

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з виконання лабораторної роботи

**ДОСЛІДЖЕННЯ СУЧАСНИХ
АЛГЕБРАЇЧНИХ КРИПТОСИСТЕМ**

Виконала студентка
групи ФІ-52мн
Балацька Вікторія

Перевірив:
Фесенко А.В

Київ — 2025

ВСТУП

Мета роботи: Дослідження особливостей реалізації сучасних алгебраїчних крипtosистем на прикладі учасників першого раунду національного конкурсу з постквантової криптографії в Кореї (КрqС).

Постановка завдання:

1. Реалізація алгоритму: розробити програмну реалізацію обраного криптографічного алгоритму та всі можливі варіанти цього алгоритму.
2. Перевірка коректності: підтвердити правильність реалізації за допомогою тестів, використовуючи офіційні тестові вектори або офіційну реалізацію.
3. Аналіз продуктивності та порівняння: знайти схожі алгоритми та провести порівняльний аналіз швидкодії за різних умов, дослідити вплив модифікацій окремих складових частин на ефективність.
4. Теоретичне дослідження: надати повний теоретичних опис алгоритму з усіма деталями та відомими результатами досліджень; провести аналіз наявних атак на обраний алгоритм та описати власні дослідження атак; виконати порівняльний аналіз обраного алгоритму зі схожими та дослідити можливість перенесення відомих атак на нього.

Хід роботи: У своїй роботі я досліджуватиму алгоритм підпису SOLMAE. SOLMAE - це схема підпису на основі решіток, що відповідає парадигмі "хеш-і-підпис" і представляється над NTRU решітками.

1 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

У цій частині звіту я проведу теоретичне дослідження алгоритму підпису SOLMAE: формальне визначення та опис алгоритму; аналіз безпеки та відомі результати; порівняльний аналіз з іншими підписами; можливі атаки та їх перенесення; аналіз продуктивності.

1.1 Формальне визначення та опис алгоритму

SOLMAE - це схема підпису на основі решітки, та розшифровується як quantum-Secure algOrithm for Long-term Message Authentication and Encryption (квантово-безпечний алгоритм для довгострокової автентифікації та шифрування повідомлень). Для ефективної реалізації, структура потребувала класу решіток, що мають ефективно обчислювальні бази з трапдорами (trapdoors) для процедури підпису, дослідивши існуючі класи решіток, автори зупинились на NTRU-решітках. У такому випадку підписання зводиться до вибірки коротких гаусових векторів у відкритій NTRU решітці. Сам алгоритм SOLMAE натхнений дизайном Falcon. Проте, порівнюючи з Falcon, тут є певні нові теоретичні основи: на високому рівні усувається властива процедура вибірки технічності і більша частина її індукованої складності з точки зору реалізації, без втрати ефективності. Простота конструкції перетворюється на швидшу роботу, але при цьому зберігаючи розміри підписів і ключів верифікації, а також надаючи додаткові функції такі як дешевше маскування та можливість паралелізації.

1.1.1 Принципи проектування

Алгоритм підпису SOLMAE побудований за парадигмою hash-then-sign на решітках та покликаний покращити ефективність і безпеку

порівняно з попередніми схемами, зокрема Falcon. Його проектування ґрунтуються на трьох ключових ідеях (зображені на рисунку 1.1):

- Гібридний семплер (Hybrid Sampler) — швидший, простіший і паралелізований спосіб генерувати гаусові вектори, що спрощує підписування та зменшує обчислювальні витрати.
- Оптимізований алгоритм генерації ключів — спеціально налаштований для покращення якості трапдорів у NTRU-решітках і підвищення рівня безпеки при збереженні продуктивності.
- Техніки стиснення даних — зменшують розмір підписів і ключів без впливу на безпеку, оптимізуючи використання пропускної здатності.

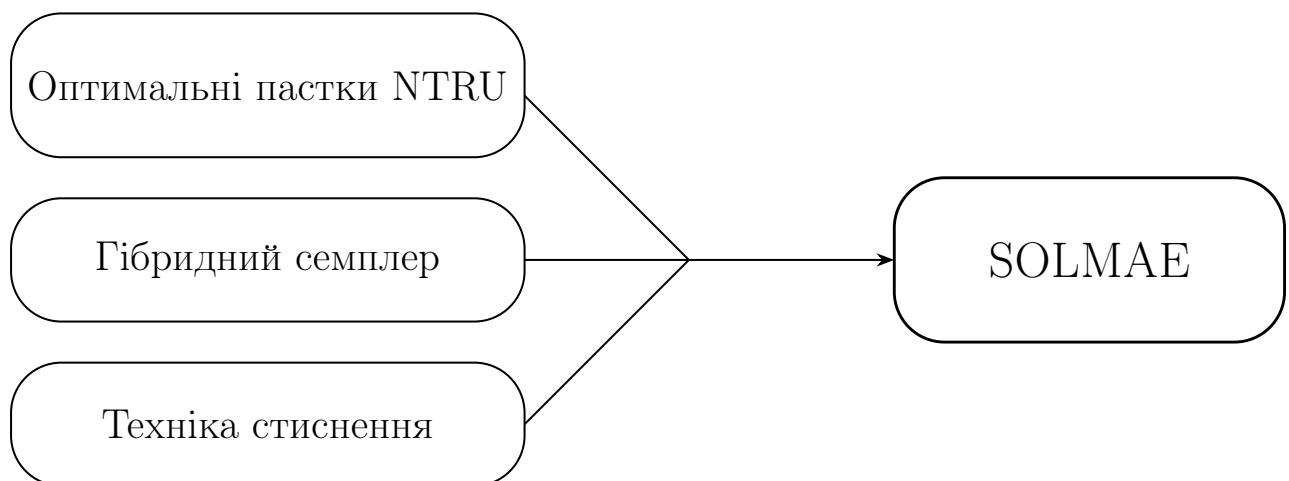


Рисунок 1.1 – Схематичне представлення основних компонентів алгоритму SOLMAE

SOLMAE використовує переваги алгебраїчної структури NTRU-решіток та поєднує сучасні підходи до побудови трапдорів і вибору гаусових вибірок. Завдяки цьому схема досягає високої швидкодії, меншого розміру підписів та ключів, а також зберігає стійкість до відомих атак на базові задачі решіток.

1.1.2 Базові поняття та позначення

Вектори позначаються жирними малими літерами та розглядаються як стовпчикові. Матриці позначаються жирними великими літерами. Коли ми говоримо, що матриця є базисом простору, маємо на увазі, що стовпчики цієї матриці утворюють базис. ℓ_2 -норма вектора $\mathbf{x} = (x_1, \dots, x_d)$ визначається як $\|\mathbf{x}\| = (\sum_i |x_i|^2)^{1/2}$, а його ℓ_∞ -норма — як $\|\mathbf{x}\|_\infty = \max_i |x_i|$.

Решітки. Решітками називають дискретну підгрупу \mathbb{R}^n . Іншими словами, решітка — це множина цілочисельних лінійних комбінацій, отриманих з базису $\mathbf{B} \subset \mathbb{R}^n$. Об'єм решітки дорівнює $\det(\mathbf{B})$ для будь-якого її базису.

