# Цифровые подписи

Захар Назаров

Июнь 2024

## Содержание

**1 Введение** **2**

**2 Термины и определения** **2**

**3 Архитектуры хеш-функций** **3**
3.1 Hash-and-Sign . . . 3
3.2 Преобразование Фиата–Шамира . . . 3

**4 DSA** **4**
4.1 Генерация цифровой подписи . . . 4
4.2 Проверка цифровой подписи . . . 5

**5 ECDSA** **5**
5.1 Генерация ключевой пары . . . 5
5.2 Генерация цифровой подписи . . . 5
5.3 Проверка цифровой подписи . . . 6

**6 ГОСТ Р 34.10-1994** **6**
6.1 Генерация цифровой подписи . . . 6
6.2 Проверка цифровой подписи . . . 6

**7 ГОСТ 34.10-2018** **7**
7.1 Генерация ключевой пары . . . 7
7.2 Генерация цифровой подписи . . . 7
7.3 Проверка цифровой подписи . . . 7

**8 Схема подписи Фиата–Шамира** **8**
8.1 Процесс взаимодействия сторон . . . 8

**9 Схема подписи Шнорра** **8**
9.1 Генерация ключей . . . 8
9.2 Генерация цифровой подписи . . . 9
9.3 Проверка цифровой подписи . . . 9

**10 Схема подписи Меркля–Лампорта** **9**
10.1 Цифровая подпись Лампарта . . . 9
10.1.1 Генерация ключей . . . 9
10.1.2 Генерация цифровой подписи . . . 9
10.1.3 Проверка цифровой подписи . . . 9
10.2 Подпись Лампарта в комбинации с деревом Меркла . . . 9
10.2.1 Генерация ключей . . . 9

10.2.2 Генерация цифровой подписи . . . 10
10.2.3 Проверка цифровой подписи . . . 10

**11 Схема подписи CFS(Courtois, Finiasz, Sendrier) 10**
11.0.1 Генерация ключей . . . 10
11.0.2 Генерация цифровой подписи . . . 10
11.0.3 Проверка цифровой подписи . . . 11
11.1 Криптоанализ . . . 11
11.1.1 Декодирующие атаки . . . 11
11.1.2 Структурные атаки . . . 11
11.1.3 Сложность . . . 11
11.2 Заключение . . . 11

**12 CRYSTALS–Dilithium. My Signature! 11**
12.0.1 Генерация ключей . . . 12
12.0.2 Генерация цифровой подписи . . . 12
12.0.3 Проверка цифровой подписи . . . 12
12.0.4 Листинг полного алгоритма . . . 13

**13 Список литературы 13**

# 1. Введение

В данной работе описываются две схемы построения цифровых подписей: "Hash-and-Sign"и "Преобразование Фиата–Шамира а также 9 конкретных цифровых подписей: DSA, ECDSA, ГОСТ Р 34.10-1994, ГОСТ 34.10-2018, Схема подписи Фиата–Шамира, Схема подписи Шнорра, Схема подписи Меркля–Лампорта, Схема подписи CFS(Courtois, Finiasz, Sendrier), CRYSTALS–Dilithium. Устройство схемы подписи CFS и ее особенности рассматриваются более детально, также расмотрены атаки на него.

# 2. Термины и определения

**Определение 2.1.** **Схема "Hash-and-Sign"**
Пусть $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ — это схема подписи для сообщений длиной $l(n)$, и пусть $\Pi_H = (\text{Gen}_H, H)$ — это хеш-функция с выходной длиной $l(n)$. Построим схему подписи $\Pi' = (\text{Gen'}, \text{Sign'}, \text{Vrfy'})$ следующим образом:

— Gen': на вход $1^n$ запускаем $\text{Gen}(1^n)$, чтобы получить $(pk, sk)$, и запускаем $\text{Gen}_H(1^n)$, чтобы получить $s$; открытый ключ равен $\langle pk, s\rangle$, а закрытый ключ равен $\langle sk, s\rangle$.

— Sign': на вход подается закрытый ключ $\langle sk, s\rangle$ и сообщение $m \in \{0,1\}^*$, на выходе $\sigma \leftarrow \text{Sign}_{sk}(H_s(m))$.

— Vrfy': на вход подается открытый ключ $\langle pk, s\rangle$, сообщение $m \in \{0,1\}^*$ и подпись $\sigma$, на выходе выдать 1, если и только если $\text{Vrfy}_{pk}(H_s(m), \sigma) \stackrel{?}{=} 1$.

Построенная $\Pi'$ и будет схемой подписи "Hash-and-Sign".

