise distribution is therefore the coefficient list of

$$C(X) = \left(\tfrac{2}{25}X^{-4} + \tfrac{4}{25}X^{-2} + \tfrac{2}{25}X^{-1} + \tfrac{9}{25} + \tfrac{2}{25}X + \tfrac{4}{25}X^2 + \tfrac{2}{25}X^4\right)^{2m} \cdot \left(\tfrac{1}{5}X^{-2} + \tfrac{1}{5}X^{-1} + \tfrac{1}{5} + \tfrac{1}{5}X + \tfrac{1}{5}X^2\right).$$

Decryption is correct when the noise lies in $[q/4 - 1]$, so setting $\alpha = q/4 - 1$ and summing $C_{-\alpha}, \ldots, C_\alpha$ gives the *exact* probability of correct decryption.

Illustration (our computation, not from the paper): $\beta = 2$, $m = 256$

We expanded $C(X)$ exactly (in integer arithmetic) for $\beta = 2$, $m = 256$. The worst-case bound is $2m\beta^2 + \beta = 2050$. The exact tail probabilities are:

t	100	200	300	400	500	600	800
$\Pr[\lvert\text{noise}\rvert > t]$	$2^{-5.2}$	$2^{-16.7}$	$2^{-34.9}$	$2^{-60.0}$	$2^{-92.2}$	$2^{-131.5}$	$2^{-231.9}$

The smallest t with $\Pr[\lvert\text{noise}\rvert > t] < 2^{-150}$ is $t = 642$ (tail $\approx 2^{-150.1}$). So $q/4 - 1 \geq 642$, i.e. $q \geq 2572$, already gives failure probability below 2^{-150}, whereas the worst-case bound would demand $q > 8200$. The exact computation buys roughly a factor of 3 in q.

What ElGamal never needed

ElGamal decryption is exact: $c_2 / c_1^s = \mu$ with no error term at all, so there is nothing to bound. In lattice encryption, correctness is a *probabilistic* statement about a sum of many small random terms, and the whole art of parameter selection is making that probability tiny while keeping q as small as possible (small q means smaller keys and a harder LWE problem).

6. The Security Proof (Done Properly)

Lyubashevsky sketches the security argument in prose. Here is the full reduction, written as clean game hops.

Theorem

If $\text{LWE}_{m,q,\beta}$ is hard, then the encryption scheme above is CPA-secure. Specifically:

$$\text{Adv}^{\text{CPA}}(\mathcal{A}) \;\leq\; 2 \cdot \text{Adv}^{\text{LWE}}_{m,q,\beta}(\mathcal{D})$$

for some efficient distinguisher $\mathcal{D}$ constructed from the CPA adversary $\mathcal{A}$.

Why no encryption oracle?

In the general CPA definition, the adversary has access to an encryption oracle. For *symmetric-key* encryption, this oracle is essential—the adversary cannot encrypt without the secret key. For *public-key* encryption, however, the oracle is redundant: the adversary already holds pk and can run the (public) encryption algorithm on any message of their choosing. The CPA game therefore reduces to a single left-or-right challenge: the adversary submits μ_0, μ_1, receives $\mathrm{Enc}(\mathrm{pk}, \mu_b)$, and guesses b. No oracle simulation is needed in the reduction.

Proof. We proceed through a sequence of games.

Game 0 – The Real CPA Game

The challenger runs honest key generation and encryption:

$$pk = (\mathbf{A},\ \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}_1)$$
$$ct = (\mathbf{u}^T = \mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T,\ \ v = \mathbf{r}^T\mathbf{t} + e_3 + \tfrac{q}{2}\mu_b)$$

The adversary $\mathcal{A}$ receives (pk, ct) and tries to guess b.

Game 1 – Replace the public key with random

Change: Replace $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}_1$ with a uniformly random $\mathbf{t} \xleftarrow{\$} \mathbb{Z}_q^m$.
Why this is hard to detect: Distinguishing $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}_1)$ from $(\mathbf{A}, \mathbf{t})$ is *exactly* the $\mathrm{LWE}_{m,q,\beta}$ problem. So:

$$\big|\Pr[\text{Game 0}] - \Pr[\text{Game 1}]\big| \ \le\ \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}\,.$$

Game 2 – Replace the ciphertext with random

Setup: After Game 1, the public key is $\mathbf{A}' = [\mathbf{A} \mid \mathbf{t}] \in \mathbb{Z}_q^{m\times(m+1)}$, which is now *uniformly random.*
Change: Replace $(\mathbf{u}^T, v) = \mathbf{r}^T\mathbf{A}' + (\mathbf{e}_2^T, e_3 + \frac{q}{2}\mu_b)$ with a uniformly random vector.
Why this is hard to detect: Given a uniformly random $\mathbf{A}'$, distinguishing $(\mathbf{A}', \mathbf{r}^T\mathbf{A}' + \mathbf{e}'^T)$ from $(\mathbf{A}', \mathbf{u}'^T)$ is *again* the LWE problem (with rows and columns swapped). So:

$$\big|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]\big| \ \le\ \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}\,.$$

Game 2: Adversary has zero advantage

In Game 2, the ciphertext $(\mathbf{u}, v)$ is uniformly random, independent of μ_b. So $\mathcal{A}$ learns nothing about b, and $\Pr[\text{Game 2 outputs } b] = 1/2$.

Combining:

$$\begin{aligned}\mathrm{Adv}^{\mathrm{CPA}}(\mathcal{A}) &= \big|\Pr[\text{Game 0}] - \tfrac{1}{2}\big| = \big|\Pr[\text{Game 0}] - \Pr[\text{Game 2}]\big| \\ &\le \big|\Pr[\text{Game 0}] - \Pr[\text{Game 1}]\big| + \big|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]\big| \\ &\le 2\cdot \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}\,. \qquad \square\end{aligned} \tag{3}$$

Compare with ElGamal's proof

ElGamal's CPA proof has the same two-hop structure, except both hops use DDH:

1. Replace (g, g^s, g^r, g^{rs}) with (g, g^s, g^r, g^u) — by DDH.
2. Now the "mask" g^u is random, so $g^u \cdot \mu_b$ hides b perfectly.

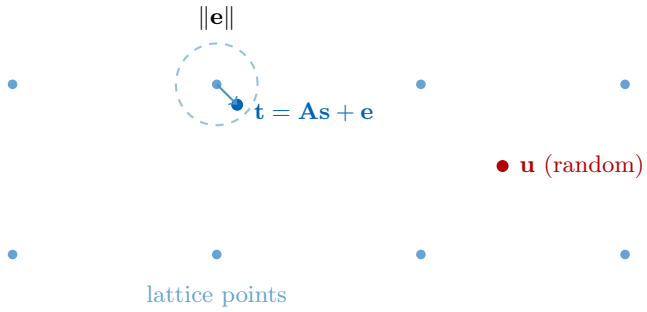
The LWE proof is structurally identical: replace the public key with random (hop 1), then the ciphertext becomes random (hop 2).

7. The Geometric Picture: Why LWE is Hard

7.1 Lattice Points and Clouds

When you publish $\mathbf{t} = \mathbf{As} + \mathbf{e}$, the vector $\mathbf{As}$ is a **lattice point** (it lies in the image of the lattice under $\mathbf{A}$), and $\mathbf{e}$ is a small perturbation pushing $\mathbf{t}$ away from that lattice point by distance $\approx \|\mathbf{e}\|$.

A uniformly random $\mathbf{u}$, on the other hand, has no reason to be near any lattice point.



7.2 Close to the Lattice, Yet Hard to Tell

Your intuition might say: "If $\mathbf{e}$ is small, $\mathbf{t}$ is very close to a lattice point, while a random $\mathbf{u}$ is far from all lattice points. So distinguishing should be *easy*!"

Information-theoretically, this is correct: for parameters that make the encryption scheme work, random cosets are much farther from the lattice than LWE cosets (Section 3.3 of the paper, and Part II below). The answer is "right there" in the data.

Computationally, nobody knows how to see it. The natural attack (Section 3.3 of the paper, Eqs. (37)–(38); §14 below) is to find *short* vectors $\mathbf{r}_1, \mathbf{r}_2$ with

$$\mathbf{r}_1^T\mathbf{A} + \mathbf{r}_2^T = \mathbf{0} \qquad \Longrightarrow \qquad \mathbf{r}_1^T\mathbf{t} = \mathbf{r}_1^T\mathbf{As} + \mathbf{r}_1^T\mathbf{e} = -\mathbf{r}_2^T\mathbf{s} + \mathbf{r}_1^T\mathbf{e}.$$

If $\mathbf{t}$ is an LWE sample, this is small (short vectors times short vectors). If $\mathbf{t}$ is random, $\mathbf{r}_1^T\mathbf{t}$ is random in $\mathbb{Z}_q$. (Note that we need the extra $\mathbf{r}_2$: for a random square $\mathbf{A}$ there is no short nonzero $\mathbf{r}$ with $\mathbf{r}^T\mathbf{A} = \mathbf{0}$ alone.) But finding short $(\mathbf{r}_1, \mathbf{r}_2)$ is a short-vector problem in a high-dimensional lattice, which is exactly the hard problem.

How the noise size enters (spelled out). For the attack to succeed, $-\mathbf{r}_2^T\mathbf{s} + \mathbf{r}_1^T\mathbf{e}$ must be noticeably smaller than q. Its size grows with $\|(\mathbf{r}_1, \mathbf{r}_2)\|$ times the size of $(\mathbf{s}, \mathbf{e})$. So the *larger* β is, the *shorter* the vectors the attacker must find, and the harder the attack. This matches the paper (Section 2.3): LWE "becomes harder as m and β/q grow," and Figure 2 of the paper, where LWE hardness increases with β.

The Hardness Intuition

Noise β (vs. q)	Distinguishing LWE	Decryption
Zero	Easy (Gaussian elimination, Section 2.2)	Always correct
Small	Harder as β/q grows (need shorter $\mathbf{r}_1, \mathbf{r}_2$)	Correct
Too large	Hardest	Fails (noise $\geq q/4$)

Security wants β/q large; correctness wants it small (roughly $2m\beta^2 + \beta < q/4$). A cryptosystem lives where both are satisfied. The paper also warns that some choices make LWE trivially hard but useless (e.g. if $n < m$ and β is large enough, $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ can be *statistically* close to uniform, and then no encryption scheme can be built on it), and some make it easy (e.g. $m = 1$).

8. Size Trade-offs, Variations and Optimizations (Sections 2.4–2.5)

The basic scheme encrypts one bit with a ciphertext of $(m+1)\log q$ bits. With $m \approx 700$ and $q \approx 2^{13}$, that is ∼9000 bits per plaintext bit. This section follows Sections 2.4 and 2.5 of the paper: amortization, then the variations and optimizations of Section 2.5.1–2.5.6.

8.1 Amortization (Section 2.4)

The basic scheme encrypts *one bit* per ciphertext, but the expensive part $\mathbf{u} \in \mathbb{Z}_q^m$ ($m \log q$ bits) is paid every time. The idea: use a secret *matrix* so that the costly part is shared across many message bits.

8.1.1 The Amortized Scheme

We want to encrypt $N = k\ell$ bits, arranged as a matrix $\mathbf{M} \in \{0,1\}^{k \times \ell}$.

Amortized LWE Encryption – $k\ell$ Bits

Key Generation:

$$\mathbf{S} \xleftarrow{\$} [\beta]^{m\times\ell}, \quad \mathbf{E}_1 \xleftarrow{\$} [\beta]^{m\times\ell}$$
$$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times m}$$
$$\mathrm{sk} = \mathbf{S}, \quad \mathrm{pk} = (\mathbf{A},\ \mathbf{T} = \mathbf{AS} + \mathbf{E}_1)$$

The public key is now $(\mathbf{A}, \mathbf{T}) \in \mathbb{Z}_q^{m\times m} \times \mathbb{Z}_q^{m\times\ell}$, costing $256 + \ell m \log q$ bits (using a seed for $\mathbf{A}$).

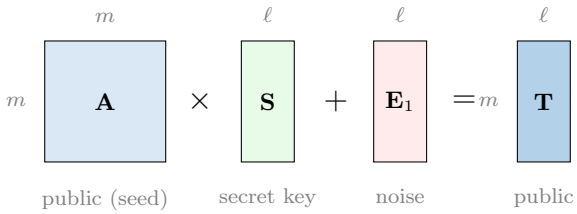
Encryption of $\mathbf{M} \in \{0,1\}^{k\times\ell}$: sample $\mathbf{R}, \mathbf{E}_2 \xleftarrow{\$} [\beta]^{k\times m}$ and $\mathbf{E}_3 \xleftarrow{\$} [\beta]^{k\times\ell}$, compute:

$$\mathbf{U} = \mathbf{RA} + \mathbf{E}_2 \in \mathbb{Z}_q^{k\times m}$$
$$\mathbf{V} = \mathbf{RT} + \mathbf{E}_3 + \left\lfloor \tfrac{q}{2} \right\rceil \cdot \mathbf{M} \in \mathbb{Z}_q^{k\times\ell}$$

Output ciphertext $(\mathbf{U}, \mathbf{V})$, costing $km \log q + k\ell \log q$ bits.

Decryption: Compute $\mathbf{V} - \mathbf{US}$ and round each entry to 0 or $\lfloor q/2 \rceil$.

Key Generation: $\mathrm{pk} = (\mathbf{A},\ \mathbf{T} = \mathbf{AS} + \mathbf{E}_1)$



Encryption: $\mathrm{ct} = (\mathbf{U},\ \mathbf{V})$

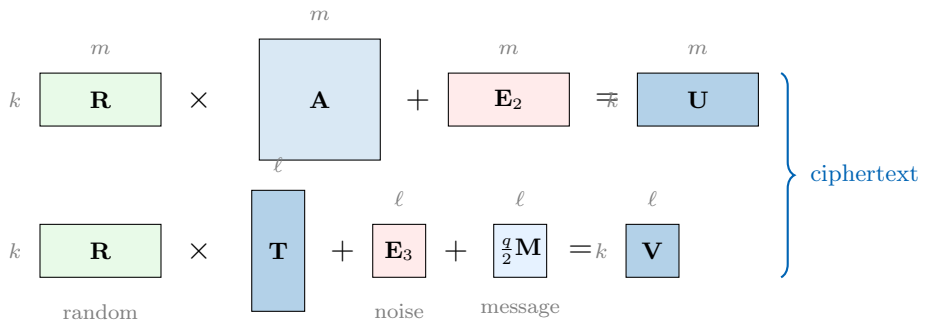


Figure 3: Amortized LWE encryption: matrix dimensions for multi-bit scheme.

The key observation: $\mathbf{U}$ ($km \log q$ bits) is the expensive part and is shared across all ℓ columns of the message. By varying k and ℓ while keeping $N = k\ell$ fixed, we trade public-key size against ciphertext size. The optimum is $k \approx \ell \approx \sqrt{N}$.

8.1.2 Correctness

Each entry (i, j) of $\mathbf{V} - \mathbf{US}$ decrypts independently:

$$\begin{aligned}(\mathbf{V} - \mathbf{US})_{ij} &= \mathbf{r}_i^T \mathbf{t}_j + (e_3)_{ij} + \tfrac{q}{2}\mu_{ij} - (\mathbf{r}_i^T \mathbf{A} + \mathbf{e}_{2,i}^T)\mathbf{s}_j \\ &= \mathbf{r}_i^T(\mathbf{A}\mathbf{s}_j + \mathbf{e}_{1,j}) + (e_3)_{ij} + \tfrac{q}{2}\mu_{ij} - \mathbf{r}_i^T \mathbf{A}\mathbf{s}_j - \mathbf{e}_{2,i}^T \mathbf{s}_j \\ &= \underbrace{\mathbf{r}_i^T \mathbf{e}_{1,j} + (e_3)_{ij} - \mathbf{e}_{2,i}^T \mathbf{s}_j}_{\text{noise}} + \frac{q}{2}\mu_{ij}\end{aligned}$$

where $\mathbf{r}_i^T$ and $\mathbf{e}_{2,i}^T$ are the i-th rows of $\mathbf{R}$ and $\mathbf{E}_2$, and $\mathbf{s}_j$ and $\mathbf{e}_{1,j}$ are the j-th columns of $\mathbf{S}$ and $\mathbf{E}_1$. This is *exactly* the same noise expression as the single-bit scheme (our Equation 1), so the parameters m, q, β are set the same way. The decryption error is now per coefficient of $\mathbf{M}$ (and these events are not independent), so the *total* decryption error is bounded by a union bound over all $k\ell$ coefficients (footnote 8 of the paper).

8.1.3 The Security Proof: Full Reduction with Hybrid Argument

Theorem (Amortized Scheme)

If $\mathrm{LWE}_{m,q,\beta}$ is hard, then the amortized encryption scheme is CPA-secure:

$$\mathrm{Adv}^{\mathrm{CPA}}(\mathcal{A}) \;\le\; \ell \cdot \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}(\mathcal{D}_1) \;+\; k \cdot \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}(\mathcal{D}_2) \;\le\; (\ell + k) \cdot \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}\,.$$

This is a security loss of $\log_2(\ell + k)$ bits compared to the single-bit scheme.

Proof. We proceed through game hops, but now each "hop" is itself a sequence of ℓ (or k) sub-hops.

Game 0 – The Real CPA Game

The challenger runs the honest amortized scheme. The adversary $\mathcal{A}$ receives:

$$\begin{aligned}\mathrm{pk} &= (\mathbf{A},\ \mathbf{T} = [\mathbf{A}\mathbf{s}_1 + \mathbf{e}_{1,1} \mid \cdots \mid \mathbf{A}\mathbf{s}_\ell + \mathbf{e}_{1,\ell}]) \\ \mathrm{ct} &= (\mathbf{U} = \mathbf{R}\mathbf{A} + \mathbf{E}_2,\ \ \mathbf{V} = \mathbf{R}\mathbf{T} + \mathbf{E}_3 + \tfrac{q}{2}\mathbf{M}_b)\end{aligned}$$

where $\mathbf{M}_b$ is the challenge message (the adversary chose $\mathbf{M}_0, \mathbf{M}_1$ and the challenger encrypted $\mathbf{M}_b$ for a random bit b).

Game 1 – Replace the public key columns with random (hybrid over ℓ steps)

Goal: Show that $(\mathbf{A}, \mathbf{T})$ is indistinguishable from $(\mathbf{A}, \mathbf{U}_{\text{rand}})$ where $\mathbf{U}_{\text{rand}} \xleftarrow{\$} \mathbb{Z}_q^{m\times\ell}$.
The column-by-column hybrid. Define intermediate distributions:

$$H_j \ : \ \big(\mathbf{A},\ \underbrace{\mathbf{u}_1, \ldots, \mathbf{u}_j}_{\text{random}},\ \underbrace{\mathbf{A}\mathbf{s}_{j+1} + \mathbf{e}_{1,j+1}, \ldots, \mathbf{A}\mathbf{s}_\ell + \mathbf{e}_{1,\ell}}_{\text{real LWE}}\big)$$

for $j = 0, 1, \ldots, \ell$, where each $\mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_q^m$.

- H_0 is the real public key (Game 0).
- H_ℓ is the fully random public key (Game 1).
- Adjacent hybrids H_{j-1} and H_j differ in **exactly one column**: the j-th column is $\mathbf{A}\mathbf{s}_j + \mathbf{e}_{1,j}$ in H_{j-1} and $\mathbf{u}_j$ in H_j.

Reduction from H_{j-1} vs. H_j to LWE: Given an LWE challenge $(\mathbf{A}, \mathbf{z})$ where $\mathbf{z}$ is either $\mathbf{A}\mathbf{s} + \mathbf{e}$ or random, the distinguisher $\mathcal{D}_1$ builds the public key by:

1. Columns $1, \ldots, j-1$: sample $\mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_q^m$ (random).
2. Column j: plug in $\mathbf{z}$ from the LWE challenge.
3. Columns $j+1, \ldots, \ell$: sample fresh $\mathbf{s}_i, \mathbf{e}_{1,i}$ and compute $\mathbf{A}\mathbf{s}_i + \mathbf{e}_{1,i}$.

If $\mathbf{z} = \mathbf{A}\mathbf{s} + \mathbf{e}$, the adversary sees H_{j-1}. If $\mathbf{z}$ is random, the adversary sees H_j. So:

$$|\Pr[H_{j-1}] - \Pr[H_j]| \ \le \ \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}(\mathcal{D}_1).$$

By the triangle inequality:

$$|\Pr[\text{Game 0}] - \Pr[\text{Game 1}]| = |\Pr[H_0] - \Pr[H_\ell]| \le \sum_{j=1}^{\ell} |\Pr[H_{j-1}] - \Pr[H_j]| \le \ell \cdot \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}\,.$$

Game 2 – Replace the ciphertext rows with random (hybrid over k steps)

After Game 1, the full public key matrix $\mathbf{A}' = [\mathbf{A} \mid \mathbf{T}] \in \mathbb{Z}_q^{m\times(m+\ell)}$ is **uniformly random**. The ciphertext is:

$$[\mathbf{U} \mid \mathbf{V}] = \mathbf{R}\mathbf{A}' + [\mathbf{E}_2 \mid \mathbf{E}_3 + \tfrac{q}{2}\mathbf{M}_b]$$

Each *row* i of this expression is an independent LWE sample: $\mathbf{r}_i^T \mathbf{A}' +$ (short noise).
The row-by-row hybrid. Define:

$$G_i \ : \ \text{rows } 1, \ldots, i \text{ are random; rows } i+1, \ldots, k \text{ are real LWE.}$$

By the same argument as above:

$$|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]| \le \sum_{i=1}^{k} |\Pr[G_{i-1}] - \Pr[G_i]| \le k \cdot \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}\,.$$

Game 2: Adversary has zero advantage

In Game 2, the ciphertext $(\mathbf{U}, \mathbf{V})$ is uniformly random, independent of $\mathbf{M}_b$. So $\Pr[\text{Game 2 outputs } b] = 1/2$.

Combining:

$$\mathrm{Adv}^{\mathrm{CPA}}(\mathcal{A}) = \left|\Pr[\text{Game } 0] - \tfrac{1}{2}\right| \le \underbrace{\ell \cdot \mathrm{Adv}^{\mathrm{LWE}}}_{\text{Hop 1: pk}} + \underbrace{k \cdot \mathrm{Adv}^{\mathrm{LWE}}}_{\text{Hop 2: ct}} = (\ell + k) \cdot \mathrm{Adv}^{\mathrm{LWE}}_{m,q,\beta}\,. \qquad \square$$

8.1.4 What the Security Loss Means

If the LWE problem offers λ bits of security (i.e. $\mathrm{Adv}^{\mathrm{LWE}} \le 2^{-\lambda}$), then the single-bit scheme gives:

$$\mathrm{Adv}^{\mathrm{CPA}} \le 2 \cdot 2^{-\lambda} \approx 2^{-\lambda} \qquad \text{(two LWE invocations)}.$$

The amortized scheme gives:

$$\mathrm{Adv}^{\mathrm{CPA}} \le (\ell + k) \cdot 2^{-\lambda} = 2^{\log_2(\ell+k)-\lambda}.$$

So the *effective* security is $\lambda - \log_2(\ell + k)$ bits instead of λ bits. (The paper states a loss of $\log \ell$ bits for the public-key hybrid; the extra k comes from our explicit hybrid over the k rows of the ciphertext.) Concretely:

- If $\ell = k = 4$ (encrypt $N = 16$ bits), you lose $\log_2(8) = 3$ bits of security.
- If $\ell = k = 16$ (encrypt $N = 256$ bits), you lose $\log_2(32) = 5$ bits.

Since practical LWE parameters target $\lambda \ge 128$ and $\ell + k$ is at most a few dozen, the loss is negligible in practice.

Is the loss real or an artifact?

The factor-ℓ loss comes from the triangle inequality: we *upper bound* the total advantage by ℓ times the per-column advantage. This is a worst-case bound. In reality, the ℓ columns use *independent* secrets $\mathbf{s}_j$, so an adversary exploiting column j gains no advantage on column j'. A tighter analysis might avoid the factor entirely, but proving this would require arguing about the *joint* distribution of all columns simultaneously rather than one at a time. As Lyubashevsky notes (footnote 7): "it's not clear in this case whether there is a real security loss or it's just an artifact of the proof."

Why the hybrid argument works column-by-column

Think of it like a combination lock with ℓ dials. We want to argue that a lock set to $(\mathrm{real}_1, \ldots, \mathrm{real}_\ell)$ looks the same as $(\mathrm{random}_1, \ldots, \mathrm{random}_\ell)$. Rather than arguing about all dials at once, we change them one at a time:

$$(\mathrm{real}, \mathrm{real}, \ldots) \to (\mathrm{rand}, \mathrm{real}, \ldots) \to (\mathrm{rand}, \mathrm{rand}, \ldots) \to \cdots \to (\mathrm{rand}, \ldots, \mathrm{rand})$$

If you can't tell the difference at any single step (because each step *is* an LWE instance), you can't tell the difference overall—but your inability to detect each step accumulates, hence the factor ℓ.

8.2 Removing the Low-Order Part (Section 2.5.1)

The ciphertext part $\mathbf{V}$ costs $N \log q$ bits. Can the encryptor send only κ bits per coefficient instead of $\log q$? Yes—at the price of one more error term in decryption.

8.2.1 The Picture: Points on a Circle

View $\mathbb{Z}_q$ as q points on a circle. Pick a subset $\mathcal{S} \subset \mathbb{Z}_q$ of size 2^κ whose points are spread as evenly as possible, so that the largest gap between neighbouring points of $\mathcal{S}$ is as small as possible. The best we can hope for is gaps of $q/2^\kappa$, and the paper's choice is (Eq. (18))

$$\mathcal{S} = \left\{ \lceil i \cdot q/2^\kappa \rfloor \ : \ 0 \le i < 2^\kappa \right\}.$$

If $2^\kappa \mid q$ all gaps are equal; otherwise they differ by at most 1, which is as good as possible. The crucial property: *every* $v \in \mathbb{Z}_q$ is within distance $\lceil q/2^{\kappa+1} \rceil$ of some element of $\mathcal{S}$ (half a gap).

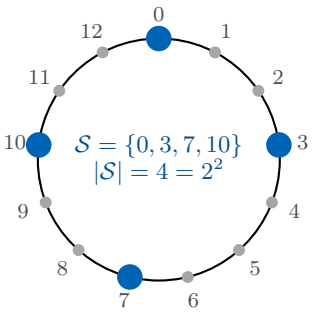


Figure 4: The paper's Figure 1: $\mathbb{Z}_{13}$ as points on a circle. With $\mathcal{S} = \{\lceil i \cdot 13/4 \rfloor : 0 \le i < 4\} = \{0, 3, 7, 10\}$ (large dots), $\mathcal{S}$ can be represented with 2 bits, and every point of $\mathbb{Z}_{13}$ is within distance $\lceil 13/8 \rceil = 2$ of an element of $\mathcal{S}$.

Checking the figure (spelled out). $i \cdot 13/4$ for $i = 0, 1, 2, 3$ is 0, 3.25, 6.5, 9.75, which rounds to $0, 3, 7, 10$ (rounding 6.5 up). The gaps are $3, 4, 3, 3$ (the last one wraps from 10 to $13 \equiv 0$). The farthest any point can be from $\mathcal{S}$ is half the largest gap, which is 2 (the point 5 sits exactly between 3 and 7).

8.2.2 HIGH and LOW

Definition: splitting v into a high part and a low part

- $\mathrm{HIGH}_{\mathcal{S}}(v)$ is the element of $\mathcal{S}$ closest to v.
- $\mathrm{LOW}_{\mathcal{S}}(v) = v - \mathrm{HIGH}_{\mathcal{S}}(v)$, which has magnitude at most $\lceil q/2^{\kappa+1} \rceil$.