Циклотомічні кільця степенів двійки. Для побудови схеми підпису SOLMAE використовується циклотомічне кільце $K = \mathbb{Q}[X]/(X^d + 1)$, де $d = 2^n$ — степінь двійки. Його цілочисельним підкільцем є $R = \mathbb{Z}[X]/(X^d + 1)$, а дійсне розширення позначається як $K_{\mathbb{R}} = \mathbb{R}[X]/(X^d + 1)$.

Поліном $f \in K_{\mathbb{R}}$ може бути представлений кількома способами:

1) Коефіцієнтне подання:

$$f = \sum_{i=0}^{d-1} f_i X^i \longleftrightarrow \mathbf{f} = (f_0, \dots, f_{d-1}).$$

2) Канонічне вкладення (також відоме як дискретне перетворення Фур'є — DFT):

$$\varphi(f) = (\varphi_1(f), \dots, \varphi_d(f)), \quad \varphi_j(f) = f(\zeta_j),$$

де $\zeta_j = e^{i(2j-1)\pi/d}$ — d -ті примітивні корені одиниці. У цьому поданні множення поліномів у кільці переходить у покомпонентне множення в \mathbb{C}^d .

3) Матриця множення:

$$[f] := \begin{pmatrix} f_0 & -f_{d-1} & \dots & -f_1 \\ f_1 & f_0 & \dots & -f_2 \\ \vdots & \vdots & \ddots & \vdots \\ f_{d-1} & f_{d-2} & \dots & f_0 \end{pmatrix}.$$

Для векторів $\mathbf{x} = (x_1, \dots, x_d)$ використовуються стандартні норми:

$$\|\mathbf{x}\|_2 = \left(\sum_i |x_i|^2 \right)^{1/2}, \quad \|\mathbf{x}\|_\infty = \max_i |x_i|.$$

Алгебраїчний метод Грама-Шмідта. Для пар (f,g) та $(F,G) \in K_{\mathbb{R}}^{2 \times 2}$ визначимо скалярний добуток: $\langle (f,g), (F,G) \rangle_K = f^*F + g^*G$, де f^* та g^* — спряжені елементи у $K_{\mathbb{R}}$.

Ортогоналізація Грама — Шмідта для пари (F,G) відносно (f,g) має вигляд:

$$(\tilde{F}, \tilde{G}) = (F, G) - \frac{\langle (f,g), (F,G) \rangle_K}{\langle (f,g), (f,g) \rangle_K} \cdot (f,g).$$

Легко перевірити, що $\langle (f,g), (\tilde{F}, \tilde{G}) \rangle_K = 0$, тобто вектори (f,g) та (\tilde{F}, \tilde{G}) є ортогональними.

Решітка NTRU. Нехай q — ціле число, а $f \in R$ таке, що f є оборотним за модулем q (еквівалентно, $\det[f]$ взаємно простий із q). Позначимо $h = g/f \bmod q$ та розглянемо *NTRU*-модуль, пов'язаний з h : $\mathcal{M}_{\text{NTRU}} = \{(u,v) \in R^2 : hu - v = 0 \bmod q\}$, а також його граткову версію $\mathcal{L}_{\text{NTRU}} = \{(\mathbf{u}, \mathbf{v}) \in \mathbb{Z}^{2d} : [h]\mathbf{u} - \mathbf{v} = 0 \bmod q\}$.

Ця гратка має об'єм q^d . Над R вона породжується парою (f,g) та будь-якими (F,G) , що задовольняють $fG - gF = q$. У такому випадку $\mathcal{L}_{\text{NTRU}}$ має базис вигляду:

$$\mathbf{B}_{f,g} = \begin{bmatrix} [f] & [F] \\ [g] & [G] \end{bmatrix}.$$

Легко перевірити, що $([h], -\text{Id}_d) \cdot \mathbf{B}_{f,g} = 0 \bmod q$, тому відкритим ключем є h .

Задача NTRU-search формулюється так: маючи $h = g/f \bmod q$, знайти будь-яку пару $(f' = x^i f, g' = x^i g)$.

У варіанті decision потрібно розрізнати $h = g/f \bmod q$ від рівномірно випадкового $h \in R_q := \mathbb{Z}[X]/(q, X^d + 1)$. Ці задачі вважаються складними при великому d .

Якість базису NTRU. Секретний базис $\mathbf{B}_{f,g}$ не може бути довільною парою, оскільки він повинен забезпечувати можливість відбору коротких гаусових векторів у гратці $\mathcal{L}_{\text{NTRU}}$ за допомогою гібридного семплінгу. Якість базису $\mathbf{B}_{f,g}$ визначається наступною величиною:

$$\mathcal{Q}(f,g) = \max_{1 \leq i \leq d/2} \max \left(\frac{|\varphi_i(f)|^2 + |\varphi_i(g)|^2}{q}, \frac{q}{|\varphi_i(f)|^2 + |\varphi_i(g)|^2} \right)^{1/2}.$$

Гаусові розподіли. Гаусова функція, центрована в точці $\mathbf{c} \in \mathbb{R}^d$ з додатно визначеною коваріаційною матрицею Σ , визначається як $\rho_{\mathbf{c},\Sigma}(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x} - \mathbf{c})^t \Sigma^{-1} (\mathbf{x} - \mathbf{c}))$.

Нормальний розподіл $\mathcal{N}_{\mathbf{c},\Sigma}$ із центром у \mathbf{c} та коваріацією Σ має щільність, пропорційну $\rho_{\mathbf{c},\Sigma}$.

Коли ми пишемо $\mathbf{x} \leftarrow \mathcal{N}_{\Sigma}^{K_{\mathbb{R}}}$, ми маємо на увазі, що відповідний d -вимірний вектор

$\frac{1}{\sqrt{d}}(\Re \varphi_1(\mathbf{x}), \Im \varphi_1(\mathbf{x}), \dots, \Re \varphi_{d/2}(\mathbf{x}), \Im \varphi_{d/2}(\mathbf{x}))$ має розподіл \mathcal{N}_{Σ} , де $\Re z, \Im z$ — це дійсна та уявна частини комплексного числа z .

Для гратки $\mathcal{L} \subset \mathbb{R}^d$ дискретний гауссовий розподіл із параметрами $\mathbf{c} \in \mathbb{R}^d$ та Σ визначається для всіх $\mathbf{x} \in \mathcal{L}$ як

$$D_{\mathcal{L},\mathbf{c},\Sigma}(\mathbf{x}) = \frac{\rho_{\mathbf{c},\Sigma}(\mathbf{x})}{\rho_{\Sigma}(\mathcal{L} - \mathbf{c})}.$$

Якщо центр $\mathbf{c} = 0$, його часто опускають. Коли $\Sigma = s^2 I$ (скалярна матриця), використовують позначення \mathcal{N}_s або $D_{\mathcal{L},s}$.