**Определение 2.2.** Эллиптической кривой над конечным полем $F_q$ называется совокупность точек $(x, y)$, которые удовлетворяют следующему уравнению:
$y^2 = x^3 + ax + b$
Над точками эллиптической кривой вводится операция сложения с помощью добавления "виртуальной точки"O. Лучше всего варианты сложения точек эллиптической кривой описываются следующей иллюстрацией:

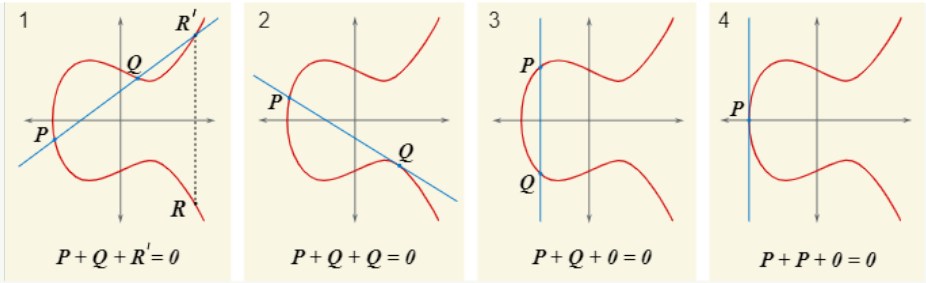


Рис. 1. Операция сложения для точек эллиптической кривой [4]

Умножение точки на целое число $Q = nP$ определяется как:

$$Q = \sum_{n} P$$

.

**Определение 2.3.** Дерево Меркла?

# 3. Архитектуры хеш-функций

Рассмотрим две основные архитектуры построения схем подписей: "Hash-and-Sign"и "преобразование Фиата–Шамира"

## 3.1. Hash-and-Sign

Пусть у нас есть схема подписи $\Pi$ сообщений длины $l$ и хеш-функция $H$, у которой длина выхода также равна $l$. Схема "Hash-and-Sign"заключается в том, что сообщение произвольной длины хешируется с помощью $H$ и затем подписывается $\Pi$.

Формальное определение содержится в главе "Термины и определение".

## 3.2. Преобразование Фиата–Шамира

В 1986 году Амос Фиат и Ади Шамир опубликовали новый протокол интерактивного взаимодействия, с помощью которого можно организовать идентификацию и схемы подписи [1]. Его криптографическая стойкость основана на сложности решения задачи нахождения квадратного корня из числа по модулю $n$, где $n$ - это произведение двух простых чисел и двоичная запись $n$ содержит не менее 512 значащих битов. Также классическая схема предполагает наличие **доверенного** центра, который будет подтверждать информацию $I$ об идентифицирующейся стороне и выдавать ей определенный секрет под эти данные. Зафиксируем:

1) Публичное число $n = pq$, где $p, q$ - простые числа. Факторизацию этого числа знает только доверенный центр.
2) Публичная псевдослучайная функция $f$.
3) Фиксированное количество $k$ маленьких значений $j$. Для простоты зафиксируем: $j = 1, \ldots, k$.
4) Количество проверок $t$.

Поэтому эта схема содержит 2 этапа: проверка идентичности доверенным центром и выдача секрета идентифицирующейся стороне, непосредственная идентификация перед контрагентом.

**Этап проверки идентичности**

1) Центр вычисляет $k$ значений $v_j = f(I, j)$.

2) Для каждого $v_j$ находится обратный к нему элемент $p = v_j^{-1}$ по модулю $n$ и затем вычисляется наименьший квадратный корень по модулю: $s_j = \sqrt{p} \bmod n$.

3) Затем безопасным путем (в оригинальной статье в виде смарт-карты) передается информация идентифицирующейся стороне: $I, s_j(j)$.

**Идентификация**
Пусть у нас есть A со смарт-картой и сторона B, перед которой A хочет идентифицироваться.

1) Сторона A отправляет I стороне B.

2) B вычисляет $v_j = f(I, j)$, где $j = 1, \ldots, k$.

3) Для i от 1 до t:

   а) Сторона A выбирает случайное $r_i \in [0, n)$ и отправляет $x_i = r_i^2$ стороне B.

   б) Сторона B отправляет рандомный бинарный вектор $(e_{i1}, \ldots, e_{ik})$ стороне A.

   в) Сторона A отправляет B:
   $y_i = r_i \prod_{e_{ij}=1} s_j \pmod n$

   г) Сторона B проверяет:
   $x_i = y_i^2 \prod_{e_{ij}=1} v_j \pmod n$

Таким образом, если все проверки B прошли успешно, то считается что A успешно идентифицировалась. В противном случае - не успешно.