Applied coefficient-wise to a matrix. In the $\mathbb{Z}_{13}$ example: $\mathrm{HIGH}(9) = 10$, $\mathrm{LOW}(9) = -1$; $\mathrm{HIGH}(12) = 0$, $\mathrm{LOW}(12) = 12 \equiv -1$.

Instead of sending $\mathbf{V} \in \mathbb{Z}_q^{k \times \ell}$, the encryptor sends $\mathbf{V}' = \mathrm{HIGH}_{\mathcal{S}}(\mathbf{V}) \in \mathcal{S}^{k \times \ell}$, which costs only κ bits per coefficient. Writing $\mathbf{E}' = \mathrm{LOW}_{\mathcal{S}}(\mathbf{V}) \in \left[\lceil q/2^{\kappa+1} \rceil\right]^{k \times \ell}$, we have $\mathbf{V} = \mathbf{V}' + \mathbf{E}'$, so decryption now computes (the paper's Eq. (19))

$$\mathbf{V}' - \mathbf{U}\mathbf{S} \ = \ (\mathbf{V} - \mathbf{U}\mathbf{S}) - \mathbf{E}' \ = \ \mathbf{R}\mathbf{E}_1 + \mathbf{E}_3 - \mathbf{E}' + \tfrac{q}{2}\mathbf{M} - \mathbf{E}_2\mathbf{S}.$$

The only change from the uncompressed decryption is the extra $-\mathbf{E}'$.

Why dropping low-order bits of V is almost free

Each coefficient of $\mathbf{E}'$ enters the noise *once*, as a single term. The other error terms $\mathbf{RE}_1$ and $\mathbf{E}_2\mathbf{S}$ are inner products of length m, which produce much larger coefficients. So in practice κ can be a small constant like 3 or 4 without increasing the decryption error much, and $\mathbf{V}$ shrinks from $N \log q$ bits to just κN bits. Assuming $\mathbf{V}$ is uniformly distributed, the exact distribution of $\mathbf{E}'$ is known, and the exact decryption error probability can be computed with the polynomial technique of §5.4.

Compressing U is harder

The same trick can be applied to $\mathbf{U}$, but any error added to $\mathbf{U}$ gets *multiplied by* $\mathbf{S}$ during decryption, giving an additional term $\mathbf{E}''\mathbf{S}$ in (19). That term is an inner product of length m, so it adds noise of the same order as $\mathbf{E}_2\mathbf{S}$. But $\mathbf{U}$ is the dominant part of the ciphertext, so even a small reduction in its bits can matter. The paper's advice: balance decryption error against ciphertext size by trial and error.

8.3 Modulus Switching / Compression / Decompression (Section 2.5.2)

Section 2.5.1 described compression with a set $\mathcal{S}$. The paper now makes it concrete with a single rounding map between moduli. Mapping to a smaller set is rounding (compression); mapping to a larger set is lifting (decompression).

Definition 3 (paper): modulus switching

For $x \in \mathbb{Z}_q$ and a positive integer p,

$$\lceil x \rfloor_{q \to p} = \left\lceil \frac{x \cdot p}{q} \right\rfloor \in \mathbb{Z}_p.$$

Well-defined (spelled out). An element of $\mathbb{Z}_q$ is really a class $x + q\mathbb{Z}$, so we must check that every representative gives the same answer in $\mathbb{Z}_p$. Replacing x by $x + qk$ gives $\frac{(x+qk)p}{q} = \frac{xp}{q} + pk$. Adding the integer pk commutes with rounding, and $pk \equiv 0 \pmod p$. So the result in $\mathbb{Z}_p$ is the same.

Lemma 1 (paper): compress-then-decompress is close to the identity

For integers $p < q$ and $x \in \mathbb{Z}_q$,

$$\left\lceil \lceil x \rfloor_{q \to p} \right\rfloor_{p \to q} = x + \eta \in \mathbb{Z}_q \qquad \text{for some } \eta \in \mathbb{Z} \text{ with } |\eta| \le \frac{q}{2p} + \frac{1}{2}.$$

Proof (the paper's, with every step shown).

1. Rounding moves a number by at most $\frac{1}{2}$, so $\lceil x \rfloor_{q \to p} = \frac{xp}{q} + \delta$ for some $\delta \in \mathbb{Q}$, $|\delta| \le \frac{1}{2}$.
2. Decompress this value:

$$\left\lceil \lceil x \rfloor_{q \to p} \right\rfloor_{p \to q} = \left\lceil \frac{\left(\frac{xp}{q} + \delta\right) q}{p} \right\rfloor = \left\lceil x + \frac{\delta q}{p} \right\rfloor = x + \frac{\delta q}{p} + \delta',$$

   where $|\delta'| \le \frac{1}{2}$ is the second rounding error.
3. So $\eta = \frac{\delta q}{p} + \delta'$ and $|\eta| \le \frac{q}{2p} + \frac{1}{2}$. □

In words: the first rounding loses at most $\frac{1}{2}$ *in units of* q/p; scaling back up turns that into at most $\frac{q}{2p}$; the second rounding adds at most $\frac{1}{2}$.

Illustration (our computation): Lemma 1 for $q = 13$, $p = 4$

Every $x \in \mathbb{Z}_{13}$, compressed to $\mathbb{Z}_4$ and decompressed back:

x	0	1	2	3	4	5	6	7	8	9	10	11	12
$\lceil x \rfloor_{13\to 4}$	0	0	1	1	1	2	2	2	2	3	3	3	0
back to $\mathbb{Z}_{13}$	0	0	3	3	3	7	7	7	7	10	10	10	0
η	0	−1	1	0	−1	2	1	0	−1	1	0	−1	1

The decompressed values are exactly $\mathcal{S} = \{0, 3, 7, 10\}$ of our Figure 4, and $\max|\eta| = 2 \le \frac{13}{8} + \frac{1}{2} = 2.125$.

8.3.1 Connecting the Two Views

The paper links Definition 3 to the $\mathcal{S}$, κ, HIGH, LOW of Section 2.5.1:

1. $2^\kappa = p$;
2. $\mathcal{S} = \{\lceil x \rfloor_{p\to q} : x \in \mathbb{Z}_p\}$;
3. $\mathrm{HIGH}_{\mathcal{S}}(x) = \lceil \lceil x \rfloor_{q\to p} \rfloor_{p\to q}$;
4. $\mathrm{LOW}_{\mathcal{S}}(x) = x - \mathrm{HIGH}_{\mathcal{S}}(x)$.

Lemma 1 then says $\mathrm{LOW}_{\mathcal{S}}(x) \in [\lceil q/2p \rceil]$. In an implementation one sends the short value $\lceil x \rfloor_{q\to p} \in \mathbb{Z}_p$ (only $\log p$ bits), not $\mathrm{HIGH}_{\mathcal{S}}(x) \in \mathbb{Z}_q$. Think of $\mathrm{HIGH}_{\mathcal{S}}(x)$ as "the decompression of the compression of x."

Decryption is compression to one bit. For $x \in \mathbb{Z}_q$, $\lceil x \rfloor_{q\to 2}$ is 0 when x is closer to 0 than to $q/2$, and 1 otherwise. (Spelled out: $2x/q$ lies in $[0, 2)$; it rounds to 0 for x near 0, to 1 for x near $q/2$, and to $2 \equiv 0 \pmod 2$ for x near q, which is again "near 0" on the circle.) So the decryption rule can be written in one line (the paper's Eq. (20)):

$$\mu = \lceil v - \mathbf{u}^T \mathbf{s} \rfloor_{q\to 2}.$$

8.3.2 Other Choices of $\mathcal{S}$

For a fixed size $|\mathcal{S}|$, item 2 is optimal for every p, q (it minimises the largest distance to $\mathcal{S}$). For particular p, q one can pick a different, equally good $\mathcal{S}$ that avoids dividing by q (usually a prime). The paper's example: $p = 4$, $q = 33$, $\mathcal{S} = \{0, 8, 16, 24\}$. Computing $\mathrm{HIGH}_{\mathcal{S}}$ then needs only reductions and divisions modulo 8, which are register shifts on a CPU. (Spelled out: the gaps are $8, 8, 8, 9$, which is as even as $33/4 = 8.25$ allows.)

The paper previews how this plays out later: in Kyber (Section 4.7) q is small, and no prime satisfies the condition needed for fast multiplication (Section 4.6) while also admitting a "nice" $\mathcal{S}$, so Kyber uses the generic $\mathcal{S}$ of item 2. In Dilithium (Section 5.7) q is larger, and a nice $\mathcal{S}$ and fast multiplication are both possible.

8.4 Learning with Rounding (Section 2.5.3)

8.4.1 The Key Observation: Noise Can Be Redundant

By the LWE assumption, $(\mathbf{A},\ \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$ looks uniform. If we round $\mathbf{t}$ to the nearest element of $\mathcal{S} \subseteq \mathbb{Z}_q$, then $(\mathbf{A},\ \mathrm{HIGH}_{\mathcal{S}}(\mathbf{t}))$ is still indistinguishable from $(\mathbf{A},\ \mathrm{HIGH}_{\mathcal{S}}(\mathbf{u}))$ for random $\mathbf{u}$.

But now look more carefully at $\mathbf{t} = \mathbf{As} + \mathbf{e}$. Since $\mathbf{e}$ has small coefficients in $[\beta]$, it might not change the rounded output at all:

$$\mathrm{HIGH}_{\mathcal{S}}(\mathbf{As} + \mathbf{e}) = \mathrm{HIGH}_{\mathcal{S}}(\mathbf{As}) \quad \text{with high probability.}$$

When this holds, the explicit noise $\mathbf{e}$ is **redundant**—the rounding itself provides all the "noise" needed for security.

8.4.2 When Does Rounding Absorb the Noise?

Consider a single coefficient of $\mathbf{As}$. The rounding $\mathrm{HIGH}_{\mathcal{S}}$ partitions $\mathbb{Z}_q$ into $|\mathcal{S}|$ bins, each of width $q/|\mathcal{S}|$. Adding $e \in [\beta]$ can only change the bin when $\mathbf{As}$ is within β of a bin boundary (i.e. a midpoint between adjacent elements of $\mathcal{S}$).

Counting the bad cases (the paper's argument, spelled out). Fix one coefficient a of $\mathbf{As}$ and one of the $|\mathcal{S}|$ midpoints. Suppose a sits at distance $\beta - i$ from that midpoint, for some $i \in \{1, \ldots, \beta\}$. Of the $2\beta + 1$ possible values of $e \in [\beta]$, exactly i push $a + e$ across the midpoint (those pointing towards it with magnitude larger than $\beta - i$). For each i there are two such positions per midpoint (one on each side), so $2|\mathcal{S}|$ positions in total. If a is uniform in $\mathbb{Z}_q$, each position has probability $1/q$. Summing over i:

$$\Pr\big[\mathrm{HIGH}_{\mathcal{S}}(a + e) \neq \mathrm{HIGH}_{\mathcal{S}}(a)\big] \;=\; \frac{2|\mathcal{S}|}{q} \sum_{i=1}^{\beta} \frac{i}{2\beta + 1} \;=\; \frac{|\mathcal{S}|\,\beta(\beta+1)}{q(2\beta+1)} \;\approx\; \frac{\beta \cdot |\mathcal{S}|}{2q}.$$

If we further assume that all n coefficients of $\mathbf{As}$ are independent, the probability that *all* of them match is (Section 2.5.3 of the paper):

$$\Pr\big[\mathrm{HIGH}_{\mathcal{S}}(\mathbf{As} + \mathbf{e}) = \mathrm{HIGH}_{\mathcal{S}}(\mathbf{As})\big] \;\approx\; \left(1 - \frac{\beta|\mathcal{S}|}{2q}\right)^{n}.$$

So whenever q is large with respect to β and $|\mathcal{S}|$, adding $\mathbf{e}$ makes no difference—and there was no reason to add it in the first place.

The LWR Problem

The **Learning with Rounding** (LWR) problem asks: given $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, distinguish

$$(\mathbf{A},\ \mathrm{HIGH}_{\mathcal{S}}(\mathbf{As})) \quad \text{from} \quad (\mathbf{A},\ \mathrm{HIGH}_{\mathcal{S}}(\mathbf{u}))$$

where $\mathbf{s} \xleftarrow{\$} [\beta]^m$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$ (Section 2.5.3 of the paper).
No explicit noise is added—the rounding *is* the noise.

8.4.3 Reduction from LWE to LWR

When rounding absorbs the noise (i.e. $\mathrm{HIGH}_{\mathcal{S}}(\mathbf{As}+\mathbf{e}) = \mathrm{HIGH}_{\mathcal{S}}(\mathbf{As})$ with high probability), there is a chain of indistinguishabilities:

$$(\mathbf{A},\ \mathrm{HIGH}_{\mathcal{S}}(\mathbf{As})) \;\approx\; (\mathbf{A},\ \mathrm{HIGH}_{\mathcal{S}}(\underbrace{\mathbf{As}+\mathbf{e}}_{\text{LWE instance}})) \;\approx_c\; (\mathbf{A},\ \mathrm{HIGH}_{\mathcal{S}}(\mathbf{u}))$$

The first step is statistical (rounding absorbs $\mathbf{e}$). The second step is computational (the LWE assumption). Together: LWR is at least as hard as LWE for these parameters.

When the reduction fails

The reduction requires q to be large with respect to β and $|\mathcal{S}|$. The paper notes that LWR is sometimes used in constructions "even when the parameters do not permit a reduction from LWE (i.e. q is not large enough)." In that case LWR is a **separate assumption**, and its relationship with LWE is unclear. (Kyber *not* do this: Section 4.7 of the paper bases its security on the hardness of a generalized-LWE problem over the ring $R_{3329,X^{256}+1}$.)

8.4.4 The Implicit Noise Viewpoint

Even without an explicit $\mathbf{e}$, there *is* noise—it's just deterministic. We can write:

$$\mathrm{HIGH}_{\mathcal{S}}(\mathbf{As}) = \mathbf{As} - \mathrm{LOW}_{\mathcal{S}}(\mathbf{As})$$

so the "implicit noise" is $\mathbf{e}_{\mathrm{impl}} = -\mathrm{LOW}_{\mathcal{S}}(\mathbf{As})$: the low-order bits that rounding discards. The best known attacks against LWR treat it as an LWE instance with this implicit noise. Crucially, the norm of $\mathbf{e}_{\mathrm{impl}}$ is typically *larger* than the noise from the LWE reduction, so LWR is believed to be at least as hard as LWE in practice.

8.4.5 Why LWR Matters: Rounding Does Double Duty

Recall the amortized ciphertext from §8.1:

$$\mathbf{V} = \mathbf{RT} + \mathbf{E}_3 + \tfrac{q}{2}\mathbf{M}$$

After compression, $\mathbf{V}$ acquires a rounding error $\mathbf{E}'$. The total noise in the compressed ciphertext is $\mathbf{E}_3 + \mathbf{E}'$.

Under the LWR assumption, the rounding error $\mathbf{E}'$ already provides sufficient noise for security. So $\mathbf{E}_3$ may be **unnecessary**—we can set $\mathbf{E}_3 = \mathbf{0}$ and rely on rounding alone. The same argument applies to $\mathbf{E}_2$ in $\mathbf{U} = \mathbf{RA} + \mathbf{E}_2$: rounding $\mathbf{U}$ to a smaller set makes $\mathbf{E}_2$ redundant.

Rounding as a noisy channel

Think of rounding like a lossy audio codec: when you compress a digital signal, the quantization itself introduces noise. If you were already planning to add dither (random noise) for security, but the codec's own quantization noise is large enough, then the separate dither step is redundant—the compression is simultaneously shrinking the data *and* providing the noise you needed. That is exactly what LWR exploits: compression and noise generation in one operation.

8.5 Encrypting More Bits per Slot (Section 2.5.4)

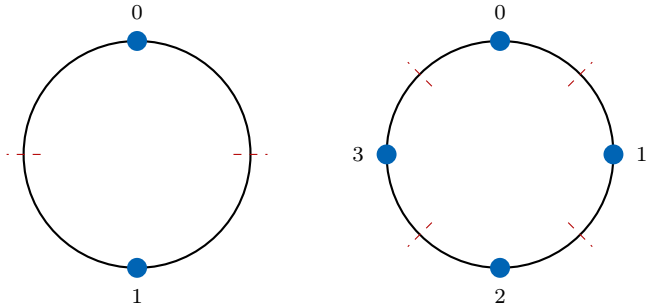
So far every coefficient of $\mathbf{M} \in \{0,1\}^{k\times\ell}$ carries one bit, encoded as 0 or $q/2$ on the circle. We could instead put b bits in each slot, packing the same $N = k\ell$ bits into

$$\mathbf{M} \in \{0, 1, \ldots, 2^b - 1\}^{(k/\sqrt{b})\times(\ell/\sqrt{b})}.$$

Two changes are needed:

- **Encryption** uses $\frac{q}{2^b}\mathbf{M}$ instead of $\frac{q}{2}\mathbf{M}$, placing the 2^b possible messages evenly around the circle.

- **Decryption** then sees $\frac{q}{2^b}\mathbf{M}$ in place of $\frac{q}{2}\mathbf{M}$ in Eq. (17), so the remaining error terms must lie in $[q/2^{b+1}]$ rather than $[q/4]$.



$b = 1$: decision regions of width $q/2$, noise $< q/4$

$b = 2$: decision regions of width $q/4$, noise $< q/8$

Figure 5: Illustration (ours): more bits per slot means more message points on the circle, so each decision region shrinks and the tolerable noise halves with every extra bit.

The trade-off. The simplest way to make decryption work again is to increase q by a factor of about 2^{b-1}, which keeps the *relative* noise $q/2^{b+1}$ the same as $q/4$ before. That can still *shrink* the keys. The public key in (13) has $256 + \ell m \log q$ bits. If q grows by 2^{b-1} but ℓ halves, the size becomes

$$256 + \frac{\ell m}{2}(\log q + b - 1),$$

which is smaller whenever $b - 1 < \log q$. (Spelled out: ℓ halves when, for instance, $b = 4$, since each matrix dimension shrinks by $\sqrt{b} = 2$. The comparison is $\frac{1}{2}(\log q + b - 1) < \log q \iff b - 1 < \log q$.)

Bigger q costs security

Increasing q while keeping everything else fixed makes LWE *easier*: the problem gets harder as the ratio β/q grows (Section 2.3), and here β/q shrinks. To compensate, one must increase β or m. The paper's advice: try a few options, "or better yet, write a script that does this for you."

8.6 LWE Encryption with a "Non-Square" Public Key (Section 2.5.5)

Every security proof so far used LWE *twice*: once to argue the public key looks uniform, and once more to argue the ciphertext looks uniform. Sometimes we want the public key (or the ciphertext) to be *truly* uniform, not just computationally, so that LWE is invoked only once.

Leftover hash lemma (as the paper states it, roughly)

If $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ with q prime, $\mathbf{s} \leftarrow [\beta]^m$, and $(2\beta+1)^m \gg q^n$, then $(\mathbf{A}, \mathbf{As})$ is statistically close to $(\mathbf{A}, \mathbf{u})$ with $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
In words: if $\mathbf{s}$ is uniform over a set that is much larger than the range of $\mathbf{s} \mapsto \mathbf{As}$, the output looks uniform.

Intuition (spelled out). The map $\mathbf{s} \mapsto \mathbf{As}$ goes from $(2\beta + 1)^m$ inputs to q^n outputs. If there are far more inputs than outputs, a random $\mathbf{A}$ spreads them almost evenly, so each output is hit roughly equally often. Compare LWE (Section 2.3), where $\mathbf{A}$ is square: there are *fewer* short secrets than outputs, $\mathbf{As}$ covers only a tiny, structured subset of $\mathbb{Z}_q^m$, and the noise $\mathbf{e}$ plus a computational assumption is what hides it.

Uniform public key (the paper's Eq. (21))

$$\mathsf{sk} : \mathbf{S} \leftarrow [\beta']^{m\times\ell}, \qquad \mathsf{pk} : \big(\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m},\ \mathbf{T} = \mathbf{AS}\big), \qquad \text{where } (2\beta'+1)^m > q^n.$$

Encryption and decryption stay exactly as in (14) and (17).

Things to notice:

- There is no $\mathbf{E}_1$: the public key is uniform by the leftover hash lemma, not by LWE. So the term $\mathbf{RE}_1$ in (17) is 0.
- Since LWE is no longer used twice, there is no reason for the key-generation bound to equal the encryption bound β. Hence the new name β'.
- Decryption is harder to get right: $\mathbf{E}_2\mathbf{S}$ in (17) grows, because m and/or β' must be larger to satisfy $(2\beta'+1)^m > q^n$.

Uniform ciphertext instead. The same principle applies after using LWE once to argue that the public key is indistinguishable from random. One chooses the dimension of $\mathbf{A}$ and the distribution of $\mathbf{R}$ so that, for uniform $[\mathbf{A} \mid \mathbf{T}] \leftarrow \mathbb{Z}_q^{n\times m}$,

$$\big([\mathbf{A} \mid \mathbf{T}],\ \mathbf{R}[\mathbf{A} \mid \mathbf{T}]\big) \quad \text{is indistinguishable from} \quad \big([\mathbf{A} \mid \mathbf{T}],\ [\mathbf{U} \mid \mathbf{V}]\big), \qquad \mathbf{U}, \mathbf{V} \text{ uniform.}$$

Why would anyone want this? The paper gives two examples, "by no means exhaustive":

- **Identity-based encryption [GPV08].** The public key of identity $x \in \{0,1\}^*$ is $(\mathbf{A}, \mathbf{t}_x)$ with $\mathbf{t}_x = \mathcal{H}(x)$ for a hash function $\mathcal{H}$ (modelled as a random oracle) into $\mathbb{Z}_q^n$. Anyone can compute it, so it cannot be derived from a secret key as in (6). Instead, a master authority holds a *trapdoor* for $\mathbf{A}$ and uses it to find a short $\mathbf{s}_x$ with $\mathbf{A}\mathbf{s}_x = \mathbf{t}_x$. Interestingly, key generation is *not* done as in (21): the secret key is created *after* the public key, which is only possible because the authority created $\mathbf{A}$ together with its trapdoor. The main result of [GPV08] gives algorithms such that the distribution of $\mathbf{A}, \mathbf{s}, \mathbf{t}$ is the same whether $\mathbf{s}$ is chosen before computing $\mathbf{t}$ or the other way round. (Spelled out: this is exactly where a uniform public key is needed, since $\mathbf{t}_x = \mathcal{H}(x)$ is uniformly random.)
- **Leakage of the encryption randomness [AGV09].** If something about $\mathbf{r}$ leaks, the leftover hash lemma still makes the ciphertext uniformly random, because enough entropy is left in $\mathbf{r}$.

What discrete log gives for free

In ElGamal, the public key g^s for uniform $s \in \mathbb{Z}_p$ is *exactly* uniform in the group, because $s \mapsto g^s$ is a bijection. No lemma needed. In lattices, short secrets are a small subset, so $\mathbf{As}$ is uniform only when there are enough of them—that is precisely the condition $(2\beta'+1)^m > q^n$, and it is paid for with a wider $\mathbf{A}$ or larger coefficients.

8.7 Different Distributions for Secret and Error (Section 2.5.6)

Another optimization [ZYF+20]: sample $\mathbf{r}, \mathbf{s} \leftarrow \psi_1$ and $\mathbf{e}_1, \mathbf{e}_2 \leftarrow \psi_2$ with $\psi_1 \neq \psi_2$.

The reasoning. Against the best known algorithms, the hardness of distinguishing $(\mathbf{A}, \mathbf{As}+\mathbf{e}_1)$ from uniform depends on the *norm* of $(\mathbf{s}, \mathbf{e}_1)$ (Section 3.3). So for a fixed total norm we are free to decide how to split it between $\mathbf{s}$ and $\mathbf{e}_1$. (This is no longer LWE exactly as defined

in Section 2.3, but the paper calls it "perhaps also a reasonable assumption.") Meanwhile, decryption needs the noise (the paper's Eq. (22))

$$\mathbf{r}^T\mathbf{e}_1 + \mathbf{e}_2^T\mathbf{s}$$

to be small, since smaller noise allows a smaller q: the LWE problem becomes harder and the public key shrinks.

Two Gaussian facts the paper uses (all vectors drawn from zero-centred normal distributions $\mathcal{N}_\sigma$; the paper ignores that these are continuous):

1. If $\mathbf{v} \leftarrow \mathcal{N}_\sigma^n$, then $\|\mathbf{v}\|$ is tightly concentrated around $\sigma\sqrt{n}$.
2. If $\mathbf{s} \leftarrow \mathcal{N}_\sigma^n$ and $\mathbf{v} \in \mathbb{R}^n$ is fixed, then $\langle \mathbf{s}, \mathbf{v}\rangle \sim \mathcal{N}_{\sigma\|\mathbf{v}\|}$.

Putting them together (spelled out). Take $\mathbf{s}, \mathbf{r} \leftarrow \mathcal{N}_{\sigma_1}^n$ and $\mathbf{e}_1, \mathbf{e}_2 \leftarrow \mathcal{N}_{\sigma_2}^n$. By fact 1, $\|\mathbf{e}_1\| \approx \sigma_2\sqrt{n}$. So by fact 2, $\mathbf{r}^T\mathbf{e}_1 \sim \mathcal{N}_{\sigma_1\sigma_2\sqrt{n}}$, and likewise for $\mathbf{e}_2^T\mathbf{s}$. Two independent normals add their *variances*: $2\sigma_1^2\sigma_2^2 n$. So (22) is roughly $\mathcal{N}_{\sigma_1\sigma_2\sqrt{2n}}$. The norm of $(\mathbf{s}, \mathbf{e}_1)$ is $\approx \sqrt{\sigma_1^2 + \sigma_2^2} \cdot \sqrt{n}$.

	$\|(\mathbf{s}, \mathbf{e}_1)\| = \|(\mathbf{r}, \mathbf{e}_2)\|$	std. dev. of (22)
$\sigma_1 = \sigma_2 = \sigma$ (Eq. (23))	$\sigma\sqrt{2n}$	$\sigma^2\sqrt{2n}$
$\sigma_1 = \frac{4}{3}\sigma$, $\sigma_2 = \frac{\sqrt{2}}{3}\sigma$ (Eq. (24))	$\sigma\sqrt{2n}$	$\frac{4\sqrt{2}}{9}\sigma^2\sqrt{2n} \approx 0.63\,\sigma^2\sqrt{2n}$

(Check: $\sigma_1^2 + \sigma_2^2 = \frac{16}{9}\sigma^2 + \frac{2}{9}\sigma^2 = 2\sigma^2$ in both rows, and $\sigma_1\sigma_2 = \frac{4\sqrt{2}}{9}\sigma^2$ in the second.) Same norm, same (heuristic) hardness, but about 37% less decryption noise.

Why unequal wins (spelled out, not in the paper). With $\sigma_1^2 + \sigma_2^2$ fixed at $2\sigma^2$, the product $\sigma_1\sigma_2$ is *largest* when $\sigma_1 = \sigma_2$ (AM–GM: $\sigma_1\sigma_2 \le \frac{1}{2}(\sigma_1^2 + \sigma_2^2)$). The balanced choice is therefore the *worst* one for decryption noise, and any imbalance helps. Pushing it too far is dangerous, though: in the limit $\sigma_2 \to 0$ the error disappears and LWE collapses to linear algebra (Section 2.2).