1.1.3 Загальний огляд схеми підпису SOLMAE

Як і в будь-якій схемі підпису нам потрібно ввести три алгоритми: генерації ключів (KeyGen), підпису (Sign) та верифікації (Verif). Алгоритм KeyGen відповідає за генерацію ключів — відкритого та особистого. Алгоритм Sign використовується для створення підпису на основі особистого ключа, тоді як Verif забезпечує перевірку правильності підпису за допомогою відкритого ключа.

У межах рамкової моделі GPV (Gentry–Peikert–Vaikuntanathan) до схеми додаються додаткові вимоги:

- визначається клас решіток, у якому алгоритм KeyGen обчислює пару, що складається з базису решітки та відповідного трапдорного елемента (trapdoor pair);
- алгоритм Sign застосовує спеціальну процедуру Sample, яка генерує випадкові вектори в цій решітці. При цьому розподіл вибірки побудований так, щоб не розкривати жодної інформації про трапдор.

Таким чином, схема SOLMAE наслідує основну структуру GPV-підписів, забезпечуючи безпечне створення та перевірку підписів на базі решіткових припущень, зокрема завдяки використанню трапдорних функцій та стохастичного семплінгу, що гарантує приховання секретної інформації.

Конструкція KeyGen

Алгоритм KeyGen у схемі SOLMAE відповідає за побудову трапдорних пар базисів решітки та попередню підготовку даних, необхідних для процедур підписування і семплінгу. Архітектурно KeyGen складається з трьох основних підзадач:

- 1) *PairGen* — генерує початкові поліномні пари (f,g) у кільці R таким чином, щоб отримана якість базису $\mathcal{Q}(f,g) = \alpha$ відповідала цільовому значенню (цей параметр залежить від обраного рівня безпеки; за замовчуванням $\alpha_{512} = 1.17$, $\alpha_{1024} = 1.64$). PairGen працює в

FFT-представленні: випадковим чином вибирає комплексні коефіцієнти у заданому кільцевому annulus і округлює їх до цілих, контролюючи похибку через підібраний радіус. Для цього використовується підпроцедура UnifCrown, яка забезпечує рівномірний розподіл у FFT-форматі.

2) *NtruSolve* — реалізує алгоритм Prest & Pornin. Вона приймає на вхід пару (f,g) і модуль q , а на виході повертає доповнення (F,G) , що задовольняє рівняння типу

$$fG - gF = q \quad \text{в кільці } R.$$

Цей алгоритм будує базис решітки $\mathcal{L}_{\text{NTRU}}$, асоційованої з публічним ключем $h = g/f \bmod q$. Реалізація *NtruSolve* оптимізована за допомогою рекурсивного використання підпільних перетворень і має практичну складність близьку до $O(d \log d)$ для параметра розміру d .

3) *Преобчислення для Sample* — після побудови базису обчислюються додаткові параметри, необхідні для стохастичного семплінгу підписів: матриці коваріацій Σ_1, Σ_2 для проміжних Gaussian-вибірок, а також проекційні вектори β_1, β_2 .

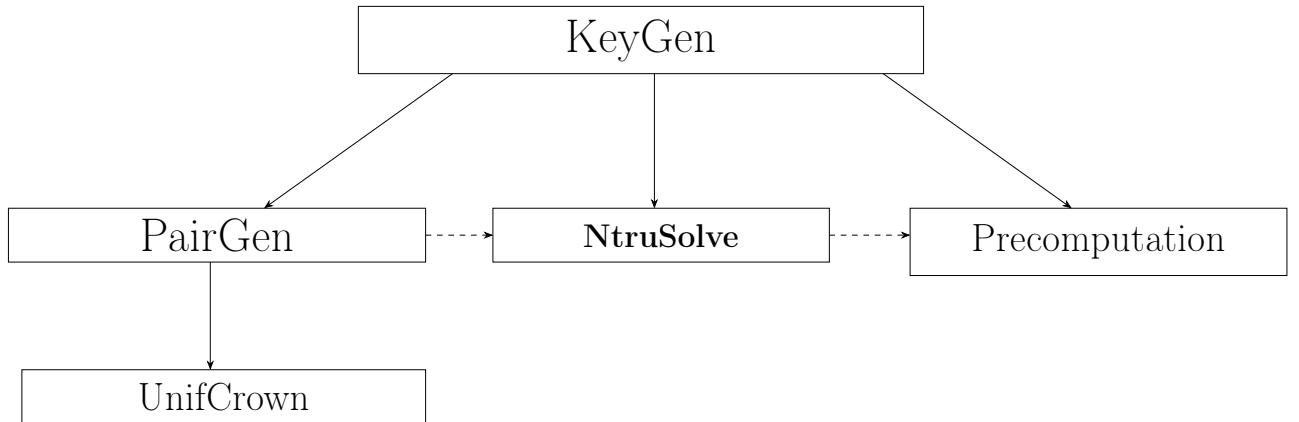


Рисунок 1.2 – Блок-схема KeyGen

Формат вихідних ключів: після виконання всіх підзадач алгоритм KeyGen повертає такі структури:

- Особистий ключ (sk):

$$\text{sk} = (b_1 = (f,g), b_2 = (F,G), \tilde{b}_2 = (\tilde{F}, \tilde{G}), \Sigma_1, \Sigma_2, \beta_1, \beta_2),$$

де \tilde{b}_2 — ортогоналізація (Gram–Schmidt) вектора (F, G) відносно (f, g) ; Σ_i та β_i — попередньо обчислені параметри для семплінгу.

– Відкритий ключ (pk):

$$\text{pk} = (h, q, \sigma_{\text{sig}}, \eta),$$

де $h = g/f \bmod q$, а σ_{sig} і η — стандартні відхилення, які визначають параметри Gaussian-семплінгу та межу *smoothing parameter*.

Якість базису та генерація пар: якість базису визначається через FFT-оцінки коефіцієнтів за формулою:

$$\alpha := \mathcal{Q}(f, g) = \max_{1 \leq i \leq d/2} \max \left(\frac{|\varphi_i(f)|^2 + |\varphi_i(g)|^2}{q}, \frac{q}{|\varphi_i(f)|^2 + |\varphi_i(g)|^2} \right)^{1/2}.$$

Базис має якість α , якщо для всіх $i \leq d/2$ виконується нерівність:

$$\frac{q}{\alpha^2} \leq |\varphi_i(f)|^2 + |\varphi_i(g)|^2 \leq \alpha^2 q.$$

Щоб забезпечити стабільність якості базису, PairGen використовує UnifCrown, яка вибирає коефіцієнти у кільці з радіусами

$$R_- = \left(\frac{1}{\alpha} + \delta \right) \sqrt{q}, \quad R_+ = (\alpha - \delta) \sqrt{q},$$

де δ — мала корекційна константа, підібрана експериментально. Наприклад, $\delta_{512} \approx 0.065$, $\delta_{1024} \approx 0.3$.

Нижче представлено основний алгоритм генерації особистого та відкритого ключів для NTRU-решіткової схеми. Алгоритм ґрунтуються на побудові базису $\mathcal{L}_{\text{NTRU}}$ з контролюваною якістю α , що використовується для забезпечення балансу між безпекою та ефективністю.