# 4. DSA

DSA (Digital Signature Algorithm) - алгоритм цифровой подписи, основанный на конструкции "Hash-and-sign". Опубликован в 1998 году и запатентован NIST. Опишем алгоритм, следуя стандарту [2].
Определим следующие параметры:

1) Простое число $p$, где $2^{L-1} < p < 2^L$, 512 <= $L$ <= 1024 и $L$ кратно 64.

2) Простое число $q$, которое является делителем $p - 1$, $2^{159} < q < 2^{160}$.

3) $g = h^{(p-1)/q} \bmod p$, где $h$ любое число от 1 до $p - 1$ и $g > 1$.

4) Случайное число $x$, сгенерированное псевдорандомно, $0 < x < q$.

5) $y = g^x \bmod p$

6) Случайное число $k$, сгенерированное псевдорандомно, $0 < k < q$.

Значения p, q, g публичны и ими может пользоваться группа пользователей. Приватный и публичный ключ пользователя x и y соответственно. Значения x и k используются для генерации подписи и остаются в секрете, k генерируется для каждой новой подписи.

## 4.1. Генерация цифровой подписи

1) $r = (g^k \bmod p) \bmod q$

2) $(k^{-1}()SHA - 1(M) + xr) \bmod q$

Значение $k^{-1}$ является мультипликативно обратным элементом к $k$ по модулю q. Если $r = 0$, то выбирается другое $k$; если $s = 0$, то выбирается другое $k$. Далее сообщение $m$ отправляется вместе с подписью $(r, S)$ получателю.

### 4.2. Проверка цифровой подписи

Получатель знает $p, q, g, y$ и получает подписанное сообщение $(m, r, s)$. Алгоритм проверки следующий:

1) Проверяется что r,s лежат в диапазоне: $0 < r, s < q$. Иначе проверка заканчивается неуспехом. Если все правильно, то проводятся дальнейшие вычисления.
2) $w = s^{-1} \bmod q$
3) $u_1 = ((SHA - 1(M))w) \bmod q$
4) $u_2 = ((r)w) \bmod q$
5) $v = (((g)^{u_1}(y)^{u_2}) \bmod p) \bmod q$

Если $v = r$, то подпись считается валидно, иначе - невалидной.

## 5. ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) - алгоритм цифровой подписи, основанный на конструкции "Hash-and-sign а также использующий эллиптические кривые.

Будем считать, что у нас есть заданная эллиптическая кривая с параметрами, которая отвечает требованиям безопасности, описанным в стандарте [3]:

1) $q$ - порядок поля $F_q$.
2) $n$ - порядок эллиптической кривой.
3) $a, b$ - параметры эллиптической кривой.
4) $G$ - базовая точка эллиптической кривой.

Начнем рассматривать взаимодействие между A и B, где A - хочет отправить подписанное сообщение.

### 5.1. Генерация ключевой пары

На входе у нас заданная эллиптическая кривая и точка $G$.

1) Выбирается случайное число $d \in [1, n - 1]$.
2) Вычисляется новая точка эллиптической кривой $Q = dG$.
3) На выход идет $d$ и $Q$.

### 5.2. Генерация цифровой подписи

На входе у нас заданная эллиптическая кривая, точка $G$, закрытый ключ $d$, хеш-функция $H$ и сообщение $m$.

1) Выбирается случайное число $k \in [1, n - 1]$.
2) Вычислить $kG = (x_1, y_1)$.
3) Вычислить $r = x_1 \bmod n$.
4) Вычислить $e = H(m)$
5) Вычислить $s = k^{-1}(e + dr) \bmod n$
6) Вернуть $(r, s)$

Если $r = 0$, то выбирается другое $k$; если $s = 0$, то выбирается другое $k$.

### 5.3. Проверка цифровой подписи

На входе у нас заданная эллиптическая кривая, точка $G$, хеш-функция $H$, сообщение $m$, открытый ключ $Q$ и подпись $(r, s)$.

1) Проверка, что $r, s \in [1, n-1]$. При неудаче - то подпись невалидна.
2) Вычислить $e = H(m)$.
3) Вычислить $w = s^{-1} \bmod n$.
4) Вычислить $u_1 = ew \bmod n, u_2 = rw \bmod n$.
5) Вычислить $X = u_1 * G + u_2 * Q = (x_2, y_2)$
6) Если $X = O$, то подпись невалидна. Иначе вычислить $v = x_2 \bmod n$

Если $v = r$, то подпись валидна, иначе - невалидна.