It is a belief, not a theorem

Whether to set $\sigma_1 \neq \sigma_2$ "depends on one's belief that the hardness of the two LWE problems (one as in Definition 2 and one where $\mathbf{s}, \mathbf{e}$ have different distributions) is equivalent."

9. Non-Interactive Key Exchange (Section 2.6)

9.1 Key Transport vs. Key Exchange

Any of the encryption schemes above immediately gives a passively-secure **key transport** protocol: party 1 sends a public key (as in (13)); party 2 picks a symmetric key (say, an AES key), encrypts it as the message $\mathbf{M}$, and sends the ciphertext (as in (14)); party 1 decrypts. For most purposes where Diffie–Hellman is used, this is good enough.

But the flow has a fixed order: party 2 cannot send anything until it has party 1's public key. In Diffie–Hellman, either party can send g^{x_i} first—the two messages are independent. A protocol with that property is **non-interactive**. The paper shows one can be built from LWE, but warns that it is much less efficient, and not worth it when the arbitrary-order property is not needed.

9.2 The Protocol (One Shared Bit)

The paper describes agreeing on one random bit; for more bits one applies the ideas of Section 2.4.

LWE non-interactive key exchange

Public setup: a random $\mathbf{A} \in \mathbb{Z}_q^{m \times m}$ that everyone trusts to be honestly generated (e.g. expanded by SHAKE from the seed 0).
Party $i \in \{1, 2\}$**:** samples $\mathbf{s}_i, \mathbf{e}_i \leftarrow [\beta]^m$ and sends

$$\text{party 1: } \mathbf{u}_1^T = \mathbf{s}_1^T \mathbf{A} + \mathbf{e}_1^T, \qquad \text{party 2: } \mathbf{u}_2 = \mathbf{A}\mathbf{s}_2 + \mathbf{e}_2.$$

Key derivation: party 1 computes $\mathbf{s}_1^T \mathbf{u}_2$, party 2 computes $\mathbf{u}_1^T \mathbf{s}_2$. Each sets its bit $b_i = 1$ if its value is closer to $q/2$ than to 0 (i.e. between $q/4$ and $3q/4$), and $b_i = 0$ otherwise.

Neither message depends on the other, so they can be sent in any order, or at the same time. Expanding (the paper's Eqs. (25)–(26)):

$$\mathbf{s}_1^T \mathbf{u}_2 = \mathbf{s}_1^T \mathbf{A}\mathbf{s}_2 + \mathbf{s}_1^T \mathbf{e}_2,$$
$$\mathbf{u}_1^T \mathbf{s}_2 = \mathbf{s}_1^T \mathbf{A}\mathbf{s}_2 + \mathbf{e}_1^T \mathbf{s}_2.$$

Both parties hold the same core value $\mathbf{s}_1^T \mathbf{A}\mathbf{s}_2$ plus a *different* small error, each of magnitude at most $m\beta^2$.

Diffie–Hellman, with noise

	Diffie–Hellman	LWE NIKE
Public parameter	generator g	matrix $\mathbf{A}$
Party 1 sends	g^{x_1}	$\mathbf{s}_1^T \mathbf{A} + \mathbf{e}_1^T$ ($\mathbf{s}_1$ on the *left*)
Party 2 sends	g^{x_2}	$\mathbf{A}\mathbf{s}_2 + \mathbf{e}_2$ ($\mathbf{s}_2$ on the *right*)
Shared value	$g^{x_1 x_2}$, **exactly** equal	$\mathbf{s}_1^T \mathbf{A}\mathbf{s}_2$ + a different small error on each side

The "left times right" trick is the same one that makes $\mathbf{r}^T \mathbf{A}\mathbf{s}$ cancel in LWE encryption. The difference is that DH commutes exactly ($(g^{x_1})^{x_2} = (g^{x_2})^{x_1}$), while the LWE parties only agree *approximately*, so they must round, and rounding can disagree.

9.3 When Do the Two Bits Disagree?

Write $c = \mathbf{s}_1^T \mathbf{A}\mathbf{s}_2$. The two parties round $c + \mathbf{s}_1^T \mathbf{e}_2$ and $c + \mathbf{e}_1^T \mathbf{s}_2$. If c is more than $m\beta^2$ away from both decision boundaries $q/4$ and $3q/4$, both errors are too small to push either value across a boundary, so $b_1 = b_2$. A mismatch is only possible when c falls into one of the two "dangerous" windows (the paper's Eq. (27), with the interval endpoints written in increasing order):

$$\Pr[b_1 \neq b_2] \;<\; \Pr\left[\mathbf{s}_1^T \mathbf{A}\mathbf{s}_2 \in \left[\tfrac{3q}{4} - m\beta^2,\ \tfrac{3q}{4} + m\beta^2\right] \text{ or } \left[\tfrac{q}{4} - m\beta^2,\ \tfrac{q}{4} + m\beta^2\right]\right].$$

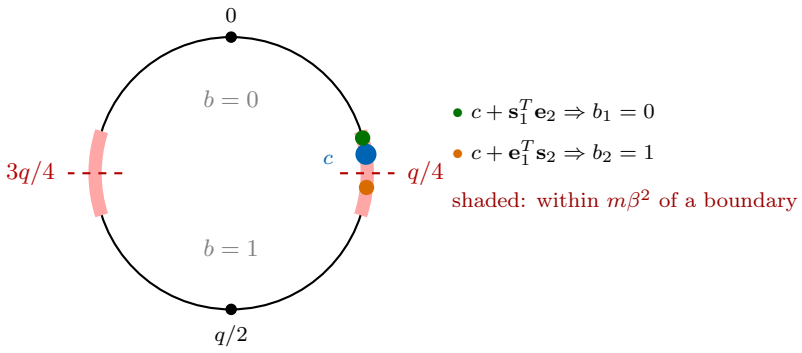


Figure 6: Why NIKE mismatches happen. The core value $c = \mathbf{s}_1^T \mathbf{A} \mathbf{s}_2$ is uniform on the circle, so it sometimes lands next to a decision boundary, and then the two parties' different errors can put their values on opposite sides.

A typo in the paper's Eq. (27)

The paper writes both intervals with their endpoints swapped, as $\left[\frac{3q}{4} + m\beta^2, \frac{3q}{4} - m\beta^2\right]$ and $\left[\frac{q}{4} + m\beta^2, \frac{q}{4} - m\beta^2\right]$. The intended windows are the ones written above, centred on the boundaries.

How likely is the dangerous window? The paper: $\mathbf{s}_1^T \mathbf{A} \mathbf{s}_2$ takes any particular value in $\mathbb{Z}_q$ with probability $1/q$, and the two windows together contain about $4m\beta^2$ values, so $\Pr[b_1 \neq b_2] \lesssim 4m\beta^2/q$. (Spelled out: $\mathbf{s}_1^T \mathbf{A} \mathbf{s}_2 = \sum_{i,j} s_{1,i} A_{ij} s_{2,j}$. If some $s_{1,i} s_{2,j} \neq 0$ is invertible mod q—e.g. q prime and $q > \beta^2$—then the uniform entry A_{ij} makes the whole sum uniform over the choice of $\mathbf{A}$. Each window has $2m\beta^2 + 1$ integer points, so strictly the bound is $(4m\beta^2 + 2)/q$.)

Why NIKE is so much worse than encryption

In encryption, the *encryptor* places the message at 0 or $q/2$—as far as possible from the boundaries—so the noise must be enormous to cause an error, and the error probability can be made exponentially small (or 0). In the NIKE, *no one* chooses the core value: c is uniform, so it lands in a boundary window with probability proportional to the window width divided by q. Using the exact techniques of §5.4 only shrinks the typical error to $O(\beta^2\sqrt{m})$; the mismatch probability stays $\Omega(\beta^2\sqrt{m}/q)$. A negligible mismatch probability therefore needs a very large q, which hurts communication (see [GdKQ+24] for concrete details), and there are technical reasons to believe this inefficiency may be intrinsic to using LWE in the "natural way" [GKRS22].

Worked example (illustration, our numbers): $q = 97$, $m = 2$, $\beta = 1$

Decision rule: $b = 1$ iff the value lies strictly between $q/4 = 24.25$ and $3q/4 = 72.75$, i.e. in $\{25, \ldots, 72\}$.

$$\mathbf{A} = \begin{pmatrix} 41 & 19 \\ 50 & 83 \end{pmatrix}, \quad \mathbf{s}_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \ \mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mathbf{s}_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \ \mathbf{e}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

- Party 1: $\mathbf{s}_1^T \mathbf{A} = (41 - 50,\ 19 - 83) = (-9, -64) \equiv (88, 33)$, so $\mathbf{u}_1^T = (89, 33)$.
- Party 2: $\mathbf{A}\mathbf{s}_2 = (-41 + 19,\ -50 + 83) = (-22, 33) \equiv (75, 33)$, so $\mathbf{u}_2 = (76, 33)$.
- Core: $c = \mathbf{s}_1^T \mathbf{A}\mathbf{s}_2 = (88, 33) \cdot (-1, 1) = -55 \equiv 42$.
- Party 1: $\mathbf{s}_1^T \mathbf{u}_2 = 76 - 33 = 43 = c + \mathbf{s}_1^T \mathbf{e}_2 = 42 + 1$. So $b_1 = 1$.
- Party 2: $\mathbf{u}_1^T \mathbf{s}_2 = -89 + 33 = -56 \equiv 41 = c + \mathbf{e}_1^T \mathbf{s}_2 = 42 - 1$. So $b_2 = 1$.

The parties hold different values (43 vs 41) but the same bit, because $c = 42$ is far from both boundaries. Here $m\beta^2 = 2$, so the paper's bound gives $\Pr[b_1 \neq b_2] \lesssim 4m\beta^2/q = 8/97 \approx 8.2\%$. Averaging over 200 random matrices $\mathbf{A}$, with all secrets and errors enumerated exhaustively, we measured a mismatch rate of about 1.8%. The bound is loose because the errors rarely reach their maximum $m\beta^2$.

9.4 Why Is It Secure? (Sketch—Not in the Paper)

The paper only calls the protocol passively secure and does not give a proof. For completeness, here is a sketch in the style of §6. We want: an eavesdropper who sees $(\mathbf{A}, \mathbf{u}_1, \mathbf{u}_2)$ cannot distinguish the key bit b_1 from a random bit.

Game 0 – Real

$\mathbf{u}_1^T = \mathbf{s}_1^T \mathbf{A} + \mathbf{e}_1^T$, $\mathbf{u}_2 = \mathbf{A}\mathbf{s}_2 + \mathbf{e}_2$, and $b_1 = \text{Round}(\mathbf{s}_1^T \mathbf{u}_2)$ (with Round the decision rule above).

Game 1 – Party 2's message becomes uniform

Replace $\mathbf{u}_2$ by a uniform vector. $(\mathbf{A}, \mathbf{A}\mathbf{s}_2 + \mathbf{e}_2)$ is an LWE instance with secret $\mathbf{s}_2$. The reduction samples $\mathbf{s}_1, \mathbf{e}_1$ itself, so it can compute $\mathbf{u}_1$ and b_1. The difference is at most Adv^{LWE}.

Game 2 – Add a small error to party 1's key computation

Compute $b_1 = \text{Round}(\mathbf{s}_1^T \mathbf{u}_2 + e_3)$ with a fresh $e_3 \leftarrow [\beta]$. Since $\mathbf{u}_2$ is now uniform and $\mathbf{s}_1 \neq \mathbf{0}$ (except with probability $(2\beta + 1)^{-m}$), $\mathbf{s}_1^T \mathbf{u}_2$ is uniform, and adding e_3 changes the rounded bit only if it lies within β of a boundary: probability about $4\beta/q$. This is a *statistical* loss of the same order as the protocol's own error rate.

Game 3 – Party 1's view becomes uniform

Now $(\mathbf{u}_1^T,\ \mathbf{s}_1^T \mathbf{u}_2 + e_3) = \mathbf{s}_1^T [\mathbf{A} \mid \mathbf{u}_2] + [\mathbf{e}_1^T \mid e_3]$, with $[\mathbf{A} \mid \mathbf{u}_2]$ uniform in $\mathbb{Z}_q^{m \times (m+1)}$. This is LWE with secret $\mathbf{s}_1$ (rows and columns swapped, as in Game 2 of §6). Replace it with a uniform $(\mathbf{u}_1^T, w)$ and set $b_1 = \text{Round}(w)$. Cost: Adv^{LWE}. Now b_1 is the rounding of a uniform $w \in \mathbb{Z}_q$, independent of the transcript and within $O(1/q)$ of an unbiased bit.

Total: the eavesdropper's advantage is at most 2 $\mathrm{Adv}^{\mathrm{LWE}} + O(\beta/q)$. The additive $O(\beta/q)$ is not negligible for practical q—another face of the inefficiency discussed above.

Compare with DH's proof

DH's key is exactly $g^{x_1 x_2}$, and a single DDH step replaces it with a random group element. The LWE proof needs *two* LWE steps (one per party, just like the two hops in the encryption proof) plus a statistical step, because the parties' views agree only up to noise.

10. Summary: What to Remember

The Five Key Ideas of LWE Encryption

1. **Structure = ElGamal with matrices.** Secret key cancels from the ciphertext via $\mathbf{r}^T\mathbf{A}\mathbf{s} - \mathbf{r}^T\mathbf{A}\mathbf{s} = 0$.
2. **Noise = security.** The error $\mathbf{e}$ prevents inversion. Without it, the scheme is trivially broken.
3. **Noise = the enemy of correctness.** The accumulated noise $\mathbf{r}^T\mathbf{e}_1 + e_3 - \mathbf{e}_2^T\mathbf{s}$ must stay below $q/4$.
4. **The security proof has two hops:** (i) public key looks random (by LWE), (ii) ciphertext looks random (by LWE again).
5. **What makes LWE hard is geometry:** distinguishing requires finding short vectors in high-dimensional lattices, which takes exponential time.

Also in Section 2 of the paper

- **Exact error bounds (2.3.2):** multiply the polynomials that encode the noise distributions; far tighter than the worst case $2m\beta^2 + \beta$ (§5.4).
- **Compression (2.5.1–2.5.2):** drop the low-order bits of $\mathbf{V}$ with $\lceil\cdot\rfloor_{q\to p}$; Lemma 1 bounds the error by $\frac{q}{2p} + \frac{1}{2}$ (§8.2, §8.3).
- **LWR (2.5.3):** rounding can replace the explicit error.
- **Variations (2.5.4–2.5.6):** more bits per slot, a uniform public key via the leftover hash lemma, different distributions for secret and error.
- **NIKE (2.6):** a Diffie–Hellman-style exchange from LWE, with an inherent mismatch probability of order $\beta^2\sqrt{m}/q$ (§9).

Part II: The Geometry Behind the Hardness

11. Linear Algebra Refresher

This section collects the background needed for the geometric view of LWE. Skip freely if comfortable.

11.1 Determinant

For a square matrix $\mathbf{B} \in \mathbb{Z}^{m\times m}$, the **determinant** $\det(\mathbf{B})$ measures how $\mathbf{B}$ scales volume. If you apply $\mathbf{B}$ to the unit hypercube in $\mathbb{R}^m$, the resulting parallelepiped has m-dimensional volume $|\det(\mathbf{B})|$.

Key properties of the determinant

- $\det(\mathbf{I}) = 1, \quad \det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B}), \quad \det(\mathbf{B}^T) = \det(\mathbf{B})$
- $\det(\mathbf{B}) = 0$ iff $\mathbf{B}$ is singular (columns are linearly dependent).
- $\det(c\,\mathbf{I}_m) = c^m$ (scaling the identity by c multiplies volume by c^m).
- Block triangular: $\det\begin{pmatrix}\mathbf{X} & \mathbf{0}\\ \mathbf{Y} & \mathbf{Z}\end{pmatrix} = \det(\mathbf{X})\cdot\det(\mathbf{Z})$
- If $\mathbf{B}$ has eigenvalues $\lambda_1,\dots,\lambda_m$: $\det(\mathbf{B}) = \prod_i \lambda_i$.

11.2 Trace

The trace is the sum of the diagonal entries: $\mathrm{tr}(\mathbf{B}) = \sum_i B_{ii}$.

The one property you need: $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$ ("cyclic property").

If $\mathbf{B}$ has eigenvalues $\lambda_1,\dots,\lambda_m$, then $\mathrm{tr}(\mathbf{B}) = \sum_i \lambda_i$. So the trace is the sum of eigenvalues, while the determinant is the product.

11.3 Quotient Groups

Given a lattice $\Lambda \subseteq \mathbb{Z}^m$, the **quotient group** $\mathbb{Z}^m/\Lambda$ is the set of all cosets of Λ. Two vectors $\mathbf{z}_1, \mathbf{z}_2$ are in the same coset iff $\mathbf{z}_1 - \mathbf{z}_2 \in \Lambda$.

For a parity-check lattice $\Lambda = \mathcal{L}_q^{\perp}(\mathbf{A}) \subseteq \mathbb{Z}^m$, two vectors are in the same coset iff $\mathbf{Az}_1 \equiv \mathbf{Az}_2 \pmod q$ (Section 3.1.1 of the paper). For our lattice $\Lambda = \mathcal{L}_q^{\perp}([\mathbf{A} \mid \mathbf{I}_n]) \subseteq \mathbb{Z}^{m+n}$ the parity-check matrix is $[\mathbf{A} \mid \mathbf{I}_n]$, so $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{Z}^{m+n}$ are in the same coset iff $[\mathbf{A} \mid \mathbf{I}_n]\mathbf{z}_1 \equiv [\mathbf{A} \mid \mathbf{I}_n]\mathbf{z}_2 \pmod q$. Each coset is labeled by $\mathbf{t} = [\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} \bmod q \in \mathbb{Z}_q^n$, and every $\mathbf{t}$ occurs (take $\mathbf{z} = (\mathbf{0}, \mathbf{t})$). There are exactly q^n cosets, and:

$$|\mathbb{Z}^{m+n}/\Lambda| = \det(\Lambda) = q^n.$$

12. Lattices as Geometric Objects

This section dissects **Section 3.1–3.2** of the paper.

12.1 Two Ways to Describe a Lattice

Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, there are two natural lattices one can associate with it:

Lattice	Definition	Intuition
$\mathcal{L}_q(\mathbf{A})$	$\{\mathbf{v} \in \mathbb{Z}^n : \mathbf{v} \equiv \mathbf{Az} \pmod q \text{ for some } \mathbf{z}\}$	The **image** of $\mathbf{A}$ mod q
$\mathcal{L}_q^\perp(\mathbf{A})$	$\{\mathbf{v} \in \mathbb{Z}^m : \mathbf{Av} \equiv \mathbf{0} \pmod q\}$	The **kernel** of $\mathbf{A}$ mod q

Why the $\perp$ symbol?

The $\perp$ stands for "perpendicular." A vector $\mathbf{v} \in \mathcal{L}_q^\perp(\mathbf{A})$ satisfies $\mathbf{Av} \equiv \mathbf{0} \pmod q$, which means $\langle \mathbf{a}_i, \mathbf{v} \rangle \equiv 0 \pmod q$ for every row $\mathbf{a}_i$ of $\mathbf{A}$. In other words, $\mathbf{v}$ is *orthogonal to all rows of* $\mathbf{A}$ in the modular sense—just like the orthogonal complement $V^\perp$ in linear algebra consists of vectors perpendicular to every vector in V.
In LWE cryptography we always work with the $\perp$ version (the kernel lattice), because both the SIS and LWE problems are naturally stated in terms of it.

More broadly, lattices can be described in two equivalent styles:

	Generator matrix	Parity-check matrix
Definition	$\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{Bz} : \mathbf{z} \in \mathbb{Z}^m\}$	$\Lambda = \mathcal{L}_q^\perp(\mathbf{A}) = \{\mathbf{v} : \mathbf{Av} \equiv \mathbf{0}\}$
Analogy	Generating matrix of a linear code	Parity-check matrix of a linear code
Used for	Computing with the lattice	Defining the cryptographic problem

The specific lattices in crypto have the form $\Lambda = \mathcal{L}_q^\perp([\mathbf{A} \mid \mathbf{I}_n])$, meaning:

$$\Lambda = \left\{(\mathbf{v}_1, \mathbf{v}_2) \in \mathbb{Z}^{m+n} : \mathbf{Av}_1 + \mathbf{v}_2 \equiv \mathbf{0} \pmod q\right\}.$$

Why $[\mathbf{A} \mid \mathbf{I}_n]$ and not just $\mathbf{A}$?

The $\mathbf{I}_n$ is there to **absorb the error $\mathbf{e}$** into the lattice framework. Recall the LWE equation:

$$\mathbf{t} = \mathbf{As} + \mathbf{e} \pmod q.$$

Rewrite this as a single matrix-vector product:

$$[\mathbf{A} \mid \mathbf{I}_n] \begin{pmatrix} \mathbf{s} \\ \mathbf{e} \end{pmatrix} \equiv \mathbf{t} \pmod q.$$

The combined vector $(\mathbf{s}, \mathbf{e}) \in [\beta]^{m+n}$ is **short**—every entry is at most β. So the LWE instance $(\mathbf{A}, \mathbf{t} = \mathbf{As} + \mathbf{e})$ is saying: *the coset of* $\mathcal{L}_q^\perp([\mathbf{A} \mid \mathbf{I}_n])$ *labeled by* $\mathbf{t}$ *contains a short representative* $(\mathbf{s}, \mathbf{e})$, i.e. it is close to the lattice.
Without $\mathbf{I}_n$, the lattice $\mathcal{L}_q^\perp(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m : \mathbf{Av} \equiv \mathbf{0}\}$ lives in $\mathbb{Z}^m$ only. You would need $\mathbf{As} \equiv \mathbf{t} \pmod q$ exactly—but that is just a linear system (solvable by Gaussian elimination), not a hard problem. The error $\mathbf{e}$ is what makes LWE hard, and $\mathbf{I}_n$ is what lets the error live inside the lattice as extra coordinates.

Lattice	Equation	Problem
$\mathcal{L}_q^\perp(\mathbf{A}) \subseteq \mathbb{Z}^m$	$\mathbf{Av} \equiv \mathbf{0}$	No room for error; $\mathbf{As} \equiv \mathbf{t}$ is easy
$\mathcal{L}_q^\perp([\mathbf{A} \mid \mathbf{I}_n]) \subseteq \mathbb{Z}^{m+n}$	$\mathbf{Av}_1 + \mathbf{v}_2 \equiv \mathbf{0}$	Error $\mathbf{e}$ becomes $\mathbf{v}_2$; finding short $(\mathbf{s}, \mathbf{e})$ is hard

Why not just use $\mathcal{L}_q^\perp(\mathbf{A})$?

The paper notes (Section 3.1) that this is "not much of a restriction." Any $\mathcal{L}_q^\perp(\mathbf{A})$ with $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ having n linearly independent columns can be rewritten in the form $\mathcal{L}_q^\perp([\mathbf{A}' \mid \mathbf{I}_n])$ by Gaussian elimination on $\mathbf{A}$. The $[\mathbf{A} \mid \mathbf{I}_n]$ form is simply more convenient: it makes the connection to LWE explicit, and gives a clean generator matrix (below).

These two representations are interchangeable. The generator matrix for $\mathcal{L}_q^\perp([\mathbf{A} \mid \mathbf{I}_n])$ is:

$$\mathbf{B} = \begin{pmatrix} -\mathbf{I}_m & \mathbf{0} \\ \mathbf{A} & q\mathbf{I}_n \end{pmatrix},$$

and one can verify $\det(\Lambda) = |\det(\mathbf{B})| = |\det(-\mathbf{I}_m)| \cdot |\det(q\mathbf{I}_n)| = 1 \cdot q^n = q^n$.

12.2 The Determinant of a Lattice

What the determinant means

$\det(\Lambda)$ is the *inverse density* of lattice points. Equivalently:

$$\det(\Lambda) = \lim_{r\to\infty} \frac{|\{\mathbf{z} \in \mathbb{Z}^m : \|\mathbf{z}\| < r\}|}{|\Lambda \cap \{\mathbf{z} : \|\mathbf{z}\| < r\}|} = \frac{\text{total integer points in a big ball}}{\text{lattice points in that ball}}.$$

Bigger determinant = fewer lattice points per unit volume = more spread out.

For our q-ary lattices: $\det(\Lambda) = q^n$. The average "spacing" between lattice points in an $(m+n)$-dimensional space is approximately $\det(\Lambda)^{1/(m+n)} = q^{n/(m+n)}$.

This quantity $q^{n/(n+m)}$ turns out to be *the* critical threshold for the existence of short lattice vectors.

12.3 Distance to the Lattice and Cosets

For a vector $\mathbf{r} \in \mathbb{Z}^m$ and a lattice Λ, the distance is:

$$\Delta_p(\mathbf{r}, \Lambda) = \min_{\mathbf{v}\in\Lambda} \|\mathbf{v} - \mathbf{r}\|_p .$$

Crucially, all vectors in the same coset have the same distance to the lattice. So distance is a property of the *coset*, not of any individual vector.

LWE in the language of lattices

Given $\mathbf{A}$ and $\mathbf{t}$, the LWE problem asks:

- **If $\mathbf{t} = \mathbf{As} + \mathbf{e}$:** The coset labeled by $\mathbf{t}$ is *close* to Λ: $\Delta_\infty^C(\mathbf{t}, \Lambda) \le \beta$ (ℓ_∞ distance)
- **If $\mathbf{t}$ is random:** A *random* coset, which is far from Λ when $\beta^{1+m/n} \ll q$ (Corollary 1)

LWE = distinguish close cosets from random cosets.

12.4 Three Lemmas: The Landscape of Short Vectors

The paper proves three lemmas that establish the "geography" of the lattice. Together they say: short vectors barely exist, but not-so-short vectors always do, and there is a sharp boundary between the two regimes.

The paper proves these for prime q only, for simplicity ("with more care, one can prove similar statements for all q").

Lemma 2 and Corollary 1 — Random cosets are far from the lattice

Lemma 2. For any prime q and any *fixed* $\mathbf{t} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\}$:

$$\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}} \left[\exists\, \mathbf{z} \in [\beta]^{n+m} \text{ s.t. } [\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} \equiv \mathbf{t} \pmod q \right] \;\leq\; \frac{(2\beta+1)^{n+m}}{q^n}.$$

Proof idea. For one fixed $\mathbf{z}$: since $\mathbf{t} \neq \mathbf{0}$, some coefficient of $\mathbf{z}$ is non-zero; say the first, z_1. Isolating the first column $\mathbf{a}$ of $\mathbf{A}$, the equation becomes $\mathbf{a} \equiv z_1^{-1}(\mathbf{t} - [\mathbf{A}' \mid \mathbf{I}_n]\mathbf{z}') \pmod q$ (z_1^{-1} exists because q is prime), which a uniform $\mathbf{a}$ satisfies with probability exactly q^{-n}. A union bound over all $(2\beta+1)^{n+m}$ vectors $\mathbf{z}$ finishes the proof.

Corollary 1. For a random coset (with $\Lambda = \mathcal{L}_q^{\perp}([\mathbf{A} \mid \mathbf{I}_n])$):

$$\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m},\, \mathbf{t} \leftarrow \mathbb{Z}_q^n} \left[\Delta_\infty^C(\mathbf{t}, \Lambda) \leq \beta \right] \;\leq\; \left(1 - |\mathbb{Z}_q^*|/q\right)^n + \frac{(2\beta+1)^{n+m}}{q^n}.$$

The first term is negligible in n for the popular choices of q: it is $(1/q)^n$ for prime q and 2^{-n} for q a power of 2. So whenever $\beta^{1+m/n} \ll q$, random cosets are more than distance β from Λ.

