УДК 519.7 DOI 10.17223/20710410/67/2

СОВРЕМЕННЫЕ ПАРАДИГМЫ ПОСТРОЕНИЯ СХЕМ ЦИФРОВОЙ ПОДПИСИ НА РЕШЁТКАХ

А. Г. Леевик*,**, Е. С. Малыгина*,***, Е. М. Мельничук*, Д. А. Набоков*

*ООО «КуАпп», г. Москва, Россия
**Университет ИТМО, г. Санкт-Петербург, Россия
***Международный научно-образовательный математический центр НГУ, Новосибирский государственный университет, г. Новосибирск, Россия

E-mail: anton.leevik@gmail.com, emalygina@qapp.tech, emelnichuk@qapp.tech, dnabokov@qapp.tech

Ввиду возникновения угрозы нарушения стойкости криптографических алгоритмов с помощью квантового вычислителя большую популярность обрело направление постквантовой криптографии — нового раздела, включающего алгоритмы и протоколы, эффективно противостоящие атакам с помощью квантового компьютера. Теория решёток на сегодняшний день является перспективным направлением постквантовой криптографии. Одними из первых криптосистем на решетках были GGH и NTRU. В их основе лежит задача о поиске ближайшего вектора решётки, а отличие схем заключается в построении решёток. На основе криптосистем GGH и NTRU была предложена схема цифровой подписи NTRUSign, взявшая лучшее у них. Другой подход к построению подписи появился в 2008 г., в его основе лежит парадигма Hash-and-Sign, а подпись для сообщения вырабатывается с помощью лазейки. Годом позже В. Любашевским был предложен ещё один подход к построению подписи на решетках. Он заключается в использовании преобразования Фиата — Шамира, однако в силу специфики решёток алгоритм формирования подписи выдаёт корректную подпись с некоторой вероятностью, поскольку в целях безопасности используется выборка отбраковки. В данной работе представлен обзор существующих парадигм построения схем цифровых подписей на решётках, а также криптографических схем, построенных на рассматриваемых парадигмах. Проведён сравнительный анализ схем, определены преимущества и недостатки каждого из подходов, определены наилучшие условия их применения.

Ключевые слова: *постквантовая криптография, теория решёток, цифровая подпись на базе решёток.*

CURRENT PARADIGMS FOR CONSTRUCTION OF LATTICE-BASED DIGITAL SIGNATURE SCHEMES

A. G. Leevik*,**, E. S. Malygina*,***, E. M. Melnichuk*, D. A. Nabokov*

*QApp, Moscow, Russia
**ITMO University, Saint Petersburg, Russia
***Mathematical Center in Akademgorodok, Novosibirsk State University, Novosibirsk, Russia

With the advent of quantum computing, research into post-quantum cryptography has gained significant attention. This is a novel branch of cryptography that utilizes

algorithms and protocols designed to withstand attacks from quantum computers. Lattice theory represents a promising area within post-quantum cryptographic research. Two early examples of lattice-based cryptosystems are the GGH and NTRU schemes. These schemes are based on the challenge of finding the closest vector in a lattice and differ primarily in the type of lattice used. The NTRUSign protocol was developed by combining the strengths of both schemes. In 2008, another approach to lattice signatures was introduced by a group of authors. It is based on the hash-and-sign paradigm, in which a signature for a message is generated using a trapdoor. A year later, V. Lyubashevsky proposed another method for constructing lattice-based signatures that utilizes the Fiat — Shamir transform. However, due to the nature of the underlying lattice structure, the algorithm for signature generation produces a correct signature only with a certain probability. This is due to the use of a rejection sampling for security purposes. This paper presents an overview of existing lattice-based signature construction approaches and cryptographic schemes that are based on these approaches. A comparative analysis was conducted on these schemes, identifying the advantages and disadvantages of each method. Based on the results, optimal conditions for the application of each approach have been determined.

Keywords: *post-quantum cryptography, lattice theory, lattice-based signature.*

Введение

С появлением квантовых компьютеров и разработкой алгоритма Шора многие современные криптографические схемы окажутся нестойкими [1], а именно схемы, в основе которых лежат задачи факторизации или дискретного логарифмирования. Эта угроза способствовала развитию нового криптографического направления — постквантовой криптографии, послужившего толчком к созданию криптографических алгоритмов, противостоящих атакам с использованием квантовых компьютеров. Среди многих постквантовых направлений криптография на базе решёток является одной из наиболее перспективных. Криптографические примитивы, в основе которых лежат решётки, могут быть использованы для создания цифровых подписей и шифрования с открытым ключом. Трудные задачи на решётках, например задача нахождения кратчайшего вектора (SVP) [2] или задача поиска ближайшего вектора (CVP) [3], считаются стойкими к квантовым атакам. Соответственно можно говорить о перспективе замены существующих схем цифровой подписи и шифрования, которые будут подвержены атакам с использованием квантовых компьютеров, с точки зрения стойкости и практичности. С практической точки зрения некоторые схемы шифрования с открытым ключом и схемы цифровой подписи, основанные на трудных решёточных задачах, в настоящее время более практичны, чем, например, традиционные схемы RSA [4].

Теоретическое направление, касающееся доказуемой стойкости схем, в основе которых лежат решётки, возникло в 1996 г. вместе с редукцией Айтая от наихудшего случая к среднему [5], что, в свою очередь, привело к рассмотрению устойчивых к коллизиям хеш-функций, которые так же трудно взломать, как решить некоторые задачи, касающиеся евклидовых решёток, в наихудшем случае. Задача Айтая в среднем случае теперь называется задачей о нахождении малых целочисленных решений (SIS). Ещё одним крупным достижением в этой области стало представление О. Регевом задачи обучения с ошибками (LWE) [6]. Отметим, что LWE в среднем сложна (к ней сводятся стандартные задачи на решётках в худшем случае) и достаточно гибка для использования в криптографии.

Основной недостаток криптографических схем, основанных на задачах LWE и SIS, заключается в их ограниченной эффективности. Ключ обычно содержит случайную матрицу над кольцом $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ для небольшого q, размер которого линеен относительно параметра стойкости. При этом размерность векторного пространства решётки должна составлять, по меньшей мере, квадрат относительно параметра стойкости. В 2002 г. Д. Миччианчо адаптировал задачу SIS к структурированным матрицам с сохранением редукции от наихудшего случая к среднему [7]. Наихудший случай — это сужение стандартной задачи на решётках к задаче на конкретном семействе циклических решёток. Структура матриц Миччианчо допускает интерпретацию в терминах арифметики кольца $\mathbb{Z}_q[x]/(x^n - 1)$, где n — размерность в наихудшем случае, а q — малое простое число. Конструкция Миччианчо приводит к семейству хеш-функций, стойких к вычислению прообраза, с квазилинейной сложностью относительно параметра n. Наибольший выигрыш обусловлен использованием дискретного преобразования Фурье для умножения многочленов. В работах [8, 9] предложено заменить кольцо на $\mathbb{Z}_q[x]/\Phi$, где многочлен Φ является неприводимым над рациональными числами и имеет малые коэффициенты (например, $\Phi = x^n + 1$, где n — степень двойки). Полученная в результате хеш-функция оказалась стойкой к нахождению коллизий относительно предполагаемой сложности модифицированной задачи усреднения, которую в настоящее время часто называют задачей нахождения малых целочисленных решений над кольцом (R-SIS). В 2009 г. В. Любашевский [10] представил эффективную цифровую подпись, доказуемо такую же надёжную, как R-SIS (в модели случайного оракула). Чуть позже в [11] был предложен вариант задачи LWE, рассматриваемый в кольцах и названный R-LWE, большая гибкость которого обеспечивает более естественные и эффективные криптографические конструкции.

На сегодняшний день криптография на основе решёток стала доступной в качестве будущей альтернативы теоретико-числовой криптографии. Следует отметить, что использование специального класса решёток, например идеальных, даёт существенное ускорение и уменьшение размеров ключей для большинства криптопротоколов, а сложные решёточные задачи, адаптированные под такой класс решёток, остаются по-прежнему трудными [11, 12].

Настоящая работа посвящена обзору существующих парадигм, на базе которых строятся цифровые подписи на решётках, а также обзору самых первых цифровых подписей и одних из последних, являющихся наиболее привлекательными с точки зрения параметров и стойкости, построенных на этих парадигмах.

Структура работы следующая. В п. 1 приводятся основные обозначения и базовые определения, связанные с задачами на решетках. Пункты 2–4 посвящены описанию основных парадигм, на основе которых строятся цифровые подписи на решётках, а именно: рассматриваются парадигмы GGH/NTRUSign, Hash-and-Sign и Фиата — Шамира и для каждой парадигмы наиболее значимые цифровые подписи. В п. 5 обсуждаются выбор параметров и оценка стойкости для решёточных схем.

1. Предварительные сведения

1.1. Обозначения

Матрицы обозначаются заглавными буквами жирным шрифтом (например, $\mathbf{A}$), векторы — строчными буквами жирным шрифтом (например, $\mathbf{v}$). Векторы являются вектор-столбцами.

Для целого (обычно простого) q пусть $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ — кольцо целых чисел по модулю q, где элементы представлены числами из диапазона $\{-\lfloor (q-1)/2 \rfloor, \ldots, \lfloor (q-1)/2 \rfloor\}$.

Пусть $\mathcal{R} = \mathbb{Z}[x]/(\varphi(x))$ — кольцо многочленов для произвольного целого n по модулю многочлена $\varphi(x)$, где $\varphi(x)$ известен из контекста; $\mathcal{R}_{\mathbb{R}} = \mathbb{R}[x]/(\varphi(x))$ — кольцо многочленов над действительными числами; $\mathcal{R}_{\mathbb{Q}} = \mathbb{Q}[x]/(\varphi(x))$ — кольцо многочленов над рациональными числами; $\mathcal{R}_q = \mathbb{Z}_q[x]/(\varphi(x))$. Для $a \in \mathbb{R}$ символ $\lfloor a \rceil$ обозначает округление до ближайшего целого, причём $\lfloor 1/2 \rceil = 1$. Символ естественным образом расширяется для матриц, векторов и многочленов, округляя их каждый коэффициент.

Для целого η обозначим S_η — множество многочленов степени меньше n (где n определено из контекста) с коэффициентами из $[-\eta, \eta] \cap \mathbb{Z}$. Для заданного вектора многочленов $\mathbf{y} = \left(\sum\limits_{0 \leqslant i < n} y_i x^i, \ldots, \sum\limits_{0 \leqslant i < n} y_{nk-n+i} x^i \right)^{\mathrm{T}} \in \mathcal{R}^k$ определим его ℓ_2-норму как $\|\mathbf{y}\|_2 = \|(y_0, \ldots, y_{nk-1})^{\mathrm{T}}\|_2$. Аналогичным образом определяются ℓ_1- и ℓ_∞-нормы. Гипершаром с центром в $\mathbf{c} \in \mathcal{R}^m$ радиуса $r > 0$ и размерности $m > 0$ называется множество $\mathscr{B}_{\mathcal{R},m}(r, \mathbf{c}) = \left\{ \mathbf{x} \in \mathcal{R}_{\mathbb{R}}^m : \|\mathbf{x} - \mathbf{c}\|_2 \leqslant r \right\}$.

Для распределения U под записью $a \leftarrow U$ будем понимать, что элемент a принимает значение в соответствии с распределением U. В записи $a \leftarrow \mathcal{I}$ элемент a принимает случайное значение из множества $\mathcal{I}$.

Определение 1. Схема цифровой подписи состоит из трёх полиномиальных алгоритмов $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, где вероятностный алгоритм $\mathsf{Gen}(1^\lambda)$ вырабатывает пару ключей $(\mathsf{sk}, \mathsf{vk})$ для параметра стойкости λ (который равен отрицательному двоичному логарифму вероятности успеха наилучшей известной атаки на схему); вероятностный алгоритм $\mathsf{Sign}(\mathsf{sk}, \mu)$ возвращает подпись σ для сообщения μ; детерминированный алгоритм $\mathsf{Verify}(\mathsf{vk}, \mu, \sigma)$ возвращает бит b.

Цифровая подпись является γ-корректной, если для любой пары ключей $(\mathsf{sk}, \mathsf{vk})$ и сообщения μ выполняется

$$\Pr[\mathsf{Verify}(\mathsf{vk}, \mu, \mathsf{Sign}(\mathsf{sk}, \mu)) = 1] \geqslant \gamma, \tag{1}$$

где вероятностное пространство задаёт алгоритм Sign. Будем говорить, что подпись корректна в (Q)ROM модели, если неравенство (1) выполняется, когда вероятность также зависит от случайности в (квантовом) случайном оракуле, моделирующим хеш-функцию, используемую в схеме.

Дадим определение двум основным моделям безопасности для схем цифровой подписи, а именно: UFCMA (existential UnForgeability under Chosen Message Attacks) и UFNMA (existential UnForgeability under No-Message Attacks).

Определение 2 (UFCMA-модель). Пусть $T, \delta \geqslant 0$, тогда схема цифровой подписи $\mathrm{sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ является (T, δ)-UFCMA стойкой в QROM-модели, если для любого квантового злоумышленника $\mathcal{A}$ с границей на время выполнения $\leqslant T$, которому предоставляется (классический) доступ к подписывающему оракулу и (квантовый) доступ к случайному оракулу H, выполняется

$$\mathsf{Adv}_{\mathsf{sig}}^{\mathsf{UFCMA}}(\mathcal{A}) = \Pr_{(\mathsf{vk},\mathsf{sk})}[\mathsf{Verify}(\mathsf{vk}, \mu^*, \sigma^*) = 1 \mid (\mu^*, \sigma^*) \leftarrow \mathcal{A}^{H, \mathsf{Sign}}(\mathsf{vk})] \leqslant \delta,$$

где вероятностное пространство задаётся случайными $\mathcal{A}$ и $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$. Злоумышленник не может вызвать оракул Sign для сообщения μ^*.

Модель UFNMA определяется схожим образом, за исключением того, что злоумышленнику не разрешается вызывать подписывающий оракул для сообщений.

1.2. Трудные задачи на решётках

Поскольку в основе алгоритмов схемы цифровой подписи лежит дискретное гауссово распределение, дадим его определение.

Определение 3. Ненормализованное гауссово распределение со стандартным отклонением $\sigma \in \mathbb{R}$ и центром $c \in \mathbb{R}^n$, вычисленное в точке $x \in \mathbb{R}^n$, определяется как

$$\rho_{\mathbf{c},\sigma}(\mathbf{x}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2}\right).$$

В случае, когда $\mathbf{c} = \mathbf{0}$, распределение обозначают $\rho_\sigma(\mathbf{x})$.

Дискретное гауссово распределение над $\mathbb{Z}^m$ с центром в точке $\mathbf{0}$ определяется как

$$D_\sigma^m(\mathbf{x}) = \frac{\rho_\sigma(\mathbf{x})}{\rho_\sigma(\mathbb{Z}^m)},$$

где $\rho_\sigma(\mathbb{Z}^m) = \sum\limits_{\mathbf{x}\in\mathbb{Z}^m} \rho_\sigma(\mathbf{x})$.

Определим MLWE- и MSIS-гипотезы для алгебраических решёток, используемых в схемах цифровой подписи.

Определение 4 (Decision-$\mathsf{MLWE}_{n,q,k,\ell,\eta}$). Для положительных целых q, k, ℓ, η и размерности n кольца $\mathcal{R}_q$ будем говорить, что преимущество злоумышленника $\mathcal{A}$, решающего задачу Decision-$\mathsf{MLWE}_{n,q,k,\ell,\eta}$, равно

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MLWE}}_{n,q,k,\ell,\eta}(\mathcal{A}) = \big|\mathsf{Pr}[b=1|\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \mathbf{b} \leftarrow \mathcal{R}_q^k; b \leftarrow \mathcal{A}(\mathbf{A},\mathbf{b})] - \\ -\mathsf{Pr}[b=1|\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; (\mathbf{s}_1,\mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s}_1+\mathbf{s}_2)]\big|.\end{aligned}$$

Запись «$s \leftarrow S$» означает, что величина s выбрана в соответствии со случайным равномерным распределением на множестве S.