## 6. ГОСТ Р 34.10-1994

ГОСТ Р 34.10-1994 - алгоритм цифровой подписи, основанный на конструкции "Hash-and-sign стандартизованный в 1994 году [5].

Пусть заданы простые числа $p$ (512 бит) и $q$ (256 бит), которые удовлетворяют условиям описанным в стандарте [5]. Пусть также есть число $a > 1$, которое представляется в виде:

$a = d^{\frac{p-1}{q}} \bmod p$ (512 бит)

### 6.1. Генерация цифровой подписи

На входе у нас заданные $p, q, a$, сообщение $m$ и хеш-функция $H$. Предварительно выбирается случайное число $x \in [0, q]$, которое будет закрытым ключом. Также считается открытый ключ по формуле: $y = a^x \bmod p$.

1) Вычислить $h = H(m) \bmod q$, если $H(M) = 0$, то $h := 0^{255}1$.
2) Выбирается случайное число $k \in [0, q]$.
3) Вычислить $r = a^k \bmod p$.
4) Вычислить $r' = r \bmod q$.
5) Если $r' = 0$, то выбрать другое $k$.
6) Вычислить $s = (xr' + kh) \bmod q$.
7) Подать на выход подпись: $(r', s)$

После выработки подписи число $k$ уничтожается.

### 6.2. Проверка цифровой подписи

На входе у нас заданные $p, q, a$, сообщение $m$, хеш-функция $H$, открытый ключ $y$ и подпись $(r', s)$.

1) Проверка, что $r', s \in (0, q)$. При неудаче - то подпись невалидна.
2) Вычислить $h = H(m) \bmod q$, если $H(M) = 0$, то $h := 0^{255}1$.
3) Вычислить $v = h^{q-2} \bmod q$.
4) Вычислить $z_1 = sv \bmod q$.
5) Вычислить $z_2 = (q - r')v \bmod q$.
6) Вычислить $u = ((a^{z_1} * y^{z_2}) \bmod p) \bmod q$.
7)

Если $u = r'$, то подпись валидна, иначе - невалидна.

# 7. ГОСТ 34.10-2018

ГОСТ Р 34.10-2018 - алгоритм цифровой подписи, основанный на конструкции "Hash-and-sign а также использующий эллиптические кривые. Является международным стандартом, принятым в 2018 году [6].

Будем считать, что у нас есть заданная эллиптическая кривая с параметрами, которая отвечает требованиям безопасности, описанным в стандарте [6]:

1) $p$ - порядок поля $F_p$.
2) $m$ - порядок эллиптической кривой.
3) $a, b$ - параметры эллиптической кривой.
4) $P$ - базовая точка эллиптической кривой.
5) $H(M)$ - хеш-функция (ГОСТ Р 34.11-2012 [Стрибог]), с выходом 256-бит.

Начнем рассматривать взаимодействие между A и B, где A - хочет отправить подписанное сообщение.

## 7.1. Генерация ключевой пары

1) Выбирается случайное число $d \in [1, q-1]$.
2) Вычисляется новая точка эллиптической кривой $Q = dP$.
3) На выход идет $d$ и $Q$.

## 7.2. Генерация цифровой подписи

На входе у нас заданная эллиптическая кривая, закрытый ключ $d$, и сообщение $m$.

1) Вычислить $h = H(m)$
2) Вычислить $e = h \bmod q$. Если $e = 0$, то $e := 1$
3) Выбирается случайное число $k \in [1, q-1]$.
4) Вычислить $C = kP = (x_1, y_1)$.
5) Вычислить $r = x_1 \bmod q$.
6) Вычислить $s = (rd + ke) \bmod q$
7) Вернуть $(r, s)$

Если $r = 0$, то выбирается другое $k$; если $s = 0$, то выбирается другое $k$.

## 7.3. Проверка цифровой подписи

На входе у нас заданная эллиптическая кривая, сообщение $m$, открытый ключ $Q$ и подпись $(r, s)$.

1) Проверка, что $r, s \in [1, q-1]$. При неудаче - то подпись невалидна.
2) Вычислить $h = H(m)$.
3) Вычислить $e = h \bmod q$. Если $e = 0$, то $e := 1$
4) Вычислить $v = e^{-1} \bmod q$.
5) Вычислить $z_1 = sv \bmod q, z_2 = -rv \bmod q$.
6) Вычислить $C = z_1 * P + z_2 * Q = (x_2, y_2)$
7) Вычислить $R = x_2 \bmod q$

Если $R = r$, то подпись валидна, иначе - невалидна.

# 8. Схема подписи Фиата–Шамира

Схема подписи Фиата–Шамира получается из преобразования Фиата–Шамира с помощью модификации.
Пусть A хочет отправить подписанное сообщение M стороне B.