A gap in the paper's Lemma 2 (our observation)

The proof assumes WLOG that the non-zero coefficient z_1 of $\mathbf{z}$ multiplies a column of $\mathbf{A}$. But $\mathbf{z}$ could be non-zero only in the $\mathbf{I}_n$ block: $\mathbf{z} = (\mathbf{0}, \mathbf{t})$ gives $[\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} = \mathbf{t}$ for *every* $\mathbf{A}$. So for $\mathbf{t} \in [\beta]^n \setminus \{\mathbf{0}\}$ the probability is 1, not $\leq (2\beta+1)^{n+m}/q^n$. The lemma (and its proof) is correct for $\mathbf{t} \notin [\beta]^n$. Corollary 1 is unaffected in substance: a random $\mathbf{t}$ lies in $[\beta]^n$ with probability $((2\beta+1)/q)^n$, which is at most the second term of its bound, so the bound at most doubles that term.

Intuition: There are $(2\beta+1)^{n+m}$ candidate short vectors that could "reach" a coset, and q^n cosets. If $(2\beta+1)^{n+m} \ll q^n$, there aren't enough short vectors to cover a random coset with noticeable probability.

Lemma 3 — The lattice probably has no short vectors

For any prime q:

$$\Pr_{\mathbf{A}}[\exists\, \mathbf{z} \in [\beta]^{n+m} \setminus \{\mathbf{0}\} : [\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} \equiv \mathbf{0}] \;\leq\; \frac{(2\beta+1)^{n+m}}{q^n}.$$

Intuition: Same counting argument, now applied to the zero coset. A random lattice probably doesn't have any short non-zero vectors when β is small enough.

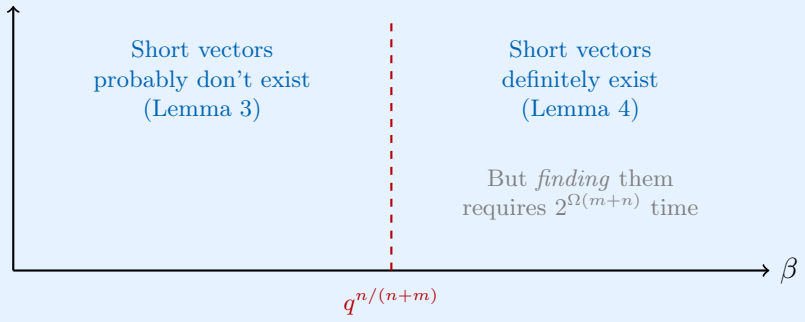
Lemma 4 — But not-so-short vectors always exist (pigeonhole)

For *any* $\mathbf{A}$:

$$\exists\, \mathbf{z} \in \left[q^{n/(n+m)}\right]^{n+m} \setminus \{\mathbf{0}\} \quad \text{such that} \quad [\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} \equiv \mathbf{0} \pmod{q}.$$

Proof: There are more than $(q^{n/(n+m)})^{n+m} = q^n$ vectors with entries in $\{0, \ldots, q^{n/(n+m)}\}$, but only q^n possible values of $\mathbf{Az} \bmod q$. By pigeonhole, two must collide: $\mathbf{Az}_1 \equiv \mathbf{Az}_2$, so $\mathbf{z}_1 - \mathbf{z}_2$ is a non-zero lattice vector.
Key point: The proof is non-constructive. It guarantees existence but gives no efficient way to *find* the vector. This gap between existence and finding is where cryptographic hardness lives.

The sharp threshold at $\beta \approx q^{n/(n+m)}$



13. The SIS Problem: Finding Short Vectors

13.1 Definition

The SIS Problem (Definition 4 in the paper)

Given a random $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, find non-zero $\mathbf{s}_1 \in [\beta]^m$, $\mathbf{s}_2 \in [\beta]^n$ such that:

$$\mathbf{As}_1 + \mathbf{s}_2 \equiv \mathbf{0} \pmod{q}.$$

Equivalently: find a short non-zero vector in $\mathcal{L}_q^{\perp}([\mathbf{A} \mid \mathbf{I}_n])$.

13.2 How Hard Is It? The LLL Family

All modern (i.e. polynomial-time) algorithms for finding short lattice vectors descend from **LLL** (Lenstra–Lenstra–Lovász, 1982). The key parameter is δ, which controls the trade-off between running time and quality of the found vector.

For a lattice of dimension $d = m + n$ with determinant q^n, the best algorithms find a vector of ℓ_2-norm approximately:

$$\|\mathbf{z}\| \;\approx\; \det(\Lambda)^{1/d} \cdot \delta^d \;=\; q^{n/(n+m)} \cdot \delta^{n+m}.$$

The first factor $q^{n/(n+m)}$ is the "ideal" shortest vector length (from Lemma 4). The second factor δ^{n+m} is the approximation penalty—how much longer than optimal the found vector is. As $\delta \to 1$, the penalty vanishes but the running time explodes.

Practical δ values

δ	Status
$\delta = 1.01$	Considered within reach
$\delta = 1.005$	May never be achieved for lattices of high-enough dimension (e.g. more than 500)

The paper calls this "a very rough rule of thumb" (Section 3.2).

It is optimal to run the algorithm on a lattice of dimension $\approx \sqrt{n \log q / \log \delta}$ (rather than the full $m + n$), giving found vectors of norm:

$$\|\mathbf{z}\| \approx 2^{2\sqrt{n \log q \log \delta}}. \tag{4}$$

13.3 SIS vs. LWE: The Dual Relationship

The SIS and LWE problems are *duals* in the following precise sense. Looking at Figure 2 of the paper:

- **LWE** gets harder as β grows (more noise = harder to distinguish from random).
- **SIS** gets easier as β grows (you're allowed to find longer vectors = easier search).
- They "cross" at approximately $\beta = q^{n/(n+m)}$.

For the signature scheme of Section 5 of the paper (Dilithium), security depends on *both* the SIS and the LWE problem (Section 3.4 of the paper). The two use very different β: in the Dilithium-like parameters of the paper's Table 2 (reproduced in §14), LWE has $\beta \in \{2, 4\}$ while SIS has $\beta \in \{2^{18}, 2^{20}\}$.

14. Solving LWE via SIS: The Attack

This is **Section 3.3** of the paper—the most important part of the hardness story. It shows concretely how an SIS solver breaks LWE.

14.1 The Strategy

Given an LWE instance $(\mathbf{A}, \mathbf{t})$ where $\mathbf{t}$ is either $\mathbf{As} + \mathbf{e}$ or random $\mathbf{u}$:

Step 1. Find short $\mathbf{r}_1 \in \mathbb{Z}^n$, $\mathbf{r}_2 \in \mathbb{Z}^m$ such that $\mathbf{r}_1^T \mathbf{A} + \mathbf{r}_2^T = \mathbf{0} \pmod q$.

This is solving SIS on $\mathbf{A}^T$—i.e., finding a short vector in $\mathcal{L}_q^\perp([\mathbf{A}^T \mid \mathbf{I}_m])$.

Step 2. Compute $\mathbf{r}_1^T \cdot \mathbf{t}$.

Step 3. Decide based on the result:

Two cases

Case 1: $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ (LWE instance).

$$\begin{aligned}\mathbf{r}_1^T\mathbf{t} &= \mathbf{r}_1^T\mathbf{A}\mathbf{s} + \mathbf{r}_1^T\mathbf{e} \\ &= -\mathbf{r}_2^T\mathbf{s} + \mathbf{r}_1^T\mathbf{e} \qquad (\text{using } \mathbf{r}_1^T\mathbf{A} = -\mathbf{r}_2^T) \\ &= \text{small} \cdot \text{small} + \text{small} \cdot \text{small} \\ &= \textbf{small.}\end{aligned}$$

Case 2: $\mathbf{t} = \mathbf{u}$ (random).

$$\mathbf{r}_1^T\mathbf{u} = \text{random element of } \mathbb{Z}_q. \qquad \textbf{(Uniformly distributed.)}$$

So: small output ⇒ LWE instance. Random output ⇒ uniform. **Distinguishing done.**

14.2 When Does the Attack Fail?

The "small" quantity $-\mathbf{r}_2^T\mathbf{s} + \mathbf{r}_1^T\mathbf{e}$ is an inner product of short vectors. Each coefficient of $\mathbf{s}, \mathbf{e}$ is uniform in $[\beta]$, with variance (the paper's Eq. (39))

$$\frac{1}{2\beta+1}\sum_{i=-\beta}^{\beta} i^2 = \frac{2}{2\beta+1}\sum_{i=1}^{\beta} i^2 = \frac{2}{2\beta+1}\cdot\frac{\beta(\beta+1)(2\beta+1)}{6} = \frac{\beta(\beta+1)}{3}.$$

Treating each coefficient as normal with that standard deviation (justified asymptotically by the central limit theorem, and "already a very good approximation" for lattice parameters), $-\mathbf{r}_2^T\mathbf{s} + \mathbf{r}_1^T\mathbf{e}$ is normal with standard deviation approximately

$$\|(\mathbf{r}_1, \mathbf{r}_2)\| \cdot \sqrt{\frac{\beta(\beta+1)}{3}}.$$

Finding $(\mathbf{r}_1, \mathbf{r}_2)$ is finding a short vector in $\mathcal{L}_q^{\perp}([\mathbf{A}^T \mid \mathbf{I}_m])$. Using Eq. (4) for its norm—with n replaced by m, because the lattice is built from $\mathbf{A}^T$—gives (the paper's Eq. (40))

$$\text{std. dev.} \;\approx\; 2^{2\sqrt{m \log q \log \delta}} \cdot \sqrt{\frac{\beta(\beta+1)}{3}}.$$

It is known [MR07] that a normal random variable with standard deviation greater than $\sqrt{3}\cdot q$, reduced modulo q, is statistically close (within $\approx 2^{-80}$) to uniform. So if (40) exceeds $\sqrt{3}\cdot q$, the attack *cannot tell the two cases apart.*

The LWE security condition

$\text{LWE}_{n,m,q,\beta}$ is secure "at least against this attack, which seems to be as good as any other known approach" whenever (the paper's Eq. (41)):

$$\boxed{\sqrt{\beta(\beta+1)} \;>\; 3q \cdot 2^{-2\sqrt{m \log q \log \delta}}}$$

(Spelled out: $\sqrt{\beta(\beta+1)/3} \cdot 2^{2\sqrt{\cdots}} > \sqrt{3}\,q$; multiply both sides by $\sqrt{3}$ and move the power of 2 across.) Given m, q, β, solve for the δ an attacker would need. By the rough rule of thumb of Section 3.2 ($\delta = 1.01$ within reach, $\delta = 1.005$ may never be achieved in dimension above about 500), the smaller the required δ, the more secure the parameters.

14.3 Practical Parameters (Section 3.4)

The paper lists the δ an attacker would need for parameters that *resemble* those of the two NIST schemes. These are illustrative $\mathrm{LWE}_{m,q,\beta}$ and $\mathrm{SIS}_{n,q,\beta}$ instances, not the exact standardized parameter sets (those come in Sections 4.7 and 5.7).

$\mathrm{LWE}_{m,q,\beta}$			
m	β	q	δ
512	2	2^{12}	1.0043
768	2	2^{12}	1.0029
1024	2	2^{12}	1.0022

Table 1: The paper's Table 1: approximate δ-hardness of $\mathrm{LWE}_{m,q,\beta}$ for parameters that resemble those used in Kyber (ML-KEM).

$\mathrm{LWE}_{m,q,\beta}$				$\mathrm{SIS}_{n,q,\beta}$			
m	β	q	δ	n	β	q	δ
1024	2	2^{23}	1.004	1024	2^{18}	2^{23}	1.0041
1280	4	2^{23}	1.003	1536	2^{20}	2^{23}	1.0032
1792	2	2^{23}	1.0023	2048	2^{20}	2^{23}	1.0025

Table 2: The paper's Table 2: approximate δ-hardness of $\mathrm{LWE}_{m,q,\beta}$ and $\mathrm{SIS}_{n,q,\beta}$ for parameters that resemble those used in Dilithium (ML-DSA).

Every required δ in both tables is below 1.005, which by the rule of thumb may never be reached in these dimensions. The paper stresses that such tables are set from the *best currently-known* lattice-reduction algorithms (cf. the Lattice Estimator [APS15]).

14.4 A Caveat: The Hardness Might Not Be Smooth

For LWE, Figure 2 of the paper shows a monotonic increase in hardness as β grows, without sudden jumps. Concretely, if $q/\beta = 2^{m/k}$ with $1 \le k \le m$ (m the lattice dimension), the best known algorithm takes time about 2^k (ignoring polynomial factors).

But a sudden jump is not out of the question. It is exactly what happens for short vectors in lattices that correspond to an **ideal of some algebraic ring** [CGS14, BS16, CDPR16, CDW17]: when $q/\beta > 2^{\sqrt{m}}$ (i.e. $k < \sqrt{m}$), the problem can be solved in quantum polynomial time instead of 2^k, but as soon as $k > \sqrt{m}$ the hardness jumps back to 2^k.

Some not-yet-invented (quantum) algorithm might similarly do much better on a particular range of q/β. The paper concludes that it "may therefore be prudent" to base cryptography on LWE with q/β as small as possible.

15. Summary of Part II (Section 3 of the Paper)

The five things to remember from the hardness story

1. **LWE is a geometric problem:** distinguish close cosets (ℓ_∞ distance $\leq \beta$) from random cosets in a q-ary lattice.
2. **The threshold is $\beta \approx q^{n/(n+m)}$:** below this, short vectors probably don't exist (Lemma 3, prime q). Above, they always do (Lemma 4, pigeonhole). The encryption scheme (with $m = n$) needs $\beta \ll \sqrt{q} = q^{n/(n+m)}$, i.e. *below* the threshold, so random cosets are far from the lattice; the SIS instances in signatures use much larger β (the paper's Table 2).
3. **SIS is the dual problem:** find a short vector in the lattice. It gets easier as β grows (opposite direction from LWE).
4. **Solving SIS breaks LWE:** find short $\mathbf{r}_1, \mathbf{r}_2$ with $\mathbf{r}_1^T\mathbf{A} + \mathbf{r}_2^T = \mathbf{0}$, compute $\mathbf{r}_1^T\mathbf{t}$. If the result is small, it's an LWE instance; if random, it's uniform.
5. **Concrete security:** parameterized by δ, the quality of lattice reduction. As a very rough rule of thumb, $\delta = 1.01$ is within reach, while $\delta = 1.005$ may never be achieved in dimension above about 500.

Next: Part III (Section 4 of the paper) — polynomial rings, NTT, and Kyber.

Part III: Encryption Over Polynomial Rings

Where we are. The LWE scheme of Section 2.3 needs a ciphertext of $(m+1)\log q$ bits to encrypt *one* bit: the expansion is linear in the security parameter. The amortized scheme of Section 2.4 brought this down to a square-root expansion, but only by blowing up the public key by the same factor. Section 4 of the paper removes the square-root blow-up altogether by doing LWE not over $\mathbb{Z}_q$ but over a **ring of polynomials**. This Part follows Sections 4.1–4.8 of the paper and ends with Kyber (ML-KEM).

16. Polynomial Rings (Section 4.1)

16.1 The Ring $\mathcal{R}_f$

The polynomial ring $\mathbb{Z}[X]$ consists of the polynomials $a(X) = \sum_i a_i X^i$ with $a_i \in \mathbb{Z}$, with the usual addition and multiplication. We usually drop the indeterminate and just write a. Three words we need:

- $\deg(a)$ is the largest i with $a_i \neq 0$;
- a is **monic** if its leading coefficient $a_{\deg(a)}$ is 1;
- a is **irreducible** if it cannot be written as $a = bc$ with $b, c \in \mathbb{Z}[X]$ and $\deg(b), \deg(c) < \deg(a)$.

The ring $(\mathcal{R}_f, +, \times)$ (paper, Section 4.1)

Fix a monic $f \in \mathbb{Z}[X]$ of degree d. The elements of $\mathcal{R}_f$ are the polynomials $a = \sum_{i=0}^{d-1} a_i X^i$ with $a_i \in \mathbb{Z}$ (degree less than d). In the lattice literature this ring is often written $\mathbb{Z}[X]/(f(X))$.

- **Addition** is coefficient-wise: $a + b = \sum_{i=0}^{d-1}(a_i + b_i)X^i$. So $\mathcal{R}_f$ under addition is just $\mathbb{Z}^d$, and multiplying by an integer is scaling a vector.
- **Multiplication** is ordinary polynomial multiplication followed by **reduction modulo** f: taking the remainder after dividing by f.

"Modulo" twice

You already know $\mathbb{Z}_q$: integers, with everything reduced modulo the *number* q. $\mathcal{R}_f$ is the same idea one level up: polynomials, with everything reduced modulo the *polynomial* f. Later we will do both at once ($\mathcal{R}_{q,f}$: coefficients mod q, polynomials mod f). In fact $\mathbb{Z}$ itself is the special case $f = X$ (or $f = X - \alpha$ for any $\alpha \in \mathbb{Z}$): every polynomial reduces to a constant. (Analogy ours.)

16.1.1 Division With Remainder Works (the paper's proof, spelled out)

Claim. Every $a \in \mathbb{Z}[X]$ can be written *uniquely* as $a = bf + r$ with $\deg(r) < d$. We then write $a \bmod f = r$.

Existence, by induction on $\deg(a)$. If $\deg(a) < d$, take $b = 0$, $r = a$. Otherwise suppose every polynomial of degree at most $k-1$ can be written this way, and let a' have degree $k \geq d$ with leading coefficient a'_k. Since f is monic, $a'_k f X^{k-d}$ has the *same* leading term $a'_k X^k$ as a'. So $a' - a'_k f X^{k-d}$ has degree at most $k-1$, and by hypothesis equals $bf + r$. Hence $a' = (a'_k X^{k-d} + b)f + r$. (This is where "monic" matters: we never have to divide by a leading coefficient, so everything stays in $\mathbb{Z}[X]$.)

Uniqueness. Suppose $bf + r = b'f + r'$ with $(b, r) \neq (b', r')$. Then $(b - b')f = r' - r$. The right side has degree less than d. If $b \neq b'$, the left side has degree $\deg(b - b') + \deg(f) \geq d$. That is a contradiction, so $b = b'$, and then $r = r'$.

The algorithm hidden in the proof. Repeatedly subtract a suitable multiple $\alpha X^i f$ to kill the leading term, until the degree drops below d.

Worked example (from the paper): $2X^3 + 8X^2 + 5X + 1 \bmod (X^2 - 2X + 1)$

Modulo f we have $X^2 \equiv 2X - 1$. So:

$$\begin{aligned} 2X^3 + 8X^2 + 5X + 1 &\equiv 2X(2X - 1) + 8X^2 + 5X + 1 \\ &= 4X^2 - 2X + 8X^2 + 5X + 1 = 12X^2 + 3X + 1 \\ &\equiv 12(2X - 1) + 3X + 1 = 27X - 11 \pmod{f}. \end{aligned}$$

(The paper writes the first step as $2(2X^2 - X)$, i.e. $2X^3 \equiv 2X \cdot X^2 \equiv 2X(2X - 1)$.)

16.2 Polynomials and Linear Algebra (Section 4.1.1)

The single most useful fact in this Part: **multiplication by a fixed polynomial is a linear map**, so it is a $d \times d$ integer matrix. Writing $b = \sum_i b_i X^i$ (the paper's Eq. (42)):

$$ab \bmod f = a \cdot \Big(\sum_{i=0}^{d-1} b_i X^i\Big) \bmod f = \sum_{i=0}^{d-1} (aX^i \bmod f)\, b_i.$$

Each $aX^i \bmod f$ has degree less than d, so it is a vector in $\mathbb{Z}^d$, and ab is the linear combination of these d vectors with weights b_i: a matrix times a vector.

Notation (the paper's Eqs. (44)–(46))

For $a \in \mathcal{R}_f$:

$$\mathcal{V}_a = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{pmatrix} \in \mathbb{Z}^d, \qquad \mathcal{M}_a = \begin{bmatrix} \mathcal{V}_a & \mathcal{V}_{aX \bmod f} & \cdots & \mathcal{V}_{aX^{d-1} \bmod f} \end{bmatrix} \in \mathbb{Z}^{d \times d},$$

so that $\mathcal{M}_a \mathcal{V}_b = \mathcal{V}_{ab}$. For a vector $\mathbf{a} \in \mathcal{R}_f^n$ and a matrix $\mathbf{A} \in \mathcal{R}_f^{n \times m}$, stack the blocks: $\mathcal{V}_{\mathbf{a}} \in \mathbb{Z}^{dn}$ and $\mathcal{M}_{\mathbf{A}} \in \mathbb{Z}^{dn \times dm}$ (block (i, j) is $\mathcal{M}_{a_{i,j}}$). Then for any $\mathbf{A} \in \mathcal{R}_f^{n \times m}$, $\mathbf{b} \in \mathcal{R}_f^m$:

$$\mathcal{M}_{\mathbf{A}} \cdot \mathcal{V}_{\mathbf{b}} = \mathcal{V}_{\mathbf{Ab}} \in \mathbb{Z}^{dn}.$$

The polynomial f is not part of the notation; it is always clear from context.

Worked example (from the paper, Eq. (43)): $(2X^2-1)(X^2-X+2) \bmod (X^3-X+1)$

Modulo $f = X^3 - X + 1$ we have $X^3 \equiv X - 1$. With $a = 2X^2 - 1$, build $\mathcal{M}_a$ column by column:

- $a = -1 + 0X + 2X^2 \;\Rightarrow\; (-1, 0, 2)$;
- $aX = 2X^3 - X \equiv 2(X-1) - X = -2 + X \;\Rightarrow\; (-2, 1, 0)$;
- $aX^2 = (aX)X \equiv -2X + X^2 \;\Rightarrow\; (0, -2, 1)$.

With $\mathcal{V}_b = (2, -1, 1)$ for $b = X^2 - X + 2$:

$$\underbrace{\begin{bmatrix} -1 & -2 & 0 \\ 0 & 1 & -2 \\ 2 & 0 & 1 \end{bmatrix}}_{\mathcal{M}_{2X^2-1}} \cdot \underbrace{\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}}_{\mathcal{V}_{X^2-X+2}} = \underbrace{\begin{bmatrix} 0 \\ -3 \\ 5 \end{bmatrix}}_{\mathcal{V}_{5X^2-3X}}.$$

Check directly: $(2X^2-1)(X^2-X+2) = 2X^4 - 2X^3 + 3X^2 + X - 2$. Using $X^3 \equiv X - 1$ and $X^4 \equiv X^2 - X$, this is $2X^2 - 2X - 2X + 2 + 3X^2 + X - 2 = 5X^2 - 3X$.

Why this matters

Everything we prove about ring elements can be translated into integer linear algebra, and vice versa. That translation is how we will connect Ring/Module-LWE back to the integer lattices of Part II (§18.2).

16.3 Coefficient Growth (Section 4.1.2)

For integers, $|ab| = |a| \cdot |b|$. In $\mathcal{R}_f$, the size of ab depends heavily on f. Since $ab = \mathcal{M}_a \mathcal{V}_b$, each coefficient of ab is a sum of d products, so

$$\|ab\|_\infty \;\le\; d \cdot \|\mathcal{M}_a\|_\infty \cdot \|\mathcal{V}_b\|_\infty ,$$

where $\|\cdot\|_\infty$ is the largest absolute coefficient. (Better ℓ_2 bounds come from the largest singular value of $\mathcal{M}_a$.) Either way, *how large the entries of* $\mathcal{M}_a$ *get* is what matters.

The best we can hope for is $\|\mathcal{M}_a\|_\infty = \|\mathcal{V}_a\|_\infty$, and the only f achieving this are $X^d \pm 1$. For $f = X^d \pm X^{d/2} + 1$ and $f = \sum_{i=0}^{d} X^i$ we still have $\|\mathcal{M}_a\|_\infty \le 2\,\|\mathcal{V}_a\|_\infty$. The paper's examples, for $a = a_0 + a_1 X + a_2 X^2 + a_3 X^3$:

$f = X^4 - 1$ (Eq. (47))

$$\begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix}$$

$f = X^4 + 1$ (Eq. (48))

$$\begin{bmatrix} a_0 & -a_3 & -a_2 & -a_1 \\ a_1 & a_0 & -a_3 & -a_2 \\ a_2 & a_1 & a_0 & -a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix}$$

$f = X^4 - X^2 + 1$ (Eq. (49), corrected)

$$\begin{bmatrix} a_0 & -a_3 & -a_2 & -a_1 - a_3 \\ a_1 & a_0 & -a_3 & -a_2 \\ a_2 & a_1 + a_3 & a_0 + a_2 & a_1 \\ a_3 & a_2 & a_1 + a_3 & a_0 + a_2 \end{bmatrix}$$

$f = X^4 + X^3 + X^2 + X + 1$ (Eq. (50))

$$\begin{bmatrix} a_0 & -a_3 & -a_2 + a_3 & -a_1 + a_2 \\ a_1 & a_0 - a_3 & -a_2 & -a_1 + a_3 \\ a_2 & a_1 - a_3 & a_0 - a_2 & -a_1 \\ a_3 & a_2 - a_3 & a_1 - a_2 & a_0 - a_1 \end{bmatrix}$$

Reading the matrices (spelled out). For $f = X^4 - 1$, $X^4 \equiv 1$: multiplying by X shifts coefficients up by one and the top one wraps around to the bottom unchanged—a *cyclic* matrix. For $f = X^4 + 1$, $X^4 \equiv -1$: the wrapped coefficient changes sign—a *negacyclic*

matrix. In both, every entry is $\pm$ some a_i, which is exactly why $\|\mathcal{M}_a\|_\infty = \|\mathcal{V}_a\|_\infty$. For the other two f, reduction combines pairs of coefficients (sums or differences), so entries can reach $2\,\|\mathcal{V}_a\|_\infty$.

A typo in the paper's Eq. (49)

The entry in row 3, column 4 of the matrix for $f = X^4 - X^2 + 1$ should be a_1; the paper prints $-a_3 + a_1$. Check: modulo f, $X^4 \equiv X^2 - 1$, $X^5 \equiv X^3 - X$, $X^6 \equiv -1$, so $aX^3 \equiv (-a_1 - a_3) - a_2X + a_1X^2 + (a_0 + a_2)X^3$, whose X^2-coefficient is a_1. (We verified all four matrices symbolically; the other entries match the paper.)

Some f make things much worse: $\|\mathcal{M}_a\|_\infty \gg \|\mathcal{V}_a\|_\infty$. Unsurprisingly this happens if f has large coefficients, but also for some f with small coefficients—e.g. $f = X^d + 2X^{d-1} + 1$ gives $\mathcal{M}_a$ with *exponentially* larger coefficients than a. Such f are useless for cryptography: products of small elements would no longer be small, and decryption noise would explode. We prefer f for which the ratio $\|\mathcal{M}_a\|_\infty \,/\, \|\mathcal{V}_a\|_\infty$ is 1 or 2.

17. The Generalized-LWE and SIS Problems (Section 4.2)

Now let the coefficients live in $\mathbb{Z}_q$: $\mathcal{R}_{q,f}$ is like $\mathcal{R}_f$ but with coefficients in $\mathbb{Z}_q$ (often written $\mathbb{Z}_q[X]/(f(X))$). For a polynomial s, "$s \leftarrow [\beta]$" means every coefficient is uniform in $[\beta]$, and $\mathbf{s} \leftarrow [\beta]^m$ means all m polynomials in $\mathbf{s}$ are sampled this way (the paper's footnote 21; the notation $[\beta]$ itself is from Section 2.1).