Определение 5 (Search-$\mathsf{MSIS}_{n,q,k,\ell,\beta}$). Для положительных целых q, k, ℓ, положительного действительного β и размерности n кольца $\mathcal{R}_q$ будем говорить, что преимущество злоумышленника $\mathcal{A}$, решающего задачу Search-$\mathsf{MSIS}_{n,q,k,\ell,\beta}$, равно

$$\mathsf{Adv}^{\mathsf{MSIS}}_{n,q,k,\ell,\beta}(\mathcal{A}) = \mathsf{Pr}\left[0 < \|\mathbf{y}\|_2 < \beta \wedge (\mathbf{A}|\mathbf{Id})\cdot\mathbf{y} = 0 \bmod q | \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \mathbf{y} \leftarrow \mathcal{A}(\mathbf{A})\right].$$

Аналогичным образом можно определить LWE- и SIS-гипотезы. В этом случае кольцо $\mathcal{R}_q$ имеет размерность $n = 1$, то есть векторы и матрицы рассматриваются над $\mathbb{Z}_q$. Для гипотез Ring-LWE и Ring-SIS имеем $k = \ell = 1$.

Определим также менее стандартную задачу SelfTargetMSIS, которая, однако, часто используется при доказательстве стойкости схем цифровой подписи, основанных на алгебраических решётках.

Определение 6 ($\mathsf{SelfTargetMSIS}_{H,n,q,k,\ell,\beta}$). Пусть $H : \{0,1\}^* \to \mathcal{R}_2$ — криптографическая хеш-функция. Для положительных целых q, k, ℓ, положительного действительного β и размерности n кольца $\mathcal{R}_q$ будем говорить, что преимущество злоумышленника $\mathcal{A}$, решающего задачу Search-$\mathsf{SelfTargetMSIS}_{H,n,q,k,\ell,\beta}$, равно

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{H,n,q,k,\ell,\beta}(\mathcal{A}) = \\ &= \mathsf{Pr}\left[\begin{array}{c} 0 \leqslant \|\mathbf{y}\|_2 \leqslant \beta \\ \wedge H(\mu\|(\mathbf{Id}|\mathbf{A})\cdot\mathbf{y}) = c \end{array} \;\middle|\; \mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}; \left(\mathbf{y} = \begin{pmatrix}\mathbf{r}\\ c\end{pmatrix}, \mu\right) \leftarrow \mathcal{A}^{|H(\cdot)\rangle}(\mathbf{A})\right].\end{aligned}$$

Задачи MSIS и SelfTargetMSIS могут быть определены и для ℓ_∞-нормы.

2. Парадигма GGH/NTRUSign

Криптосистемы GGH [13] и NTRUEncrypt [14] — одни из первых, в основе которых лежат сложные задачи на решётках, в частности задача об аппроксимации ближайшего вектора. Разница между этими схемами заключается в том, что последнюю можно рассматривать как спецификацию первой. Криптосистема GGH включает схему цифровой подписи, что легло в основу NTRUSign [15], которая сохраняет почти весь дизайн GGH, но с применением NTRU-решёток, используемых в NTRUEncrypt. Предшественник схемы NTRUSign, NSS [16], был взломан К. Джентри и др. [17, 18]. NTRUSign постигла та же участь, атаки представлены в [19], где показано восстановление секретного ключа с экспериментами на 400 подписях. Поскольку модификация NTRUSign с учётом контрмер и версия с возмущениями также были взломаны [20], интерес к этой схеме был потерян из-за непрактичности её применения. Однако недавние исследования [21] дают некоторые надежды относительно будущего этой схемы.

2.1. Схема цифровой подписи NTRUSign

В основе схемы цифровой подписи лежит задача об аппроксимации ближайшего вектора в NTRU-решётке (APPR-CVP).

Определение 7 (Задача APPR-CVP). Пусть $\mathcal{R} = \mathbb{Z}[x]/(\varphi(x))$, где $\varphi(x) = x^n - 1$; $\mathscr{L}$ — $\mathcal{R}$-модуль ранга r; $\mathbf{m} \in \mathcal{R}_{\mathbb{R}}^r$ — произвольный вектор. Вектор $\mathbf{v} \in \mathscr{L}$ называется решением задачи APPR-CVP, если $\|\mathbf{v} - \mathbf{x}\| < \mathcal{N}$ для некоторого малого $\mathcal{N} \in \mathbb{R}$.

На множестве $\mathcal{R} = \mathbb{Z}[x]/(x^n - 1)$ будем использовать операции сложения и свёрточного умножение (свёртку). Под свёрткой двух многочленов f и g будем понимать вычисление коэффициента при x^k в $f * g$:

$$(f * g)_k = \sum_{i+j=k \,(\mathrm{mod}\, N)} f_i \cdot g_j, \quad 0 \leqslant k < N,$$

где f_i и g_j — коэффициенты f и g соответственно для $i, j = 1, \ldots, N$.

Рассмотрим $q \in \mathbb{Z}$, $h \in \mathcal{R}$ и обозначим $M_{h,q} = \{(u, v) \in \mathcal{R}^2 : v = u * h \mod q\}$. Это множество является решёткой размерности $2N$. Если f и g соответствуют бинарным многочленам из $\mathcal{R}$ и d_f, d_g — количество отличных от нуля коэффициентов f и g соответственно, то нормы $\|f\|$, $\|g\|$ и $\|(f, g)\|$ определяются следующим образом:

$$\|f\| = \sqrt{d_f \left(1 - \frac{d_f}{N}\right)}, \quad \|g\| = \sqrt{d_g \left(1 - \frac{d_g}{N}\right)}, \quad \|(f, g)\| = \sqrt{\|f\|^2 + \|g\|^2}.$$

Генерация ключей. Зададим целые $N, q, B \geqslant 0$ и вектор t. Далее сгенерируем секретный и открытый базисы решётки. Для этого положим $i = B$ и пока $i \geqslant 0$:

- выберем случайным образом бинарные многочлены $f, g \in \mathcal{R}$ так, чтобы d_f и d_g были отличны от нуля;
- найдём малые $F, G \in \mathcal{R}$ (т. е. $\|F\|, \|G\| = \mathcal{O}(\sqrt{N})$), такие, что $f * G - F * g = q$;
- если t имеет стандартную форму, то положим $f_i = f$ и $f'_i = F$. Если t имеет траспонированную форму, то положим $f_i = f$ и $f'_i = g$. Также положим $h_i = f_i^{-1} * f'_i \mod q$ и $i = i - 1$.

Открытым ключом являются параметры N, q, d_f, d_g, B и $h = h_0 = f_0^{-1} * f'_0 \mod q$, а секретным — множество $\{f_i, f'_i, h_i : i = 0, \ldots, B\}$.

Формирование подписи. Для подписания необходима хеш-функция $H : \mathcal{D} \to \mathcal{R}$ для некоторого множества $\mathcal{D}$. В действительности $H = H_2 \circ H_1$, где H_1 — стандартная безопасная хеш-функция, получающая на выходе β бит $H_1(\mathcal{D})$. Функция $H_2 : \mathbb{Z}_2^\beta \to \mathbb{Z}_q^N$

задаёт дайджест сообщения $m = H_2(H_1(\mathcal{D}))$. Также нам потребуется функция нормы $\|\cdot\| : \mathcal{R}^2 \to \mathbb{R}$ и граница нормы $\mathcal{N} \in \mathbb{R}$. Для $(s,t) \in \mathcal{R}^2$ определим величину $\|(s \bmod q, r \bmod q)\|$, являющуюся минимальным значением среди $\|(s + k_1 q, r + k_2 q)\|$ для $k_1, k_2 \in \mathcal{R}$.

Процесс подписания сообщения $M \in \mathcal{D}$ с использованием секретного ключа $\{f_i, f_i', h_i : i = 0, \ldots, B\}$ выглядит следующим образом:

1) Положить $r = 0$.
2) Положить $s = 0$ и $i = B$. Представить r в виде битовой строки, вычислить $m_0 = H(M\|r)$ и положить $m = m_0$.
3) Пока $i \geqslant 1$:
   — вычислить $x = \lfloor -(1/q)m * f_i' \rceil$, $y = \lfloor -(1/q)m * f_i \rceil$, $s_i = x * f_i + y * f_i'$;
   — вычислить $m = s_i * (h_i - h_{i-1}) \bmod q$;
   — положить $s = s + s_i$, $i = i - 1$.
4) Вычислить $x = \lfloor -(1/q)m * f_0' \rceil$, $y = \lfloor -(1/q)m * f_0 \rceil$, $s_0 = x * f_0 + y * f_0'$ и положить $s = s + s_0$.
5) Вычислить $b = \|(s, (s * h - m_0) \bmod q)\|$. Если $b \geqslant \mathcal{N}$, то положить $r = r + 1$ и перейти на шаг 2.

Подписью является тройка (M, r, s).

Проверка. На вход подаются подпись (M, r, s) и открытый ключ h. Величина r кодируется в битовую строку и вычисляются значения $m = H(M\|r)$ и $b = \|(s, (s * h - m) \bmod q)\|$. Если $b < \mathcal{N}$, то подпись верна, в противном случае подпись отклоняется.

Безопасность. Восстановление ключа из публичной информации эквивалентно нахождению малых векторов в NTRU-решётке, что, в свою очередь, приводит к сложной задаче, на которой основана NTRUEncrypt. В транспонированной решётке ситуация для злоумышленника ещё сложнее, поскольку целевые малые векторы рассматриваются приближённо к гауссовой эвристике и, как следствие, на практике их сложно найти с помощью редукции решётки. Для подписи хорошо использовать редуцированный базис, однако нахождение такого базиса — трудная задача для больших N.

Рекомендованными параметрами для NTRUSign являются

$$(N, q, d_f, d_g, B, t, \mathcal{N}) = (251, 128, 73, 71, 1, \text{“transpose”}, 310);$$

время работы алгоритмов подписи указано в табл. 1.

Таблица 1

Время, затраченное на генерацию ключей, подпись и проверку NTRUSign

Схема	Генерация ключей, мс	Подпись, мс	Проверка, мс
NTRUSign-251	180 000	500	300

Первые работы, касающиеся того, что схемы цифровой подписи GGH и NTRUSign могут быть нестойкими, представили К. Джентри и М. Шидло [17, 18], которые заметили, что при каждой подписи происходит утечка некоторой информации о секретном ключе. Однако, как продемонстрировали П. Нгуен и О. Регев несколькими годами позже [19], эта утечка информации действительно приводит к атаке на схему. Точнее, они показали, что при наличии достаточного количества пар «сообщение — подпись» можно восстановить закрытый ключ. Более того, их атака достаточно эффективна и была

реализована и применена к большинству вариантов параметров в GGH и NTRUSign; тем самым установлено, что эти схемы цифровой подписи не являются стойкими на практике.

Идея атаки довольно проста. Основное наблюдение состоит в том, что разность $(m-s)$, полученная из пары «сообщение — подпись» (m,s), распределена равномерно в $\{\mathbf{Bx} : \mathbf{x} \in [-1/2, 1/2]^n\}$, где $\mathbf{B}$ — базис решётки. Следовательно, при достаточном количестве таких пар возникает следующая алгоритмическая задача, называемая задачей скрытого параллелепипеда: при множестве случайных точек, равномерно распределённых по неизвестному n-мерному параллелепипеду, необходимо восстановить этот параллелепипед или его приближение. Эффективное решение этой задачи составляет атаку.

Наилучшими мерами противодействия атаке являются методы возмущения [22, 23]. Они изменяют процесс генерации подписи таким образом, что скрытый параллелепипед заменяется значительно более сложным телом, что, по-видимому, предотвращает атаки описанного типа. Основной недостаток возмущений заключается в том, что они замедляют генерацию подписи и увеличивают размер секретного ключа. Однако NTRUSign с возмущениями не имеет никакого доказательства безопасности.

2.2. Модификации NTRUSign

Разработка схем цифровой подписи на решётках тесно связана с редукцией вектора по модулю фундаментального параллелепипеда секретного базиса [15]. Использование такого подхода привело к утечке секретной информации, а именно формы параллелепипеда [19]. NTRUSign — чрезвычайно эффективная схема, как следствие, интерес к разработке мер противодействия атакам возрастает, однако сами меры не имеют особого успеха [20].

Схема цифровой подписи № 1

В [21] представлен альтернативный подход к устранению утечки NTRUSign. Вместо модификаций используемых решёток и алгоритмов авторы рассматривают классическую негерметичную схему цифровой подписи NTRUSign и скрывают её с помощью гауссова шума, используя методы по аналогии со схемой цифровой подписи Любашевского.

Секретный ключ $\mathbf{S}$ представляется в виде матрицы из множества $\{-d, \ldots, d\}^{m\times k}$. Сообщение хешируется в вектор $\mathbf{c} \in \{-1, 0, 1\}^k$, такой, что $\|\mathbf{c}\|_1 \leqslant \kappa$, а подпись состоит из элемента $\mathbf{Sc}$, сдвинутого на маску $\mathbf{y} \leftarrow D_\sigma^m$, где D_σ^m является дискретным гауссовым распределением размерности m со стандартным отклонением σ.

Вместо того чтобы скрыть $\mathbf{Sc}$, авторы получают негерметичную схему цифровой подписи из NTRUSign, а затем используют эту подпись в качестве секретной и скрывают её с помощью корректно подобранного $\mathbf{y}$. Важный нюанс заключается в выборе размерности m и стандартного отклонения σ: если σ выбрано слишком маленьким, то секрет не будет скрыт должным образом, как следствие, представленная модификация не будет стойкой относительно атаки на NTRUSign. Если σ слишком велико, то схема потеряет практическую эффективность.

В отличие от других доказуемо стойких схем цифровой подписи, например таких, как GPV [24] или [25], исходная схема цифровой подписи NTRU не претерпевает изменений, за исключением более консервативного выбора общедоступных параметров, и тем самым сохраняет присущий ей размер и вычислительную эффективность. Однако маскировка подписи обходится дорого. В табл. 2 приведены размеры предлагаемой подписи и ключей для различных уровней стойкости.

Подпись скрывается с помощью шумоподавления в предположении, что не существует никаких структурных атак на открытый ключ NTRU, поскольку за последнее десятилетие не опубликовано никаких известных структурных атак на NTRU-решётки и что такие сложные задачи, как SIS, решаются не проще, чем в случайных решётках.

Таблица 2

Размеры ключей и подписи модифицированной версии NTRUSign

	Откр. ключ (биты)	Секр. ключ (биты)	Подпись (биты)
Уровень стойкости 100			
Размер	10 700	6 900	1 400
Скорость	12 700	6 900	1 400
Уровень стойкости 112			
Размер	12 300	7 700	1 550
Скорость	14 600	7 700	1 550
Уровень стойкости 128			
Размер	14 500	8 700	1 750
Скорость	17 100	8 700	1 750
Уровень стойкости 160			
Размер	16 800	9 900	2 000
Скорость	19 800	9 900	2 000

В силу гибридности схемы можно выделить три типа потенциальных атак:

1. Атака на NTRU-решётку заключается в попытке найти секретный ключ (f, g) только из открытого ключа $h = g * f^{-1}$. Однако эту задачу невозможно решить, поскольку отсутствует теоретическое доказательство неразрешимости этой атаки.
2. Попытка выявить исходную NTRU-подпись внутри построенной (скрытой) подписи, чтобы затем перейти к атаке по аналогии с [20]. Эту задачу можно решить с помощью методов, описанных в [26].
3. Если злоумышленнику удастся создать подделку за полиномиальное время, то эту подделку можно использовать для решения задачи SIS.

Стоит отметить, что некоторые атаки на NTRUSign могут снизить уровень стойкости схемы, а также то, что методы редукции решётки обречены на провал либо из-за того, что короткий вектор оказывается слишком длинным, либо из-за увеличения вычислительной сложности [20, 27–29].