Додаткові підпроцедури PairGen та NtruSolve відповідають за генерацію пар поліномів та знаходження розв'язків відповідного рівняння в кільці \mathbb{R}_q .

Algorithm 1.1 KeyGen

1: **Input:** modulus q , target quality $1 < \alpha$, parameters $\sigma_{\text{sig}}, \eta > 0$
 2: **Output:** basis $((f,g),(F,G)) \in R^2$ of $\mathcal{L}_{\text{NTRU}}$ with $\mathcal{Q}(f,g) = \alpha$

▷ Secret basis computation

3: **repeat**

4: $\mathbf{b}_1 \leftarrow (f,g) \leftarrow \text{PAIRGEN}(q,\alpha,R_-,R_+)$

5: **until** f is invertible mod q

6: $\mathbf{b}_2 \leftarrow (F,G) \leftarrow \text{NTRUSOLVE}(q,f,g)$

▷ Public key quantities

7: $h \leftarrow g/f \bmod q$

8: $\gamma \leftarrow 1.1 \cdot \sigma_{\text{sig}} \cdot \sqrt{2d}$

// tolerance for signature length

9: $\beta_1 \leftarrow \frac{1}{\langle \mathbf{b}_1, \mathbf{b}_1 \rangle_K} \cdot \mathbf{b}_1$

10: $\Sigma_1 \leftarrow \sqrt{\frac{\sigma_{\text{sig}}}{\langle \mathbf{b}_1, \mathbf{b}_1 \rangle_K} - \eta^2}$

11: $\tilde{\mathbf{b}}_2 \leftarrow \mathbf{b}_2 - \langle \mathbf{b}_1, \mathbf{b}_2 \rangle \cdot \mathbf{b}_1$

12: $\beta_2 \leftarrow \frac{1}{\langle \tilde{\mathbf{b}}_2, \tilde{\mathbf{b}}_2 \rangle_K} \cdot \tilde{\mathbf{b}}_2$

13: $\Sigma_2 \leftarrow \sqrt{\frac{\sigma_{\text{sig}}}{\langle \tilde{\mathbf{b}}_2, \tilde{\mathbf{b}}_2 \rangle_K} - \eta^2}$

14: $sk \leftarrow (\mathbf{b}_1, \mathbf{b}_2, \tilde{\mathbf{b}}_2, \Sigma_1, \Sigma_2, \beta_1, \beta_2)$

15: $pk \leftarrow (q, h, \sigma_{\text{sig}}, \eta, \gamma)$

16: **return** (sk, pk)

Algorithm 1.2 PairGen

1: **Input:** modulus q , target quality $1 < \alpha$, radii $0 < R_- < R_+$
 2: **Output:** pair (f,g) with $\mathcal{Q}(f,g) = \alpha$

3: **for** $i = 1$ to $d/2$ **do**

4: $(x_i, y_i) \leftarrow \text{UNIFCROWN}(R_-, R_+)$

5: $\theta_x, \theta_y \leftarrow \mathcal{U}(0,1)$

6: $\varphi_{f,i} \leftarrow |x_i| e^{2\pi i \theta_x}, \quad \varphi_{g,i} \leftarrow |y_i| e^{2\pi i \theta_y}$

7: **end for**

8: $(f^{\mathbb{R}}, g^{\mathbb{R}}) \leftarrow \text{FFT}^{-1}((\varphi_{f,i})_{i \leq d/2}), \text{FFT}^{-1}((\varphi_{g,i})_{i \leq d/2})$

9: $(f, g) \leftarrow (|f^{\mathbb{R}}|)_{i \leq d/2}, (|g^{\mathbb{R}}|)_{i \leq d/2}$

10: $(\varphi(f), \varphi(g)) \leftarrow (\text{FFT}(f), \text{FFT}(g))$

11: **for** $i = 1$ to $d/2$ **do**

12: **if** $\frac{\alpha^2}{q} > |\varphi_i(f)|^2 + |\varphi_i(g)|^2$ **or** $\alpha^2 q < |\varphi_i(f)|^2 + |\varphi_i(g)|^2$ **then**

13: **restart**

14: **end if**

15: **end for**

16: **return** (f, g)

▷ see UnifCrown

▷ reject and resample

Конструкція Sign

Нагадаємо, що решітки NTRU живуть у просторі \mathbb{R}^{2d} . Їхня структура спрощує обчислення прообразу. Дійсно, підписувачу достатньо обчислити $\mathbf{m} = \mathbf{H}(M) \in \mathbb{R}^d$, оскільки $\mathbf{c} = (0, \mathbf{m})$ є коректним прообразом, який задовільняє рівняння $(h, 1) \cdot \mathbf{c} = \mathbf{m}$.

Ще одна важлива властивість полягає в тому, що лише перша половина підпису $(\mathbf{s}_1, \mathbf{s}_2) \in \mathcal{L}_{\text{NTRU}}$ передається разом із повідомленням, оскільки перевірник може самостійно відновити другу половину за співвідношенням $h\mathbf{s}_1 = \mathbf{s}_2 \bmod q$. Це значно зменшує обсяг передаваних даних і спрощує перевірку.

Оскільки підписи мають вигляд векторів Гауса, їх можна ефективно стискати. Найстарші біти (MSB) кожної координати кодуються за допомогою унарного або Гаффманівського кодування, як у схемі Falcon. Додаткове стиснення (на 7–12%) можна досягти за допомогою пакетного кодування (наприклад, ANS).

Щоб уникнути детермінізму у виходах, до повідомлення додається випадкова «сіль» $r \in \{0,1\}^k$: хеш обчислюється не від M , а від $(r \| M)$. Отже, підпис має вигляд:

$$\text{sig} = (r, \text{Compress}(\mathbf{s}_1)).$$

Специфікації алгоритму Sign: алгоритм підпису Sign виконує обчислення хешу та перетворення Фур'є, а сам процес вибірки гаусових векторів делегується процедурі Sample. Сіль $r \in \{0,1\}^k$ визначається з консервативного рівня безпеки та максимальної кількості запитів $q_s = 2^{64}$, що дає $k = 320 = 256 + \log q_s$.

Параметр η вибирається як верхня межа для параметра згладжування $\eta_\varepsilon(\mathbb{Z}^d)$ при $\varepsilon = 2^{-41}$:

$$\eta = \frac{1}{\pi} \sqrt{\frac{1}{2} \log \left(2d \left(1 + \frac{1}{\varepsilon} \right) \right)}.$$

Для $d = 512$ отримуємо $\eta_{512} \approx 1,338$, а для $d = 1024 - \eta_{1024} \approx 1,351$. Результатом є гаусів вектор $\mathbf{v} \in \mathcal{L}_{\text{NTRU}}$, центрований у $\mathbf{c} = (0, \mathbf{H}(r \| M))$.