— В данном случае теперь у нас нет доверенного центра, сторона A сама определяет число $n = pq$.

— Затем A генерирует секретную последовательность из z чисел $x_0, \ldots, x_z$ таких, что: $n$ div $2^{64}$ <= $x_i$ <= $n - 1$ и $GCD(x_i, n) = 1$, где $GCD(a, b)$ - наибольший общий делитель a и b.

— Далее A генерирует публичную последовательность $y_0, \ldots, y_z$, которая вычисляется по формуле:
$y_i = x_i^{-2} \bmod n$.

Пусть A хочет отправить сообщение M стороне B. Пусть у нас также задана хеш-функция $H$.

## 8.1. Процесс взаимодействия сторон

1) Сторона A выбирает случайное $k \in [0, n)$ и отправляет $R = k^2$ стороне B.
2) Сторона B отправляет рандомный бинарный вектор $(e_{i1}, \ldots, e_{iz})$ стороне A.
3) Сторона A вычисляет хеш $E = H(M||R)$.
4) Сторона A вычисляет: $S = k \prod_{e_{ij}=1} x_i \pmod n$.
5) Сторона A отправляет $(M, E, S)$ стороне B.
6) Сторона B вычисляет: $R' = S^2 \prod_{e_{ij}=1} y_j \pmod n$
7) Сторона B вычисляет: $E' = H(M||R')$.
8) Если $E = E'$, то подпись валидна, иначе - невалидна.

Отметим, что сторона B знает публичную последовательность $y_0, \ldots, y_z$ стороны A (аналог публичного ключа). Также интересно, что для одного и того же сообщения для разных посылок, будут разные подписи из-за генерации $k$.

# 9. Схема подписи Шнорра

Схема подписи Шнорра - схема подписи, которая является модификацией схемы подписи Фиата-Шамира.

## 9.1. Генерация ключей

1) Сторона A выбирает простое число $p$ длиной 1024 бит.
2) Сторона A выбирает простое число $q$, такое что оно является делителем числа $p - 1$. Размер $q$ равен 160 бит.
3) Сторона A выбирает $g > 1$, такое что $g^q = 1 \bmod p$.
4) Сторона A выбирает случайное число $w < q$.
5) Сторона A вычисляет $y = g^{q-w} \bmod p$.

Таким образом у A появляется открытый ключ $(p, q, g, y)$ и секретный ключ $w$.
Пусть A хочет отправить сообщение M стороне B. Пусть у нас также задана хеш-функция $H$.

### 9.2. Генерация цифровой подписи

1) Сторона A выбирает случайное число $r < q$ и вычисляет $x = g^r \bmod p$.
2) Сторона A вычисляет $S_1 = H(M||x)$.
3) Сторона A вычисляет $S_2 = r + wS_1 \bmod q$.
4) Сторона A отправляет стороне B подписанное сообщение $(M, S_1, S_2)$.

### 9.3. Проверка цифровой подписи

1) Сторона B вычисляет $X = (g^{S_2} g^{S_1}) \bmod p$.
2) Сторона B вычисляет $S' = H(M||X)$

Если $S' = S_1$, то подпись валидна, иначе - невалидна.

## 10. Схема подписи Меркля–Лампорта

Схема подписи Меркля–Лампорта - это схема цифровой подписи, основанная на использовании дерева Меркла и на одноразовой цифровой подписи Лампарта [7].

### 10.1. Цифровая подпись Лампарта

Пусть A хочет отправить сообщение $M$ стороне B. Пусть задано положительное число $k$, $M \in \{0,1\}^k$. Пусть у нас также задана хеш-функция $H$.

#### 10.1.1. Генерация ключей

Генерируется $2r$ значений $y_{ij}$, где $1 <= i <= k$, $0 <= j <= 1$. Также вычисляется для всех $i, j : z_{ij} = H(y_{ij})$.
Набор $y_{ij}$ - это закрытый ключ. Набор $z_{ij}$ - это открытый ключ.

#### 10.1.2. Генерация цифровой подписи

Пусть $M = m_1||m_2|| \ldots ||m_k$. Подпись сообщения $sign$ считается как: $sign = (y_{1,m_1}, \ldots, y_{k,m_k}) = (s_1, \ldots, s_k)$

#### 10.1.3. Проверка цифровой подписи

Если для всех $1 <= i <= k$ верно: $f(s_i) = z_{i,m_i}$, то подпись $sign$ считается валидной, иначе - невалидной.