Definition 5 (paper): $\mathcal{R}_{q,f}$-LWE$_{n,m,\beta}$

For positive integers m, n, q, $\beta < q$, and a ring $\mathcal{R}_{q,f}$, distinguish
1. $(\mathbf{A}, \mathbf{As} + \mathbf{e})$, where $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n\times m}$, $\mathbf{s} \leftarrow [\beta]^m$, $\mathbf{e} \leftarrow [\beta]^n$, from
2. $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n\times m}$ and $\mathbf{u} \leftarrow \mathcal{R}_{q,f}^{n}$.

As before, n has no known effect on hardness unless it is large, so we usually write $\mathcal{R}_{q,f}$-LWE$_{m,\beta}$. This definition and the cryptosystem below follow the line of work [LPR10, BV11, LPR13b, LS15], which related its security to worst-case problems on certain lattices.

Definition 6 (paper): $\mathcal{R}_{q,f}$-SIS$_{n,m,\beta}$

For a random $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n\times m}$, find $\mathbf{s}_1 \in [\beta]^m$ and $\mathbf{s}_2 \in [\beta]^n$, not both $\mathbf{0}$, with $\mathbf{As}_1 + \mathbf{s}_2 = \mathbf{0} \pmod q$. (Generalizing [PR06, LM06, LS15].)

Names. In the literature these are called Ring-LWE / Ring-SIS or Module-LWE / Module-SIS. In the NIST names ML-KEM and ML-DSA, "ML" stands for "Module Lattice". (Spelled out, standard usage: $d = 1$, $f = X$ gives back plain LWE over $\mathbb{Z}_q$; a single column of ring elements ($m = 1$) over a large ring is usually called "Ring-LWE"; a small $k \times k$ matrix of ring elements, as in Kyber, is "Module-LWE".)

Same problem, bigger "numbers"

Definitions 5–6 are the paper's Definitions 1 (LWE) and 4 (SIS) with every integer replaced by a ring element. Nothing about the *shape* of the problem changes—just as ElGamal over an elliptic curve is the same scheme as ElGamal over $\mathbb{Z}_p^*$, with a different group underneath. (Analogy ours.)

18. Generalized-LWE Encryption (Section 4.3)

The scheme is "virtually identical" to Section 2.3.1, with $\mathbb{Z}$ replaced by $\mathcal{R}_f$. The payoff: the message μ is now a ring element, so it carries d bits.

Generalized-LWE encryption (the paper's Eqs. (51)–(52))

Key generation:

$$\mathsf{sk} : \mathbf{s} \leftarrow [\beta]^m, \qquad \mathsf{pk} : \big(\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{m\times m},\ \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}_1\big), \quad \mathbf{e}_1 \leftarrow [\beta]^m.$$

Encryption of $\mu \in \mathcal{R}_f$ with coefficients in $\{0,1\}$: sample $\mathbf{r}, \mathbf{e}_2 \leftarrow [\beta]^m$, $e_3 \leftarrow [\beta]$, output

$$\Big(\mathbf{u}^T = \mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T,\ \ v = \mathbf{r}^T\mathbf{t} + e_3 + \tfrac{q}{2}\mu\Big).$$

Decryption: compute $v - \mathbf{u}^T\mathbf{s}$ and round each of its d coefficients to 0 or $q/2$.

Correctness (Eqs. (53)–(54)). Exactly as in Part I, the $\mathbf{r}^T\mathbf{A}\mathbf{s}$ terms cancel:

$$v - \mathbf{u}^T\mathbf{s} = \mathbf{r}^T(\mathbf{A}\mathbf{s} + \mathbf{e}_1) + e_3 + \tfrac{q}{2}\mu - (\mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T)\mathbf{s} = \mathbf{r}^T\mathbf{e}_1 + e_3 + \tfrac{q}{2}\mu - \mathbf{e}_2^T\mathbf{s}.$$

(Spelled out: the cancellation uses that ring multiplication is associative and commutative, so $(\mathbf{r}^T\mathbf{A})\mathbf{s} = \mathbf{r}^T(\mathbf{A}\mathbf{s})$ exactly as for integer matrices.)

Security is based on $\mathcal{R}_{q,f}$-LWE$_{m,\beta}$, and the argument is identical to the one for LWE$_{m,q,\beta}$: the same two game hops as §6, with ring elements in place of integers.

Decryption error. Leaving out $\frac{q}{2}\mu$, the noise in vector form is (the paper's Eq. (55))

$$\mathcal{M}_{\mathbf{r}^T}\mathcal{V}_{\mathbf{e}_1} + \mathcal{V}_{e_3} - \mathcal{M}_{\mathbf{e}_2^T}\mathcal{V}_{\mathbf{s}},$$

and the techniques of Section 2.3.2 (§5.4) give the probability that none of the d coefficients exceeds $q/4$. When $f = X^d \pm 1$ this is *the same computation as over the integers* for each coefficient: every row of $\mathcal{M}_{\mathbf{r}^T}$ and $\mathcal{M}_{\mathbf{e}_2^T}$ contains each coefficient exactly once (up to sign), so each noise coefficient is again a sum of independent products (look at the matrices (47) and (48)). A union bound over the d coefficients then bounds the probability that *any* of them is too large. For other f the technique still applies if the matrix-vector product can be rewritten as a sum of independent random variables.

18.1 Optimizations and Efficiency (Section 4.3.1)

The main advantage: to encrypt a longer message we no longer need to enlarge the public key as in Section 2.4. A degree-d ring naturally carries d message bits, so with $d \geq 256$—the length of an AES key, the typical thing one encrypts with public-key encryption—we get an optimally small public key. There is no need to pack several bits per coefficient (Section 2.5.4). The compression of Section 2.5.1 is still very useful and applies exactly as before, and LWR (Section 2.5.3) and the NIKE (Section 2.6) are defined analogously over $\mathcal{R}_f$.

Sizes, spelled out (our accounting)

With $\mathbf{A}$ generated from a 256-bit seed (Section 2.4):

	public key	ciphertext
Plain LWE, 1 bit (Section 2.3.1)	$256 + m \log q$	$(m+1)\log q$
Plain LWE, $N = k\ell$ bits (Section 2.4)	$256 + \ell m \log q$	$km \log q + k\ell \log q$
Ring, d bits (Section 4.3)	$256 + md \log q$	$(m+1)\, d \log q$

The ring version looks like the 1-bit plain scheme with every element d times bigger—but it carries d bits instead of 1. The ring's LWE dimension is dm (§18.2), so to match a plain LWE dimension of several hundred, m itself can be tiny (Kyber: $m = k \in \{2, 3, 4\}$ with $d = 256$).

ElGamal with bigger payloads

In ElGamal, a ciphertext $(g^r, h^r \mu)$ carries one group element's worth of message. Plain LWE is like an ElGamal whose "group elements" carry only one bit each, which is why its ciphertexts balloon. The ring version restores the ElGamal-like ratio: one ciphertext "element" carries a full 256-bit key. (Analogy ours.)

18.2 Security and Connection to Integer Lattices (Section 4.3.2)

The matrix view (46) translates Generalized-LWE into integer LWE. Distinguishing $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$ from uniform with $\mathbf{A} \in \mathcal{R}_{q,f}^{n \times m}$ is the same as distinguishing

$$\big(\mathcal{M}_{\mathbf{A}},\ \mathcal{V}_{\mathbf{t}} = \mathcal{M}_{\mathbf{A}}\mathcal{V}_{\mathbf{s}} + \mathcal{V}_{\mathbf{e}}\big) \quad \text{from} \quad \big(\mathcal{M}_{\mathbf{A}},\ \mathcal{V}_{\mathbf{u}}\big),$$

a problem over $\mathbb{Z}_q$. So we can attack it exactly as in Section 3.3, by looking for a short vector in (compare the paper's Eq. (56), $\mathcal{L}_q^{\perp}([\mathbf{A}^T \mid \mathbf{I}_m])$)

$$\mathcal{L}_q^{\perp}\big([\mathcal{M}_{\mathbf{A}}^T \mid \mathbf{I}_{dm}]\big),$$

where $\mathcal{M}_{\mathbf{A}}^T$ is a $dm \times dn$ integer matrix, so the lattice has dimension $d(m+n)$.

The important quantity is $d \times$ (matrix dimension)

If the algebraic structure of $\mathcal{R}_f$ has no weaknesses (see §20), finding a short vector in this lattice is as hard as in the plain $\mathrm{LWE}_{n',m',q,\beta}$ lattice with $n' = dn$ and $m' = dm$. So for $\mathcal{R}_{q,f}$-$\mathrm{LWE}_{n,m,\beta}$ the important value is dm (degree of f times the number of *columns* of $\mathbf{A}$), and for $\mathcal{R}_{q,f}$-$\mathrm{SIS}_{n,m,\beta}$ it is dn (degree times the number of *rows*).

(Spelled out: this is why Table 1's plain LWE with $m = 512, 768, 1024$ resembles Kyber, which uses $d = 256$ and $k = 2, 3, 4$: $dk = 512, 768, 1024$.)

19. NTRU (Section 4.4)

NTRU [HPS98] was the first truly efficient lattice-based encryption scheme, and the first to use polynomial rings—specifically $\mathcal{R}_{q,X^d-1}$. It was proposed as a **trapdoor one-way function**, which can be seen as a OW-CPA cryptosystem (the paper's footnote 13: an attacker holding the public key cannot recover the message of a ciphertext of a randomly chosen message). A simple modification gives CPA security [SS11], but usually the one-way function suffices, since there are black-box transformations from such primitives to CCA-secure encryption (cf. [Den02]).

19.1 The NTRU Trapdoor One-Way Function (Section 4.4.1)

Search Generalized-LWE. So far we used the *decision* version. The *search* version: given $(a, as+e)$ with $a \leftarrow \mathcal{R}_{q,f}$ and $s, e \leftarrow [\beta]$, find e. If a is invertible, finding e immediately gives $s = (as+e-e)a^{-1}$.

NTRU is the same problem (with $n = m = 1$) except that a is not uniform: it is $a = pg_1g_2^{-1}$ for small $g_1, g_2 \leftarrow [\beta]$ (g_2 invertible) and an integer $p = 2\beta+1$ relatively prime to q ($\beta = 1$, i.e. $p = 3$, is popular). The assumption is that search is still hard for this special a.

Definition 7 (paper): the NTRU problem

Let $p = 2\beta+1$. Given $(a, as+e)$ where $a = pg_1g_2^{-1}$ for $g_1, g_2, s, e \leftarrow [\beta]$ and g_2 invertible in $\mathcal{R}_{q,f}$ and $\mathcal{R}_{p,f}$, find e.

The trapdoor one-way function (Eqs. (57)–(61))

Key generation: secret small invertible $g_1, g_2 \leftarrow [\beta]$, with g_2 invertible in both $\mathcal{R}_{q,f}$ and $\mathcal{R}_{p,f}$. Public key $a = pg_1g_2^{-1}$; secret key g_2.
Evaluate on $s, e \leftarrow [\beta]$: $b = as + e \in \mathcal{R}_{q,f}$.
Invert with the trapdoor g_2:

$$g_2 b \bmod p = pg_1 s + g_2 e \bmod p = g_2 e \bmod p, \tag{59}$$

$$(g_2 b \bmod p)\, g_2^{-1} \bmod p = e, \tag{60}$$

$$(b-e)a^{-1} = s. \tag{61}$$

Why this works (the subtle part). Two moduli, q and p, are relatively prime—"very rarely an idea that leads to any meaningful results," as the paper says. Three facts make it work:

1. **No wraparound.** $g_2 b = g_2(pg_1g_2^{-1}s + e) = pg_1 s + g_2 e$ in $\mathcal{R}_{q,f}$. But g_1, g_2, s, e and p are so small relative to q that every coefficient of $pg_1 s + g_2 e$ lies strictly between $-q/2$ and $q/2$. So reading $g_2 b$ with centred coefficients gives $pg_1 s + g_2 e$ *over* $\mathcal{R}_f$, i.e. over the integers, not just mod q.
2. **Reducing mod p kills the a-part.** An equation over $\mathcal{R}_f$ stays true modulo p, and $pg_1 s \equiv 0 \pmod p$. Only $g_2 e \bmod p$ survives. Multiplying by g_2^{-1} in $\mathcal{R}_{p,f}$ gives $e \bmod p$.
3. **Mod p is enough.** Since $p = 2\beta+1$, the elements of $[\beta] = \{-\beta, \ldots, \beta\}$ are exactly one representative of each residue mod p. So $e \bmod p$ determines e, and then (61) gives s.

Worked example (illustration, our numbers): $f = X^4+1$, $q = 97$, $\beta = 1$, $p = 3$

Take $g_1 = 1-X+X^3$ and $g_2 = 1+X-X^3$. In both $\mathcal{R}_{97,f}$ and $\mathcal{R}_{3,f}$, $g_2^{-1} = -1+X-X^3$ (one checks $g_2 \cdot g_2^{-1} = 1$). Then

$$a = 3g_1g_2^{-1} \bmod 97 = 88 + 6X + 91X^3.$$

Evaluate at $s = X - X^2 + X^3$, $e = 1 + X^2 - X^3$: $b = as + e = 1 + 82X + 22X^2 + 81X^3$. Invert: $g_2 b \bmod 97$, centred, is $2 + 8X - 9X^2 + 5X^3$, which equals $3g_1s + g_2e$ computed over the integers (no wraparound: for *all* $s, e \in [1]^4$ the largest coefficient of $3g_1s + g_2e$ is $12 < 97/2$). Reducing mod 3 gives $2 + 2X + 2X^3$. Multiplying by g_2^{-1} in $\mathcal{R}_{3,f}$ and centring gives $1 + X^2 - X^3 = e$, and $(b - e)a^{-1} = X - X^2 + X^3 = s$.

A lattice RSA

The paper frames NTRU as a trapdoor one-way function, the same role RSA plays in discrete-log-free public-key crypto: anyone can evaluate $b = as + e$ (like $x^e \bmod N$), but only the holder of the trapdoor g_2 (like the factorization of N) can invert it. LWE encryption, by contrast, follows the ElGamal template. (Analogy ours.)

19.2 Security (Section 4.4.2)

Attacking (s, e) directly. If we ignore the special form of a, recovering s, e from (58) is the same as attacking a Generalized-LWE public key: find a short vector in (the paper's Eq. (62))

$$\mathcal{L}_q^{\perp}\big([\mathcal{M}_a \mid \mathcal{V}_b \mid \mathbf{I}_d]\big).$$

Attacking the key. One can also try to recover g_1, g_2 (or other short polynomials related to them) from $a = pg_1g_2^{-1}$ by finding a short vector in (Eq. (63))

$$\mathcal{L}_q^{\perp}\big([\mathcal{M}_{p^{-1}a} \mid \mathbf{I}_d]\big).$$

(Spelled out: $p^{-1}a \cdot g_2 - g_1 = 0$, so $(g_2, -g_1)$ is a short vector in this lattice.) The two lattices differ only by the one extra column $\mathcal{V}_b$ in (62). It was therefore somewhat surprising that when q is much larger than β (but not so large that generic lattice reduction clearly works), finding short vectors in (63) can be *significantly easier* than in (62) [ABD16, CJL16, KF17]. These attacks do not affect NTRU's actual parameters, but they rule out the NTRU assumption in advanced primitives (e.g. FHE) that need large moduli and small noise. There, schemes based on Generalized-LWE, whose security rests essentially on (62), are used.

20. Exploiting Algebraic Structure for Attacks (Section 4.5)

Take $f = X^d - 1$. Since $X - 1$ divides $X^d - 1$, there is a ring homomorphism $\mathcal{R}_{q,f} \to \mathcal{R}_{q,X-1} = \mathbb{Z}_q$:

$$a = \sum_{i=0}^{d-1} a_i X^i \;\longmapsto\; a' = \sum_{i=0}^{d-1} a_i \in \mathbb{Z}_q.$$

(Spelled out: this is "evaluate at $X = 1$". It respects multiplication because $f(1) = 0$, so reducing modulo f does not change the value at 1.) The special feature: *small coefficients stay small*—the image is at most d times larger.

The attack (adapted by the paper from [PR06, LM06

)] Given an $\mathcal{R}_{q,f}$-LWE$_{n,m,\beta}$ instance $\mathbf{A} \in \mathcal{R}_{q,f}^{n\times m}$, $\mathbf{t} \in \mathcal{R}_{q,f}^{n}$, map everything through the homomorphism to $\mathbf{A}' \in \mathbb{Z}_q^{n\times m}$, $\mathbf{t}' \in \mathbb{Z}_q^n$. If $\mathbf{s} \in [\beta]^m$, $\mathbf{e} \in [\beta]^n$ with $\mathbf{As} + \mathbf{e} = \mathbf{t}$ really exist, then

$$\mathbf{A}'\mathbf{s}' + \mathbf{e}' = \mathbf{t}' \quad \text{with} \quad \mathbf{s}' \in [d\beta]^m,\ \mathbf{e}' \in [d\beta]^n.$$

Because m and n are fairly small, this is a *small-dimensional* LWE problem, which can be solved as in Section 3.3.

What made it work: a homomorphism to a smaller ring that does not increase coefficients by much. If f had the factor $X - 2$ instead, the homomorphism would be $a \mapsto \sum_i a_i 2^i$, and the image would be exponentially large in d—useless to the attacker.

Irreducible over $\mathbb{Z}$, not over $\mathbb{Z}_q$

Interestingly, the worst-case to average-case reductions for $\mathcal{R}_{q,f}$-LWE$_{m,\beta}$ and $\mathcal{R}_{q,f}$-SIS$_{n,m,\beta}$ [PR06, LM06, LPR13a, LS15, PRS17] only require f to be irreducible (or, for SIS, to have a large-degree irreducible factor) over $\mathbb{Z}[X]$, *not* over $\mathbb{Z}_q[X]$. And as the next section shows, it is actually very advantageous for efficiency if f has *many* low-degree factors in $\mathbb{Z}_q[X]$.

Pohlig–Hellman

The discrete-log analogue: if the group order has small factors, Pohlig–Hellman projects the problem into small subgroups and solves it there. Here the "projection" is a ring homomorphism to a small ring. In both cases, the defence is to make sure no *cheap* projection exists—for lattices, a factor like $X - 1$ over $\mathbb{Z}$ that keeps coefficients small. (Analogy ours.)

21. Algebraic Structure for Efficiency: the NTT (Section 4.6)

The most expensive operation in the scheme is multiplying polynomials in $\mathcal{R}_{q,f}$. Schoolbook multiplication of degree-d polynomials takes $O(d^2)$ operations; Karatsuba and Toom–Cook take roughly $O(d^{1.5})$. The **Number Theoretic Transform** (NTT) needs as few as $O(d \log d)$ operations over $\mathbb{Z}_q$. It is the FFT, performed over the field $\mathbb{Z}_q$ instead of over the complex numbers. Its use for lattice primitives began with the SWIFFT hash function [LMPR08] and is now standard in encryption [ADPS16, BDK$^+$18] and signatures [DKL$^+$18, PFH$^+$17].

21.1 One Level: Split With the Chinese Remainder Theorem

Work in $\mathbb{Z}_q[X]/(X^d + \alpha)$ with d a power of 2. Suppose $-\alpha$ has a square root r in $\mathbb{Z}_q$. Then

$$X^d + \alpha \equiv (X^{d/2} - r)(X^{d/2} + r) \pmod{q},$$

and by the Chinese Remainder Theorem we can compute $ab \bmod (X^d + \alpha)$ by (Eqs. (64)–(66)):

1. reducing: $(a \bmod X^{d/2} - r,\ a \bmod X^{d/2} + r)$ and the same for b;
2. multiplying component-wise: $(ab \bmod X^{d/2} - r,\ ab \bmod X^{d/2} + r)$, two multiplications in half-size rings;

3. reconstructing $ab \bmod X^d + \alpha$.

(Spelled out: the two factors are coprime, since their difference is the constant $2r$, which is invertible in $\mathbb{Z}_q$ when, e.g., q is an odd prime and $r \neq 0$. That is exactly what CRT needs.)

Lemma 5 (paper): one level costs n additions and $n/2$ multiplications

Suppose $X^n + \alpha \equiv (X^{n/2} - r)(X^{n/2} + r) \pmod q$ and define

$$\phi : \mathbb{Z}_q[X]/(X^n + \alpha) \to \mathbb{Z}_q[X]/(X^{n/2} - r) \times \mathbb{Z}_q[X]/(X^{n/2} + r),$$
$$\phi(a) = \big(a \bmod X^{n/2} - r,\ a \bmod X^{n/2} + r\big).$$

If r and $r^{-1} \bmod q$ are precomputed, then ϕ and $2 \cdot \phi^{-1}$ can each be computed with n additions/subtractions and $n/2$ multiplications over $\mathbb{Z}_q$.

Proof (with the reduction spelled out). Write $a = \sum_{i=0}^{n-1} a_i X^i = \sum_{i<n/2} a_i X^i + X^{n/2} \sum_{i<n/2} a_{i+n/2} X^i$. Modulo $X^{n/2} - r$ we may replace $X^{n/2}$ by r; modulo $X^{n/2} + r$, by $-r$. So the two reductions $\sum b_i X^i$ and $\sum c_i X^i$ are

$$b_i = a_i + r \cdot a_{i+n/2}, \qquad c_i = a_i - r \cdot a_{i+n/2} \qquad (0 \le i < n/2).$$

That is $n/2$ multiplications (each $r \cdot a_{i+n/2}$ is used twice) and n additions/subtractions. In reverse:

$$2a_i = b_i + c_i, \qquad 2a_{i+n/2} = r^{-1}(b_i - c_i),$$

again $n/2$ multiplications and n additions/subtractions. □

A small inconsistency in the paper

The paper's proof of Lemma 5 says the reverse direction takes "$2n$ additions (or subtractions)", while the lemma states n. Counting the formulas above gives n ($n/2$ additions for the $2a_i$ and $n/2$ subtractions for the $2a_{i+n/2}$), matching the statement.

Why $2 \cdot \phi^{-1}$ and not ϕ^{-1}? It saves multiplications. The recursion runs this step $\log d$ times, and the factor 2 accumulates to $2^{\log d} = d$. So at the very end we multiply once by d^{-1}, instead of multiplying by 2^{-1} at every level.

21.2 Recursing All the Way Down

If $X^{d/2} - r$ factors further as $(X^{d/4} - s)(X^{d/4} + s)$ (and similarly $X^{d/2} + r = (X^{d/4} - t)(X^{d/4} + t)$), we can compute the half-size products recursively. With $T(d)$ the cost of one multiplication in $\mathbb{Z}_q[X]/(X^d + \alpha)$, and A, M the costs of one addition and one multiplication in $\mathbb{Z}_q$, the paper's recurrence is (Eq. (67))

$$T(d) = 2 \cdot T(d/2) + 2d \cdot A + d \cdot M.$$

If d is a power of 2 and -1 and $-\alpha$ have d-th roots in $\mathbb{Z}_q$, the recursion continues down to linear factors (footnote 14 of the paper proves the invariant: at every level the factors have the form $X^k \pm s^l r^k$). The solution is (Eq. (68))

$$T(d) = d \cdot T(1) + 2d \log d \cdot A + d \log d \cdot M, \qquad T(1) = M,$$

so a product costs as few as $2d \log d$ additions and $d(\log d + 1)$ multiplications.

(Spelled out: unrolling, level j has 2^j subproblems of size $d/2^j$, each costing $2(d/2^j)A + (d/2^j)M$; every level therefore costs $2dA + dM$ in total, there are $\log d$ levels, and at the bottom d products of constants cost $d \cdot T(1)$. Counting every transform separately—forward for a and for b, then one reconstruction—gives a larger constant, but the same $O(d \log d)$.)

When does it split all the way? For $\alpha = 1$, the ring $\mathcal{R}_{q,X^d+1} = \mathbb{Z}_q[X]/(X^d + 1)$, splitting $X^d + 1$ into linear factors needs -1 to have a d-th root, i.e. $\mathbb{Z}_q^*$ must contain a $2d$-th root of unity. That happens whenever q is a prime with $q \equiv 1 \pmod{2d}$ (Lemma 7). If q does not satisfy this, the NTT still works but stops earlier. For example, if $q \equiv 1 \pmod{d}$, $X^d + 1$ splits into quadratic factors $X^2 - r_i$, and the base case is a multiplication in $\mathbb{Z}_q[X]/(X^2 - r_i)$, which costs a small constant number of operations (5 multiplications and 2 additions). Because there is one level fewer, the total is virtually the same.

Worked example (illustration, our numbers): the full NTT for $q = 17$, $d = 4$

Here $17 \equiv 1 \pmod 8$, so $X^4 + 1$ splits completely. Since $4^2 = 16 \equiv -1$, $2^2 = 4$ and $8^2 = 64 \equiv 13 \equiv -4 \pmod{17}$:

$$X^4 + 1 \equiv (X^2 - 4)(X^2 + 4) \equiv (X - 2)(X + 2)(X - 8)(X + 8) \pmod{17}.$$

Take $a = 1 + 2X + 3X^2 + 4X^3$ and $b = 5 + 6X + 7X^2$.

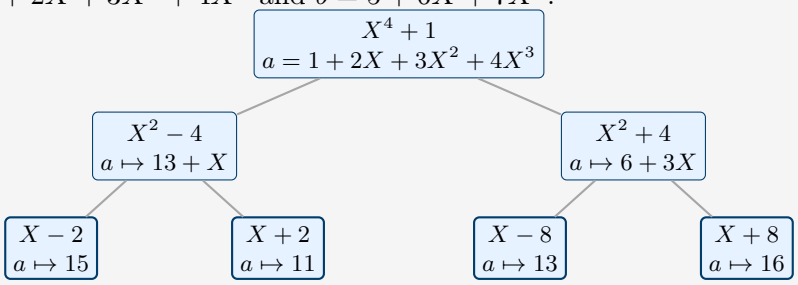


Figure 7: Illustration (ours): the CRT tree of the NTT for $q = 17$, $d = 4$. Each step is one application of Lemma 5; the leaves are $a(2), a(-2), a(8), a(-8)$ mod 17.

- Level 1 ($r = 4$): $b_i = a_i + 4a_{i+2}$ gives $(1 + 12,\ 2 + 16) \equiv (13, 1)$; $c_i = a_i - 4a_{i+2}$ gives $(1 - 12,\ 2 - 16) \equiv (6, 3)$.
- Level 2 ($r = 2$ on the left, $r = 8$ on the right): $13 \pm 2 \cdot 1 = 15, 11$; $6 \pm 8 \cdot 3 = 30, -18 \equiv 13, 16$.
- So $\hat{a} = (15, 11, 13, 16)$ and likewise $\hat{b} = (11, 4, 8, 14)$. Pointwise product: $\hat{a}\hat{b} = (12, 10, 2, 3)$.
- Reverse, level 2 (using $2^{-1} = 9$, $8^{-1} = 15$): $(12 + 10,\ 9(12 - 10)) \equiv (5, 1)$ and $(2 + 3,\ 15(2 - 3)) \equiv (5, 2)$.
- Reverse, level 1 (using $4^{-1} = 13$): $(5{+}5,\ 1{+}2,\ 13(5{-}5),\ 13(1{-}2)) \equiv (10, 3, 0, 4)$. This is $4ab$.
- Multiply by $d^{-1} = 4^{-1} = 13$: $ab = 11 + 5X + X^3$.

Direct check: schoolbook multiplication modulo $X^4 + 1$ and 17 gives the same $11 + 5X + 0X^2 + X^3$.