Параметр σ_{sig} визначає очікувану відстань від \mathbf{v} до центра:

$$\sigma_{\text{sig}} = \eta \cdot \mathcal{Q}(f,g) \cdot \sqrt{q}.$$

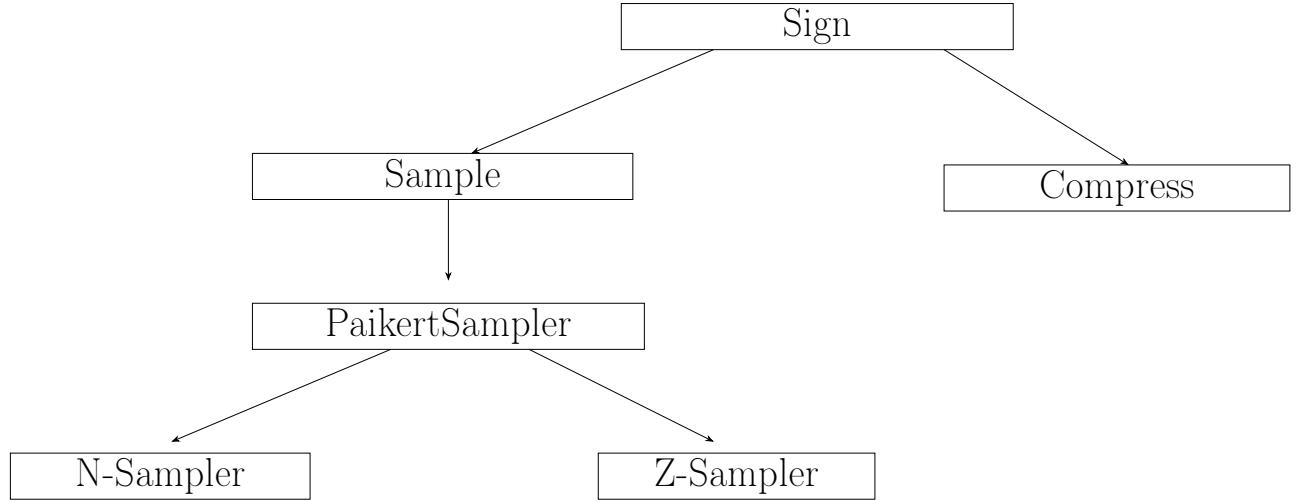


Рисунок 1.3 – Блок-схема Sign

Algorithm 1.3 SIGN

1: **Вхід:** повідомлення $M \in \{0,1\}^*$, секретний ключ $\text{sk} = ((f,g),(F,G),(\tilde{F},\tilde{G}),\sigma_{\text{sig}},\Sigma_1,\Sigma_2,\eta)$, параметр відхилення $\gamma > 0$.

2: **Вихід:** пара $(r, \text{Compress}(\mathbf{s}_1))$, де $r \in \{0,1\}^{320}$ і $\|(\mathbf{s}_1, \mathbf{s}_2)\| \leq \gamma$.

3: $r \leftarrow \mathcal{U}(\{0,1\}^{320})$

4: $\mathbf{c} \leftarrow (0, H(r \| M))$

5: $\hat{\mathbf{c}} \leftarrow \text{FFT}(\mathbf{c})$

6: **repeat**

7: $(\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2) \leftarrow \hat{\mathbf{c}} - \text{SAMPLE}(\hat{\mathbf{c}}, \text{sk})$ \triangleright вибірка з $D_{\mathcal{L}_{\text{NTRU}}, \mathbf{c}, \sigma_{\text{sig}}}$

8: **until** $\|(\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2)\|^2 \leq \gamma^2$

9: $\mathbf{s}_1 \leftarrow \text{FFT}^{-1}(\hat{\mathbf{s}}_1)$

10: $s \leftarrow \text{Compress}(\mathbf{s}_1)$

11: **return** (r, s)

Вибірка векторів підпису в алгоритмі 1.3 реалізується каскадом процедур Sample, PeikertSampler та Z-Sampler, як показано на рисунку 1.3 у

специфікації.

Конструкція Sampler

Нашою другою головною відмінністю від Falcon є використання гібридного семплера (hybrid sampler) для процедури Sample. Щоб зрозуміти цю зміну, розглянемо короткий огляд граткового гаусівського семплінгу, оскільки вибірка підпису фактично є ядром усієї конструкції типу fast-GPV.

Такі алгоритми зазвичай розглядаються як рандомізовані процедури декодування (randomized decoding procedures). Бабай описав два класичних методи граткового декодування:

- Round-off decoding — коли координати цілі просто округлюються до найближчого цілого у базисі гратки;
- Nearest-plane decoding — більш точний підхід, який ітеративно коригує координати за допомогою ортогоналізації Грама–Шмідта.

Після рандомізації округлення з використанням гаусівських цілих (Gaussian integers) отримують або семплер Пайкерта (PeikertSampler), або семплер Кляйна (KleinSampler). Перший є більш ефективним для граток типу NTRU, але має проблему нестачі добрих пасток (trapdoors), що збільшує довжину підписів. Другий, навпаки, забезпечує майже оптимальну довжину підписів, але вимагає складної структури дерева, що робить реалізацію більш обтяжливою.

Гібридний семплер (Hybrid sampler) — це проміжний варіант, який поєднує ефективність і короткість підписів. Він виконує ортогоналізацію Грама–Шмідта та рандомізоване декодування на рівні кільця (at the ring level), завдяки чому алгоритм можна вважати комбінацією підходів Кляйна та Пайкерта.

Для NTRU-граток це означає, що потрібно виконати лише два кроки декодування (замість $2d$, як у FFO-семплері). Рандомізація обробляється за допомогою алгоритму PeikertSampler, який працює у квазі-лінійному часі, що забезпечує кращу загальну продуктивність. Завдяки новому алгоритму KeyGen, тепер можливо ефективно створювати добре пастки, тож усі переваги гібридного семплера реалізуються повною мірою: він простіший у

реалізації, ефективніший, легше паралелізується та не потребує деревоподібних структур.

Специфікація Sample: процедура Sample є реалізацією гібридного семплера і може розглядатися як двокрокове рандомізоване декодування, у якому рандомізація відбувається на рівні кільця або в d -вимірному просторі. Усі операції виконуються у Фур'є-області (Fourier domain).

Algorithm 1.4 Sample

Require: Цільовий вектор $c = (0, c') \in K_R^2$, кортеж секретного ключа $sk = (b_1 = (f, g), b_2 = (F, G), \tilde{b}_2 = (\tilde{F}, \tilde{G}), \sigma_{sig}, \Sigma_1, \Sigma_2, \beta_1, \beta_2)$.

Ensure: Вектор $v \in L_{NTRU}$ з розподілом, статистично близьким до $D_{L_{NTRU}, c, \sigma_{sig}}$.