### 10.2. Подпись Лампарта в комбинации с деревом Меркла

В подписи Лампарта открытый ключ может использоваться только 1 раз. С помощью дерева Меркла можно создать схему подписи, где открытый ключ может использоваться несколько раз (при большом дереве, можно считать, что ключ можно использовать неограниченное количество раз).

#### 10.2.1. Генерация ключей

Зададим некоторое $n > 1$. Сгенерируем $2^n$ пар ключей Лампарта $(X_i, Y_i)$ $(1 <= i <= 2^n)$, где $X_i$ - закрытый ключ, $Y_i$ - открытый ключ.

Затем посчитаем для каждого открытого ключа посчитаем его хеш: $h_i = H(Y_i)$, $1 <= i <= 2^n$.

Построим дерево Меркла с листьями $(h_1, \ldots, h_{2^n})$. Пусть корень дерева $pub$ будет открытым ключом для новой схемы подписи.

#### 10.2.2. Генерация цифровой подписи

Выберем любую пару ключей $X_i$, $Y_i$, например $X_d$, $Y_d$. Подпишем ею сообщение $M$, получим подпись $sig'$. Возьмем хеш от публичного ключа $h_d = H(Y_d)$. В дереве Меркла это будет лист. Встанем на этот лист и возьмем соседний элемент (назовем его $a_0$), который тоже ведет к родителю $h_d$. Встанем теперь на родителя $h_d$ и также возьмем элемент (назовем его $a_1$), который ведет к родителю родителя $h_d$. И так далее, пока не дойдем до вершины. Последний взятый элемент $a_{n-1}$ будет иметь родителя $pub$.

Составим итоговую подпись сообщения $M$ в новой схеме:
$sig = (siq', Y_d, a_0, \ldots, a_{n-1})$

#### 10.2.3. Проверка цифровой подписи

Сначала проверяется одноразовая подпись $sig'$. Если она валидна, то идем дальше, если нет - $sig$ также считается невалидной. Если $sig'$ валидна, то считаем $A_0 = H(Y_d)$. Затем считаем $A_i = H(A_{i-1}||a_i)$ для $0 <= i <= n-1$.

Если $A_{n-1} = pub$, то подпись валидна, иначе - невалидна.

## 11. Схема подписи CFS(Courtois, Finiasz, Sendrier)

Схема подписи CFS(Courtois, Finiasz, Sendrier) - алгоритм цифровой подписи, основанный на конструкции "Hash-and-sign а также использующий линейные коды. Этот алгоритм был предложен в 2001 году [8] исследователями Courtois, Finiasz и Sendrier.

Ее безопасность опирается на сложность декодирования, а также на сложность восстановления структуры линейного кода.

Пусть A хочет отправить сообщение $M$ стороне B.

Зафиксируем:

— Пусть заданы целые числа $n, k$, которые определяют размерность линейного кода $C[n, k]$.

— Пусть у нас задана хеш-функция $H$ с выходом размера $n - k$.

— Пусть у нас задана случайная бинарная невырожденная квадратная матрица $S$ порядка $(n - k)$ .

— Пусть у нас задана случайная бинарная матрица перестановки $P$ порядка $n$.

Будем считать, что вышеописанные параметры отвечают требованиями безопасности, описанным в [8].

#### 11.0.1. Генерация ключей

Секретным ключом является линейный код $C_0[n, k]$ с генератором $G_0$ и матрицей проверки $H_0$.
Публичным ключом является: $H = V * H_0 * P$.

#### 11.0.2. Генерация цифровой подписи

1) Вычисляется хеш $s = H(M)$.
2) Вычисляются $s_i = h(s||i)$, $i = 0, 1, 2, \ldots$, до тех пор пока какое-то $s_{i_0}$ не станет декодируемым для $C_0$. Запомним этот $s_{i_0}$.
3) Затем с помощью обратной функции(trapdoor function) мы находим такое $z$, что $H * z^T = s_{i_0}$.
4) На выход идет подпись $(z, i_0)$.

#### 11.0.3. Проверка цифровой подписи

1) Вычисляется хеш $s = H(M)$.

2) Вычислить $s_1 = H * z^T$

3) Вычислить $s_2 = h(s||i_0)$

Если $s_1 = s_2$, то подпись валидна, иначе - нет. Для оптимизации разработчики преобразуют $z$, чтобы подпись была более компактная.

### 11.1. Криптоанализ

Как говорилось выше, безопасность этой схемы опирается на сложность декодирования, а также на сложность восстановления структуры линейного кода. Поэтому рассмотрим атаки именно на эти аспекты.