Схема цифровой подписи № 2

Д. Штеле и Р. Штейнфельд получили доказуемо стойкую схему, очень близкую к NTRUSign (с теоретической точки зрения) [30]. Они модифицировали схему NTRUSign, чтобы сделать её доказуемо стойкой для использования в стандартной модели (соответственно в модели случайного оракула) относительно предполагаемой квантовой (соответственно классической) сложности стандартных задач на решётках в наихудшем случае, ограниченных семейством решеток в круговых полях. Авторы показали, как расширить секретный ключ схемы NTRUEncrypt до секретного ключа подписи. Стойкость схемы следует из уже доказанной сложности задач R-SIS и R-LWE.

Для модифицированной схемы NTRUSign была применена техника, описанная в [15]. Отметим, что заявленный подход является эвристическим. Например, мы выбираем секретный ключ для зашифрования и отклоняем выборку до тех пор, пока не будут выполняться некоторые свойства (в первую очередь, взаимная простота двух секретных многочленов над $\mathbb{Z}[x]/(x^n - 1)$). Однако как влияет использование такого

подхода на стойкость, детально не исследовано. Авторы показывают, что в модифицированной схеме вероятность отклонения достаточно далека от 1, если использовать дзета-функцию Дедекинда в круговых полях. Кроме того, стойкость схемы следует из сложности задачи R-SIS даже при использовании дополнительной отбраковки.

Модифицированная схема NTRUSign устойчива к атакам на классическом компьютере в модели случайного оракула (предполагающей наихудшую сложность стандартных решёточных задач для идеальных решёток). Из-за использования случайного оракула отсюда непосредственно не следует, остаётся ли это доказательство значимым в случае квантовых атак. Как указано в [31], следует быть крайне осторожным, используя случайный оракул в квантовой модели.

3. Парадигма Hash-and-Sign

Схемы цифровой подписи, основанные на парадигме Hash-and-Sign, берут свое начало из работы У. Диффи и М. Хеллмана [32]. Концепция построения следует критерию, согласно которому сообщение M должно быть хешировано перед тем, как быть подписанным. Таким образом, для подписания сообщения сначала необходимо вычислить его хеш, чтобы получить величину $h = H(M)$, которая должна находиться в диапазоне функции лазейки f (пример такой функции — функция RSA). Затем результат хеширования подписывается $\sigma = f^{-1}(h)$, а алгоритм проверки проверяет условие $f(\sigma) = H(M)$, чтобы выяснить, является ли (σ, M) допустимой парой.

Такая теория стала основой для моделирования хеш-функций с помощью случайного оракула [33]. Если f является перестановочной лазейкой, то схема экзистенциально устойчива к атаке на выбранное сообщение. Приложение решёток к такому типу схем на основе парадигмы Hash-and-Sign заключается в том, что короткий базис решётки мог бы обеспечить такую функцию лазейки. Это породило первое предложение — схему GPV цифровой подписи, основанную на сложных задачах на решётках [24]. Основными в схеме являются:

— построение функций лазейки со свойством, что каждое выходное значение имеет несколько прообразов;
— использование гауссовой выборки;
— использование модульных решёток.

Более поздняя схема Миччианчо и Пайкерта [34] также использует парадигму Hash-and-Sign, но с более эффективной лазейкой, нежели в GPV. Улучшения в алгоритме генерации ключей были внесены Дж. Олвеном и К. Пайкертом [35].

3.1. Схема цифровой подписи GPV

GPV — схема цифровой подписи на решётках, стойкость которой основана на сложности решения задачи SIS. В алгоритмах схемы цифровой подписи используется функция лазейки, дискретное гауссово распределение и модульные решётки.

Зададим целые значения n, q (полиномиально зависящее от n), $m = \Omega(n \log q)$ и $s \geqslant \geqslant 2\sqrt{n \log q}$ — стандартное отклонение. В оригинальной работе [24] в качестве функции лазейки предложено использовать короткий базис решётки.

Определение 8. Для матрицы $\mathbf{A} \in \mathbb{Z}^{n \times m}$, определяющей решётку

$$\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{Az} = 0 \pmod q\} \supseteq q\mathbb{Z}^m,$$

матрица $\mathbf{T} \in \mathbb{Z}^{n \times m}$ является лазейкой, если:

— $\mathbf{AT} = 0 \pmod q$;
— $\mathbf{T}$ имеет полный ранг над $\mathbb{Z}$;

— каждый столбец $\mathbf{t}_i \in \mathbb{Z}_q^m$ матрицы $\mathbf{T}$ является коротким.

Генерация ключей. На этапе генерации по заданным параметрам создаётся пара $(\mathsf{vk}, \mathsf{sk})$, где

— открытый ключ vk составляют описанная выше матрица $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ и такая функция $f_{\mathbf{A}} : \mathbb{Z}^m \to \mathbb{Z}^n$, что для $\mathbf{x} \in \mathbb{Z}^m$

$$\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \pmod{q};$$

— секретный ключ $\mathsf{sk} = \mathbf{T}$, являющийся коротким базисом решётки $\Lambda^{\perp}(A)$, выступает в роли лазейки для решения задачи SIS и используется для выбора $\mathbf{x}'$, такого, что $\mathbf{x}' = f_{\mathbf{A}}^{-1}(\mathbf{u})$ с вероятностью $e^{-\|\mathbf{x}'\|^2/\sigma^2}$.

Фактически алгоритм выработки открытого/секретного ключей заключается в генерации односторонней функции с лазейкой. Для этого используется алгоритм $\mathsf{TrapGen}(1^n)$, возвращающий пару $(\mathbf{A}, \mathbf{T})$, где $\mathbf{A}$ описывает одностороннюю функцию $f_{\mathbf{A}}$, а $\mathbf{T}$ — лазейка.

Формирование подписи. Для подписания сообщения необходима хеш-функция $H : \{0,1\}^\star \to \mathbb{Z}_q^n$. Перед подписанием сообщение M хешируется:

$$\mathbf{y} = H(M) \in \mathbb{Z}_q^n.$$

Так как хеш-значение — это вектор, его можно записать в виде $\mathbf{y} = \mathbf{A}\mathbf{x}$. Далее используем короткий базис $\mathbf{T}$ и функцию $f_{\mathbf{A}}^{-1}$, чтобы получить прообраз, соответствующий полученному хеш-значению:

$$\mathbf{x}' = f_{\mathbf{A}}^{-1}(\mathbf{u}).$$

Для этого используется рандомизированный вариант алгоритма “nearest plane” [24] — алгоритм 1 ($\mathsf{SampleD}$).

Алгоритм 1. SampleD

Вход: $\mathbf{T}, \mathbf{s}, \mathbf{c}$.
Выход: Вектор $\mathbf{v}_0$.

$\mathbf{v}_n := \mathbf{0}$.
$\mathbf{c}_n := \mathbf{c}$.
Для $i = n, \ldots, 1$:
  $c'_i := \langle \mathbf{c}_i, \widetilde{\mathbf{t}_i} \rangle / \langle \widetilde{\mathbf{t}_i}, \widetilde{\mathbf{t}_i} \rangle \in \mathbb{R}$;
  $s'_i := s / \|\widetilde{\mathbf{t}_i}\|_2 > 0$;
  $z_i \sim D_{\mathbb{Z}, s'_i, c'_i}$;
  $\mathbf{c}_{i-1} := c_i - z_i \mathbf{b}_i$;
  $\mathbf{v}_{i-1} := \mathbf{v}_i + z_i \mathbf{b}_i$.
Вернуть $\mathbf{x}' := \mathbf{v}_0$.

Полученный таким образом вектор $\mathbf{x}'$ является подписью σ_M сообщения M.

Проверка. Для проверки подписи достаточно вычислить хеш-значение $\mathbf{y} = H(M)$, убедиться, что подпись является коротким вектором, то есть $\|\sigma_M\|_2 \leqslant 6n \log q$, и проверить, что выполняется равенство $\mathbf{A} \cdot \sigma_M = \mathbf{y}$.

Безопасность GPV. Стойкость схемы GPV основывается на сложности задачи SIS. В оригинальной работе [24] невозможность подделки подписи доказывается в модели EUF-CMA. Изложим суть доказательства.

Пусть алгоритм $\mathcal{A}$ позволяет подделать подпись с вероятностью $e = e(n)$. Построим алгоритм $\mathcal{B}$, решающий задачу SIS с вероятностью $p = e(n)(1 - 2^{-\Omega(n)})$, используя для этого алгоритм $\mathcal{A}$ и симулятор $\mathcal{S}$. Принимая на вход матрицу $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathcal{B}$ должен найти ненулевой вектор $\mathbf{z} \in \mathbb{Z}_q^m$, такой, что $||\mathbf{z}||_2 < \beta$ и $\mathbf{Az} = 0 \pmod q$. Поскольку алгоритм $\mathcal{A}$ требует взаимодействия с оракулом подписи GPV и случайным оракулом, алгоритм $\mathcal{B}$ использует симулятор $\mathcal{S}$, чтобы симулировать ответы на запросы $\mathcal{A}$, играя роль «реального» алгоритма. Для корректной работы алгоритма необходимо, чтобы распределение ответов симулятора $\mathcal{S}$ и реальных оракула подписи и случайного оракула было неотличимо.

В рамках модели безопасности алгоритм $\mathcal{A}$ в ходе работы может осуществлять два типа запросов: на хеш (то есть обращение к случайному оракулу) и на подпись (обращение к оракулу подписи). Симулятор $\mathcal{S}$ в зависимости от запроса поступает следующим образом:

— при обращении к случайному оракулу: на каждый запрос $H(M)$, если до этого такого запроса не было, выбирается $\mathbf{x}_M \leftarrow D_{\mathbb{Z}^m,\sigma}$, вычисляется $\mathbf{u}_M = \mathbf{Ax}_M \bmod q$ и возвращается $H(M) = \mathbf{u}_M$. При этом тройка элементов $(M, \mathbf{u}_M, \mathbf{x}_M)$ сохраняется в памяти для будущих запросов;

— при обращении к оракулу подписи: для каждого запроса на подпись $\mathsf{Sign}(M)$ мы предполагаем с высокой вероятностью, что предварительно был сделан запрос на подпись $H(M)$. Поэтому мы можем извлечь соответствующую тройку $(M, \mathbf{u}_M, \mathbf{x}_M)$ и вернуть значение $\mathbf{x}_M$ в качестве подписи.

После коммуникации с симулятором $\mathcal{S}$ алгоритм $\mathcal{A}$ с вероятностью успеха e возвращает подделку $(M^\star, \mathbf{x}^\star)$ для сообщения $M^\star$, для которого не было сделано запросов на подпись. После получения подделки алгоритм $\mathcal{B}$ делает запрос к оракулу подписи симулятора $\mathcal{S}$ для сообщения $M^\star$ и получает значение $\mathbf{x}_{M^\star}$. Таким образом, получаем равенство

$$H(M^\star) = \mathbf{Ax}^\star = \mathbf{Ax}_{M^\star}.$$

При этом так как $\Pr[\mathbf{x}^\star = \mathbf{x}_{M^\star}] \leqslant 2^{-\Omega(n)}$ [36], то $(\mathbf{x}^\star - \mathbf{x}_{M^\star})$ — решение задачи SIS с вероятностью $1 - 2^{-\Omega(n)}$.

3.2. Схема цифровой подписи Falcon

Falcon — схема цифровой подписи, использующая парадигму Hash-and-Sign на базе NTRU-решёток. Falcon использует преимущества NTRUSign и GPV, за счёт чего удаётся сделать размеры ключей и подписи компактными, сохраняя доказуемую стойкость схемы.

В схеме Falcon в качестве модуля используется число $q = 12\,289$. Зададим целое $n = 2^k$ и круговой многочлен $\phi(x) = x^n + 1$, задающий кольцо $\mathcal{R} = \mathbb{Z}[x]/(\phi(x))$. При этом ϕ полностью раскладывается в $\mathbb{Z}_q[x]$. Определим границу $\lfloor \beta^2 \rfloor > 0$ и стандартное отклонение σ. Кроме того, зададим промежуток $[\sigma_{\min}, \sigma_{\max}]$, в котором будут варьироваться все значения σ' при выборке целого числа $z \sim D_{\mathbb{Z},\sigma',\mu}$ с помощью подпроцедуры $\mathsf{SamplerZ}$ алгоритма $\mathsf{ffSampling}$. Байтовую длину подписи обозначим через $\mathsf{sbytelen}$.

Генерация ключей (алгоритм 2). Процесс выработки ключевой пары в схеме цифровой подписи Falcon состоит из двух частей:

1) Решение NTRU-уравнения: ищутся многочлены $f, g, F, G \in \mathcal{R}$, удовлетворяющие NTRU-уравнению

$$fG - gF = q \pmod \phi.$$

2) Построение бинарного дерева Falcon T, которое удовлетворяет следующим условиям:
   — значением корня T.value дерева T высоты k является многочлен $l \in \mathcal{R}_{\mathbb{Q}}$, где $n = 2^k$;
   — левый и правый потомки дерева T, обозначаемые как T.left и T.right, являются бинарными деревьями Falcon высоты $k - 1$.

Для построения дерева T используется $\mathbf{LDL}^\star$-разложение матриц, где $\mathbf{L} \in \mathcal{R}_{\mathbb{Q}}^{n \times n}$ — нижняя треугольная матрица с единичными элементами на диагонали; $\mathbf{L}^\star$ — матрица, эрмитово сопряжённая матрице $\mathbf{L}$; $\mathbf{D} \in \mathcal{R}_{\mathbb{Q}}^{n \times n}$ — диагональная матрица.

Непосредственно нам понадобится $\mathbf{LDL}^\star$-разложение матрицы $\mathbf{G} = \mathbf{BB}^\star$, где

$$\mathbf{B} = \begin{bmatrix} g & -f \\ G & -F \end{bmatrix};$$

$\mathbf{B}^\star$ — матрица, эрмитово сопряжённая матрице $\mathbf{B}$. В алгоритме генерации ключей схемы цифровой подписи Falcon для построения T используется функция ffLDL*, которая принимает на вход матрицу $\mathbf{G}$.

Алгоритм ffLDL* построения дерева T работает рекурсивно и состоит из двух частей:

1) на первом этапе вычисляется $\mathbf{LDL}^\star$-разложение матрицы $\mathbf{G} = \mathbf{BB}^\star$, которая подаётся на вход в алгоритм в FFT-представлении, то есть каждый элемент матрицы $\mathbf{G}$ представлен FFT-формой:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 \\ L_{21} & 1 \end{bmatrix} \cdot \begin{bmatrix} D_{11} & 0 \\ 0 & D_{22} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ L_{21}^\star & 1 \end{bmatrix},$$

   где $L_{ij}, D_{ij} \in \mathcal{R}_{\mathbb{Q}}$. Значение L_{21} сохраняется в корне дерева T.value;
2) на втором этапе определяются значения левого и правого потомков корня T.value из элементов $D_{11}, D_{22} \in \mathbb{Q}[x]/(x^n + 1)$ матрицы $\mathbf{D}$. Равенство $D_{ii} = d_{i1}(x^2) + x d_{i2}(x^2)$, $i \in 1, 2$, позволяет задать функцию

$$\mathsf{splitfft}(D_{ii}) = (d_{i1}, d_{i2}) \in \left(\mathbb{Q}[x]/(x^{n/2} + 1)\right)^2.$$

   Из полученных элементов d_{ij} строятся две новые матрицы:

$$\mathbf{G}_{\text{left}} = \begin{bmatrix} d_{11} & d_{12} \\ d_{12}^\star & d_{11} \end{bmatrix}, \quad \mathbf{G}_{\text{right}} = \begin{bmatrix} d_{21} & d_{22} \\ d_{22}^\star & d_{21} \end{bmatrix}.$$

   Для матрицы $\mathbf{G}_{\text{left}}$ вызывается алгоритм ffLDL*, чтобы определить значение T.left и всех его потомков по рекурсии. Аналогично матрица $\mathbf{G}_{\text{right}}$ используется для вычисления значения T.right и всех его потомков.

Заметим, что алгоритм ffLDL* строит ненормализованное дерево T. Нормализация относительно σ происходит на шагах 6–7 алгоритма генерации ключей.