- 1: $t \leftarrow c, v \leftarrow 0$
- 2: **for** $i = 2$ to 1 **do**
- 3: $t_i \leftarrow \langle \beta_i, t \rangle_K$
- 4: $z_i \leftarrow PeikertSampler(t_i, \Sigma_i, \eta)$
- 5: $t \leftarrow t - z_i b_i, v \leftarrow v + z_i b_i$
- 6: **end for**
- 7: **return** v

Рандомізація виконується через виклик PeikertSampler, який породжує елементи в R з гаусівським розподілом і відповідними коваріаційними матрицями у Фур'є-області. Міжпроміжні стандартні відхилення визначаються як:

$$\Sigma_i = \sqrt{\frac{\sigma_{sig}}{\langle b_i, \tilde{b}_i \rangle_K} - \eta^2}.$$

Ці параметри у Фур'є-області відповідають додатно визначеним матрицям з діагональними елементами, що дозволяє легко обчислювати квадратні корені координатно.

Конструкція Verify

Заключний етап схеми є, на щастя, значно простішим для опису. Після отримання підпису (r, s) і повідомлення M перевірник відновлює s до полінома s_1 , а потім обчислює $c = (0, \mathcal{H}(r\|M))$. Далі перевірник формує

повекторну сигнатуру $\mathbf{v} = (s_1, s_2)$. Якщо \mathbf{v} є дійсним підписом, виконується тотожність перевірки:

$$(h, -1) \cdot (c - \mathbf{v}) = -\mathcal{H}(r \| M) - hs_1 + s_2 \bmod q = 0,$$

що еквівалентно обчисленню:

$$s_2 = \mathcal{H}(r \| M) + hs_1 \bmod q.$$

Це обчислення виконується в кільці R_q і може бути реалізоване дуже ефективно при правильному виборі модуля q , використовуючи Number Theoretic Transform (NTT). Наразі використовується стандартний параметр (як у Falcon): $q = 12289$, оскільки множення у форматі NTT передбачає лише d цілих множень у $\mathbb{Z}/q\mathbb{Z}$.

Останнім кроком є перевірка нерівності:

$$\|(s_1, s_2)\|^2 \leq \gamma^2,$$

де γ — це межа відхилення (rejection bound). Підпис приймається лише в цьому випадку.

Обґрунтування параметра γ : параметр γ походить із математичного очікування довжини векторів, які видає процедура Sample. Оскільки ці вектори є гаусівськими за природою, вони концентруються навколо свого стандартного відхилення. Параметр «запасу» $\tau = 1.042$ налаштований так, щоб приблизно 90% векторів, згенерованих Sample, проходили перевірку. Таким чином:

$$\gamma = \tau \cdot \sigma_{sig} \cdot \sqrt{2d}.$$

Algorithm 1.5 Verif

Require: Підпис (r, s) на повідомленні M , відкритий ключ $\mathbf{pk} = h$, межа γ .

Ensure: Рішення про прийняття або відхилення підпису.

```

1:  $s_1 \leftarrow \text{Decompress}(s)$ 
2:  $c \leftarrow \mathcal{H}(r\|M)$ 
3:  $s_2 \leftarrow c + hs_1 \bmod q$ 
4: if  $\|(s_1, s_2)\|^2 > \gamma^2$  then
5:   return Reject
6: end if
7: return Accept

```

Пояснення:

- Функція $\text{Decompress}(s)$ відновлює поліном s_1 з компресованої форми підпису.
- Обчислення $c = \mathcal{H}(r\|M)$ створює хеш повідомлення та частини підпису.
- Потім обчислюється $s_2 = c + hs_1 \bmod q$, що відновлює другу частину вектору підпису.
- Якщо евклідова норма $\|(s_1, s_2)\|^2$ перевищує γ^2 , підпис відхиляється, інакше приймається.

Таким чином, перевірка є простою, обчислювально ефективною (особливо у форматі NTT), і не потребує складних структур або багаторазових раундів.

Специфікації допоміжних алгоритмів: нижче наведено кілька допоміжних алгоритмів, які використовуються під час виконання процедур KeyGen, Sign та Verif. Вони забезпечують ефективну роботу у Фур'є-області, рівномірне вибіркове породження точок, гаусівське семплінг та стиснення підписів.

Algorithm 1.6 FFT

Require: Поліном $f \in K_{\mathbb{R}} = \mathbb{R}[X]/(X^d + 1)$ **Ensure:** FFT-представлення полінома f

- 1: $\zeta \leftarrow \exp(i\pi/d)$
 - 2: $\varphi(f) \leftarrow (f(\zeta^1), f(\zeta^3), \dots, f(\zeta^{2d-1}))$
 - 3: **return** $\varphi(f)$
-

Пояснення: алгоритм FFT (швидке перетворення Фур'є) переводить поліном f з простору коефіцієнтів у спектральну область $K_{\mathbb{R}}$, що дозволяє ефективно виконувати множення та інші операції у кільці R_q .

Algorithm 1.7 FFT^{-1}

Require: Вектор $c = (c_0, \dots, c_{d-1}) \in \mathbb{C}^d$ та умова $c_{d-1-i} = \overline{c_i}$ **Ensure:** Поліном $f \in K_{\mathbb{R}}$ та $c = \varphi(f)$

- 1: $\zeta \leftarrow \exp(i\pi/d)$
 - 2: $\mathbf{V} \leftarrow (\zeta^{jk})_{j \in \mathbb{Z}_d, k \in \mathbb{Z}_d}$
 - 3: $f \leftarrow \frac{1}{d} \cdot \mathbf{V} \cdot c$
 - 4: **return** f
-

Пояснення: зворотне перетворення Фур'є відновлює початковий поліном з його спектрального подання. Воно використовується після виконання операцій у Фур'є-просторі для повернення до базового кільця.

Algorithm 1.8 UnifCrown

Require: Параметри $0 < R_- < R_+$ **Ensure:** Точка (x, y) з рівномірним розподілом у кільцевій області $A(R_-, R_+)$

- 1: $u_\rho, u_x, u_y \leftarrow \mathcal{U}(0, 1)$
 - 2: $\rho \leftarrow \sqrt{R_-^2 + u_\rho(R_+^2 - R_-^2)}$
 - 3: $x \leftarrow \rho \cdot \cos(2\pi u_x)$
 - 4: $y \leftarrow \rho \cdot \sin(2\pi u_y)$
 - 5: **return** (x, y)
-

Пояснення: UnifCrown генерує рівномірно розподілену випадкову точку в площинному кільці з радіусами R_- та R_+ . Цей метод потрібен для

симетричного задання випадкових шумів у процедурах гаусівського вибіркового породження.

Специфікація: першим кроком у процедурі PeikertSampler є вибірка безперервного гаусівського збурення з еліптичним коваріаційним матричним параметром E . Якщо Σ — це матриця, для якої $\Sigma^t \Sigma = E$, тоді $\mathcal{N}_E = \Sigma \cdot \mathcal{N}_1$. У Фур'є-просторі матриця коваріації є діагональною з додатними елементами:

$$\Sigma = \sqrt{E},$$

де квадратний корінь береться поелементно по діагоналі.