#### 11.1.1. Декодирующие атаки

Наиболее известная атака методом декодирования была предложена Каньто и Шабо [10], и её асимптотическая временная сложность составляет примерно $(n/log_2(n))^{f(t)}$, где $f(t) = \lambda * t - c$ является аффинной функцией с λ, не сильно меньшим единицы, и $c$ — константой, лежащей в пределах от 1 до 2.

Хорошие оценки асимптотического поведения сложности лучших известных общих методов декодирования приведены Баргом в [11]. В действительности, когда скорость $(R = \frac{k}{n}$ кода стремится к 1, временная и объем памяти становится $(2^{n(1-R)/2(1+o(1))})$, что для кодов Гоппа даёт $(n^{t(1/2+o(1))})$.

#### 11.1.2. Структурные атаки

Немного известно о различимости кодов Гоппы. На практике единственная структурная атака [12] заключается в перечислении всех кодов Гоппы и затем тестировании их на эквивалентность с открытым ключом. Существует $(2^{tm}/t)$ бинарных кодов Гоппы, исправляющих t ошибок, длины $(n = 2^m)$, но из-за свойств кодов Гоппа нужно проверить только один из $(mn^3)$, и, наконец, сложность проверки эквивалентности не может быть ниже $(n(tm)^2)$ (методом Гауссова исключения). В итоге, совокупная сложность структурной атаки не будет меньше чем $(tmn^{t-2})$ элементарных операций.

#### 11.1.3. Сложность

Пусть $n = 2^m$, $k = n - tm$, $d >= 2t + 1$ для кода Гоппа. Тогда сложность лучших атак будет:

— Для декодирующей атаки: $2^{tm(1/2+o(1))}$

— Для структурной атаки: tm $2^{m(t-2)}$

### 11.2. Заключение

**Вывод:**

Схема цифровой подписи CFS (Courtois, Finiasz, Sendrier) основана на методе "Hash-and-sign"и использует линейные коды для обеспечения безопасности. Безопасность данной схемы зависит от сложности декодирования и восстановления структуры линейного кода, что делает её устойчивой к широкому спектру атак. Основные атаки для схемы это декодирующие и структурные атаки. Временная и пространственная сложность этих атак варьируется в зависимости от параметров кода, однако лучшие известные атаки всё ещё имеют достаточно высокую вычислительную сложность.

## 12. CRYSTALS–Dilithium. My Signature!

CRYSTALS–Dilithium - постквантовый алгоритм цифровой подписи, опубликованный в 2021 году [9].

### 12.0.1. Генерация ключей

— Генерация матрицы A размера $k * l$, элементами которой являются полиномы в кольце $R_q = Z_q[n]/(X^n + 1)$. Эта матрица - первая часть публичного ключа.

— Генерация двух случайных векторов $(s_1, s_2)$ размеров l и k соответственно. Элементом вектора являются также полиномы в кольце $R_q$ (с коэффициентами у многочленов не больше η.

— Вычислить $t = A * s_1 + s_2$.

Таким образом,
Пара (A, t) - публичный ключ.
Пара (A, t, $s_1$, $s_2$) - приватный ключ.

Пусть A хочет отправить сообщение M стороне B. Пусть у нас также задана хеш-функция $H$.
Определим вспомогательные функции:

$\underline{\mathsf{Power2Round}_q(r, d)}$
$r := r \bmod^+ q$
$r_0 := r \bmod^\pm 2^d$
**return** $\big((r - r_0)/2^d, r_0\big)$

$\underline{\mathsf{MakeHint}_q(z, r, \alpha)}$
$r_1 := \mathsf{HighBits}_q(r, \alpha)$
$v_1 := \mathsf{HighBits}_q(r + z, \alpha)$
**return** $[\![r_1 \neq v_1]\!]$

$\underline{\mathsf{UseHint}_q(h, r, \alpha)}$
$m := (q - 1)/\alpha$
$(r_1, r_0) := \mathsf{Decompose}_q(r, \alpha)$
**if** $h = 1$ and $r_0 > 0$ **return** $(r_1 + 1) \bmod^+ m$
**if** $h = 1$ and $r_0 \leq 0$ **return** $(r_1 - 1) \bmod^+ m$
**return** $r_1$

$\underline{\mathsf{Decompose}_q(r, \alpha)}$
$r := r \bmod^+ q$
$r_0 := r \bmod^\pm \alpha$
**if** $r - r_0 = q - 1$
  **then** $r_1 := 0$; $r_0 := r_0 - 1$
**else** $r_1 := (r - r_0)/\alpha$
**return** $(r_1, r_0)$

$\underline{\mathsf{HighBits}_q(r, \alpha)}$
$(r_1, r_0) := \mathsf{Decompose}_q(r, \alpha)$
**return** $r_1$

$\underline{\mathsf{LowBits}_q(r, \alpha)}$
$(r_1, r_0) := \mathsf{Decompose}_q(r, \alpha)$
**return** $r_0$

Рис. 2. Вспомогательные функции для CRYSTALS–Dilithium [9].