Формирование подписи (алгоритм 3). Для подписания вычисляется хеш-значение $c \in \mathcal{R}_q$ от сообщения $\mathbf{m}$, дополненного случайно сгенерированной солью $\mathbf{r} \in \{0,1\}^{320}$. В схеме Falcon для этого используется функция HashToPoint, принимающая на вход $\mathbf{r}||\mathbf{m}$, а также значения q и n. Следующим шагом с помощью алгоритма семплирования ffSampling [37], реализующего выборку с использованием лазейки “fast Fourier nearest plane” [38], находим значения $s_1, s_2 \in \mathcal{R}$, такие, что $s_1 + s_2 h = c \pmod q$. “Fast Fourier

Алгоритм 2. Выработка ключей Falcon

Вход: $\phi \in \mathbb{Z}[x]$, q.
Выход: (sk, vk).
1: $f, g, F, G := \mathsf{NTRUGen}(\phi, q)$.
2: $\mathbf{B} := \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$.
3: $\hat{\mathbf{B}} := \begin{bmatrix} \mathsf{FFT}(g) & -\mathsf{FFT}(f) \\ \mathsf{FFT}(G) & -\mathsf{FFT}(F) \end{bmatrix}$.
4: $\mathbf{G} := \hat{\mathbf{B}} \times \hat{\mathbf{B}}^{\star}$.
5: $T := \mathsf{ffLDL}^{\star}(\mathbf{G})$.
6: **Для** $leaf \in T$:
7: $\quad leaf.\text{value} := \sigma / \sqrt{leaf.\text{value}}$
8: $sk := (\hat{\mathbf{B}}, \mathbf{T})$.
9: $h := gf^{-1} \bmod q$.
10: $pk := h$.

nearest plane" представляет собой рекурсивный вариант алгоритма семплирования, предложенного П. Клейном [39] и использованного в оригинальной схеме цифровой подписи GPV [24].

Однако, в отличие от алгоритма выборки Клейна, алгоритм, предложенный в схеме цифровой подписи Falcon в качестве лазейки, вместо одной матрицы $\mathbf{L}$ использует дерево T, состоящее из набора таких матриц.

На последнем шаге алгоритма формирования подписи s_2 сжимается в битовую строку $\mathbf{s}$ посредством функции $\mathsf{Compress}$ и подписью становится пара $(\mathbf{r}, \mathbf{s})$.

Алгоритм 3. Формирование подписи Falcon

Вход: сообщение $\mathbf{m}$, закрытый ключ sk, ограничение $\lfloor \beta^2 \rfloor$.
Выход: подпись ω.
1: $\mathbf{r} \xleftarrow{\$} \{0,1\}^{320}$.
2: $c := \mathsf{HashToPoint}(\mathbf{r}||\mathbf{m}, q, n)$.
3: $\mathbf{t} := \left(-\frac{1}{q}\mathsf{FFT}(c) \cdot \mathsf{FFT}(F), \frac{1}{q}\mathsf{FFT}(c) \cdot \mathsf{FFT}(f) \right)$.
4: **Выполнить**
5: $\quad$ **Выполнить**
6: $\quad\quad \mathbf{z} := \mathsf{fftSampling}(\mathbf{t}, \mathbf{T})$,
7: $\quad\quad \mathbf{s} := (\mathbf{t} - \mathbf{z})\hat{\mathbf{B}}$
8: $\quad$ **до тех пор пока** $||\mathbf{s}||_2^2 > \lfloor \beta^2 \rfloor$;
9: $\quad (s_1, s_2) := \mathsf{invFFT}(\mathbf{s})$,
10: $\quad \mathbf{s} := \mathsf{Compress}(s_2, 8 \cdot \mathsf{sbytelen} - 328)$
11: **до тех пор пока** $(\mathbf{s} = \perp)$.
12: **Вернуть** $\omega := (\mathbf{r}, \mathbf{s})$.

Проверка (алгоритм 4). Чтобы проверить подпись $\omega = (\mathbf{r}, \mathbf{s})$, проверяющая сторона начинает с вычисления хеш-значения $c \in \mathcal{R}_q$, используя $\mathbf{r}||\mathbf{m}$ и алгоритм $\mathsf{HashtoPoint}$. После этого к $\mathbf{s}$ применяется функция $\mathsf{Decompress}$ для получения значения $s_2 \in \mathcal{R}_q$,

что, в свою очередь, позволяет вычислить $s_1 = c - s_2 h \mod q$. На последнем этапе проверяется условие $||(s_1, s_2)||_2 \leqslant \lfloor \beta^2 \rfloor$.

Безопасность. Как уже отмечалось, стойкость схемы GPV, лежащей в основе цифровой подписи Falcon, базируется на сложности задачи SIS в модели ROM [24]. В схеме Falcon используются NTRU-решётки, поэтому доказательство стойкости, представленное для схемы GPV, адаптируется для NTRU-решёток. Фактически злоумышленнику $\mathcal{A}$ для подделки подписи некоторого сообщения $M^\star$ при заданных $y \in \mathcal{R}$ и матрицы $\mathbf{A} = \begin{bmatrix} 1 & h \end{bmatrix}$, являющейся открытым ключом, необходимо найти такие короткие $s_1, s_2 \in \mathcal{R}$, что выполняется

$$\mathbf{As} = \begin{bmatrix} 1 & h \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = s_1 + s_2 h = y \pmod q.$$

Алгоритм 4. Проверка подписи Falcon

Вход: сообщение $\mathbf{m}$, подпись $\omega = (\mathbf{r}, \mathbf{s})$, открытый ключ pk, ограничение $\lfloor \beta^2 \rfloor$.
Выход: принять или отклонить подпись.

1: $c := \mathsf{HashToPoint}(\mathbf{r}||\mathbf{m}, q, n)$.
2: $s_2 := \mathsf{Decompress}(\mathbf{s}, 8 \cdot \mathsf{sbytelen} - 328)$.
3: **Если** $s_2 = \perp$, **то**
4: отклонить подпись.
5: $s_1 := c - s_2 h \mod q$.
6: **Если** $||(s_1, s_2)||_2^2 \leqslant \lfloor \beta^2 \rfloor$, **то**
7: принять подпись,
8: **иначе**
9: отклонить подпись.

Помимо подделки, одно из направлений возможных атак на схему цифровой подписи Falcon связано с восстановлением секретного ключа. Для проведения данной атаки применяются алгоритмы редукции решёток, в частности алгоритм DBKZ [40], а определение параметров, при которых схема соответствует уровням стойкости, осуществляется с помощью методологии coreSVP hardness (см. далее п. 5). Параметры для обеспечения стойкости схемы Falcon для разных уровней представлены в табл. 3.

Таблица 3

Параметры для цифровой подписи Falcon

Схема	Уровень стойкости (NIST)	n	q	σ	$\sigma_{\min}$	$\sigma_{\max}$	$\lfloor \beta^2 \rfloor$
Falcon-512	I	512	12289	165,736617183	1,277833697	1,8205	34034726
Falcon-1024	V	1024	12289	168,388571447	1,298280334	1,8205	70265242

4. Парадигма Фиата — Шамира

Альтернативный способ построения схемы цифровой подписи заключается в том, чтобы сначала построить некоторый протокол аутентификации сторон, а затем преобразовать его в схему цифровой подписи с помощью преобразования Фиата — Шамира [41, 42]. Одной из известных схем классической криптографии, построенной на таком подходе, является схема Шнорра [43]. Перенос схемы Шнорра на аппарат теории решёток, а также её адаптация к сложным задачам на решётках были осуществлены впервые В. Любашевским в работах [10, 26, 44].

В [10] за основу взята задача о коротком целочисленном решении SIS. Первоначально протокол аутентификации сторон строится на основе решётки, в которой вызов (challenge) рассматривается как многочлен некоторого кольца $\mathcal{R}$. Стойкость протокола основана на сложности нахождения аппроксимированного кратчайшего вектора в стандартной модели, а также в модели случайного оракула.

Затем протокол аутентификации сторон преобразуется в схему цифровой подписи путём использования хеш-функции h из стойкого к коллизиям семейства хеш-функций $\mathcal{H}$ для выработки вызова на стороне подписывающего. В схеме цифровой подписи также оптимизируются параметры протокола. Стойкость схемы цифровой подписи зависит от сложности поиска коллизий в определённых семействах хеш-функций. Злоумышленник, способный подделать подпись, может затем использовать это для поиска коллизии хеш-функции, выбранной случайным образом из $\mathcal{H}$. Следовательно, подделка подписи и нахождение коллизии случайно выбранной $h \in \mathcal{H}$ эквивалентно нахождению коротких векторов в решётке над кольцом $\mathcal{R}$.

Разница со схемой Шнорра заключается в том, что для сохранения сложности задачи SIS секретные векторы необходимо брать малыми по норме, из чего следует, что и нормы векторов подписи также малы. Из этого можно сделать вывод, что, в отличие от схемы Шнорра, где распределение значения подписи равновероятно по всему $\mathbb{Z}_q$, вектор подписи при переносе на решётки не распределён равномерно по всему пространству $\mathbb{Z}_q^n$, а подчиняется некоему закону распределения $\mathcal{Q}$. Для обеспечения стойкости схемы цифровой подписи требуется, чтобы распределение выходных векторов не зависело от секретных значений, поэтому при выработке вектора подписи из исходного распределения $\mathcal{Q}$ с некоторой вероятностью он отклоняется и процесс выработки начинается сначала. Отказ происходит таким образом, чтобы распределение выходных векторов подписи не зависело от секретного значения. Такая парадигма получила название «парадигма Фиата — Шамира с прерываниями», а алгоритм выработки элементов из заданного распределения называется алгоритмом выработки отбраковки.

В [26] В. Любашевским предложены модификации его ранних работ. Наиболее значительным изменением является адаптация задачи Ring-SIS к задаче Ring-LWE, что существенно уменьшает размеры подписи и ключей, тем самым повышая её эффективность. Второе улучшение касается алгоритма формирования подписи, который включает асимптотически более короткие подписи. Здесь требуется более сложная выборка отбраковки, чтобы подпись была независима от секретного ключа, а также выборка из нормального распределения, где необходимы высокоточные вычисления. В работе доказано, что схема является устойчивой относительно известных атак, её стойкость основана на сложности нахождения коротких векторов в решётке в наихудшем случае.

На парадигме Фиата — Шамира с прерываниями построена схема цифровой подписи Dilithium [45], которая выбрана Национальным институтом стандартов и технологий США в качестве нового стандарта цифровой подписи [46]. В новом конкурсе на выбор дополнительной схемы цифровой подписи также представлены схемы на решётках, построенные на данной парадигме, среди них выделяется схема HAETAE [47], в которой авторы смогли уменьшить размер подписи по сравнению с Dilithium на 30 %.

4.1. Схема цифровой подписи Любашевского

Опишем общий подход к построению схемы цифровой подписи, а также рассмотрим различные варианты, предложенные самим Любашевским либо другими авторами.

Пусть q — простое число, задающее поле вычетов $\mathbb{Z}_q$. Несмотря на то, что некоторые варианты схем Любашевского построены в кольцах многочленов $\mathcal{R}_q = \mathbb{Z}_q[x]/(p(x))$,

где $p(x)$ — многочлен степени n, существует биективное отображение $\sigma : \mathcal{R}_q \to \mathbb{Z}_q^n$, таким образом, будем рассматривать построение схемы на целочисленных решётках как общий случай. Пусть k, l, m — размерности матриц и векторов, зависящие от уровня стойкости схемы; $\mathcal{Q}, \mathcal{C}$ — распределения маскирующего вектора $\mathbf{y}$ и секретного ключа $\mathbf{s}$ соответственно; $\mathcal{P}$ — целевое распределение вектора $\mathbf{z}$ на выходе алгоритма.

Генерация ключей. Первоначально выбирается открытая матрица $\mathbf{A} \in \mathbb{Z}_q^{k \times l}$, являющаяся общей для всех пользователей системы (может быть вычислена предварительно). Секретным ключом является матрица $\mathbf{S} \leftarrow \mathcal{C}^{l \times m}$; открытым — значение $\mathbf{T} = \mathbf{AS} \bmod q \in \mathbb{Z}_q^{k \times m}$.

Формирование подписи. Сначала вырабатывается маскирующий вектор $\mathbf{y}$ из распределения $\mathcal{Q}^l$. Далее для сообщения μ подписывающий вычисляет хеш, используя стойкую к коллизиям хеш-функцию H, получает значение $\mathbf{c} = H(\mathbf{Ay} \bmod q, \mu) \in \mathbb{Z}_q^m$ и вычисляет потенциальный вектор подписи $\mathbf{z} = \mathbf{Sc} + \mathbf{y} \in \mathbb{Z}_q^l$. Поскольку вектор $\mathbf{y}$ выбран из распределения $\mathcal{Q}$, а значение $\mathbf{S}$ не меняется при генерации новой подписи, можно заметить, что распределение вектора $\mathbf{z}$ равно распределению $\mathcal{Q}$, сдвинутому на $\mathbf{Sc}$ (будем обозначать как $\mathcal{Q}_{+\mathbf{Sc}}$). Чтобы выходное распределение вектора $\mathbf{z}$ было равно $\mathcal{P}$, применим процедуру выборки отбраковки [48], в результате которой в качестве подписи возвратим значение $(\mathbf{z}, \mathbf{c})$.

Выборка отбраковки. Пусть даны две функции плотности распределения вероятностей p_s и p_t; выборка отбраковки — это способ получения элементов из p_t при наличии доступа к элементам из p_s. Для p_s и p_t существует такое значение $M \in \mathbb{R}$, что для любого x имеет место $p_t(x) \leqslant M \cdot p_s(x)$. Тогда если значение z получено из p_s и принято с вероятностью $\dfrac{p_t(z)}{M \cdot p_s(z)}$, то распределение принятого z в точности равно p_t, а среднее количество прерываний для генерации z равно M. В рамках алгоритма формирования подписи вектор $\mathbf{z}$ принимается с вероятностью $\dfrac{\mathcal{P}(\mathbf{z})}{M \cdot \mathcal{Q}_{+\mathbf{Sc}}(\mathbf{z})}$.

Проверка. На входе пара $(\mathbf{z}, \mathbf{c})$ и открытый ключ $\mathbf{T}$. Сначала проверяется, что вектор $\mathbf{z}$ имеет малую норму, то есть $\|\mathbf{z}\| \leqslant \beta$ (в зависимости от спецификации схемы может использоваться как ℓ_2-, так и ℓ_∞-норма) для β, выбранного в зависимости от распределения $\mathcal{P}$. Если эта проверка пройдена, то проверяется условие $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc} \bmod q, \mu)$. Если равенство выполнено, то подпись принимается.

Безопасность. Стойкость схемы Любашевского основывается на сложности задач SIS и LWE. Защищённость подписи от подделки докажем в модели EUF-CMA. В рамках данной модели злоумышленник $\mathcal{A}$ может отправлять q_s запросов на подпись, а также q_h запросов на хеширование выбранных им сообщений. Злоумышленник выигрывает, если на выходе он выдаёт корректную подделку подписи для сообщения, для которого не отправлял запроса на подпись. В рамках теоретического доказательства приведём алгоритм симуляции подписи, а также алгоритм сведения задачи подделки подписи к задаче SIS.

Пусть существует алгоритм $\mathcal{B}$, на вход которого подаётся матрица $\mathbf{A}$. На выходе $\mathcal{B}$ должен вернуть короткий вектор $\mathbf{v}$, такой, что $\mathbf{Av} = \mathbf{0}$. Для решения задачи SIS $\mathcal{B}$ использует алгоритм $\mathcal{A}$, решающий задачу подделки подписи. $\mathcal{B}$ выбирает параметры q, k, l, m для схемы цифровой подписи и из экземпляра задачи SIS генерирует открытый и секретный ключи: для этого $\mathcal{B}$ выбирает матрицу $\mathbf{S} \leftarrow \mathcal{C}^{l \times m}$ и вычисляет $\mathbf{T} = \mathbf{AS} \bmod q$. Алгоритм $\mathcal{A}$ получает на вход от $\mathcal{B}$ матрицы $\mathbf{A}$ и $\mathbf{T}$ и на выходе выдаёт подделку подписи $(\mathbf{z}, \mathbf{c})$. В ходе работы $\mathcal{A}$ может делать два типа запросов: на хеш и на подпись. Для обработки запросов от $\mathcal{A}$ алгоритм $\mathcal{B}$ использует симулятор $\mathcal{S}$,

который симулирует схему цифровой подписи таким образом, что при создании подписи не используется секретный ключ. На вход симулятору подаются матрицы $\mathbf{A}, \mathbf{T}$, а алгоритм $\mathcal{A}$ отправляет $\mathcal{S}$ свои запросы. При обработке запросов на подпись и на хеш $\mathcal{S}$ использует случайный оракул H, который заменяет хеш-функцию (алгоритм 5).