Other rings. f need not be $X^d + 1$. Some other cyclotomic polynomials have a similar factorization tree: for certain $d = 2^k \cdot 3$ and primes q, $f = X^d - X^{d/2} + 1$ factors as $(X^{d/2} - r_1)(X^{d/2} - r_2)$ modulo q, and the recursion proceeds, bottoming out at degree-3 rather than linear factors, almost as efficiently [LS19]. For arbitrary f one can multiply in $\mathbb{Z}_q[X]$ using a ring $\mathcal{R}_{q,X^d+1}$ with d large enough ($d > 2(\deg(f) - 1)$) that the reduction modulo $X^d + 1$ never happens, and then reduce modulo f; this is very often still the fastest

method [CHK+21].

21.3 Useful Algebraic Properties of $\mathcal{R}_{q,X^d+1}$ (Section 4.6.1)

We want X^d+1 *irreducible over* $\mathbb{Z}[X]$ (for security, §20) but *highly reducible over* $\mathbb{Z}_q[X]$ (for the NTT). Both are possible.

Lemma 6 (paper)

X^d+1 is irreducible over $\mathbb{Z}[X]$ if and only if d is a power of 2.

Proof. Let Φ_k be the k-th cyclotomic polynomial; recall $X^n-1=\prod_{k|n}\Phi_k(X)$. Since $(X^d+1)(X^d-1)=X^{2d}-1$,

$$X^d+1=\frac{X^{2d}-1}{X^d-1}=\frac{\prod_{k|2d}\Phi_k(X)}{\prod_{k|d}\Phi_k(X)}=\prod_{k\,:\,k|2d,\ k\nmid d}\Phi_k(X).$$

If $d=2^\ell$, the only divisor of $2d=2^{\ell+1}$ that does not divide d is $2^{\ell+1}$ itself, so $X^d+1=\Phi_{2d}(X)$, which is irreducible (all cyclotomic polynomials are). If $d=2^\ell d'$ with $d'>1$ odd, then both $2d$ and $2d/d'=2^{\ell+1}$ divide $2d$ but not d, so Φ_{2d} and $\Phi_{2d/d'}$ are two distinct factors. □

(Spelled out, for $d=3$: $X^3+1=(X+1)(X^2-X+1)=\Phi_2\Phi_6$.)

Lemma 7 (paper): factoring mod q

Let $d\ge k\ge 1$ with $k\mid d$, and let $q\equiv 1 \pmod{2k}$ be prime. Then there are k distinct $r_i\in\mathbb{Z}_q^*$ with $r_i^k\equiv -1 \pmod q$ such that

$$X^d+1\equiv\prod_{i=1}^{k}(X^{d/k}-r_i)\pmod q. \tag{69}$$

Proof. $\mathbb{Z}_q^*$ is cyclic of order $q-1$, and $2k\mid q-1$, so there is an r of order exactly $2k$; then $r^k\equiv -1$ (it squares to 1 but is not 1). The k odd powers r^{2i+1}, $i\in\{0,\dots,k-1\}$, are distinct and satisfy $(r^{2i+1})^k=(r^k)^{2i+1}\equiv -1$. So they are the k roots of X^k+1, and $X^k+1\equiv\prod_{i=0}^{k-1}(X-r^{2i+1})$. Substituting $X^{d/k}$ for X gives (69). □

(The paper's proof reads "for all odd $i\in\{0,\dots,k-1\}$"; it is the *exponents* $2i+1$ that are odd, for all i.) In the $q=17$ example above, $k=d=4$ and $r=2$ has order 8; its odd powers $2,8,2^5\equiv 15\equiv -2,2^7\equiv 9\equiv -8$ are exactly the four roots we used.

Lemma 8 (paper): never a field

Let q be an odd prime and d a multiple of 4. Then X^d+1 factors into at least 2 polynomials over $\mathbb{Z}_q[X]$. So $\mathcal{R}_{q,X^d+1}$ is never a field.

Proof. If $q\equiv 1 \pmod 4$, use Lemma 7 with $k=2$. If $q\equiv 3 \pmod 4$, then for every $x\in\mathbb{Z}_q^*$ exactly one of $x,-x$ is a square (footnote 17 of the paper). Let $b=1$ if 2 is a square and $b=-1$ if -2 is, and let $r^2\equiv 2b$. Then, using $b^2=1$ and $r^2=2b$,

$$\begin{aligned}X^d+1&\equiv X^d+(2b-r^2)X^{d/2}+1\equiv(X^{d/2}+b)^2-r^2X^{d/2}\\&\equiv\left(X^{d/2}+b+rX^{d/4}\right)\left(X^{d/2}+b-rX^{d/4}\right)\pmod q.\end{aligned}$$

□

The paper notes this lemma is not used later, but it matters for some advanced applications: to get an "almost-field" one can choose parameters where $X^d + 1$ factors into just two irreducibles $X^{d/2} \pm r$ [LS18].

The FFT you already know

The NTT is the FFT with $\mathbb{Z}_q$ in place of $\mathbb{C}$: "evaluate at the $2d$-th roots of unity, multiply pointwise, interpolate back." The leaves in our Figure 7 are exactly a evaluated at the four primitive 8-th roots of unity mod 17. There is no discrete-log counterpart to this speed-up; the paper notes (Section 4.8) that NTT-based operations are very fast compared with the exponentiations or elliptic-curve operations of classical cryptography. (Analogy ours.)

22. CRYSTALS-Kyber (ML-KEM) (Section 4.7)

Kyber puts everything together. It is the Generalized-LWE scheme of §18 over the ring $\mathcal{R}_{3329,X^{256}+1}$, with ciphertext compression (Sections 2.5.1–2.5.2), NTT multiplication, and secrets drawn from a **binomial** distribution instead of the uniform one—simply because it is easier to sample.

Definition 8 (paper): the binomial distribution ψ_η

Sample $a_1, \ldots, a_\eta, b_1, \ldots, b_\eta \leftarrow \{0,1\}$ and output $\sum a_i - \sum b_i$. For a polynomial, $a \leftarrow \psi_\eta$ means every coefficient is sampled independently from ψ_η; for a vector of k polynomials, $\mathbf{a} \leftarrow \psi_\eta^k$ means every entry is sampled this way. (The paper writes "according to ψ_k" at the end of the definition; it means ψ_η.)

Spelled out. ψ_η ranges over $\{-\eta, \ldots, \eta\}$ with $\Pr[x] = \binom{2\eta}{\eta+x}/4^\eta$:

x	-3	-2	-1	0	1	2	3
ψ_2 $(\times 16)$		1	4	6	4	1	
ψ_3 $(\times 64)$	1	6	15	20	15	6	1

Sampling needs only 2η random bits and some additions, with no rejection. (Section 2.3 of the paper already noted that "some practical implementations, notably Kyber, use the binomial distribution" because generating a string of bits and adding them up is sometimes faster than generating a uniform element of $[\beta]$.)

The paper's Figure 3: CRYSTALS-Kyber CPA-secure encryption

Public parameters: $k, \eta_1, \eta_2, d_u, d_v \in \mathbb{Z}^+$.

CPA-KeyGen
- $\mathbf{A} \leftarrow \mathcal{R}_{3329,X^{256}+1}^{k \times k}$
- $(\mathbf{s}, \mathbf{e}) \leftarrow \psi_{\eta_1}^k \times \psi_{\eta_1}^k$
- $\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e}$
- $pk = (\mathbf{A}, \mathbf{t})$, $sk = \mathbf{s}$

CPA-Encrypt(pk, m)
- $(\mathbf{r}, \mathbf{e}_1, e_2) \leftarrow \psi_{\eta_1}^k \times \psi_{\eta_2}^k \times \psi_{\eta_2}$
- $\mathbf{u}^T := \lceil \mathbf{r}^T \mathbf{A} + \mathbf{e}_1^T \rfloor_{q \to 2^{d_u}}$
- $v := \left\lceil \mathbf{r}^T \mathbf{t} + e_2 + \frac{q-1}{2} m \right\rfloor_{q \to 2^{d_v}}$
- $ct = (\mathbf{u}, v)$

CPA-Decrypt(sk, ct)
- $\mathbf{u}' := \lceil \mathbf{u} \rfloor_{2^{d_u} \to q}$
- $v' := \lceil v \rfloor_{2^{d_v} \to q}$
- $m' := \lceil v' - \mathbf{u}'^T \mathbf{s} \rfloor_{q \to 2}$

Reading Figure 3. k is the main security parameter varied between levels (the module rank: $\mathbf{A}$ is $k \times k$). η_1, η_2 set the noise. d_u, d_v are the log of the size of the set $\mathcal{S}$ (Figure 1, Eq. (18)) that the two ciphertext parts are rounded to; they determine the ciphertext size and the decryption error. Key generation is (51) with binomial secrets; encryption computes

the uncompressed ciphertext (52) and compresses it (Sections 2.5.1–2.5.2); decryption is (54), with the compression function recovering the 0/1 coefficients as in (20). (Spelled out: $q = 3329$ is odd, so $q/2$ is not an integer; the message is scaled by $\frac{q-1}{2} = 1664$ instead.)

	k	η_1	η_2	d_u	d_v	decryption error	pk size	ciphertext size
Kyber-512	2	3	2	10	4	2^{-139}	800 B	768 B
Kyber-768	3	2	2	10	4	2^{-164}	1184 B	1088 B
Kyber-1024	4	2	2	11	5	2^{-174}	1568 B	1568 B

Table 3: The paper's Table 3: parameters of the three Kyber instantiations. Their security is "supposedly" no worse in practice than AES-128, AES-192 and AES-256 respectively.

Checking the sizes (spelled out). Each coefficient mod $3329 < 2^{12}$ takes 12 bits. The public key is a 32-byte seed for $\mathbf{A}$ plus $\mathbf{t}$ (k polynomials of 256 coefficients): $32 + 384k$ bytes, i.e. $800, 1184, 1568$. The ciphertext is $\mathbf{u}$ ($k \cdot 256$ coefficients of d_u bits) plus v (256 coefficients of d_v bits): $32kd_u + 32d_v$ bytes, i.e. $640+128 = 768$, $960+128 = 1088$, and $1408+160 = 1568$. All match Table 3.

22.1 Correctness and Decryption Error

Write e' and $\mathbf{e}''$ for the compression errors (their coefficients are the η of Lemma 1). Then

$$\lceil v' - \mathbf{u}'^T\mathbf{s} \rfloor_{q\to 2} = \left\lceil \mathbf{r}^T\mathbf{t} + e_2 + \tfrac{q-1}{2}m + e' - \left(\mathbf{r}^T\mathbf{A} + \mathbf{e}_1^T + \mathbf{e}''^T\right)\mathbf{s} \right\rfloor_{q\to 2},$$

and replacing $\mathbf{t}$ by $\mathbf{As} + \mathbf{e}$ (the $\mathbf{r}^T\mathbf{As}$ terms cancel), the paper's Eq. (70):

$$= \left\lceil \mathbf{r}^T\mathbf{e} + e_2 + \tfrac{q-1}{2}m + e' - (\mathbf{e}_1 + \mathbf{e}'')^T\mathbf{s} \right\rfloor_{q\to 2}.$$

The result is m if every coefficient of (Eq. (71))

$$\mathbf{r}^T\mathbf{e} + e_2 + e' - (\mathbf{e}_1 + \mathbf{e}'')^T\mathbf{s}$$

is less than $q/4$ in magnitude. This is the computation of §5.4, adapted to rings as in (55), with two extra terms e' and $\mathbf{e}''$ from compression. Assuming the compressed values are uniform, the distribution of each coefficient of e' is that of $\lceil \lceil x \rfloor_{q\to 2^{d_v}} \rfloor_{2^{d_v}\to q} - x$ for uniform $x \in \mathbb{Z}_q$ (and with d_u for $\mathbf{e}''$). Compute the exact error probability of one coefficient, then multiply by 256 (union bound).

Illustration (our computation): reproducing Table 3's decryption errors

We ran exactly this recipe: exact convolution of the distributions of $\mathbf{r}^T\mathbf{e}$ ($256k$ products of ψ_{η_1} values), $e_2 \sim \psi_{\eta_2}$, e' (compression error of v), and $(\mathbf{e}_1 + \mathbf{e}'')^T\mathbf{s}$ ($256k$ products of ψ_{η_2} + compression error, times ψ_{η_1}), then $\Pr[|\text{noise}| > 832] \times 256$.

	ours	Table 3
Kyber-512	$2^{-139.1}$	2^{-139}
Kyber-768	$2^{-165.2}$	2^{-164}
Kyber-1024	$2^{-175.2}$	2^{-174}

Agreement to within about one bit; the small gaps likely come from details such as the exact decision threshold and rounding convention.

22.2 Security

Kyber's security is based on the hardness of $\mathcal{R}_{3329,X^{256}+1}$-LWE$_{k,\psi_2}$; the proof is exactly that of Sections 2.3.1 and 4.3. One subtlety: in Kyber-512, $\eta_1 = 3$ but $\eta_2 = 2$. So:

- distinguishing the public key $(\mathbf{A}, \mathbf{t})$ from uniform rests on $\mathcal{R}$-LWE$_{k,\psi_3}$;
- distinguishing the ciphertext from uniform rests on a "hybrid" distribution ($\mathbf{r}$ from ψ_3, $\mathbf{e}_1, e_2$ from ψ_2), which is at least as hard as $\mathcal{R}$-LWE$_{k,\psi_2}$.

But the scheme does not output $\mathbf{r}^T\mathbf{A} + \mathbf{e}_1^T$; it outputs the *rounded* $\lceil \mathbf{r}^T\mathbf{A} + \mathbf{e}_1^T \rfloor_{q \to 2^{d_u}}$, which adds extra error as in LWR (Section 2.5.3). The combined error of ψ_2 plus compression from 3329 down to $2^{d_u} = 1024$ values is actually somewhat larger than ψ_3. So technically the ciphertext's security rests on $\mathcal{R}$-LWE$_{k,\psi_2}$, but in practice one gets a few extra bits of heuristic security and the problem should be as hard as $\mathcal{R}$-LWE$_{k,\psi_3}$. The price for using ψ_3 for $\mathbf{s}, \mathbf{e}, \mathbf{r}$ is a larger decryption error.

22.3 Computational Efficiency

- **Seed instead of A.** $\mathbf{A}$ is uniform, so store a 256-bit seed ρ and set $\mathbf{A} = \mathcal{H}(\rho)$ for an extendable hash such as SHAKE. The public key is just $(\rho, \mathbf{t})$.
- **NTT parameters.** $3329 \equiv 1 \pmod{256}$, so by Lemma 7 (with $k = 128$) $X^{256} + 1$ splits into 128 quadratics $X^2 - r_i$. For $a \in \mathcal{R}_{X^{256}+1}$ its **NTT representation** is (Eq. (72))
$$\hat{a} = \big(a \bmod X^2 - r_1,\ \ldots,\ a \bmod X^2 - r_{128}\big),$$
and converting a to $\hat{a}$ (and back) takes $O(d \log d)$ operations. Why not a prime that splits $X^{256} + 1$ into linear factors? None of similar size exists: that would need $q \equiv 1 \pmod{512}$ (footnote 18; spelled out: $3328 = 2^8 \cdot 13$, divisible by 256 but not 512). And the quadratic base case costs virtually nothing extra (§21).
- **Sample A directly in NTT form.** NTT is a bijection, so a uniform $\hat{\mathbf{A}}$ is as good as a uniform $\mathbf{A}$. Also store $\mathbf{t}$ in NTT form: no inverse NTT after computing $\mathbf{As}$, and encryption needs $\hat{\mathbf{t}}$ anyway for $\mathbf{r}^T\mathbf{t}$—a double win. Since $\mathbf{A}$ has k^2 polynomials, this avoids k^2 NTTs. (The paper says $\mathbf{A}$ "consists of k polynomials"; a $k \times k$ matrix has k^2, as the paper's own "k^2 NTT computations" implies.)
- **What cannot be done in NTT form.** $\mathbf{s}, \mathbf{e}, \mathbf{r}, \mathbf{e}_1, e_2$ are not uniform, so they must be sampled normally and transformed. Compression $\lceil \cdot \rfloor_{q \to p}$ is coefficient-wise and cannot be applied in NTT form either.

23. From CPA Encryption to a CCA-KEM (Section 4.8)

Definitions: KEM, CPA-KEM, CCA-KEM

A **key encapsulation mechanism** (KEM) lets two parties agree on a random shared key. It has three algorithms:

- KEM-KeyGen outputs a public key and a secret key;
- KEM-Encaps takes the public key and outputs a shared key and a ciphertext;
- KEM-Decaps takes the ciphertext and the secret key and outputs the same shared key.

CPA-secure: given the public key and the ciphertext, the adversary cannot distinguish the shared key from uniform. Any CPA-secure encryption gives one: encrypt a random message and use it as the key.

CCA-secure: the same indistinguishability, even when the adversary may also query a *decapsulation oracle* on any ciphertext except the challenge.

The Fujisaki–Okamoto (FO) transform. The intuition is to make the decapsulation oracle *useless*: it should output something other than $\perp$ only on ciphertexts whose message the adversary already knows. Two ingredients achieve this:

1. derive the encryption randomness *from the message* (so encryption becomes deterministic—harmless, because we only ever encrypt random messages; footnote 20);
2. on decapsulation, decrypt, *re-encrypt*, and reject if the result differs from the received ciphertext.

The paper's Figure 4: CCA-secure KEM via the Fujisaki–Okamoto transform

Public parameters: those of the CPA scheme of Figure 3. $\mathcal{H}$ and $\mathcal{G}$ are modelled as random oracles.

KEM-KeyGen	KEM-Encaps(pk)	KEM-Decaps(sk, c, h, z)
$(pk, sk) \leftarrow$ CPA-KeyGen	$m \leftarrow \{0,1\}^{256} \in \mathcal{R}_{X^{256}+1}$	$m' :=$ CPA-Decrypt(sk, c)
$pk := (\mathbf{A}, \mathbf{t})$, $sk := \mathbf{s}$	$(K, \rho) := \mathcal{H}(m, pk) \in \{0,1\}^{512}$	$(K', \rho') := \mathcal{H}(m', pk)$
	$c :=$ CPA-Encrypt(pk, m, ρ)	$c' :=$ CPA-Encrypt(pk, m', ρ')
	Shared key $:= K$, ctxt $:= c$	if $c \neq c'$ then $K' := \perp$
		Shared key $:= K'$

The Decaps inputs h and z belong to the ML-KEM modifications described below (pre-hashed public key and implicit rejection). The extra input $\rho \in \{0,1\}^{256}$ to CPA-Encrypt is the random coins used to generate $(\mathbf{r}, \mathbf{e}_1, e_2)$. (Hashing m together with pk, rather than using m as the key directly, is for slightly more advanced security notions, and is good practice in general; footnote 19.)

Why the re-encryption check makes the oracle useless (informal, not in the paper). To get a non-$\perp$ answer on some c, the check requires $c =$ CPA-Encrypt(pk, m', ρ') with $(K', \rho') = \mathcal{H}(m', pk)$. In the random-oracle model, the only practical way to produce such a c is to have queried $\mathcal{H}(m', pk)$ and run the encryption yourself—in which case you already know m' and K', and the oracle tells you nothing new. A simulator can therefore answer decapsulation queries by scanning the adversary's hash queries, without the secret key. (The rigorous proof, including the effect of the small decryption error, is beyond the paper and this guide; this is one reason Section 2.3.2 aims for error probabilities like 2^{-150}.)

Why CPA is not enough: malleability

Plain ElGamal is malleable: from (c_1, c_2) anyone can make $(c_1, c_2 \cdot g)$, which decrypts to $\mu \cdot g$. A decryption oracle for the mauled ciphertext then reveals μ. LWE encryption is malleable in the same way: adding $\frac{q}{2}$ to v flips the message bit. FO's re-encryption check kills every such mauled ciphertext, because it is not the deterministic encryption of its own decryption. (Analogy ours.)

23.1 Modifications Specific to Lattice Encryption

The standardized ML-KEM changes Figure 4 in two ways.

- **Pre-hashing the public key.** Unlike in discrete-log schemes, the lattice public key is large (≈ 1 KB), and NTT arithmetic is fast compared with exponentiation or elliptic-curve operations. So hashing pk inside Encaps and Decaps is noticeable: somewhere between 30 and 50 percent of the running time with AVX-2 instructions. Since Decaps typically runs more often than KeyGen (a party's public key may be fixed), KeyGen computes and stores $h = \mathcal{G}(pk)$, and Encaps and Decaps use h instead of pk as input to $\mathcal{H}$ (nothing is saved in Encaps, but Decaps hashes 32 bytes instead of ≈ 1 KB).
- **Implicit rejection.** Kyber never outputs $\perp$: on a mismatch it outputs a random-looking key computed as a hash of the input ciphertext and a secret random value created at key generation. The paper notes the reasoning is "somewhat technical" and "it is not really clear whether this adds any security in practice."

24. Summary of Part III (Section 4 of the Paper)

What to remember from Section 4

1. **Rings pack d bits per element.** Replacing $\mathbb{Z}_q$ by $\mathcal{R}_{q,f}$ removes the ciphertext blow-up without growing the public key; the security-relevant dimension becomes $d\times$(matrix dimension).
2. **Multiplication by a is the matrix $\mathcal{M}_a$.** For $f = X^d \pm 1$ it is (nega)cyclic, so products of small elements stay small.
3. **Choose f irreducible over $\mathbb{Z}$** (e.g. $X^d + 1$ with d a power of 2, Lemma 6) to avoid cheap homomorphisms such as evaluation at 1 (Section 4.5)—**but highly reducible mod q** (Lemma 7) so the NTT computes products in $O(d \log d)$.
4. **NTRU** is a trapdoor one-way function $b = as + e$ with a special $a = pg_1 g_2^{-1}$; inverting uses two coprime moduli and the absence of wraparound.
5. **Kyber** = Module-LWE over $\mathcal{R}_{3329,X^{256}+1}$ + binomial noise + compression + NTT, turned into a CCA-secure KEM by the Fujisaki–Okamoto transform.

Next: Part IV (Section 5 of the paper) — digital signatures from Σ-protocols and Dilithium.

Part IV: Digital Signatures from Σ-Protocols

The road map is Schnorr's. In a Schnorr signature [Sch89], the public key is g, h in a finite group and the signature is a non-interactive zero-knowledge proof of knowledge (ZKPoK) of x with $g^x = h$. It is built in two steps: (1) a 3-move interactive Σ-protocol that is an honest-verifier ZKPoK of x; (2) the Fiat–Shamir transform, which makes it non-interactive. Section 5 of the paper follows exactly this road map with $\mathcal{R}_{q,f}$-LWE and $\mathcal{R}_{q,f}$-SIS, and ends with Dilithium (ML-DSA).

25. The Statements and Witnesses (Section 5.1)

Start from an $\mathcal{R}_{q,f}$-LWE$_{n,m,\beta}$ instance: $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n\times m}$, $\mathbf{s}_1 \leftarrow [\beta]^m$, $\mathbf{s}_2 \leftarrow [\beta]^n$, and $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. The public key is $(\mathbf{A}, \mathbf{t})$; the secret key is $(\mathbf{s}_1, \mathbf{s}_2)$. (Here the "error" of LWE is called $\mathbf{s}_2$ and is part of the secret.)

The difficulty. Unlike discrete log, we must prove two things: the algebraic relation $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t}$ *and* that the coefficients of $\mathbf{s}_i$ are small (ideally in $[\beta]$, but $[\bar{\beta}]$ for some $\bar{\beta}$ a little larger is fine). (Spelled out: without the size condition the statement is trivial: any $\mathbf{s}_1$ works with $\mathbf{s}_2 = \mathbf{t} - \mathbf{A}\mathbf{s}_1$.)

The trick: prove a relaxed statement. The most efficient signatures from Σ-protocols do not prove knowledge of small $\mathbf{s}_1, \mathbf{s}_2$ with (the paper's Eq. (73))

$$\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t},$$

but of $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$ with coefficients in a somewhat larger interval, together with a small ring element $\bar{c}$, such that (Eq. (74))

$$\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\,\mathbf{t}.$$

This still means something:

Lemma 9 (paper): the relaxed statement is still hard

Suppose an algorithm, given $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n\times m}$ and $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ for $\mathbf{s}_1 \leftarrow [\beta]^m$, $\mathbf{s}_2 \leftarrow [\beta]^n$, outputs $\bar{\mathbf{s}}_1 \in [\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 \in [\bar{\beta}]^n$ and $\bar{c} \in [2]$ with $\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\,\mathbf{t}$. Then there is another algorithm, with the same running time and success probability, that solves either $\mathcal{R}_{q,f}$-LWE$_{n,m,\beta}$ or $\mathcal{R}_{q,f}$-SIS$_{n,m+1,\bar{\beta}}$.
(Implicit, as in the paper: the output must be non-trivial, $\bar{c} \neq 0$—otherwise all zeros would do. In use $\bar{c} \in \bar{\mathcal{C}}$, which is non-zero by (76).)

Proof (spelled out). Given a uniformly random SIS instance $\bar{\mathbf{A}} = [\mathbf{A} \mid \mathbf{t}] \in \mathcal{R}_{q,f}^{n\times(m+1)}$, hand $(\mathbf{A}, \mathbf{t})$ to the algorithm.

- By $\mathcal{R}_{q,f}$-LWE$_{n,m,\beta}$, a uniform $\mathbf{t}$ is indistinguishable from $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$, which is what the algorithm expects. If it behaves differently on uniform $\mathbf{t}$, that difference is itself an LWE distinguisher.
- If it succeeds, rearrange: $\mathbf{A}\bar{\mathbf{s}}_1 - \bar{c}\,\mathbf{t} + \bar{\mathbf{s}}_2 = \mathbf{0}$, i.e. $[\mathbf{A} \mid \mathbf{t}] \begin{pmatrix} \bar{\mathbf{s}}_1 \\ -\bar{c} \end{pmatrix} + \bar{\mathbf{s}}_2 = \mathbf{0}$. The vector $(\bar{\mathbf{s}}_1, -\bar{c})$ has coefficients in $[\bar{\beta}]$ (as $\bar{c} \in [2]$, assuming $\bar{\beta} \geq 2$), and it is non-zero when $\bar{c} \neq 0$. That is an $\mathcal{R}_{q,f}$-SIS$_{n,m+1,\bar{\beta}}$ solution for $\bar{\mathbf{A}}$. □

Why "$\bar{c}\,\mathbf{t}$" is harmless in discrete log too

In the Schnorr extraction, one gets $g^{\bar{z}} = h^{\bar{c}}$ with $\bar{c} = c - c' \neq 0$, i.e. knowledge of $\bar{z}$ with $g^{\bar{z}} = h^{\bar{c}}$—also a "relaxed" statement. In a prime-order group it costs nothing, because $\bar{c}$ is invertible: $x = \bar{z}/\bar{c}$. In lattices, dividing by $\bar{c}$ would destroy smallness, so we keep the relaxed form (74) and use Lemma 9 instead.

25.1 The Challenge Space (Section 5.1.1)

The coefficients of $\bar{c}$ (and of $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$) depend on the challenge space, so we want challenges with small norms. Over $\mathcal{R}_f$ with $\deg f = d$, let η be the smallest integer with $2^\eta \binom{d}{\eta} > 2^{256}$ (assuming d is large enough), and define (Eqs. (75)–(76))

$$\mathcal{C} = \{c \in [1] : \|c\|_1 = \eta\}, \qquad \bar{\mathcal{C}} = \{\bar{c} = c_1 - c_2 : c_1 \neq c_2 \in \mathcal{C}\}.$$

So $\mathcal{C}$ is the set of polynomials with exactly η non-zero coefficients, each ± 1, and $|\mathcal{C}| = 2^\eta \binom{d}{\eta}$: choose the η positions, then a sign for each. (Allowing fewer non-zero coefficients would complicate sampling without increasing the size by much.) Footnote 22 of the paper: to sample, start from a length-d vector with η entries equal to 1 (the rest 0), shuffle it (e.g. Fisher–Yates), then randomly negate each 1.