Algorithm 1.9 \mathcal{N} -Sampler

Require: Ступінь d кільця R

Ensure: Дві змінні x, y з нормальним розподілом \mathcal{N}_d

- 1: $u_\rho, u_\theta \leftarrow \mathcal{U}(0, 1)$
 - 2: $\rho \leftarrow \sqrt{-2d \ln u_\rho}$
 - 3: $x \leftarrow \rho \cdot \cos(2\pi u_\theta)$
 - 4: $y \leftarrow \rho \cdot \sin(2\pi u_\theta)$
 - 5: **return** (x, y)
-

Пояснення: цей алгоритм реалізує класичний метод Бокса–Мюллера для породження двох гаусівських випадкових змінних із нульовим середнім та дисперсією, пропорційною d . Використовується у внутрішньому циклі семплера Пайкерта.

Algorithm 1.10 Compress

Require: Поліном $s = \sum_{i=0}^{d-1} s_i X^i \in R = \mathbb{Z}[X]/(X^d + 1)$, ціле число $slen$

Ensure: Стиснене представлення str полінома s довжиною $slen$, або \perp

```

1:  $str \leftarrow \{\}$ 
2: for  $i = 0$  to  $d - 1$  do
3:    $str \leftarrow (str \| b)$ , де  $b = 1$  якщо  $s_i < 0$ , інакше  $b = 0$ 
4:    $str \leftarrow (str \| b_6 b_5 \dots b_0)$ , де  $b_j = (|s_i| \gg j) \& 0x1$ 
5:    $k \leftarrow |s_i| \gg 7$ 
6:    $str \leftarrow (str \| 0^k 1)$ 
7: end for
8: if  $|str| > slen$  then
9:    $str \leftarrow \perp$ 
10: else
11:    $str \leftarrow (str \| 0^{slen - |str|})$ 
12: end if
13: return  $str$ 

```

Пояснення: Алгоритм стискає коефіцієнти полінома s у бітовий рядок обмеженої довжини $slen$. Коефіцієнти зберігаються у двійковому вигляді з позначенням знаку. Якщо довжина перевищує ліміт, повертається \perp (недопустиме значення). Ця функція використовується під час створення коротких підписів у схемах типу Falcon.

Algorithm 1.11 Decompress

Require: Бітовий рядок str довжиною $slen$

Ensure: Поліном $s = \sum_{i=0}^{d-1} s_i X^i \in R = \mathbb{Z}[X]/(X^d + 1)$ або \perp

```

1: if  $|str| \neq slen$  then
2:   return  $\perp$ 
3: end if
4: for  $i = 0$  to  $d - 1$  do
5:    $s'_i \leftarrow \sum_{j=0}^6 2^{6-j} str[1 + j]$ 
6:    $k \leftarrow 0$ 
7:   while  $str[8 + k] = 0$  do
8:      $k \leftarrow k + 1$ 
9:   end while
10:   $s_i \leftarrow (-1)^{str[0]} \cdot (s'_i + 2^7 k)$ 
11:  if  $s_i = 0$  і  $str[0] = 1$  then
12:    return  $\perp$ 
13:  end if
14:   $str \leftarrow str[9 + k :]$ 
15: end for
16: if  $|str| \neq 0$  then
17:   return  $\perp$ 
18: end if
19: return  $s = \sum_{i=0}^{d-1} s_i X^i$ 

```

Пояснення: алгоритм Decompress виконує зворотну операцію до алгоритму Compress, відновлюючи вихідний поліном s із його стислого бітового представлення. На початку перевіряється, чи довжина входного рядка str збігається із зазначеним параметром $slen$. Якщо ні — алгоритм завершується з помилкою, повертаючи \perp .

Далі для кожного коефіцієнта s_i виконується поетапне відновлення: спершу зчитується його абсолютна величина s'_i , яка кодується 7 бітами (2^{6-j}), а потім визначається кількість послідовних нульових бітів k , що додають старші біти до значення коефіцієнта. Після цього встановлюється знак коефіцієнта через перший біт $str[0]$.

Якщо виявлено некоректну комбінацію ($s_i = 0$ та $str[0] = 1$), алгоритм зупиняється і повертає \perp , що свідчить про помилку у структурі стиснених даних. У кінці перевіряється, чи не залишилось невикористаних бітів у рядку str . Успішне завершення повертає відновлений поліном:

$$s = \sum_{i=0}^{d-1} s_i X^i.$$

Таким чином, алгоритм Decompress забезпечує точне відновлення коефіцієнтів полінома, що дозволяє перейти від компактного бітового представлення підпису до його початкової математичної форми.

2 БЕЗПЕКА

2.1 Модель для редукції решіток

У подальшому аналізі автори дотримуються так званого припущення геометричної прогресії (Geometric Series Assumption, GSA), яке стверджує, що для приведеного базису норми векторів Грама–Шмідта зменшуються за геометричним законом. Більш формально, це припущення може бути реалізоване для самодуального алгоритму редукції решіток BKZ (або DBKZ) Міччанчо та Вальтера.

Алгоритм DBKZ із розміром блоку β для решітки \mathcal{L} рангу n повертає базис $(\mathbf{b}_1, \dots, \mathbf{b}_n)$, який задовольняє співвідношення:

$$\|\mathbf{b}_i^*\| = \delta_\beta^{d-2(i-1)} \det(\mathcal{L})^{1/n},$$

де \mathbf{b}_i^* — це i -й вектор Грама–Шмідта базису, а

$$\delta_\beta = \left(\frac{(\pi\beta)^{1/\beta} \cdot \beta}{2\pi e} \right)^{\frac{1}{2(\beta-1)}}.$$

Ця модель використовується для оцінки стійкості решіткових схем до редукційних атак, оскільки вона описує, як довжини векторів у базисі змінюються в процесі редукції, та дозволяє наближено оцінити силу найкоротшого вектора в решітці.

2.2 Атака на відновлення ключа (Key recovery attack)

Атака на відновлення ключа полягає у пошуку приватного секретного ключа (тобто пар поліномів $f, g \in R^2$) з відкритих даних, зокрема з публічних елементів q та h . Найпотужніші відомі атаки в цьому класі реалізуються через редукцію решітки: будують алгебраїчну решітку над кільцем R , породжену векторами $(q, 0)$ та $(h, 1)$ (тобто публічний базис

NTRU-ключа), і намагаються витягнути з цієї решітки вектор секрету $\mathbf{s} = (\mathbf{g}, \mathbf{f})$ серед усіх векторів з нормою, обмеженою $\|\mathbf{s}\| = \sqrt{2d}\sigma$ (або еквівалентний вектор, наприклад $(\mu g, \mu f)$ для будь-якої одиниці μ поля).