Алгоритм 5. Симуляция подписи

Вход: $\mathbf{A}, \mathbf{T}, \mu, M, \kappa$.
Выход: $(\mathbf{z}, \mathbf{c})$.

1: $\mathbf{c} \xleftarrow{\$} \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}, \|\mathbf{v}\|_1 \leqslant \kappa\}$.
2: $\mathbf{z} \xleftarrow{\$} \mathcal{P}$.
3: С вероятностью $1/M$:
4: Установить $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.
5: Вернуть $(\mathbf{z}, \mathbf{c})$.

Важным аспектом при доказательстве является неразличимость между выходами алгоритмов $\mathcal{S}$ и реальной схемы цифровой подписи. Данное условие является обязательным, так как в противном случае $\mathcal{A}$ отличит выход симулированной от действительной схем и закончит работу, если получит симулированную подпись от $\mathcal{S}$. Неразличимость распределений выходов таких алгоритмов определяется с помощью разных статистических методов, например статистической разностью (в оригинальной работе [26]) и дивергенцией (дивергенция Реньи используется в [47]).

В [26] доказывается неразличимость распределений выходов реальной схемы и симулятора $\mathcal{S}$ при использовании в качестве $\mathcal{P}, \mathcal{Q}$ дискретных гауссовых распределений. Далее доказательство происходит следующим образом: пусть $\mathcal{A}$ с некоторой вероятностью вернул подделку подписи $(\mathbf{z}, \mathbf{c})$ для сообщения μ. Допустим, что $\mathcal{A}$ не обращался с запросом на хеш для сообщения $\mathbf{w} = \mathbf{Az} - \mathbf{Tc}$, тогда вероятность того, что он сможет вычислить $\mathbf{c} = H(\mathbf{w}, \mu)$, равна $1/|D_H|$, где D_H — область значений H. Эта вероятность пренебрежимо мала, поэтому будем считать, что $\mathcal{A}$ делал запрос на хеш. Поскольку мы знаем нужный запрос, то для него и последующих запросов заново выработаем новые ответы и запустим алгоритм $\mathcal{A}$ снова. Тогда по лемме разветвления [49] получаем другую подделку подписи $(\mathbf{z}', \mathbf{c}')$ для сообщения μ. Поскольку $\mathbf{c}, \mathbf{c}'$ являются результатами выхода $H(\mathbf{Aw}, \mu)$, а $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc}, \mu)$ и $\mathbf{c}' = H(\mathbf{Az}' - \mathbf{Tc}', \mu)$, то $\mathbf{Az} - \mathbf{Tc} = \mathbf{Az}' - \mathbf{Tc}'$. В итоге получим

$$\mathbf{A}(\mathbf{z} - \mathbf{z}' + \mathbf{Sc}' - \mathbf{Sc}) = \mathbf{0}.$$

Вектор $\mathbf{z} - \mathbf{z}' + \mathbf{Sc}' - \mathbf{Sc}$ является малым по норме, поэтому осталось доказать, что $\mathbf{z} - \mathbf{z}' + \mathbf{Sc}' - \mathbf{Sc} \neq \mathbf{0}$. Пусть i — это позиция, в которой различаются векторы $\mathbf{c}$ и $\mathbf{c}'$. По лемме из [26] с вероятностью $1 - 2^{-100}$ существует секретный ключ $\mathbf{S}'$, отличающийся от $\mathbf{S}$ только в i-м столбце, и $\mathbf{AS} = \mathbf{AS}'$. Тогда если $\mathbf{z} - \mathbf{z}' + \mathbf{Sc}' - \mathbf{Sc} = \mathbf{0}$, то существует такой $\mathbf{S}'$, что $\mathbf{z} - \mathbf{z}' + \mathbf{S}'\mathbf{c}' - \mathbf{S}'\mathbf{c} \neq \mathbf{0}$. Поскольку $\mathcal{A}$ неизвестен секретный ключ $\mathbf{S}$, мы получим решение SIS с вероятностью $1/2$. Таким образом, задача подделки подписи не легче, чем задача SIS.

Другой задачей для злоумышленника является восстановление ключа. Он может это сделать двумя способами: 1) либо решить уравнение $\mathbf{AS} = \mathbf{T} \pmod q$, 2) либо найти секретный ключ из значений $(\mathbf{z}, \mathbf{c})$, где $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$. Первый вариант является сложным, так как требует решения задачи ISIS (Inhomogeneous SIS), которая не легче SIS, второй вариант также сложный, так как требует решения задачи Search-LWE.

Выбор распределений $\mathcal{P}$ и $\mathcal{Q}$. Этот выбор важен с точки зрения компактности получаемой подписи, а также сложности программной реализации схемы. Так, например, в выбранной в качестве стандарта схеме цифровой подписи Dilithium [45] используется равномерное распределение в гиперкубе. Благодаря этому достигается простота программной реализации схемы цифровой подписи и стойкость к атакам по побочным каналам. В работах [26, 50] используется дискретное гауссово распределение, из-за чего размеры подписи удаётся уменьшить, однако ввиду сложности реализации гауссова распределения это приводит к раскрытию секретной информации при использовании атак по побочным каналам [51–53]. Кроме того, гауссово распределение не ограничено и требуется делать дополнительные проверки, что подпись корректна. Новым подходом стало использование равномерного распределения в n-мерном гипершаре для генерации подписи [54]. В [54] показано, что ожидаемый размер подписи, полученной при выборке из гипершара, равен ожидаемому размеру подписи, полученной из дискретного гауссова распределения, однако первая выборка реализуется проще, чем вторая. С применением результатов [54] была разработана и предложена для рассмотрения на новом этапе конкурса NIST схема цифровой подписи HAETAE [47]. Благодаря использованию равномерного распределения из гипершара, в HAETAE удалось уменьшить размер подписи на 29–39 % и размер открытого ключа на 20–25 % в зависимости от уровня стойкости.

4.2. Схема цифровой подписи Dilithium

Схема цифровой подписи Dilithium выбрана Национальным институтом стандартов и технологий США в качестве одного из новых стандартов цифровой подписи [46]. Схема базируется на работах В. Любашевского, но использует модульные решётки. Таким образом, стойкость схемы основывается на модульных вариантах задач SIS и LWE [12, 55]. В схеме Dilithium для выработки секретных и маскирующих векторов используется равномерное распределение в гиперкубе. Такой выбор распределения обусловлен простотой реализации и защитой от атак по побочным каналам. Опишем упрощённую (менее эффективную) версию, являющуюся несколько изменённой версией схемы из [56].

Генерация ключей (алгоритм 6). Алгоритм генерирует матрицу $\mathbf{A}$ размера $k \times \ell$, являющуюся открытой; каждый её элемент является многочленом в кольце $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$, где $q = 2^{23} - 2^{13} + 1$, $n = 256$. После этого случайным образом выбираются секретные векторы $\mathbf{s}_1$ и $\mathbf{s}_2$. Каждая компонента этих векторов является элементом из множества S_η, то есть элементом $\mathcal{R}_q$ малой нормы, не превышающей η. В качестве нормы элемента $\mathcal{R}_q$ вычисляется его бесконечная норма $\|a\|_\infty = \max\limits_i |a_i|$, где $a \in \mathcal{R}_q$, то есть каждый коэффициент выбираемого многочлена лежит в $\{-\eta, \ldots, \eta\}$. Множество S_η^k называется k-мерным гиперкубом, а векторы выбираются из него случайно и равномерно. Вторая часть открытого ключа равна $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. Все алгебраические операции выполняются над кольцом $\mathcal{R}_q$ (алгоритм 6).

Алгоритм 6. Выработка ключей Dilithium

1: $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}$.
2: $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k$.
3: $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$.
4: **Вернуть** $(\mathbf{A}, \mathbf{t})$ — открытый ключ, $(\mathbf{s}_1, \mathbf{s}_2)$ — секретный ключ.

Формирование подписи (алгоритм 7). Подписывающий генерирует маскирующий вектор $\mathbf{y}$, состоящий из многочленов из $\mathcal{R}_q$ с коэффициентами меньше γ_1. Параметр γ_1 достаточно велик, чтобы конечная подпись не раскрывала секретный ключ (т. е. алгоритм формирования подписи основан на нулевом разглашении), но в то же время достаточно мал, чтобы подпись было сложно подделать. Затем подписывающий вычисляет $\mathbf{Ay}$ и берёт в качестве $\mathbf{w}_1$ старшие биты компонентов вычисленного вектора. В частности, каждая компонента w в $\mathbf{Ay}$ может быть записана в каноническом виде $w = w_1 \cdot 2\gamma_2 + w_0$, где $|w_0| \leqslant \gamma_2$. Таким образом, $\mathbf{w}_1$ — это вектор, содержащий все значения w_1. Вызов c задаётся как хеш сообщения M и вектора $\mathbf{w}_1$ и представляет собой многочлен из $\mathcal{R}_q$ с элементами ± 1 в количестве τ и остальными элементами, равными нулю. Причина такого распределения заключается в том, что c имеет малую норму и является элементом кольца размера $\log_2 \binom{256}{\tau} + \tau$, находящегося в диапазоне от 128 до 256. Сама подпись вычисляется как $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$.

Для избежания зависимости $\mathbf{z}$ от секретного ключа используется выборка отбраковки. Параметр β задан как максимально возможный коэффициент $c\mathbf{s}_i$. Поскольку количество ± 1 в c равно τ, а максимальный коэффициент в $\mathbf{s}_i$ равен η, очевидно, что $\beta \leqslant \tau \cdot \eta$. Если какой-либо коэффициент $\mathbf{z}$ больше, чем $\gamma_1 - \beta$, то подпись отклоняется и процедура генерации подписи запускается сначала. Кроме того, если младшие биты какого-либо коэффициента вектора $\mathbf{Az} - c\mathbf{t}$ больше, чем $\gamma_2 - \beta$, процедура генерации подписи перезапускается. Первая проверка необходима для обеспечения стойкости, вторая — не только для стойкости, но и для корректности.

Алгоритм 7. Формирование подписи Dilithium

Вход: Сообщение M, открытый ключ $(\mathbf{A}, \mathbf{t})$, секретный ключ $(\mathbf{s}_1, \mathbf{s}_2)$.
Выход: Подпись $\sigma = (\mathbf{z}, c)$ сообщения M.

1: $\mathbf{z} := \perp$.
2: **Пока** $\mathbf{z} = \perp$:
3: $\quad \mathbf{y} \leftarrow S^{\ell}_{\gamma_1 - 1}$;
4: $\quad \mathbf{w}_1 := \mathsf{HighBits}(\mathbf{Ay}, 2\gamma_2)$;
5: $\quad c(\in B_\tau) := H(M \| \mathbf{w}_1)$;
6: $\quad \mathbf{z} := \mathbf{y} + c\mathbf{s}_1$.
7: $\quad$ **Если** $\|\mathbf{z}\|_\infty \geqslant \gamma_1 - \beta$ или $\|\mathsf{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geqslant \gamma_2 - \beta$, **то** $\mathbf{z} := \perp$.
8: **Вернуть** $\sigma = (\mathbf{z}, c)$.

Проверка (алгоритм 8). Проверяющий сначала вычисляет $\mathbf{w}'_1$ в качестве старших битов вектора $\mathbf{Az} - c\mathbf{t}$, а затем принимает подпись, если все коэффициенты $\mathbf{z}$ меньше $(\gamma_1 - \beta)$ и если c является хешем сообщения M и вектора $\mathbf{w}'_1$. Отметим, что $\mathbf{Az} - c\mathbf{t} = \mathbf{Ay} - c\mathbf{s}_2$. Кроме того, $\mathsf{HighBits}(\mathbf{Ay}, 2\gamma_2) = \mathsf{HighBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)$. Это верно потому, что действительная подпись содержит $\|\mathsf{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta$. Поскольку мы знаем, что коэффициенты $c\mathbf{s}_2$ меньше β, добавления $c\mathbf{s}_2$ недостаточно, чтобы вызвать какие-либо переносы путём увеличения любого коэффициента младшего порядка до величины не менее γ_2.

Алгоритм 8. Проверка подписи Dilithium

Вход: Сообщение M, открытый ключ $(\mathbf{A}, \mathbf{t})$, подпись $\sigma = (\mathbf{z}, c)$.
Выход: Принять или отклонить подпись.
1: $\mathbf{w}'_1 := \mathsf{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$.
2: **Если** $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ и $c = H(M\|\mathbf{w}'_1)$, **то**
принять подпись.

Безопасность. Наследуя идею доказательства из [26], стойкость схемы Dilithium доказывается похожим образом в модели случайного оракула, основываясь на сложности двух задач. Первая задача — это задача распознавания LWE над кольцами многочленов, в рамках которой требуется различить экземпляр задачи LWE $(\mathbf{A}, \mathbf{t} = \mathbf{As_1} + \mathbf{s_2})$ и пару $(\mathbf{A}, \mathbf{u})$, где $\mathbf{u}$ выбран случайно. Второй является задача SelfTargetMSIS [57], в рамках которой требуется найти короткий вектор $\begin{bmatrix} \mathbf{z} \\ c \\ \mathbf{v} \end{bmatrix}$ и сообщение μ, удовлетворяющие равенству

$$H\left(\mu \| [\mathbf{A}|\mathbf{t}|\mathbf{I}] \cdot \begin{bmatrix} \mathbf{z} \\ c \\ \mathbf{v} \end{bmatrix}\right) = c.$$

Для модели случайного оракула можно также произвести сведение, используя лемму о разветвлении [49], от задачи MSIS к задаче SelfTargetMSIS, сводя тем самым стойкость схемы Dilithium к сложности задач MSIS и MLWE. Однако в модели QROM мы не можем воспользоваться данной леммой, поэтому сложность подделки подписи основана на задачах MLWE и SelfTargetMSIS. Обоснование сложности задачи и соотвественно стойкости схемы опирается на следующие причины:

1) на данный момент не существует схем цифровой подписи, построенных с помощью преобразования Фиата — Шамира из сигма-протокола, которые являются стойкими в модели ROM и нестойкими в модели QROM;
2) есть возможность выбрать такой набор параметров, при котором задача SelfTargetMSIS станет информационно-теоретически сложной и стойкость схемы будет основана только на задаче MLWE. Но при таком наборе параметров размеры подписи и открытого ключа увеличиваются в несколько раз [57].

Однако в недавней работе [58] показано, что если сигма-протокол коллапсирует и имеет свойство special soundness, то схема цифровой подписи, полученная с помощью преобразования Фиата — Шамира, будет стойкой в модели QROM. Свойство special soundness сигма-протокола Dilithium обеспечивается сложностью задачи MSIS, а в работах [58, 59] показано, что сигма-протокол коллапсирует. Таким образом, можно считать, что безопасность схемы для модели QROM обоснована.

4.3. Схема цифровой подписи HAETAE

HAETAE — схема постквантовой цифровой подписи на решётках, представленная на дополнительном конкурсе NIST. HAETAE также построена на парадигме Фиата — Шамира с прерываниями [10, 26], а её стойкость основана на сложности решения модульных версий задач SIS и LWE [12, 55]. Несмотря на то, что HAETAE частично похожа на схему CRYSTALS-Dilithium, она имеет следующие основные отличия:

1) вместо унимодального распределения для генерации выборки отбраковки HAETAE использует бимодальное распределение, как в схеме цифровой подписи BLISS [50];

2) выборки случайных векторов и векторов отбраковки осуществляются с помощью использования многомерного гипершара, а не гиперкуба.