(Spelled out: for $d = 256$ the rule gives $\eta = 60$, since $2^{60}\binom{256}{60} \approx 2^{257}$. Note $\bar{c} \in [2]$, which is why Lemma 9 has $\bar{c} \in [2]$, and $\|\bar{c}\|_1 \leq 2\eta$.)

26. The Basic Σ-Protocol (Section 5.2)

The paper's Figure 5: the basic zero-knowledge proof (from [Lyu09

)] Private: $\mathbf{s}_1 \in [\beta]^m$, $\mathbf{s}_2 \in [\beta]^n$. Public: $\mathbf{A} \in \mathcal{R}_{q,f}^{n \times m}$, $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_{q,f}^n$.

Prover		Verifier
$\mathbf{y}_1 \leftarrow [\gamma + \bar{\beta}]^m$, $\mathbf{y}_2 \leftarrow [\gamma + \bar{\beta}]^n$		
$\mathbf{w} := \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2$	$\xrightarrow{\mathbf{w}}$	
	$\xleftarrow{c}$	$c \leftarrow \mathcal{C}$
$\mathbf{z}_1 := c\mathbf{s}_1 + \mathbf{y}_1$, $\mathbf{z}_2 := c\mathbf{s}_2 + \mathbf{y}_2$		
if $\mathbf{z}_1 \notin [\bar{\beta}]^m$ or $\mathbf{z}_2 \notin [\bar{\beta}]^n$ then $(\mathbf{z}_1, \mathbf{z}_2) := \perp$	$\xrightarrow{(\mathbf{z}_1, \mathbf{z}_2)}$	
		accept iff $\mathbf{z}_1 \in [\bar{\beta}]^m$, $\mathbf{z}_2 \in [\bar{\beta}]^n$ and $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} = \mathbf{w}$

It is a ZKPoK of $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 \in [2\bar{\beta}]^n$ and $\bar{c} \in \bar{\mathcal{C}}$ satisfying (74). γ is defined in Lemma 10; $\bar{\beta}$ controls how often $\perp$ is sent.

Completeness (spelled out). $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} = \mathbf{A}(c\mathbf{s}_1 + \mathbf{y}_1) + c\mathbf{s}_2 + \mathbf{y}_2 - c(\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2) = \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2 = \mathbf{w}$. So whenever the prover does not abort, the verifier accepts.

What is unusual: rejection sampling. The protocol does *not* have perfect completeness. To keep the output small *and* independent of the secret, the prover checks that all coefficients of $\mathbf{z}_i$ lie in a certain range and otherwise sends $\perp$ and restarts. More complicated rejection steps (sampling and rejecting according to a discrete Gaussian) give slightly smaller outputs [Lyu12, DDLL13], but they are harder to implement: a slightly incorrect implementation may leak the secret key, and they may be harder to protect against side channels. For a widely used primitive, the simpler range check may be preferable. The price of rejection sampling is that signing time becomes a random variable—but one independent of $\mathbf{s}_1, \mathbf{s}_2$.

The Schnorr correspondence. Mask $(\mathbf{y}_1, \mathbf{y}_2)$, commit $(\mathbf{w})$, challenge (c), respond with "mask + challenge × secret" $(\mathbf{z}_i = c\mathbf{s}_i + \mathbf{y}_i)$—then the extra rejection step. Fiat–Shamir will replace c by a hash of the message and the first message.

Hashing the first message. A common trick is to send a hash of the first message instead of the message itself; the verifier recomputes it from the later messages. With rejection sampling this has an extra benefit: it lets one simulate transcripts in which a rejection occurs. It is unnecessary for the signature's security, though, because the signer's aborted attempts are never seen. So the paper presents the protocol without hashing and proves zero-knowledge only for non-aborting transcripts.

26.1 Honest-Verifier Zero-Knowledge (Section 5.2.1)

Lemma 10 (paper)

Let $\gamma \in \mathbb{Z}^+$ be such that $cs \in [\gamma]$ for all polynomials $s \in [\beta]$ and $c \in \mathcal{C}$ (footnote 23: when $\|c\|_1 = \eta$ and $s \in [\beta]$, simply $\gamma = \eta\beta$). Then for all $\mathbf{s}_i, c$ as in Figure 5:

$$\Pr_{\mathbf{y}_1,\mathbf{y}_2}\left[(\mathbf{z}_1, \mathbf{z}_2) \neq \perp\right] = \left(\frac{2\bar{\beta}+1}{2(\bar{\beta}+\gamma)+1}\right)^{d(m+n)}, \tag{77}$$

$$\forall\, \mathbf{z}_1' \in [\bar{\beta}]^m, \mathbf{z}_2' \in [\bar{\beta}]^n : \Pr_{\mathbf{y}_1,\mathbf{y}_2}\left[(\mathbf{z}_1, \mathbf{z}_2) = (\mathbf{z}_1', \mathbf{z}_2') \,\middle|\, (\mathbf{z}_1, \mathbf{z}_2) \neq \perp\right] = \left(\frac{1}{2\bar{\beta}+1}\right)^{d(m+n)}. \tag{78}$$

In words: the abort probability does not depend on the secret, and an accepted response is *uniform* on $[\bar{\beta}]^{d(m+n)}$. (The paper's (78) quantifies over $\mathbf{z}_1' \in [\beta]^m$, $\mathbf{z}_2' \in [\beta]^n$; it must be $[\bar{\beta}]$, as in (79) and the proof.)

Proof, coefficient by coefficient. View everything as integer vectors of length $d(m+n)$ (§16): $\mathbf{z} = \mathcal{V}_{c\mathbf{s}} + \mathcal{V}_{\mathbf{y}}$. Fix a target coefficient $\nu_z \in [\bar{\beta}]$. If the corresponding coefficient of $c\mathbf{s}$ is $\nu_s \in [\gamma]$, the mask coefficient must be exactly $\nu_y = \nu_z - \nu_s$. Since $\nu_z \in [\bar{\beta}]$ and $\nu_s \in [\gamma]$, we have $\nu_z - \nu_s \in [\bar{\beta}+\gamma]$—exactly the range ν_y is drawn from. So this happens with probability exactly $\frac{1}{2(\bar{\beta}+\gamma)+1}$, *whatever* ν_s is. Hence (Eq. (79))

$$\forall\, \mathbf{z}' \in [\bar{\beta}]^{d(m+n)} : \quad \Pr_{\mathbf{y}}[\mathbf{z} = \mathbf{z}'] = \left(\frac{1}{2(\bar{\beta}+\gamma)+1}\right)^{d(m+n)}.$$

Summing over the $(2\bar{\beta}+1)^{d(m+n)}$ acceptable values gives (77), and dividing (79) by (77) gives (78). □

A typo in the paper's proof of Lemma 10

The first sentence on page 44 says the i-th coefficient of $\mathbf{z}$ equals a particular ν_z with probability "exactly $\frac{1}{2\bar{\beta}+1}$". That is the *conditional* probability given no abort, i.e. (78). The unconditional probability, which is what the argument shows and what (79) states, is $\frac{1}{2(\bar{\beta}+\gamma)+1}$.

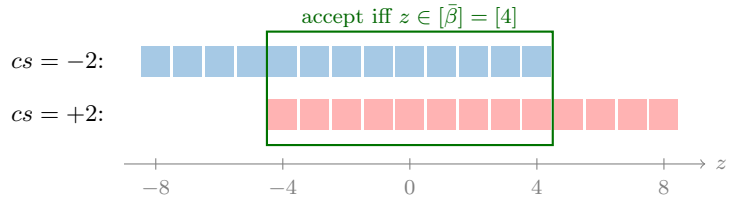


Figure 8: Illustration (ours) of rejection sampling with $\gamma = 2$, $\bar{\beta} = 4$, one coefficient. The mask y is uniform on $[\gamma + \bar{\beta}] = [6]$ (13 values), so $z = cs + y$ is uniform on a window *shifted by the secret*—blue for $cs = -2$, red for $cs = +2$. Unconditioned, z leaks cs (e.g. $z = -8$ is only possible for $cs = -2$). Keeping only $z \in [4]$ leaves exactly the 9 values $-4, \ldots, 4$ for *every* $cs \in [2]$, each with probability $1/13$: acceptance probability $9/13 = \frac{2\bar{\beta}+1}{2(\bar{\beta}+\gamma)+1}$, and the accepted z is uniform on $[4]$ regardless of the secret.

How often does the prover abort? (Eq. (80))

$$\left(\frac{2\bar{\beta}+1}{2(\bar{\beta}+\gamma)+1}\right)^{d(m+n)} > \left(\frac{\bar{\beta}}{\bar{\beta}+\gamma}\right)^{d(m+n)} = \left(1+\frac{\gamma}{\bar{\beta}}\right)^{-d(m+n)} \approx e^{-\gamma d(m+n)/\bar{\beta}}.$$

(Spelled out: the inequality is $(2\bar{\beta}+1)(\bar{\beta}+\gamma) > \bar{\beta}(2\bar{\beta}+2\gamma+1)$, which after expanding reduces to $\gamma > 0$; the approximation is $(1+x)^{-N} \approx e^{-xN}$ for small x.) So $\bar{\beta} = \gamma d(m+n)$ gives success probability about $1/e$, i.e. an expected $e \approx 2.7$ repetitions. A smaller $\bar{\beta}$ is possible at the cost of more repetitions.

The simulator. Choose $\mathbf{z}_1 \leftarrow [\bar{\beta}]^m$, $\mathbf{z}_2 \leftarrow [\bar{\beta}]^n$, $c \leftarrow \mathcal{C}$ uniformly, set $\mathbf{w} := \mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t}$, and output $(\mathbf{w}, c, \mathbf{z}_1, \mathbf{z}_2)$. This is *exactly* the distribution of a non-aborting real transcript: c is uniform in both; by Lemma 10 the accepted $(\mathbf{z}_1, \mathbf{z}_2)$ is uniform for any c; and $\mathbf{w}$ is determined by the other values. So the protocol is HVZK when $\perp$ is not sent.

No timing leak

The probability of sending $\perp$—and hence the number of restarts—is independent of the secret (Lemma 10). A running time that depended on the secret would open side-channel attacks; this protocol is immune to that particular attack.

26.2 Proof of Knowledge (Section 5.2.2)

The usual rewinding argument (the paper's Figure 6): run the prover until it sends $\mathbf{w}$; give it challenge c and get $(\mathbf{z}_1, \mathbf{z}_2)$; rewind to just after $\mathbf{w}$ and give it $c' \neq c$ to get $(\mathbf{z}_1', \mathbf{z}_2')$. If both transcripts verify, $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} = \mathbf{A}\mathbf{z}_1' + \mathbf{z}_2' - c'\mathbf{t}$ (both equal $\mathbf{w}$), which simplifies to (Eq. (81))

$$\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_1') + (\mathbf{z}_2 - \mathbf{z}_2') = (c - c')\mathbf{t}.$$

This is (74) with $\bar{\mathbf{s}}_1 = \mathbf{z}_1 - \mathbf{z}_1' \in [2\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 = \mathbf{z}_2 - \mathbf{z}_2' \in [2\bar{\beta}]^n$, $\bar{c} = c - c' \in [2]$ (the differences of two elements of $[\bar{\beta}]$ lie in $[2\bar{\beta}]$).

26.3 Putting It All Together (Sections 5.2.3–5.2.4)

HVZK says an adversary learns nothing from non-aborted transcripts; PoK says an adversary who can impersonate the prover can produce a solution to (74), which by Lemma 9 solves Ring-LWE or Ring-SIS. Together: assuming both are hard, no one can impersonate the prover even after watching valid interactions. The Fiat–Shamir transform then gives a signature scheme secure in the random oracle model based on Ring-SIS and Ring-LWE.

Setting the parameters. Extraction gives $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$ with coefficients in $[2\bar{\beta}]$ and $\|\bar{c}\|_1 \leq 2\eta$. By Lemma 9, if $\mathcal{R}_{q,f}$-LWE$_{n,m,\beta}$ is hard, this means solving $\mathcal{R}_{q,f}$-SIS with $m+1$ columns. The optimal setting makes the two problems equally hard—on the same vertical line in the paper's Figure 2. $\bar{\beta}$ is dictated by Lemma 10: by (80), set $\bar{\beta} \approx \gamma d(m+n)$ with $\gamma = \eta\beta$ when $\|c\|_1 \leq \eta$. So $\bar{\beta}$ is roughly a factor $\eta d(m+n)$ larger than β.

(Note: the paper writes the SIS problem here as $\mathcal{R}_{q,f}$-SIS$_{n,m+1,\bar{\beta}}$, but since the extracted coefficients lie in $[2\bar{\beta}]$, applying Lemma 9 gives bound $2\bar{\beta}$—as the paper itself writes in Section 5.7, $\mathcal{R}_{q,f}$-SIS$_{n,m+1,2\bar{\beta}}$.)

27. Analogies to Discrete-Log Schemes: Schnorr, Okamoto, Katz–Wang (Section 5.3)

The paper notes this section is not needed for the sequel (footnote 24). Its point: by changing only β (and the derived $\bar{\beta}$), the lattice protocol mimics three different discrete-log schemes with different security properties. The Schnorr-like variant, with β unconstrained, is the most efficient.

The common blueprint. In all of them there is a homomorphic one-way function family $\mathcal{F}$; the public key is $f, f(x)$ for a random $f \in \mathcal{F}$ and random secret x. The prover sends $w = f(y)$ for a random mask y and answers challenge c with $z = y + xc$. Different properties come from changing the relation between the size of the domain and the range of f.

27.1 Schnorr

Schnorr identification

Public key g, $h = g^x$. Prover: random y, sends $w = g^y$. Verifier: random c. Prover: $z = y + xc$. Verifier checks $g^z = h^c \cdot w$.

(The paper writes the check as $g^z = t^c \cdot w$, using t for h.)

Security. Given a discrete-log challenge (g, h), use it as the public key. Without x, one can still simulate honest transcripts by picking random z, c and setting $w = g^z/h^c$. If the adversary can impersonate, rewinding gives (w, c, z) and (w, c', z') with $g^z = h^c w$ and $g^{z'} = h^{c'} w$, so $g^{\bar{z}} = h^{\bar{c}}$ with $\bar{z} = z - z'$, $\bar{c} = c - c'$, and the discrete log is $\bar{z}/\bar{c}$.

The lattice analogue (Figure 5): public key $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$; extract $\bar{\mathbf{s}}_i, \bar{c}$ satisfying (74); Lemma 9 turns that into Ring-SIS for $[\mathbf{A} \mid \mathbf{t}]$. One difference: in Schnorr, the public key (g, g^x) *is* random; in lattices we needed Ring-LWE to argue that the public key looks random.

	Schnorr	Lattice (Figure 5)
One-way function	$x \mapsto g^x$	$(\mathbf{s}_1, \mathbf{s}_2) \mapsto \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
Commitment	$w = g^y$	$\mathbf{w} = \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2$
Response	$z = y + xc$	$\mathbf{z}_i = \mathbf{y}_i + c\mathbf{s}_i$, **plus rejection**
Check	$g^z = h^c w$	$\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} = \mathbf{w}$, **plus size check**
Extracted	$x = \bar{z}/\bar{c}$	$(\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{c})$ with (74)
Assumption	discrete log	Ring-LWE (key looks random) + Ring-SIS

(Spelled out: why does Schnorr need no rejection? Because y is uniform in $\mathbb{Z}_p$, the response $y + xc$ is *exactly* uniform, whatever x is. Lattices need z small, so y cannot be uniform over everything, and rejection sampling restores "uniform whatever the secret".)

27.2 Okamoto

The Okamoto protocol [Oka92] can be proven secure against *active* adversaries (who choose challenges maliciously). Since Fiat–Shamir only needs HVZK, this stronger property is not really needed for signatures in practice. (Footnote 25: there is no reduction from discrete log or DDH to the actively secure Schnorr scheme, though it can be proven secure under certain "knowledge assumptions" [BP02].)

Okamoto identification

Public key: random (g_1, g_2) and $h = g_1^{x_1} g_2^{x_2}$. Prover sends $w = g_1^{y_1} g_2^{y_2}$; on challenge c replies $z_i = y_i + cx_i$ $(i \in \{1, 2\})$. Verifier checks $g_1^{z_1} g_2^{z_2} = h^c \cdot w$.

Security without simulation. Given a discrete-log instance (g_1, g_2) with $g_2 = g_1^x$ for unknown x, the reduction picks its own valid secret key (x_1, x_2), sets $h = g_1^{x_1} g_2^{x_2}$, and *honestly* runs the protocol for the adversary. If the adversary then impersonates, rewinding gives (Eqs. (82)–(83))

$$h^{\bar{c}} = g_1^{\bar{z}_1} g_2^{\bar{z}_2} \quad \Longrightarrow \quad 1 = g_1^{\bar{z}_1 - x_1 \bar{c}} g_2^{\bar{z}_2 - x_2 \bar{c}}.$$

As long as the exponents $\bar{z}_i - x_i \bar{c}$ are not both 0, this gives x with $g_1^x = g_2$. Why not both 0? Many secret keys fit h: any (x_1', x_2') with $x_1' + x \cdot x_2' = x_1 + x \cdot x_2$. Each is equally likely to be the one in use, and the transcripts are identically distributed for all of them (footnote 26: the z_i are uniform because the y_i are, and w is determined by the z_i and c), even with adversarial c. If the impersonator produced $\bar{z}_i = x_i \bar{c}$, it would know $x_i = \bar{z}_i / \bar{c}$—but the key in use is information-theoretically hidden, so even an all-powerful impersonator cannot hit it with non-negligible probability. (In Schnorr this fails: $h = g^x$ has only one secret key.)

The lattice analogue: choose β so large that the public key $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$ does *not*, with high probability, determine the secret. If $(2\beta + 1)^{n+m} > q^n \cdot 2^{128/d}$, any (even all-powerful) algorithm recovers the exact $(\mathbf{s}_1, \mathbf{s}_2)$ with probability only 2^{-128}. (Footnote 27: the optimal guesser outputs the most likely preimage for each $\mathbf{t}$; there are q^{nd} values of $\mathbf{t}$, so it outputs at most q^{nd} of the $(2\beta + 1)^{(n+m)d}$ equally likely preimages, a fraction $q^{nd}/(2\beta + 1)^{(n+m)d} < 2^{-128}$.) Lemma 10 shows the transcripts leak nothing about the $\mathbf{s}_i$. But, as with Okamoto vs. Schnorr, the requirement on β makes it less efficient.

27.3 Katz–Wang

The Katz–Wang protocol [KW03, Section 3] is based entirely on DDH and its proof needs *no rewinding*, which makes the reduction tighter (footnote 28: rewinding loses a factor of the number of random-oracle queries). This matters more in the quantum random oracle model (QROM), where quantum adversaries cannot simply be rewound. But the paper notes that tightness does not seem to affect the actual security of the signature schemes, so Okamoto and Katz–Wang, and their lattice versions, remain mostly of theoretical interest.

Katz–Wang identification

Public key: random (g_1, g_2) and $(h_1 = g_1^x, h_2 = g_2^x)$. Prover: random y, sends $(w_1 = g_1^y, w_2 = g_2^y)$; on challenge c replies $z = y + cx$. Verifier checks $g_i^z = h_i^c \cdot w_i$ for both i.

Reduction from DDH. Given (g_1, g_2, h_1, h_2) (either a DDH tuple or random; footnote 29 relates this to the usual (g, g^a, g^b, g^{ab}) form), publish it as the public key and simulate transcripts as for Schnorr. If the tuple is random, write $h_i = g_i^{x_i}$ with $x_1 \neq x_2$ and $w_i = g_i^{r_i}$.

A valid z must satisfy $z = x_1 c + r_1$ and $z = x_2 c + r_2$, so

$$\Pr_c\big[\exists z:\ g_1^z = h_1^c w_1 \wedge g_2^z = h_2^c w_2\big] = \Pr_c\big[c = (r_2 - r_1)/(x_1 - x_2)\big] = 1/|\mathcal{C}|.$$

So on a random tuple even an unbounded adversary almost never succeeds; whether the adversary succeeds therefore decides DDH.

The lattice analogue: make $\bar{\beta}$ small enough that, for a uniformly random public key $(\mathbf{A}, \mathbf{t})$, valid responses $\mathbf{z}_1, \mathbf{z}_2$ information-theoretically do not exist. Then the adversary's success distinguishes random $(\mathbf{A}, \mathbf{t})$ from $(\mathbf{A}, \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$—exactly Ring-LWE. We want: for all $\mathbf{w}$, if the key is random, at most one challenge c admits valid $\mathbf{z}_i$ (coefficients in $[\bar{\beta}]$, $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 = \mathbf{t}c + \mathbf{w}$). If two did, subtracting gives (Eq. (84))

$$\mathbf{A}\bar{\mathbf{z}}_1 + \bar{\mathbf{z}}_2 = \mathbf{t}\bar{c}, \qquad \bar{\mathbf{z}}_i = \mathbf{z}_i - \mathbf{z}_i',\ \bar{c} = c - c',$$

and an argument like Lemma 2 shows such $\bar{\mathbf{z}}_i$ with coefficients in $[2\bar{\beta}]$ do not exist for a random $(\mathbf{A}, \mathbf{t})$ with high probability. (Footnote 30: over $\mathcal{R}_{q,f}$ this needs the polynomials in $[2\bar{\beta}]$ to be invertible, which can be ensured by choosing the ring's parameters, cf. [LS18, Corollary 1.2].) For this, $\bar{\beta}$ must be somewhat less than $q^{n/(n+m)}$ (as in Lemma 2), so β must be even smaller, with the relation fixed by Lemma 10. Again, less efficient than the Schnorr analogue.

27.4 Comparing Efficiency and Security

Recall Figure 2 of the paper: LWE gets harder and SIS easier as β grows, crossing near $q^{n/(n+m)}$. The signature needs LWE hard at β and SIS hard at $\bar{\beta}$ (with $\bar{\beta} \gg \beta$, by Lemma 10). The optimal Schnorr-like setting puts β and $\bar{\beta}$ on *opposite* sides of the crossing. Okamoto needs $\beta > q^{n/(n+m)}$ (so the key does not determine the secret), pushing both to the right; Katz–Wang needs $\bar{\beta} < q^{n/(n+m)}$, pushing both to the left. Either way one of the two problems gets easier, so n and m must grow to compensate.

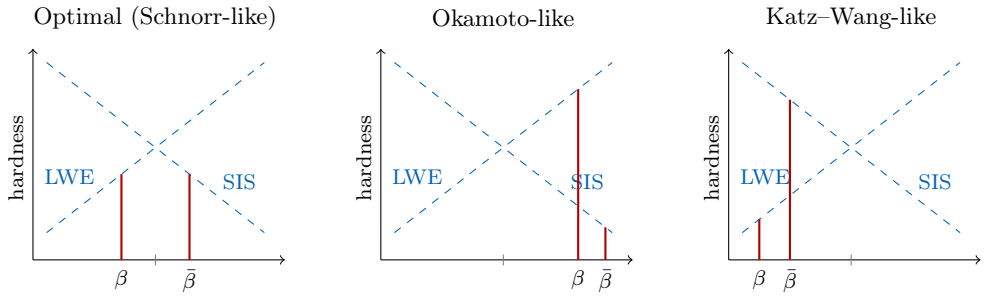


Figure 9: After the paper's Figure 7 (redrawn): LWE hardness at β and SIS hardness at $\bar{\beta}$ (red bars) for the three variants. The security level is the *lower* of the two bars (our reading). The Schnorr-like choice straddles the crossing and gets the highest minimum; the Okamoto- and Katz–Wang-like constraints put both parameters on one side, so one problem becomes easier.

28. Reducing the Proof Size (Section 5.4)

Goal: stop the prover from sending $\mathbf{z}_2$.

Two easy tricks that clash.

- In Figure 5 the verifier can *recompute* $\mathbf{z}_2 = \mathbf{w} - \mathbf{A}\mathbf{z}_1 + c\mathbf{t}$, so $\mathbf{z}_2$ need not be sent.

- Alternatively, the prover can send the short hash $\rho = \mathcal{H}(\mathbf{w})$ instead of $\mathbf{w}$, and the verifier checks $\mathcal{H}(\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t}) = \rho$. Not sending $\mathbf{w}$ essentially halves the signature size, and the paper notes that the recompute-$\mathbf{z}_2$ trick is not compatible with using the protocol efficiently or with Fiat–Shamir (Section 5.6).

But with the hash, $\mathbf{z}_2$ can no longer be recomputed. So it seems the prover must send either $\mathbf{w}$ or $\mathbf{z}_2$.

The insight: make $\mathbf{w}$ not need $\mathbf{z}_2$. In verification, $\mathbf{z}_2$ is small, so with good probability it does not affect the *high-order bits* of $\mathbf{w}$. Likewise $\mathbf{y}_2$ does not affect the high-order bits of $\mathbf{A}\mathbf{y}_1 + \mathbf{y}_2$. So define $\mathbf{w}$ as the high-order bits of $\mathbf{A}\mathbf{y}_1$, and let the verifier check that the high-order bits of $\mathbf{A}\mathbf{z}_1 - c\mathbf{t}$ equal $\mathbf{w}$. Then $\mathcal{H}(\mathbf{A}\mathbf{z}_1 - c\mathbf{t}) = \mathcal{H}(\mathbf{w})$ holds too, and neither $\mathbf{z}_2$ nor $\mathbf{w}$ needs to be sent. The prover still needs $\mathbf{s}_2$, though, to keep the protocol zero-knowledge. This is in the spirit of the bit-dropping of Section 2.5.1; the idea of not sending $\mathbf{z}_2$ comes from [GLP12, BG14], and Figure 8 is due to [BG14].

Notation (from Section 2.5.1). $\mathcal{S} \subset \mathbb{Z}_q$ has size 2^κ with neighbouring points $\approx q/2^\kappa$ apart (Eq. (18)); $w = \mathrm{HIGH}_\mathcal{S}(w) + \mathrm{LOW}_\mathcal{S}(w)$ with $\mathrm{LOW}_\mathcal{S}(w) \in [q/2^{\kappa+1}]$, applied coefficient-wise to vectors over $\mathcal{R}_{q,f}$. Let $\delta_\mathcal{S}$ be the largest integer such that the sets $s_i + [\delta_\mathcal{S}]$, $s_i \in \mathcal{S}$, are disjoint. For equidistant points, $\delta_\mathcal{S} \approx q/2^{\kappa+1}$, which is also (within 1) the largest possible $|\mathrm{LOW}_\mathcal{S}(w)|$. The key observation, for all positive $\gamma < \delta_\mathcal{S}$ and $s \in [\gamma]$ (Eq. (85)):

$$\mathrm{LOW}_\mathcal{S}(w) \in [\delta_\mathcal{S} - \gamma] \implies \mathrm{HIGH}_\mathcal{S}(w) = \mathrm{HIGH}_\mathcal{S}(w + s).$$

(Spelled out: w sits within $\delta_\mathcal{S} - \gamma$ of its nearest point s_i; moving it by at most γ keeps it within $\delta_\mathcal{S}$ of s_i, i.e. inside $s_i + [\delta_\mathcal{S}]$, which contains no other point's neighbourhood—so s_i is still the nearest.)

The paper's Figure 8: zero-knowledge proof with a smaller output

Private: $\mathbf{s}_1 \in [\beta]^m$, $\mathbf{s}_2 \in [\beta]^n$. Public: $\mathbf{A} \in \mathcal{R}_{q,f}^{n \times m}$, $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$.