Щоб уникнути перебору по всій сфері радіуса $\sqrt{2d}\sigma$ (яка за Гаусовською евристикою містить близько $(2d\sigma^2/q)^d$ векторів), застосовується так званий projection trick. Детальніше процедура виглядає так. Нехай β — розмір блоку алгоритму DBKZ. Спочатку редукують публічний базис цим алгоритмом і позначають отримані вектори як $[\mathbf{b}_1, \dots, \mathbf{b}_{2d}]$. Якщо вдається відновити проекцію секретного вектора на підпростір

$$\mathcal{P} = (\text{Span}(\mathbf{b}_1, \dots, \mathbf{b}_{2d-\beta}))^\perp,$$

то за допомогою алгоритму Babai nearest plane можна за поліноміальний час підняти цю проекцію до вектора потрібної норми в оригінальній решітці. Таким чином, достатньо знайти проекцію секретного вектора серед найкоротших векторів решітки, породженої останніми β векторами редукованого базису, проектованими на \mathcal{P} .

Класично, сіовання на цій проекції відновлює всі вектори нормою менші за $\sqrt{\frac{4}{3}}\ell$, де ℓ — норма $2d - \beta$ -го вектора Грама–Шмідта $\mathbf{b}_{2d-\beta}^*$ редукованого базису. За припущенням GSA маємо оцінку для ℓ :

$$\ell = \sqrt{q} \delta_\beta^{-2d+2\beta+2} \approx \left(\frac{\beta}{2\pi e} \right)^{1-\frac{d}{\beta}},$$

де δ_β — параметр, визначений для DBKZ (див. п. 5.1).

Далі, вважаючи, що \mathbf{s} поводиться як випадковий вектор з нормою $\|\mathbf{s}\| = \sqrt{2d}\sigma$, та застосовуючи GSA для оцінки норм векторів Грама–Шмідта $[\mathbf{b}_1^*, \dots, \mathbf{b}_{2d-\beta}^*]$, норма проекції \mathbf{s} на простір \mathcal{P} приблизно дорівнює

$$\sqrt{\frac{\beta}{2d}} \|\mathbf{s}\| = \sqrt{\frac{\beta}{2d}} \cdot \sqrt{2d}\sigma = \beta^{1/2} \sigma.$$

Отже, ми зможемо відновити проекцію серед просіяних векторів тоді й

тільки тоді, коли

$$\beta^{1/2}\sigma \leq \sqrt{\frac{4}{3}}\ell.$$

Після піднесення в квадрат ця умова дає обмеження для σ^2 :

$$\sigma^2 \leq \frac{4q}{3\beta} \delta_\beta^{4(\beta+1-d)}. \quad (2.1)$$

Формула (2.1) дозволяє порівняти параметри схеми (зокрема σ , q , d та вибір β в DBKZ) із практичними можливостями атак редукції решіток. Вибір параметрів системи має забезпечувати, щоб права частина в (2.1) була настільки малою, що існуючі алгоритми редукції не можуть задовольнити цю нерівність для реалістичних β (тобто атака буде надто дорогою).

2.3 Підробка підпису через зведення до Approx-CVP

Як підпис у парадигмі Hash-and-Sign, підробка підпису зводиться до підведення вектора решітки \mathbf{v} на обмежену відстань від випадкової точки \mathbf{x} . Ця задача Approx-CVP (Approximate Closest Vector Problem) може бути розв'язана у рамках так званого Nearest-Cospace підходу, розвинутого в. Під припущенням GSA теорема стверджує, що якщо виконано умову

$$\|\mathbf{x} - \mathbf{v}\| \leq (\delta_\beta^{2d} q^{1/2}),$$

то декодування може бути виконане за полігональний час (у кількості викликів оракула CVP в вимірності β).

Стандартна оптимізація цієї атаки полягає в роботі не з повною решіткою, а з підрешіткою, породженою підмножиною векторів публічного базису. Із практичної точки зору цікавим підходом є ігнорування перших $k \leq d$ векторів базису: вимірність простору при цьому зменшується на k , але ми мусимо працювати з решіткою з більшою коваріантною (covolume), що призводить до іншого глобального критерію декодування.

Після такого зменшення вимірності загальна умова декодування стає

трохи ускладненою, і зводиться до оцінки

$$\|\mathbf{x} - \mathbf{v}\| \leq \min_{k \leq d} \left(\delta_\beta^{2d-k} q^{\frac{d}{2d-k}} \right).$$

У зв'язку з цим для параметра відсікання (rejection bound) γ необхідно забезпечити:

$$\gamma \geq \min_{k \leq d} \left(\delta_\beta^{2d-k} q^{\frac{d}{2d-k}} \right). \quad (2.2)$$

Інтуїтивно, права частина нерівності (2.2) описує компроміс між зменшенням вимірності (що полегшує декодування) та збільшенням коваріанту підрешітки (що робить задачу складнішою). Для захисту схеми потрібно так підібрати параметри (q , d , σ , а також допустимий β для атак BKZ), щоб права частина у (2.2) залишалася меншою за типове значення γ (параметра відсікання підписів), тобто щоб існуючі алгоритми декодування/редукції не могли знайти відповідний вектор \mathbf{v} для реалістичних β і k .

Формула (2.2) дає практичний критерій для налаштування параметрів: оцінюючи праву частину при різних β (який визначає ефективність DBKZ на практиці) і для різних k , можна визначити, чи існує комбінація параметрів атаки, що задовольняє нерівність. В роботах на цю тему типовим є обчислення мінімального β , при якому рівняння виконується для деякого k — це дозволяє отримати практичну оцінку стійкості проти підробки.

2.3.1 Інші атаки на SOLMAE

У цьому підпункті перелічуємо інші можливі типи атак на підпис, які, однак, для використованого набору параметрів вважаються неістотними.

2.3.2 Алгебраїчні атаки.

Як зауважено при розробці NTRU-подібних схем (наприклад Falcon або ModFalcon), в модулях над коволюшн-кільцем \mathcal{R} присутня багата алгебраїчна структура. Однак наразі немає відомого способу покращити загальні алгоритми (зокрема редукцію решіток та пов'язані методи) більш ніж на поліноміальний множник, використовуючи цю структуру. Тобто можливі алгебраїчні пришвидшення існують локально, але поки не дають асимптотичного прориву, який би став критичним для безпеки SOLMAE.

2.3.3 Overstretched NTRU-подібні атаки.

Коли модуль q значно більший за амплітуди коефіцієнтів секретного ключа NTRU, атаки на ключ на основі редукції решітки можуть давати кращі результати. Такий режим називають “overstretched NTRU”: для бінарних секретів проблема погіршується, коли

$$q > (2d)^{2.83}.$$

Проте для типових параметрів SOLMAE (аналогічних до параметрів Falcon та інших кандидатів NIST) цей ефект не призводить до практичної вразливості — навіть суттєве покращення алгоритмів у цій області навряд чи зробить атаку реалістичною.

2.3.4 Гібридні атаки.

Існують гібридні підходи типу “meet-in-the-middle” (наприклад ідея Odlyzko) або більш сучасні гібридні атаки Howgrave–Graham, які поєднують meet-in-the-middle з відновленням ключа через редукцію решітки. Ці методи були успішно застосовані проти деяких реалізацій NTRU, особливо коли

секрети є розрідженими поліномами. У SOLMAE секрети є щільними елементами кільця \mathcal{R} , отже цей клас атак втрачає основну перевагу (спарсність), і їхній вплив на вибір параметрів SOLMAE є незначним.