Опишем некоторые предварительные сведения, необходимые для понимания работы схемы HAETAE.

Улучшение компактности. Выбор распределений для выборки случайных векторов и векторов отбраковки существенно влияет на размер подписи [54]. Dilithium использует дискретные равномерные распределения в гиперкубах, что упрощает реализацию схемы. Однако такие распределения далеки от оптимальных с точки зрения результирующих размеров подписей. В HAETAE существует компромисс: немного теряя в простоте реализации, получаются более компактные подписи.

Равномерное распределение на гипершаре. Гауссово распределение превосходит равномерное распределение на гипершаре с точки зрения компактности подписи [54]. Однако у гауссова распределения есть два недостатка:

— шаг отбраковки включает вычисление некоторой трансцендентной функции на входных данных, которая зависит от секретного ключа. Это является громоздким для реализации и чувствительным к атакам по побочным каналам [52];
— поскольку итоговая подпись связана с гауссовым распределением, существует ненулевая вероятность, что подпись будет слишком большой и не пройдёт проверку.

Как следствие, в качестве альтернативного выбора в [54] представлены равномерные распределения на гипершарах, приводящие к более компактным подписям, нежели гауссовы распределения и равномерные распределения на гиперкубе. Таким образом, шаг отбраковки заключается всего лишь в проверке того, достаточно ли малы евклидовы нормы рассматриваемых векторов.

Бимодальное распределение. Модификация схемы цифровой подписи Любашевского [10, 26] представлена в [50] и использует бимодальное распределение при генерации самой подписи. Таким образом, подпись имеет вид $\mathbf{y}+(-1)^b\mathbf{Sc}$, где вектор $\mathbf{y}$ выбран из фиксированного распределения, а величина $b \in \{0,1\}$ выбрана равномерно. Подпись отклоняется для заданного целевого распределения, не зависящего от секрета. Чтобы убедиться, что проверка прошла успешно, все вычисления выполняются по модулю $2q$, а генерация ключа приводит к равенству $\mathbf{AS} = q\mathbf{Id}$. Оказывается, что эта модификация может привести к более компактным подписям, чем унимодальный случай.

Выбор кольца и модуля. Оригинальный дизайн схемы цифровой подписи BLISS [50] основан на задачах Ring-LWE и Ring-SIS, а алгоритм генерации ключей — на соотношениях полиномов по аналогии с NTRU. Для обеспечения бóльшей гибкости без потери эффективности реализации в HAETAE рассматриваются модульные решётки (как в Dilithium) с фиксированным кольцом многочленов $\mathcal{R} = \mathbb{Z}[x]/(x^{256}+1)$ для всех уровней стойкости.

В качестве модуля q выбирается простое число, такое, что операции в кольце $\mathcal{R}$ могут быть эффективно реализованы, а алгоритм декомпозиции битов работает правильно. Для операций в кольце $\mathcal{R}$ используется теоретико-числовое преобразование (NTT), а мультипликативная группа $\mathbb{Z}_q^*$ имеет элемент порядка 512, что эквивалентно условию $q \equiv 1 \pmod{512}$. В HAETAE $q = 64513$.

Генерация ключей. Ключевой парой схемы HAETAE является $(\mathbf{A}, \mathbf{s})$, где $\mathbf{A} \in \mathcal{R}_p^{k\times(k+l)}$ — открытый ключ; $\mathbf{s} \in \mathcal{R}_p^{k+l}$ — секретный, причём $\mathbf{As} = -\mathbf{As} \pmod p$. Для генерации такой пары полагают $p = 2q$ и стремятся к выполнению условия $\mathbf{As} = q\mathbf{j} \pmod{2q}$, где $\mathbf{j} = (1,0,0,\ldots,0)^{\mathrm{T}}$.

Матрица $\mathbf{A}$ и вектор $\mathbf{s}$ получаются следующим образом: генерируется MLWE-выборка $\mathbf{b}-\mathbf{a} = \mathbf{A}_0\mathbf{s}_0+\mathbf{e}_0 \bmod q$, где $\mathbf{A}_0 \leftarrow U(\mathcal{R}_q^{k\times(l-1)})$; $\mathbf{a} \leftarrow U(\mathcal{R}_q^k)$; $(\mathbf{s}_0, \mathbf{e}_0) \leftarrow U(S_\eta^{l-1}\times S_\eta^k)$. Для любого $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_0$ определим $\mathbf{A} = (2(\mathbf{a} - \mathbf{b}_1) + q\mathbf{j}|2\mathbf{A}_0|2\mathbf{I}_k)$ и $\mathbf{s} = (1|\mathbf{s}_0|\mathbf{e}_0 - \mathbf{b}_0)$.

Для разложения nk-мерного вектора $\mathbf{b}$ с координатами в $[0, q-1]$ выбираем $\mathbf{b}_0$ с координатами в $\{-1, 0, 1\}$ таким образом, что если координата $\mathbf{b}$ нечётная, то она округляется до ближайшего значения, кратного 4. Далее можем записать $\mathbf{b} = \mathbf{b}_0 + 2\mathbf{b}_1$, где $\mathbf{b}_1$ кодируется с использованием $\lceil \log_2 q - 1 \rceil$ бит на координату и $\mathbf{b}_0 = (-1)^{\lceil \mathbf{b}/2 \rceil \bmod 2}\mathbf{b} \bmod 2$.

Критическим шагом в алгоритме генерации ключей является ограничение нормы $\|\mathbf{s}c\|_2$, где $\mathbf{s}$ генерируется, как и ранее, а $c \in \mathcal{R}$ — многочлен с коэффициентами в $\{0, 1\}$ и имеет $\leqslant \tau$ ненулевых коэффициентов для некоторого τ. Чем ниже эта граница, тем меньше подпись, что, в свою очередь, приводит к усложнению подделки.

Формирование и проверка подписи (алгоритмы 9 и 10). Подпись сообщения M состоит из двух компонент: $c = H(\mathbf{A}\lfloor \mathbf{y} \rceil \bmod 2q, M)$ и $\mathbf{z} = \lfloor \mathbf{y} \rceil \pm c\mathbf{s}$. Иногда вектор $\mathbf{z}$ отклоняется и процесс подписания выполняется снова. Отметим, что $\mathbf{Az} = \mathbf{A}\lfloor \mathbf{y} \rceil + qc\mathbf{j} \bmod 2q$ не зависит от знака, выбранного для $c\mathbf{s}$. На этапе верификации подписи проверяется согласованность пары $(\mathbf{z}, c)$ и малый размер нормы вектора $\mathbf{z}$.

Алгоритм 9. Формирование подписи

Вход: Секретный ключ $(\mathbf{A}, \mathbf{s})$, сообщение M.
Выход: $\sigma = (\lfloor \mathbf{z} \rceil, c)$.
1: $\mathbf{y} \leftarrow U(\mathscr{B}_{(1/N)\mathcal{R},(k+\ell)}(B))$.
2: $\mathbf{w} \leftarrow \mathbf{A}\lfloor \mathbf{y} \rceil$.
3: $c := H(\mathbf{w}, M) \in \mathcal{R}_2$.
4: $\mathbf{z} := y + (-1)^b c\mathbf{s}$ для $b \leftarrow U(\{0, 1\})$.
5: **Если** $\|\mathbf{z}\|_2 > B'$, **то**
начать сначала,
6: **иначе**
7: **Если** $\|2\mathbf{z} - \mathbf{y}\|_2 < B$, **то**
начать сначала с вероятностью $1/2$.
8: **Вернуть** $\sigma = (\lfloor \mathbf{z} \rceil, c)$.

Алгоритм 10. Проверка подписи

Вход: Открытый ключ $\mathbf{A}$, сообщение M, подпись $\sigma = (\tilde{\mathbf{z}}, c)$.
Выход: $\sigma = (\lfloor \mathbf{z} \rceil, c)$.
1: $\tilde{\mathbf{w}} := \mathbf{Az} - qc\mathbf{j} \bmod 2q$.
2: **Вернуть** $c = H(\tilde{\mathbf{w}}, M)$ и $\|\tilde{\mathbf{z}}\| < B + \sqrt{n(k+\ell)}/2$.

Безопасность. Стойкость схемы HAETAE основывается на сложности задач MLWE и BimodalSelfTargetMSIS [47]. Суть задачи BimodalSelfTargetMSIS состоит в поиске короткого вектора $\mathbf{y}$, значения c и сообщения μ, таких, что $H((\mathbf{Ay} - qc\mathbf{j}) \bmod 2q, \mu) = c$, где матрица $\mathbf{A} = (2\mathbf{b} + q\mathbf{j} \mid 2\mathbf{A}_0 \mid 2\mathbf{I}_k) \bmod 2q$. В [47] представлено сведение задачи MSIS к задаче BimodalSelfTargetMSIS, подобно сведению MSIS к SelfTargetMSIS [45]. Доказательство использует идеи из [60] и сводит стойкость в модели UFCMA к стойкости в модели UFNMA. Для этого требуется, чтобы соответствующая схеме цифровой подписи протокол аутентификации сторон обладал высокой мин-энтропией обязательств

(commitment min-entropy), а также свойством нулевого разглашения честного проверяющего (honest-verifier zero-knowledge), что доказывается для протокола аутентификации сторон, построенного на основе HAETAE.

5. Оценка стойкости и подбор параметров

Обоснование стойкости криптографических схем состоит из двух частей: теоретического доказательства стойкости схемы и её практического криптоанализа. Теоретическое доказательство необходимо для обоснования сложности взлома схемы. Так, в ходе доказательства производится сведение задачи взлома к вычислительно сложной математической задаче. Практический анализ стойкости схемы представляет собой оценку её стойкости к ряду самых успешных существующих на данный момент атак, исходя из которой подбираются параметры, обеспечивающие заданный уровень безопасности. Так как стойкость рассмотренных схем цифровой подписи основывается на сложности задач SIS и LWE в различных формулировках, одним из главных видов атак на такие схемы являются атаки, направленные на поиск коротких векторов в решётке.

На сегодняшний день лучшим алгоритмом редукции базиса и нахождения коротких векторов является BKZ [61]. Алгоритм BKZ на вход принимает базис решётки, а также параметр b (размер блока) и на выходе возвращает редуцированный базис. В рамках работы алгоритма происходит обращение к оракулу, решающему задачу SVP для решётки размерности b. Количество обращений к оракулу полиномиально, поэтому сложность алгоритма BKZ зависит от сложности алгоритма, решающего SVP и используемого оракулом. Отсюда следует, что в рамках оценки стойкости схемы рассматривается сложность не всего алгоритма BKZ, а только одного вызова алгоритма, решающего SVP. Очевидно, что при увеличении значения b увеличивается время работы алгоритма и уменьшается предполагаемая норма получаемого кратчайшего вектора на его выходе. Таким образом, для оценки стойкости схемы необходимо сначала определить размер блока b, при котором можно будет схему взломать, а далее оценить сложность алгоритма решения SVP для такой размерности, эта сложность и является уровнем стойкости схемы. Такой подход к оценке практической стойкости криптографических схем на решётках предложен в работе [62] и называется методологией coreSVP hardness.

В настоящее время сложность лучшего классического алгоритма [63] решения задачи SVP равна $\approx 2^{c_C \cdot b}$, где $c_C = \log_2 \sqrt{3/2} \approx 0{,}292$. Лучший квантовый алгоритм [64] решает SVP за время $\approx 2^{c_Q \cdot b}$, где $c_Q = \log_2 \sqrt{13/9} \approx 0{,}265$. Также считается, что даже с улучшениями вряд ли сложность квантового алгоритма решения SVP сможет быть меньше $\sqrt{4/3}^{b+o(b)} \approx 2^{0{,}2075\, b}$.

Теперь опишем подход, с помощью которого реализуется атака на экземпляр задачи LWE $(\mathbf{A}, \mathbf{t})$, где $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$; $\mathbf{A} \in \mathbb{Z}^{n \times m}$; $\mathbf{s} \in \mathbb{Z}^m$; $\mathbf{t}, \mathbf{e} \in \mathbb{Z}^n$. Для LWE существуют два основных вида атак — это атаки на основную решётку и дуальную к ней. Для реализации первой атаки построим следующую решётку:

$$\Lambda = \{\mathbf{x} \in \mathbb{Z}^{m+n+1} : (\mathbf{A}| - \mathbf{I}_m| - \mathbf{t})\mathbf{x} = \mathbf{0} \pmod q\}.$$

Значение $d = m + n + 1$ является размерностью решётки, q^m — её объёмом. Для данной решётки кратчайшим вектором является вектор $\mathbf{v} = (\mathbf{s}, \mathbf{e}, 1)$. Для случайных решёток $\mathcal{L}$ существует гауссова эвристика — это некоторое предположение, полученное эвристическим методом, которое примерно оценивает $\lambda_1(\mathcal{L})$ в случайных решётках,

и данная эвристика равна $\lambda_1(\mathcal{L}) \approx \sqrt{\frac{d}{2\pi e}} \cdot \mathrm{Vol}(\Lambda)^{1/d}$, где d — размерность решётки. Поскольку для решётки Λ значение $\lambda_1(\Lambda) = \|\mathbf{v}\|$ значительно меньше, чем эвристика случайной решётки, которой равен второй минимум $\lambda_2(\Lambda)$, перед нами лежит экземпляр задачи unique-SVP. Для решения задачи и оценки сложности алгоритма BKZ опишем некоторые предположения и определения, на которых строится оценка.

Предположение о геометрической прогрессии [65]. Длины векторов Грама — Шмидта после редукции базиса решётки удовлетворяют следующему соотношению:

$$\|\mathbf{b}_i^*\| = \alpha^{i-1} \cdot \|\mathbf{b}_1\|, \quad 0 < \alpha < 1.$$

Определение 9 (корневой эрмитовый фактор [66]). Корневой эрмитовый фактор базиса $\mathbf{B}$ решётки $\mathcal{L}$ определяется как

$$\delta(\mathbf{B}) = \left(\frac{\|\mathbf{b}_1\|}{\mathrm{Vol}(\mathcal{L})^{1/d}} \right)^{1/d}.$$

Для BKZ-β редуцированного базиса случайной решётки, согласно гауссовой эвристике, выполняется следующее соотношение [66]:

$$\lim_{d\to\infty} \delta(\mathbf{B}) = \left(\frac{\beta}{2\pi e} (\pi\beta)^{1/\beta} \right)^{1/(2(\beta-1))}.$$

Пусть $\delta_0 = \left(\frac{\beta}{2\pi e} (\pi\beta)^{1/\beta} \right)^{1/(2(\beta-1))}$. Выразим значение $\|\mathbf{b}_1\|$ как $\|\mathbf{b}_1\| = \delta_0^d \cdot \mathrm{Vol}(\mathcal{L})^{1/d}$. Объединяя предположение о геометрической прогрессии, значение корневого эрмитового фактора и выражение $\mathrm{Vol}(\mathcal{L}) = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|$, получаем значение $\alpha = \delta_0^{-2d/(d-1)} \approx \delta_0^{-2}$. Таким образом, имеем следующую зависимость: $\|\mathbf{b}_i^*\| = \delta_0^{2-2i} \cdot \mathrm{Vol}(\mathcal{L})^{1/d}$.

Для редуцированного базиса мы считаем, что выполняется предположение о геометрической прогрессии, поэтому можем предсказать предполагаемое значение $\|\mathbf{b}_i^*\|$ после запуска на базисной матрице алгоритма BKZ с параметром блока β. Так как значение длины проекций векторов уменьшается к последним векторам матрицы, возьмём последний блок $(\mathbf{b}_{d-\beta}, \ldots, \mathbf{b}_d)$: в данном блоке оракул SVP найдёт вектор с проекцией, примерно равной $\|\mathbf{b}_{d-\beta+1}^*\| \approx \delta_0^{2\beta-d} \cdot \mathrm{Vol}(\Lambda)^{1/d}$. Однако если проекция искомого вектора $\mathbf{v}$ к первым $d-\beta$ векторам окажется меньше, то оракул непременно её найдёт. Таким образом, для успешного нахождения вектора $\mathbf{v}$ требуется, чтобы

$$\|\mathbf{v}^*\| \leqslant \delta_0^{2\beta-d} \cdot \mathrm{Vol}(\Lambda)^{1/d} = \delta_0^{2\beta-d} \cdot q^{m/d}. \tag{2}$$

Найдя наименьшее значение β, такое, чтобы выполнялось (2), можем оценить сложность алгоритма SVP: $2^{0{,}292\,\beta}$ и $2^{0{,}265\,\beta}$ для классического и квантового вычислителя соответственно.