Prover		Verifier
$\mathbf{y} \leftarrow [\gamma + \bar{\beta}]^m$, $\mathbf{w} := \mathrm{HIGH}_\mathcal{S}(\mathbf{A}\mathbf{y})$	$\xrightarrow{\mathbf{w}}$	
	$\xleftarrow{c}$	$c \leftarrow \mathcal{C}$
$\mathbf{z} := c\mathbf{s}_1 + \mathbf{y}$ if $\mathbf{z} \notin [\bar{\beta}]^m$ or $\mathrm{LOW}_\mathcal{S}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2) \notin [\delta_\mathcal{S} - \gamma]^n$ then $\mathbf{z} := \perp$	$\xrightarrow{\mathbf{z}}$	accept iff $\mathbf{z} \in [\bar{\beta}]^m$ and $\mathrm{HIGH}_\mathcal{S}(\mathbf{A}\mathbf{z} - c\mathbf{t}) = \mathbf{w}$

A ZKPoK of $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 \in [q/2^\kappa]^n$ and $\bar{c} \in \bar{\mathcal{C}}$ satisfying (74).

28.1 Correctness (Section 5.4.1)

If $\mathbf{z} \neq \perp$ we need $\mathrm{HIGH}_\mathcal{S}(\mathbf{A}\mathbf{y}) = \mathrm{HIGH}_\mathcal{S}(\mathbf{A}\mathbf{z} - c\mathbf{t})$. Now (Eq. (86))

$$\mathbf{A}\mathbf{z} - c\mathbf{t} = \mathbf{A}(c\mathbf{s}_1 + \mathbf{y}) - c(\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2) = \mathbf{A}\mathbf{y} - c\mathbf{s}_2.$$

The prover only continues if $\mathrm{LOW}_\mathcal{S}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2) \in [\delta_\mathcal{S} - \gamma]^n$. By (85), applied to $w = \mathbf{A}\mathbf{y} - c\mathbf{s}_2$ and $s = c\mathbf{s}_2 \in [\gamma]$, $\mathrm{HIGH}_\mathcal{S}(\mathbf{A}\mathbf{y}) = \mathrm{HIGH}_\mathcal{S}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2)$, which is $\mathrm{HIGH}_\mathcal{S}(\mathbf{A}\mathbf{z} - c\mathbf{t})$.

28.2 Zero-Knowledge (Section 5.4.2)

Again we simulate non-aborting transcripts. By Lemma 10, conditioned on $\mathbf{z} \in [\bar{\beta}]^m$, $\mathbf{z}$ is uniform. The simulator picks $\mathbf{z} \leftarrow [\bar{\beta}]^m$ and $c \leftarrow \mathcal{C}$, checks $\mathrm{LOW}_\mathcal{S}(\mathbf{A}\mathbf{z} - c\mathbf{t}) \in [\delta_\mathcal{S} - \gamma]^n$ (retrying if not), then sets $\mathbf{w} := \mathrm{HIGH}_\mathcal{S}(\mathbf{A}\mathbf{z} - c\mathbf{t})$ and outputs $(\mathbf{w}, c, \mathbf{z})$. $\mathbf{z}$ has the right

distribution after the first check, and the second check is *exactly the same* in the real protocol and the simulation (by (86), $\mathbf{Ay} - c\mathbf{s}_2 = \mathbf{Az} - c\mathbf{t}$, computable from public data).

Why the odd-looking check (87) is needed

Completeness alone would be satisfied by a different check: the prover could directly test whether the verifier will accept, i.e. $\mathbf{w} = \mathrm{HIGH}_{\mathcal{S}}(\mathbf{Az} - c\mathbf{t})$. But the simulator cannot perform *that* test: it does not know $\mathbf{w}$ until after it has fixed $\mathbf{z}$ and passed the check. The check (Eq. (87))

$$\mathrm{LOW}_{\mathcal{S}}(\mathbf{Ay} - c\mathbf{s}_2) \in [\delta_{\mathcal{S}} - \gamma]^n$$

is chosen because it guarantees correctness *and* the simulator can perform it. Replacing it by the direct acceptance check would lose the zero-knowledge property.

28.3 The Probability of $\perp$ (Section 5.4.3)

As in Lemma 10, $\Pr_{\mathbf{y}}[\mathbf{z} \in [\bar{\beta}]^m] = \left(\frac{2\bar{\beta}+1}{2(\beta+\gamma)+1}\right)^{dm} \approx e^{-\gamma dm/\bar{\beta}}$. For the second check, the paper makes the *heuristic* assumption that $\mathbf{Ay} - c\mathbf{s}_2$ is uniform in $\mathcal{R}_{q,f}^n$, so $\mathrm{LOW}_{\mathcal{S}}$ of it is uniform in $[\delta_{\mathcal{S}}]^n$, and it lands in $[\delta_{\mathcal{S}} - \gamma]^n$ with probability (Eq. (88))

$$\left(\frac{2(\delta_{\mathcal{S}} - \gamma) + 1}{2\delta_{\mathcal{S}} + 1}\right)^{dn} > \left(1 - \frac{\gamma}{\delta_{\mathcal{S}}}\right)^{dn} \approx e^{-\gamma dn/\delta_{\mathcal{S}}}.$$

Combining (Eq. (89)):

$$\Pr_{\mathbf{y}}[\mathbf{z} \neq \perp] \approx e^{-\gamma d(m/\bar{\beta} + n/\delta_{\mathcal{S}})}.$$

Larger $\bar{\beta}$ and $\delta_{\mathcal{S}}$ mean fewer aborts, but larger extracted $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$ (so an easier SIS problem). The heuristic is only used to *estimate* the abort rate; that the abort probability is independent of the secret needs no heuristic, because $\mathbf{z}$ is independent of the secret (Lemma 10) and $\mathrm{LOW}_{\mathcal{S}}(\mathbf{Ay} - c\mathbf{s}_2) = \mathrm{LOW}_{\mathcal{S}}(\mathbf{Az} - c\mathbf{t})$ is a function of $\mathbf{z}$, c and public data.

28.4 Proof of Knowledge (Section 5.4.4)

Rewinding gives $(\mathbf{w}, c, \mathbf{z})$ and $(\mathbf{w}, c', \mathbf{z}')$ with $\mathrm{HIGH}_{\mathcal{S}}(\mathbf{Az} - c\mathbf{t}) = \mathrm{HIGH}_{\mathcal{S}}(\mathbf{Az}' - c'\mathbf{t})$. Writing each side as HIGH + LOW, with LOW parts in $[q/2^{\kappa+1}]^n \approx [\delta_{\mathcal{S}}]^n$, and subtracting (Eq. (90)):

$$\mathbf{A}(\mathbf{z} - \mathbf{z}') - (c - c')\mathbf{t} = \mathrm{LOW}_{\mathcal{S}}(\mathbf{Az} - c\mathbf{t}) - \mathrm{LOW}_{\mathcal{S}}(\mathbf{Az}' - c'\mathbf{t}) \in [q/2^{\kappa}]^n \approx [2\delta_{\mathcal{S}}]^n,$$

which is (74) with $\bar{\mathbf{s}}_1 = \mathbf{z} - \mathbf{z}'$, $\bar{c} = c - c'$ and $\bar{\mathbf{s}}_2$ set to the difference of the two LOW parts (up to sign).

29. Reducing the Public Key Size (Section 5.5)

Same idea, applied to $\mathbf{t}$. Write $\mathbf{t} = \mathrm{HIGH}_{\mathcal{T}}(\mathbf{t}) + \mathrm{LOW}_{\mathcal{T}}(\mathbf{t})$ for a set $\mathcal{T} \subset \mathbb{Z}_q$ of 2^{ℓ} elements about $q/2^{\ell}$ apart. Since $\mathbf{Az}_1 \approx c\mathbf{t} = c(\mathrm{HIGH}_{\mathcal{T}}(\mathbf{t}) + \mathrm{LOW}_{\mathcal{T}}(\mathbf{t})) \approx c \cdot \mathrm{HIGH}_{\mathcal{T}}(\mathbf{t})$, the verifier does not need the low bits of $\mathbf{t}$ to verify *approximately*. So publish only $\mathbf{t}_1 = \mathrm{HIGH}_{\mathcal{T}}(\mathbf{t})$ ($nd\ell$ bits instead of $nd \log q$), with $\mathbf{t} = \mathbf{t}_1 + \mathbf{t}_0$, $\mathbf{t}_0 = \mathrm{LOW}_{\mathcal{T}}(\mathbf{t})$. The techniques are from [DKL+18].

The verifier can now only compute $\mathbf{Az} - c\mathbf{t}_1 = \mathbf{Az} - c\mathbf{t} + c\mathbf{t}_0$. For verification we would need (Eq. (91))

$$\mathrm{HIGH}_{\mathcal{S}}(\mathbf{Az} - c\mathbf{t}) = \mathrm{HIGH}_{\mathcal{S}}(\mathbf{Az} - c\mathbf{t} + c\mathbf{t}_0).$$

Why not reuse the previous trick? We could force $c\mathbf{t}_0$ to be tiny and use (85). But that is wasteful and unlikely to save much. We bounded $c\mathbf{s}_2$ by γ because $\mathbf{s}_2$ had to stay *secret*. $\mathbf{t}_0$ need not be secret—we only want the verifier not to need it. So it is fine for the verifier to compute something that depends on $\mathbf{t}_0$, such as $\mathbf{Az} - c\mathbf{t} + c\mathbf{t}_0$, as long as it can then derive $\mathrm{HIGH}_{\mathcal{S}}(\mathbf{Az} - c\mathbf{t})$.

The hint. If $c\mathbf{t}_0 \in [\delta_{\mathcal{S}}]^n$, a verifier who knows $\mathrm{HIGH}_{\mathcal{S}}(\mathbf{Az} - c\mathbf{t} + c\mathbf{t}_0)$ needs only *one extra bit per coefficient*. If an integer point v lies between s_i and s_{i+1} in $\mathcal{S}$ and we add $v' \in [\delta_{\mathcal{S}}]$, the closest point to $v + v'$ can only be s_i or s_{i+1}. The prover, who knows v and v', tells the verifier which. Cost: dn extra "hint" bits in the signature, in exchange for a much smaller public key.

HINT and USEHINT

$\mathrm{HINT}(\mathbf{Az} - c\mathbf{t}_1, c\mathbf{t}_0) = \mathbf{h} \in \{0,1\}^{dn}$ is the vector of hints needed to recover $\mathrm{HIGH}_{\mathcal{S}}(\mathbf{Az} - c\mathbf{t})$ from $\mathbf{Az} - c\mathbf{t}_1$. $\mathrm{USEHINT}(\mathbf{Az} - c\mathbf{t}_1, \mathbf{h})$ computes $\mathbf{v} = \mathbf{Az} - c\mathbf{t}_1$ and, for each coefficient (lying between some s_i and s_{i+1}), outputs the closer point if the hint bit is 0 and the farther one if it is 1. Usually the hint bit is 0, so $\mathbf{h}$ is stored as the list of positions where it is 1. For every $\mathbf{v}$ and $\mathbf{h}$ (Eq. (92)):

$$\mathbf{v} - \mathrm{USEHINT}(\mathbf{v}, \mathbf{h}) \in [q/2^{\kappa}]^n \approx [2\delta_{\mathcal{S}}]^n,$$

because USEHINT always outputs one of the two points adjacent to $\mathbf{v}$, which are $q/2^{\kappa}$ apart.

The paper's Figure 9: smaller proof *and* smaller public key

Public: $\mathbf{A}$, $\mathbf{t}_1 = \mathrm{HIGH}_{\mathcal{T}}(\mathbf{t})$ ($\mathbf{t} = \mathbf{As}_1 + \mathbf{s}_2$ is not used by the verifier).

Prover		Verifier
$\mathbf{y} \leftarrow [\gamma + \bar{\beta}]^m$, $\mathbf{w} := \mathrm{HIGH}_{\mathcal{S}}(\mathbf{Ay})$	$\xrightarrow{\mathbf{w}}$	
	$\xleftarrow{c}$	$c \leftarrow \mathcal{C}$
$\mathbf{z} := c\mathbf{s}_1 + \mathbf{y}$ if $\mathbf{z} \notin [\bar{\beta}]^m$ or $\mathrm{LOW}_{\mathcal{S}}(\mathbf{Ay} - c\mathbf{s}_2) \notin [\delta_{\mathcal{S}} - \gamma]^n$: $(\mathbf{z}, \mathbf{h}) := \perp$ if $c\mathbf{t}_0 \notin [\delta_{\mathcal{S}}]^n$: $(\mathbf{z}, \mathbf{h}) := \perp$		
if $(\mathbf{z}, \mathbf{h}) \neq \perp$: $\mathbf{h} := \mathrm{HINT}(\mathbf{Az} - c\mathbf{t}_1, c\mathbf{t}_0)$	$\xrightarrow{(\mathbf{z},\mathbf{h})}$	accept iff $\mathbf{z} \in [\bar{\beta}]^m$ and $\mathrm{USEHINT}(\mathbf{Az} - c\mathbf{t}_1, \mathbf{h}) = \mathbf{w}$

We require that, with high probability over c and $\mathbf{t}$, $c \cdot \mathrm{LOW}_{\mathcal{T}}(\mathbf{t}) \in [\delta_{\mathcal{S}}]^n$. It is a ZKPoK of $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 \in [q/2^{\kappa-1}]^n$ and $\bar{c} \in \bar{\mathcal{C}}$ with (Eq. (93))

$$\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\,\mathbf{t}_1.$$

To relate (93) to the original $\mathbf{t}$ and (74): substitute $\mathbf{t}_1 = \mathbf{t} - \mathbf{t}_0$ to get $\mathbf{A}\bar{\mathbf{s}}_1 + (\bar{\mathbf{s}}_2 + \bar{c}\mathbf{t}_0) = \bar{c}\mathbf{t}$. So the bound on $\bar{\mathbf{s}}_2$ grows by the largest possible $\bar{c}\mathbf{t}_0$ over $\bar{c} \in \bar{\mathcal{C}}$.

29.1 Correctness and Zero-Knowledge (Section 5.5.1)

Although the verifier does not need $\mathbf{t}_0$, it *should not* be considered secret: the hint $\mathbf{h}$ leaks information about it. Think of it as: the verifier knows all of $\mathbf{t}$ but only uses $\mathbf{t}_1$. Zero-knowledge then follows from that of Figure 8 (Section 5.4.2): the only new output, $\mathbf{h}$, can be computed from $\mathbf{z}$ and $\mathbf{t}$. Correctness holds whenever $c\mathbf{t}_0 \in [\delta_{\mathcal{S}}]^n$. If $\mathbf{t}_0$ is such that this sometimes fails, security is unaffected; the prover just restarts more often.

(Footnote 31 of the paper: a 1-bit hint in fact works when $c\mathbf{t}_0 \in [2\delta_{\mathcal{S}} - 1]$, since $v + [2\delta_{\mathcal{S}} - 1]$

contains at most two points of $\mathcal{S}$. Observed independently by Jonathan Katz and in [BDL24], this lets $\mathbf{t}_0$ be about twice as large, compressing the public key further, at a small cost in signature size because the hint vector becomes uniformly random rather than sparse. It came after ML-DSA was standardized, so it is not in the standard.)

29.2 Proof of Knowledge (Section 5.5.2)

Rewinding gives $(\mathbf{w}, c, (\mathbf{z}, \mathbf{h}))$ and $(\mathbf{w}, c', (\mathbf{z}', \mathbf{h}'))$ with (Eq. (94)) $\mathrm{USEHINT}(\mathbf{Az} - c\mathbf{t}_1, \mathbf{h}) = \mathrm{USEHINT}(\mathbf{Az}' - c'\mathbf{t}_1, \mathbf{h}')$. By (92), $\mathbf{Az} - c\mathbf{t}_1$ and $\mathbf{Az}' - c'\mathbf{t}_1$ are each within $[q/2^\kappa]^n$ of this common value, so (Eq. (95))

$$\mathbf{A}(\mathbf{z} - \mathbf{z}') - (c - c')\mathbf{t}_1 \in [q/2^{\kappa-1}]^n \approx [4\delta_{\mathcal{S}}]^n,$$

which is knowledge of $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{c}$ as in (93).

30. Digital Signatures (Section 5.6)

The paper's Figure 10: the Fiat–Shamir signature scheme

Private: $\mathbf{s}_1 \leftarrow [\beta]^m$, $\mathbf{s}_2 \leftarrow [\beta]^n$. Public: $\mathbf{A}$, $\mathbf{t}_1 = \mathrm{HIGH}_{\mathcal{T}}(\mathbf{t})$.
Sign(μ):

1. $\mathbf{y} \leftarrow [\gamma + \bar{\beta}]^m$;
2. $c := \mathcal{H}(\mathrm{HIGH}_{\mathcal{S}}(\mathbf{Ay}), \mu, \mathbf{A}, \mathbf{t}) \in \mathcal{C}$;
3. $\mathbf{z} := c\mathbf{s}_1 + \mathbf{y}$;
4. if $\mathbf{z} \notin [\bar{\beta}]^m$ or $\mathrm{LOW}_{\mathcal{S}}(\mathbf{Ay} - c\mathbf{s}_2) \notin [\delta_{\mathcal{S}} - \gamma]^n$, RESTART;
5. if $c\mathbf{t}_0 \notin [\delta_{\mathcal{S}}]^n$, RESTART;
6. $\mathbf{h} := \mathrm{HINT}(\mathbf{Az} - c\mathbf{t}_1, c\mathbf{t}_0)$; output $(\mathbf{z}, \mathbf{h})$.

Verify$(\mu, (\mathbf{z}, \mathbf{h}))$: accept iff $\mathbf{z} \in [\bar{\beta}]^m$ and $\mathcal{H}(\mathrm{USEHINT}(\mathbf{Az} - c\mathbf{t}_1, \mathbf{h}), \mu, \mathbf{A}, \mathbf{t}) = c$.
As good practice, the public key is an input to $\mathcal{H}$. This prevents some malleability attacks, where a signature for one public key lets one create a signature for a closely related public key.

(The paper's Figure 10 lists $(\mathbf{z}, \mathbf{h})$ as the output, while the verifier's check uses c; the signature therefore also contains c—as the size count in Section 5.7 confirms: "the signature consists of the vector $\mathbf{z}_1$, the challenge c, and the hint vector $\mathbf{h}$". Similarly, both signer and verifier hash "$\mathbf{A}, \mathbf{t}$", although the verifier is only given $\mathbf{t}_1$ (the figure itself says $\mathbf{t}$ is not used by the verifier); read this input as "the public key".)

What changes with Fiat–Shamir. The challenge is now a hash of the (high bits of the) commitment, the message and the public key. Because the protocol is no longer interactive, the signer never has to send $\perp$: it simply restarts until rejection sampling succeeds. That is why it was enough to prove zero-knowledge only for non-aborting transcripts. Correctness and simulatability follow from Figure 9 plus the random-oracle heuristic, and the generic properties of Fiat–Shamir let us extract from a successful forger the same $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{c}$ satisfying (93) as in Section 5.5.2. By Lemma 9, that is as hard as Ring-LWE or Ring-SIS.

Schnorr signatures, restated

Schnorr signature: $c = \mathcal{H}(g^y, \mu)$, $z = y + xc$, signature (c, z), check $c = \mathcal{H}(g^z h^{-c}, \mu)$. Dilithium's shape is identical: $c = \mathcal{H}(\text{HIGH}(\mathbf{Ay}), \mu, pk)$, $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$, check $c = \mathcal{H}(\text{HIGH}(\mathbf{Az} - c\mathbf{t}), \ldots)$. The three lattice-only additions: the rejection step (keeps $\mathbf{z}$ independent of $\mathbf{s}_1$), taking high bits (so $\mathbf{s}_2$ never needs to be sent), and the hint (so the low bits of $\mathbf{t}$ need not be published).

31. CRYSTALS-Dilithium (ML-DSA) (Section 5.7)

The paper instantiates Figure 10 with parameters very close to [DKL+18] at NIST Level 3, (conservatively) estimated at 192 bits of security.

q	$2^{23} - 2^{13} + 1$
$f(X)$	$X^{256} + 1$
β	4
(n, m)	$(6, 5)$
$\mathcal{C}$	$c \in [1]$ with 49 coefficients ± 1 and 207 zeros
γ	$49 \cdot 4 = 196$
$\bar{\beta}$	$2^{19} - \gamma - 1$
$\mathcal{S}$	$\{i \cdot (q-1)/16 \mid 0 \le i \le 15\}$
$\delta_{\mathcal{S}}$	$(q-1)/32 - 1$
$\mathcal{T}$	$\{i \cdot 2^{13} \mid 0 \le i \le (q-1)/2^{13}\}$

Table 4: The paper's Table 4: sample parameters very similar to the NIST Level 3 (AES-192-equivalent) CRYSTALS-Dilithium set [DKL+18]. (The paper's table labels the $\bar{\beta}$ row as β; the value $2^{19} - \gamma - 1$ is $\bar{\beta}$.)

Reading the parameters.

- $q \equiv 1 \pmod{512}$, which allows efficient NTT (Section 4.6). (Spelled out: $q - 1 = 2^{13} \cdot 1023$, so by Lemma 7 $X^{256} + 1$ splits into linear factors and the NTT runs all the way down.)
- Challenges have 49 non-zeros, so $\gamma = \eta\beta = 49 \cdot 4$ (footnote 23). (Spelled out: this gives $|\mathcal{C}| = 2^{49}\binom{256}{49} \approx 2^{225}$, fewer than the 2^{256} of Section 5.1.1's generic definition, which would need $\eta = 60$; for a scheme targeting 192-bit security, 2^{225} challenges are still plenty.)
- $\mathcal{S}$ has 16 points, $(q-1)/16$ apart, so every point of $\mathbb{Z}_q$ is at most $(q-1)/32+1$ from $\mathcal{S}$.
- $\mathcal{T}$ has points 2^{13} apart, so every coefficient of $\mathbf{t}_0$ lies in $[2^{12}]$. Hence $c\mathbf{t}_0 \in [\delta_{\mathcal{S}}]^n$ with high probability. In fact always: footnote 32 notes $\|\mathbf{t}_0\|_\infty \le 2^{12}$ and $\|c\|_1 = 49$, so $\|c\mathbf{t}_0\|_\infty \le 49 \cdot 2^{12} = 200{,}704 < \delta_{\mathcal{S}} = 261{,}887$.

Signing time. From (89), the probability that a signing attempt does not restart is about

$$e^{-196 \cdot 256 \cdot (5/2^{19} + 6 \cdot 32/(q-1))} \approx 0.2,$$

so on average about 5 attempts are needed. Signing is noticeably slower than verification, but optimized implementations still sign in well under a millisecond on a standard PC. (Our check: the exponent is ≈ 1.63, giving 0.196, i.e. ≈ 5.1 attempts; the exact product form of (77) and (88) gives the same 0.196.)

Sizes.

- **Public key:** a 256-bit seed ρ for $\mathbf{A}$ plus $\mathbf{t}_1 = \mathrm{HIGH}_{\mathcal{T}}(\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$. Each of the $6 \cdot 256$ coefficients of $\mathbf{t}_1$ is one of 1024 points of $\mathcal{T}$, i.e. 10 bits: $32 + 6 \cdot 256 \cdot 10/8 = 1952$ bytes. (The paper says "the set $\mathcal{S}$ can be described with 10 bits"; it means $\mathcal{T}$, which has $(q-1)/2^{13} + 1 = 1024$ elements, whereas $\mathcal{S}$ has 16.)
- **Signature:** $\mathbf{z}_1$ has coefficients in $[\bar{\beta}]$ (the paper writes $[\beta]$), so 20 bits each: $256 \cdot 5 \cdot 20$ bits $= 3200$ bytes. The challenge c is about 256 bits (32 bytes); the hint $\mathbf{h}$ has $256 \cdot 6$ bits (192 bytes). Total: about 3424 bytes.

31.1 Some Optimizations in ML-DSA

- **Hash the message once.** If μ' is long, recomputing $\mathcal{H}(\mathrm{HIGH}_{\mathcal{S}}(\mathbf{A}\mathbf{y}), \mu, \mathbf{A}, \mathbf{t})$ after every restart is wasteful. So first hash μ' with the public key using SHA-512 into a 512-bit digest μ, and sign the digest.
- **Power-of-two masking range.** $\mathbf{y}$ is resampled on every restart. $[\gamma + \bar{\beta}] = [2^{19} - 1]$ has $2^{20} - 1$ values; sampling instead from $\{-(2^{19} - 1), \ldots, 2^{19} - 1, 2^{19}\}$ gives exactly 2^{20} values, i.e. exactly 20 random bits.
- **Compact hint.** With high probability $\mathbf{h}$ has at most 55 ones. Instead of the $256 \cdot 6 = 1536$-bit string, send the positions of the ones (8 bits each: $8 \cdot 55$ bits) plus the boundaries between the 6 polynomials ($5 \cdot 6 = 30$ bits): 470 bits in total. (The paper says "positions of the 0's" and "naive $256 \cdot 8 = 1536$"; the positions are those of the non-zero entries, and $1536 = 256 \cdot 6$.) The signature drops to about 3290 bytes. The signer must restart if $\mathbf{h}$ has more than 55 ones, but experimentally this is very rare.

(Our check of the sizes: 1952 bytes; $3200 + 32 + 192 = 3424$ bytes; and $3424 - 192 + 470/8 \approx 3290.75$ bytes.)

31.2 Security

By Section 5.5, Lemma 9 and (93), Dilithium's security rests on $\mathcal{R}_{q,f}$-LWE$_{n,m,\beta}$ and $\mathcal{R}_{q,f}$-SIS$_{n,m+1,2\bar{\beta}}$:

- LWE: the public key $(\mathbf{A}, \mathbf{t})$ is indistinguishable from uniform;
- SIS: a forger would yield a solution to (93) with $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]$ and $\bar{\mathbf{s}}_2 \in [4\delta_{\mathcal{S}}]$ (see (95)), both roughly $[2^{20}]$ here.

This is the middle row of the paper's Table 2 (reproduced in §14): LWE with dimension $dm = 256 \cdot 5 = 1280$ and $\beta = 4$; SIS with dimension $dn = 256 \cdot 6 = 1536$ and $\beta = 2^{20}$. The required δ is about the same for both (1.003 and 1.0032)—the balance that §26 said is optimal.

32. Summary of Part IV (Section 5 of the Paper)

What to remember from Section 5

1. **Same road map as Schnorr:** Σ-protocol (mask, commit, challenge, respond $\mathbf{z} = \mathbf{y} + c\mathbf{s}$) plus Fiat–Shamir.
2. **Prove a relaxed statement** $\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\,\mathbf{t}$; by Lemma 9 producing it still means solving Ring-LWE or Ring-SIS.
3. **Rejection sampling** (Lemma 10) makes accepted responses uniform on $[\bar{\beta}]$ *whatever the secret*, with a secret-independent acceptance probability $\approx e^{-\gamma d(m+n)/\bar{\beta}}$.
4. **High bits and hints** remove $\mathbf{z}_2$ from the signature (Section 5.4) and the low bits of $\mathbf{t}$ from the public key (Section 5.5); the check (87) gives correctness while remaining simulatable, which keeps the scheme zero-knowledge.
5. **Dilithium (ML-DSA):** $q = 2^{23} - 2^{13} + 1$, $(n, m) = (6, 5)$ over $X^{256} + 1$; about 5 signing attempts; 1952-byte public key; about 3290-byte signatures; security balanced between $\mathcal{R}$-LWE and $\mathcal{R}$-SIS.

This completes the guide to Sections 2–5 of Lyubashevsky's tutorial.