Дуальная атака на LWE [62] состоит из поиска короткого вектора в дуальной решётке $\Lambda' = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^m \times \mathbb{Z}^n : \mathbf{A}^T\mathbf{x} = \mathbf{y} \pmod q\}$. Найдя короткий вектор $(\mathbf{w}, \mathbf{v}) \in \Lambda'$ длины l, можем вычислить значение $\langle \mathbf{w}, \mathbf{b} \rangle = \langle \mathbf{v}, \mathbf{s} \rangle + \langle \mathbf{w}, \mathbf{e} \rangle$, которое тоже является малым значением и распределено согласно гауссовому распределению со стандартным отклонением $l\sigma$, если пара $(\mathbf{A}, \mathbf{b})$ является экземпляром задачи LWE, в противном случае распределение равномерное. Расстояние между двумя такими распределениями ограничено значением $\varepsilon = 4\exp(-2\pi^2\tau^2)$, где $\tau = l\sigma/q$. Нахождение такого вектора

даёт ε-преимущество для решения задачи Decision-LWE. Пусть $l = \|\mathbf{b}_1\|$ после применения BKZ. Зная, что Λ' имеет размерность $d = m + n$ и объём q^n, получаем значение $l = \delta^{d-1} q^{n/d}$. Таким образом, для получения преимущества ε требуется запустить алгоритм BKZ с блоком β, таким, что

$$-2\pi^2\tau^2 \geqslant \ln(\varepsilon/4).$$

Методология coreSVP hardness используется для анализа безопасности и подбора параметров для большинства существующих схем цифровой подписи на решетках, в частности, с её помощью подобраны параметры для схем Dilithium, HAETAE и Falcon. В рамках оценки стойкости анализировались две атаки: на подделку подписи и на восстановление секретного ключа. Для каждой атаки подобрано значение блока β, для которого соблюдается уровень стойкости, определённый Национальным институтом стандартов и технологий США. Значения блоков и соответствующие им уровни стойкости схем в битах как против классического, так и против квантового злоумышленника представлены в табл. 4.

Таблица 4

Значения блоков BKZ и оценки стойкости схем цифровой подписи Falcon, Dilithium, HAETAE

Схема	Восстановление ключа			Подделка подписи		
	β	Классическая сложность	Квантовая сложность	β	Классическая сложность	Квантовая сложность
Уровень стойкости I (NIST)						
Falcon	458	133	121	411	120	108
Уровень стойкости II (NIST)						
Dilithium	423	123	112	423	123	112
HAETAE	428	125	109	409	119	105
Уровень стойкости III (NIST)						
Dilithium	638	186	169	624	182	165
HAETAE	810	236	208	617	180	158
Уровень стойкости V (NIST)						
Falcon	936	273	248	952	277	252
Dilithium	909	265	241	863	252	229
HAETAE	988	288	253	878	256	225

Производительность схем, а также размеры открытого и секретного ключей и подписи также зависят от требуемого уровня стойкости и для схем Dilithium, Falcon и HAETAE представлены в табл. 5. Сравнивая схемы по разным показателям эффективности, нельзя определённо сказать, какая из них лучшая, так как каждая имеет свои преимущества и недостатки.

Таблица 5

Показатели эффективности схем цифровой подписи Falcon, Dilithium, HAETAE

Схема	Открытый ключ (байты)	Секретный ключ (байты)	Подпись (байты)	Формирование подписи (циклы)	Проверка подписи (циклы)
Уровень стойкости I (NIST)					
Falcon	897	1 281	666	1 009 764	81 036
Уровень стойкости II (NIST)					
Dilithium	1 312	2 528	2 420	333 013	118 412
HAETAE	992	1 376	1 463	6 253 166	387 594
Уровень стойкости III (NIST)					
Dilithium	1 952	4 000	3 293	529 106	179 424
HAETAE	1 472	2 080	2 337	9 472 724	718 010
Уровень стойкости V (NIST)					
Falcon	1 793	2 305	1 280	2 053 080	160 596
Dilithium	2 592	4 864	4 595	642 192	279 936
HAETAE	2 080	2 720	2 908	8 989 980	913 378

Заключение

Рассмотрев существующие подходы к построению схем цифровой подписи на решётках, а также сравнив количественные показатели флагманских схем, приходим к выводу, что лучшего подхода на данный момент не существует, а каждый из существующих обладает своими преимуществами и недостатками. Парадигма GGH/NTRUSign сейчас отошла на второй план ввиду того, что главная схема NTRUSign оказалась небезопасной [19], а новые модификации проигрывают в эффективности с точки зрения размеров ключей и подписи другим схемам на решётках.

Схемы цифровой подписи, построенные на двух других парадигмах, оказались более успешными, и две из них — Falcon и Dilithium — выбраны в качестве нового стандарта цифровой подписи США [46]. Анализируя оба подхода, нельзя прийти к однозначному мнению, что один из них лучше, чем другой, а сравнивая показатели эффективности схем — представителей данных парадигм, можно сделать вывод, что выбор схемы зависит от условий её использования. Так, схема цифровой подписи Falcon обладает достаточно малым размером подписи и малым временем её проверки, однако время создания подписи в несколько раз выше, чем у схемы Dilithium. Важным аспектом является также факт, что при формировании подписи производятся операции с числами с плавающей точкой, данное свойство не позволяет использовать Falcon на системах, не поддерживающих такие операции, а также ведёт к утечке информации по побочным каналам; при проверке подписи операции с числами с плавающей точкой не производятся. Использование схемы Falcon вместе с математическим сопроцессором сильно ускоряет алгоритм формирования подписи: общее время работы алгоритмов создания подписи и её проверки становится меньше, чем у схемы Dilithium [67], однако авторы в [67] замечают, что при использовании математического сопроцессора операции с числами с плавающей точкой всё равно не являются константными по времени, что может потенциально привести к атакам по побочным каналам. Исходя из особенностей схемы Falcon, рекомендуется использовать её в случае, когда клиенту требуется только хранить и проверять подпись, в то время как формирование подписи можно производить на производительном сервере [67–69]; также рекомендуется делать это в оффлайн-режиме [70] в целях недопущения атак по побочным каналам.

Схема цифровой подписи Dilithium является более универсальной, по сравнению со схемой Falcon она обладает рядом преимуществ: во-первых, Dilithium не использует

арифметику с плавающей точкой и дизайн схемы позволяет применять маскирование и выполнение операций за константное время [45, 69], что обеспечивает определённый уровень защиты от атак по побочным каналам; во-вторых, схему Dilithium легче программно реализовать [45, 69] безопасным образом, то есть обеспечив защиту от атак по побочным каналам, а также легче отслеживать ошибки реализации. Главное преимущество Dilithium — это баланс между размерами ключей и подписей и временем работы алгоритмов, что позволяет схеме быть универсальной. В Falcon, а также во многих других постквантовых схемах, в отличие от Dilithium, упор делается на один из показателей в ущерб другим, что сужает круг ситуаций, в которых целесообразно использовать данную схему. Сейчас схема Dilithium начинает активно внедряться в различные информационные системы в качестве альтернативы классическим схемам цифровой подписи, так, например, исследователи компании Google предлагают использовать Dilithium и SPHINCS+ в качестве основных схем цифровой подписи [71].

Схема HAETAE похожа на Dilithium, однако в силу нововведений вырабатывает подписи меньших размеров, чем Dilithium. Программная реализация схемы требует значительных доработок в целях повышения скорости выполнения алгоритмов, однако без учёта производительности данная схема выглядит перспективно и может в будущем заменить Dilithium.

ЛИТЕРАТУРА

1. *Shor P. W.* Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer // SIAM Rev. 1995. No. 41. P. 303–332.
2. *Ajtai M., Kumar R., and Sivakumar D.* A sieve algorithm for the shortest lattice vector problem // Proc. STOC'01. Hersonissos, Greece, 2001. P. 601–610.
3. *Dinur I., Kindler G., and Safra S.* Approximating CVP to within almost-polynomial factors is NP-hard // Combinatorica. 2003. V. 23. P. 205–243.
4. *Roy S. S., Vercauteren F., Mentens N., et al.* Compact ring-LWE cryptoprocessor // LNCS. 2014. V. 8731. P. 371–391.
5. *Ajtai M.* Generating hard instances of lattice problems // Proc. STOC'96. Philadelphia, Pennsylvania, USA, 1996. P. 99–108.
6. *Regev O.* On lattices, learning with errors, random linear codes, and cryptography // Proc. STOC'05. Baltimore, MD, USA. 2005. P. 84–93.
7. *Micciancio D.* Generalized compact knapsacks, cyclic lattices, and efficient one-way functions // Comput. Complexity. 2007. No. 16(4). P. 365–411.
8. *Peikert C. and Rosen A.* Lattices that admit logarithmic worst-case to average-case connection factors // Proc. STOC'07. San Diego, California, USA. 2007. P. 478–487.
9. *Lyubashevsky V. and Micciancio D.* Generalized compact knapsacks are collision resistant // LNCS. 2006. V. 4052. P. 144–155.
10. *Lyubashevsky V.* Fiat — Shamir with aborts: Applications to lattice and factoring-based signatures // LNCS. 2009. V. 5912. P. 598–616.
11. *Lyubashevsky V., Peikert C., and Regev O.* On ideal lattices and learning with errors over rings // J. ACM. 2010. V. 60. P. 43:1–43:35.
12. *Langlois A. and Stehlé D.* Worst-case to average-case reductions for module lattices // Des. Codes Cryptography. 2015. V. 75. P. 565–599.
13. *Goldreich O., Goldwasser S., and Halevi S.* Public-key cryptosystems from lattice reduction problems // LNCS. 1997. V. 1294. P. 112–131.
14. *Hoffstein J., Pipher J., and Silverman J. H.* NTRU: A ring-based public key cryptosystem // LNCS. 1998. V. 1423. P. 267–288.

15. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* NTRUSign: Digital signatures using the NTRU lattice // LNCS. 2003. V. 2612. P. 122–140.
16. *Hoffstein J., Pipher J., and Silverman J. H.* NSS: An NTRU lattice-based signature scheme // LNCS. 2001. V. 2045. P. 211–228.
17. *Gentry C., Jonsson J., Stern J., and Szydlo M.* Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001 // LNCS. 2001. V. 2248. P. 1–20.
18. *Gentry C. and Szydlo M.* Cryptanalysis of the revised NTRU signature scheme // LNCS. 2002. V. 2332. P. 299–320.
19. *Nguyen P. Q. and Regev O.* Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures // J. Cryptology. 2009. No. 22(2). P. 139–160.
20. *Ducas L. and Nguyen P. Q.* Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures // LNCS. 2012. V. 7658. P. 433–450.
21. *Melchor C. A., Boyen X., Deneuville J.-C., and Gaborit P.* Sealing the leak on classical NTRU signatures // LNCS. 2014. V. 8772. P. 1–21.
22. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* NTRUSIGN: Digital signatures using the NTRU lattice // LNCS. 2003. V. 2612. P. 122–140.
23. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* Performance Improvements and a Baseline Parameter Generation Algorithm for NTRUSign. Cryptology ePrint Archive. Paper 2005/274. `https://eprint.iacr.org/2005/274`.
24. *Gentry C., Peikert C., and Vaikuntanathan V.* Trapdoors for hard lattices and new cryptographic constructions // Proc. STOC'08. Victoria, British Columbia, Canada, 2008. P. 197–206.
25. *Stehlé D. and Steinfeld R.* Making NTRU as secure as worst-case problems over ideal lattices // LNCS. 2011. V. 6632. P. 27–47.
26. *Lyubashevsky V.* Lattice signatures without trapdoors // LNCS. 2012. V. 7237. P. 738–755.
27. *Gama N. and Nguyen P. Q.* Predicting lattice reduction // LNCS. 2008. V. 4965. P. 31–51.
28. *Howgrave-Graham N.* A hybrid lattice-reduction and meet-in-the-middle attack against NTRU // LNCS. 2007. V. 4622. P. 150–169.
29. *May A. and Silverman J. H.* Dimension reduction methods for convolution modular lattices // LNCS. 2001. V. 2146. P. 110–125.
30. *Stehlé D. and Steinfeld R.* Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices. Cryptology ePrint Archive. Paper 2013/004. `https://eprint.iacr.org/2013/004`.
31. *Boneh D., Dagdelen Ö., Fischlin M., et al.* Random oracles in a quantum world // LNCS. 2011. V. 7073. P. 41–69.
32. *Diffie W. and Hellman M. E.* New directions in cryptography // IEEE Trans. Inform. Theory. 1976. No. 22(6). P. 644–654.
33. *Bellare M. and Rogaway P.* Random oracles are practical: A paradigm for designing efficient protocols // Proc. CCS'93. Fairfax, Virginia, USA, 1993. P. 62–73.
34. *Micciancio D. and Peikert C.* Trapdoors for lattices: Simpler, tighter, faster, smaller // LNCS. 2012. V. 7237. P. 700–718.
35. *Alwen J. and Peikert C.* Generating shorter bases for hard random lattices // Theory Comput. Syst. 2011. No. 48(3). P. 535–553.
36. *Peikert C. and Rosen A.* Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices // LNCS. 2006. V. 3876. P. 145–166.
37. *Fouque P.-A., Hoffstein J., Kirchner P., et al.* Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. Specification v1.2. `https://falcon-sign.info/falcon.pdf`. 2020.

38. *Ducas L. and Prest T.* Fast Fourier orthogonalization // Proc. ISSAC'16. Waterloo, ON, Canada, 2016. P. 191–198.
39. *Klein P. N.* Finding the closest lattice vector when it's unusually close // Proc. SODA'00. San Francisco, California, USA, 2000. P. 937–941.
40. *Micciancio D. and Walter M.* Practical, predictable lattice basis reduction // LNCS. 2016. V. 9665. P. 820–849.
41. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems // LNCS. 1987. V. 263. P. 186–194.
42. *Abdalla M., An J. H., Bellare M., and Namprempre C.* From identification to signatures via the Fiat — Shamir transform: Minimizing assumptions for security and forward-security // LNCS. 2002. V. 2332. P. 418–433.
43. *Schnorr C. P.* Efficient signature generation by smart cards // J. Cryptology. 1991. V. 4. No. 3. P. 161–174.
44. *Lyubashevsky V.* Fiat — Shamir with aborts: Applications to lattice and factoring-based signatures // LNCS. 2009. V. 5912. P. 598–616.
45. *Ducas L., Kiltz E., Lepoint T., et al.* CRYSTALS-Dilithium: A lattice-based digital signature scheme // IACR Trans. Cryptographic Hardware and Embedded Systems. 2018. No. 1. P. 238–268.
46. *Alagic G., Apon D., Cooper D., et al.* Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. NIST Interagency/Internal Report (NISTIR). 2022. Report 8413. https://csrc.nist.gov/pubs/ir/8413/upd1/final.
47. *Cheon J. H., Choe H., Devevey J., et al.* HAETAE: Shorter Lattice-Based Fiat — Shamir Signatures. Cryptology ePrint Archive. Paper 2023/624. https://eprint.iacr.org/2023/624.
48. *Devroye L.* General principles in random variate generation // Non-Uniform Random Variate Generation. N.Y.: Springer, 1986. P. 27–82.
49. *Bellare M. and Neven G.* Multi-signatures in the plain public-key model and a general forking lemma // Proc. CCS'06. Alexandria, Virginia, USA, 2006. P. 390–399.
50. *Ducas L., Durmus A., Lepoint T., and Lyubashevsky V.* Lattice signatures and bimodal gaussians // LNCS. 2013. V. 8042. P. 40–56.
51. *Groot Bruinderink L., Hülsing A., Lange T., and Yarom Y.* Flush, Gauss, and reload — a cache attack on the BLISS lattice-based signature scheme // LNCS. 2016. V. 9813. P. 323–345.
52. *Espitau T., Fouque P. A., Gérard B., and Tibouchi M.* Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers // Proc. CCS'17. Dallas, Texas, USA, 2017. P. 1857–1874.
53. *Pessl P., Bruinderink L. G., and Yarom Y.* To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures // Proc. CCS'17. Dallas, Texas, USA, 2017. P. 1843–1855.
54. *Devevey J., Fawzi O., Passelègue A., and Stehlé D.* On rejection sampling in Lyubashevsky's signature scheme // LNCS. 2022. V. 13794. P. 34–64.
55. *Brakerski Z., Gentry C., and Vaikuntanathan V.* (Leveled) fully homomorphic encryption without bootstrapping // Proc. ITCS'12. Cambridge, Massachusetts, 2012. P. 309–325.
56. *Bai S. and Galbraith S. D.* An improved compression technique for signatures based on learning with errors // LNCS. 2014. V. 8366. P. 28–47.
57. *Kiltz E., Lyubashevsky V., and Schaffner C.* A concrete treatment of Fiat — Shamir signatures in the quantum random-oracle model // LNCS. 2018. V. 10822. P. 552–586.

58. *Don J., Fehr S., Majenz C., and Schaffner C.* Security of the Fiat — Shamir transformation in the quantum random-oracle model // LNCS. 2019. V. 11693. P. 356–383.
59. *Liu Q. and Zhandry M.* Revisiting post-quantum Fiat — Shamir // LNCS. 2019. V. 11693. P. 326–355.
60. *Devevey J., Fallahpour P., Passelègue A., and Stehlé D.* A detailed analysis of Fiat — Shamir with aborts // LNCS. 2023. V. 14085. P. 327–357.
61. *Schnorr C. P. and Euchner M.* Lattice basis reduction: Improved practical algorithms and solving subset sum problems // Math. Programming. 1994. V. 66. P. 181–199.
62. *Alkim E., Ducas L., Pöppelmann T., and Schwabe P.* Post-quantum key exchange: A new hope // Proc. SEC'16. Austin, TX, USA, 2016. P. 327–343.
63. *Becker A., Ducas L., Gama N., and Laarhoven T.* New directions in nearest neighbor searching with applications to lattice sieving // Proc. SODA'16. Arlington, Virginia, 2016. P. 10–24.
64. *Laarhoven T., Mosca M., and Van De Pol J.* Finding shortest lattice vectors faster using quantum search // Des. Codes Cryptography. 2015. V. 77. P. 375–400.
65. *Schnorr C. P.* Lattice reduction by random sampling and birthday methods // LNCS. V. 2607. P. 145–156.
66. *Chen Y.* Réduction de réseau et sécurité concrete du chiffrement completement homomorphe. Thése de doctorat. Paris, 2013. (in French)
67. *Howe J. and Westerbaan B.* Benchmarking and analysing the NIST PQC lattice-based signature schemes standards on the ARM Cortex M7 // LNCS. 2023. V. 14064. P. 442–462.
68. *Vidaković M. and Miličević K.* Performance and applicability of post-quantum digital signature algorithms in resource-constrained environments // Algorithms. 2023. V. 16. No. 11:518.
69. *Lyubashevsky V.* CRYSTALS-Dilithium Round 3 Presentation. `https://csrc.nist.gov/presentations/2021/crystals-dilithium-round-3-presentation`. 2021.
70. *Westerbaan B.* NIST's Pleasant Post-Quantum Surprise. `https://blog.cloudflare.com/nist-post-quantum-surprise/`. 2022.
71. *Schmieg S., Kolbl S., and Endignoux G.* Google's Threat Model for Post-Quantum Cryptography. `https://bughunters.google.com/blog/5108747984306176/google-s-threat-model-for-post-quantum-cryptography`. 2024.

REFERENCES

1. *Shor P. W.* Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev., 1995, no. 41, pp. 303–332.
2. *Ajtai M., Kumar R., and Sivakumar D.* A sieve algorithm for the shortest lattice vector problem. Proc. STOC'01, Hersonissos, Greece, 2001, pp. 601–610.
3. *Dinur I., Kindler G., and Safra S.* Approximating CVP to within almost-polynomial factors is NP-hard. Combinatorica, 2003, vol. 23, pp. 205–243.
4. *Roy S. S., Vercauteren F., Mentens N., et al.* Compact ring-LWE cryptoprocessor. LNCS, 2014, vol. 8731, pp. 371–391.
5. *Ajtai M.* Generating hard instances of lattice problems. Proc. STOC'96, Philadelphia, Pennsylvania, USA, 1996, pp. 99–108.
6. *Regev O.* On lattices, learning with errors, random linear codes, and cryptography. Proc. STOC'05, Baltimore, MD, USA, 2005, pp. 84–93.
7. *Micciancio D.* Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Comput. Complexity, 2007, no. 16(4), pp. 365–411.

8. *Peikert C. and Rosen A.* Lattices that admit logarithmic worst-case to average-case connection factors. Proc. STOC'07, San Diego, California, USA, 2007, pp. 478–487.
9. *Lyubashevsky V. and Micciancio D.* Generalized compact knapsacks are collision resistant. LNCS, 2006, vol. 4052, pp. 144–155.
10. *Lyubashevsky V.* Fiat — Shamir with aborts: Applications to lattice and factoring-based signatures. LNCS, 2009, vol. 5912, pp. 598–616.
11. *Lyubashevsky V., Peikert C., and Regev O.* On ideal lattices and learning with errors over rings. J. ACM, 2010, vol. 60, pp. 43:1–43:35.
12. *Langlois A. and Stehlé D.* Worst-case to average-case reductions for module lattices. Des. Codes Cryptography, 2015, vol. 75, pp. 565–599.
13. *Goldreich O., Goldwasser S., and Halevi S.* Public-key cryptosystems from lattice reduction problems. LNCS, 1997, vol. 1294, pp. 112–131.
14. *Hoffstein J., Pipher J., and Silverman J. H.* NTRU: A ring-based public key cryptosystem. LNCS, 1998, vol. 1423, pp. 267–288.
15. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* NTRUSign: Digital signatures using the NTRU lattice. LNCS, 2003, vol. 2612, pp. 122–140.
16. *Hoffstein J., Pipher J., and Silverman J. H.* NSS: An NTRU lattice-based signature scheme. LNCS, 2001, vol. 2045, pp. 211–228.
17. *Gentry C., Jonsson J., Stern J., and Szydlo M.* Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001. LNCS, 2001, vol. 2248, pp. 1–20.
18. *Gentry C. and Szydlo M.* Cryptanalysis of the revised NTRU signature scheme. LNCS, 2002, vol. 2332, pp. 299–320.
19. *Nguyen P. Q. and Regev O.* Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J. Cryptology, 2009, no. 22(2), pp. 139–160.
20. *Ducas L. and Nguyen P. Q.* Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. LNCS, 2012, vol. 7658, pp. 433–450.
21. *Melchor C. A., Boyen X., Deneuville J.-C., and Gaborit P.* Sealing the leak on classical NTRU signatures. LNCS, 2014, vol. 8772, pp. 1–21.
22. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* NTRUSIGN: Digital signatures using the NTRU lattice. LNCS, 2003, vol. 2612, pp. 122–140.
23. *Hoffstein J., Howgrave-Graham N., Pipher J., et al.* Performance Improvements and a Baseline Parameter Generation Algorithm for NTRUSign. Cryptology ePrint Archive, paper 2005/274. `https://eprint.iacr.org/2005/274`.
24. *Gentry C., Peikert C., and Vaikuntanathan V.* Trapdoors for hard lattices and new cryptographic constructions. Proc. STOC'08, Victoria, British Columbia, Canada, 2008, pp. 197–206.
25. *Stehlé D. and Steinfeld R.* Making NTRU as secure as worst-case problems over ideal lattices. LNCS, 2011, vol. 6632, pp. 27–47.
26. *Lyubashevsky V.* Lattice signatures without trapdoors. LNCS, 2012, vol. 7237, pp. 738–755.
27. *Gama N. and Nguyen P. Q.* Predicting lattice reduction. LNCS, 2008, vol. 4965, pp. 31–51.
28. *Howgrave-Graham N.* A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. LNCS, 2007, vol. 4622, pp. 150–169.
29. *May A. and Silverman J. H.* Dimension reduction methods for convolution modular lattices. LNCS, 2001, vol. 2146, pp. 110–125.
30. *Stehlé D. and Steinfeld R.* Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices. Cryptology ePrint Archive, paper 2013/004. `https://eprint.iacr.org/2013/004`.

31. *Boneh D., Dagdelen Ö., Fischlin M., et al.* Random oracles in a quantum world. LNCS, 2011, vol. 7073, pp. 41–69.
32. *Diffie W. and Hellman M. E.* New directions in cryptography. IEEE Trans. Inform. Theory, 1976, no. 22(6), pp. 644–654.
33. *Bellare M. and Rogaway P.* Random oracles are practical: A paradigm for designing efficient protocols. Proc. CCS'93, Fairfax, Virginia, USA, 1993, pp. 62–73.
34. *Micciancio D. and Peikert C.* Trapdoors for lattices: Simpler, tighter, faster, smaller. LNCS, 2012, vol. 7237, pp. 700–718.
35. *Alwen J. and Peikert C.* Generating shorter bases for hard random lattices. Theory Comput. Syst., 2011, no. 48(3), pp. 535–553.
36. *Peikert C. and Rosen A.* Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. LNCS, 2006, vol. 3876, pp. 145–166.
37. *Fouque P.-A., Hoffstein J., Kirchner P., et al.* Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. Specification v1.2. `https://falcon-sign.info/falcon.pdf`, 2020.
38. *Ducas L. and Prest T.* Fast Fourier orthogonalization. Proc. ISSAC'16, Waterloo, ON, Canada, 2016, pp. 191–198.
39. *Klein P. N.* Finding the closest lattice vector when it's unusually close. Proc. SODA'00, San Francisco, California, USA, 2000, pp. 937–941.
40. *Micciancio D. and Walter M.* Practical, predictable lattice basis reduction. LNCS, 2016, vol. 9665, pp. 820–849.
41. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems. LNCS, 1987, vol. 263, pp. 186–194.
42. *Abdalla M., An J. H., Bellare M., and Namprempre C.* From identification to signatures via the Fiat — Shamir transform: Minimizing assumptions for security and forward-security. LNCS, 2002, vol. 2332, pp. 418–433.
43. *Schnorr C. P.* Efficient signature generation by smart cards. J. Cryptology, 1991, vol. 4, no. 3, pp. 161–174.
44. *Lyubashevsky V.* Fiat — Shamir with aborts: Applications to lattice and factoring-based signatures. LNCS, 2009, vol. 5912, pp. 598–616.
45. *Ducas L., Kiltz E., Lepoint T., et al.* CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR Trans. Cryptographic Hardware and Embedded Systems, 2018, no. 1, pp. 238–268.
46. *Alagic G., Apon D., Cooper D., et al.* Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. NIST Interagency/Internal Report (NISTIR), 2022, Report 8413. `https://csrc.nist.gov/pubs/ir/8413/upd1/final`.
47. *Cheon J. H., Choe H., Devevey J., et al.* HAETAE: Shorter Lattice-Based Fiat — Shamir Signatures. Cryptology ePrint Archive, paper 2023/624, `https://eprint.iacr.org/2023/624`.
48. *Devroye L.* General principles in random variate generation. Non-Uniform Random Variate Generation, N.Y., Springer, 1986, pp. 27–82.
49. *Bellare M. and Neven G.* Multi-signatures in the plain public-key model and a general forking lemma. Proc. CCS'06, Alexandria, Virginia, USA, 2006, pp. 390–399.
50. *Ducas L., Durmus A., Lepoint T., and Lyubashevsky V.* Lattice signatures and bimodal gaussians. LNCS, 2013, vol. 8042, pp. 40–56.
51. *Groot Bruinderink L., Hülsing A., Lange T., and Yarom Y.* Flush, Gauss, and reload — a cache attack on the BLISS lattice-based signature scheme. LNCS, 2016, vol. 9813, pp. 323–345.

52. *Espitau T., Fouque P. A., Gérard B., and Tibouchi M.* Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. Proc. CCS'17, Dallas, Texas, USA, 2017, pp. 1857–1874.
53. *Pessl P., Bruinderink L. G., and Yarom Y.* To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures. Proc. CCS'17, Dallas, Texas, USA, 2017, pp. 1843–1855.
54. *Devevey J., Fawzi O., Passelègue A., and Stehlé D.* On rejection sampling in Lyubashevsky's signature scheme. LNCS, 2022, vol. 13794, pp. 34–64.
55. *Brakerski Z., Gentry C., and Vaikuntanathan V.* (Leveled) fully homomorphic encryption without bootstrapping. Proc. ITCS'12, Cambridge, Massachusetts, 2012, pp. 309–325.
56. *Bai S. and Galbraith S. D.* An improved compression technique for signatures based on learning with errors. LNCS, 2014, vol. 8366, pp. 28–47.
57. *Kiltz E., Lyubashevsky V., and Schaffner C.* A concrete treatment of Fiat — Shamir signatures in the quantum random-oracle model. LNCS, 2018, vol. 10822, pp. 552–586.
58. *Don J., Fehr S., Majenz C., and Schaffner C.* Security of the Fiat — Shamir transformation in the quantum random-oracle model. LNCS, 2019, vol. 11693, pp. 356–383.
59. *Liu Q. and Zhandry M.* Revisiting post-quantum Fiat — Shamir. LNCS, 2019, vol. 11693, pp. 326–355.
60. *Devevey J., Fallahpour P., Passelègue A., and Stehlé D.* A detailed analysis of Fiat — Shamir with aborts. LNCS, 2023, vol. 14085, pp. 327–357.
61. *Schnorr C. P. and Euchner M.* Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Math. Programming, 1994, vol. 66, pp. 181–199.
62. *Alkim E., Ducas L., Pöppelmann T., and Schwabe P.* Post-quantum key exchange: A new hope. Proc. SEC'16, Austin, TX, USA, 2016, pp. 327–343.
63. *Becker A., Ducas L., Gama N., and Laarhoven T.* New directions in nearest neighbor searching with applications to lattice sieving. Proc. SODA'16, Arlington, Virginia, 2016, pp. 10–24.
64. *Laarhoven T., Mosca M., and Van De Pol J.* Finding shortest lattice vectors faster using quantum search. Des. Codes Cryptography, 2015, vol. 77, pp. 375–400.
65. *Schnorr C. P.* Lattice reduction by random sampling and birthday methods. LNCS, vol. 2607, pp. 145–156.
66. *Chen Y.* Réduction de réseau et sécurité concrete du chiffrement completement homomorphe. Thése de doctorat. Paris, 2013. (in French)
67. *Howe J. and Westerbaan B.* Benchmarking and analysing the NIST PQC lattice-based signature schemes standards on the ARM Cortex M7. LNCS, 2023, vol. 14064, pp. 442–462.
68. *Vidaković M. and Miličević K.* Performance and applicability of post-quantum digital signature algorithms in resource-constrained environments. Algorithms, 2023, vol. 16, no. 11:518.
69. *Lyubashevsky V.* CRYSTALS-Dilithium Round 3 Presentation. `https://csrc.nist.gov/presentations/2021/crystals-dilithium-round-3-presentation`, 2021.
70. *Westerbaan B.* NIST's Pleasant Post-Quantum Surprise. `https://blog.cloudflare.com/nist-post-quantum-surprise/`, 2022.
71. *Schmieg S., Kolbl S., and Endignoux G.* Google's Threat Model for Post-Quantum Cryptography. `https://bughunters.google.com/blog/5108747984306176/google-s-threat-model-for-post-quantum-cryptography`, 2024.