### 12.0.2. Генерация цифровой подписи

1) Генерация вектора-маски $y \in S^l_{\gamma_1 - 1}$, который состоит из многочленов с коэффициентами меньше, чем $\gamma_1$. Считаем, что $\gamma_1$ отвечает требованиям безопасности, описанным в [9].

2) Вычисляем $w_1 = HighBits(Ay, 2 * \gamma_2)$. Считаем, что $\gamma_2$ отвечает требованиям безопасности, описанным в [9].

3) Вычислить $c = H(M||w_1)$, $c \in B_{60}$. $c$ - это многочлен в $R_q$, имеющий ровно 60 коэффициентов, которые равны 1 или -1, остальные - нули.

4) $\beta = max(c * s_1, c * s_2)$

5) Если один из коэффициентов z, больше чем $\gamma_1 - \beta$, то генерация подписи начинается заново.

6) Если все коэффициенты младших битов $Az - ct$, больше чем $\gamma_2 - \beta$, то генерация подписи начинается заново.

7) На выход идет подпись $(z, c)$.

### 12.0.3. Проверка цифровой подписи

1) Вычислить $w'_1 = HighBits(Az - ct, 2\gamma_2$.

2) Если один из коэффициентов z, больше чем $\gamma_1 - \beta$, то подпись невалидна.

3) Вычислить $c' = H(M||w'_1)$

Если c’ = c, то подпись валидна, иначе - нет.

#### 12.0.4. Листинг полного алгоритма

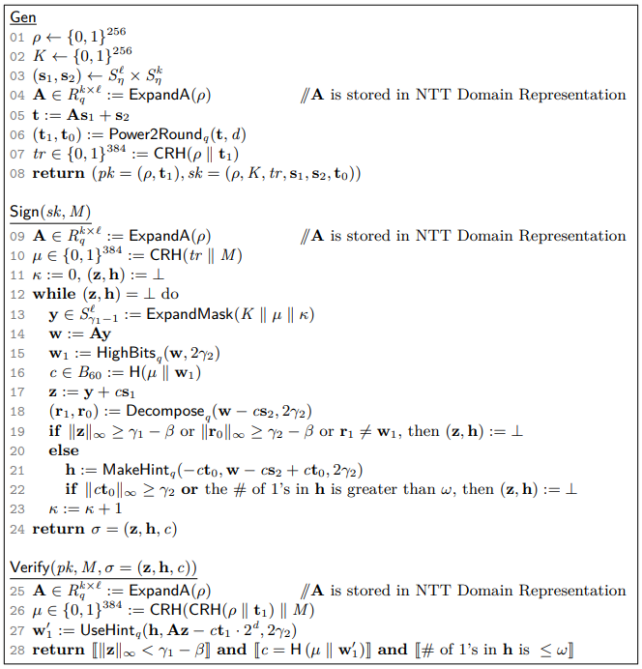


Рис. 3. Листинг CRYSTALS–Dilithium [9]

## 13. Список литературы

[1] Fiat A., Shamir A. How to prove yourself: Practical solutions to identification and signature problems //Conference on the theory and application of cryptographic techniques. – Berlin, Heidelberg : Springer Berlin Heidelberg, 1986. – С. 186-194.

[2] Digital signature standart (DSS) // FIPS PUB 186-1. – 1998

[3] ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA). // American Bankers Association. - 1998

[4] Elliptic curve group law // URL: https://commons.wikimedia.org/wiki/File:ECClines-2.0.svg

[5] ГОСТ 34.11-94 «Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма» // URL: https://files.stroyinf.ru/Data2/1/4294820/4294820322.pdf

[6] ГОСТ 34.11-94 «Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма» // URL: https://protect.gost.ru/v.aspx?control=7id=232149

[7] Lamport L. Constructing digital signatures from a one way function. – 1979.

[8] Courtois N. T., Finiasz M., Sendrier N. How to achieve a McEliece-based digital signature scheme //Advances in Cryptology—ASIA 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7. – Springer Berlin Heidelberg, 2001. – С. 157-174.

[9] Ducas L. et al. CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR TCHES 2018 (1), 238–268 (2018).

[10] Canteaut A. A new algorithm for finding minimum-weight words in large linear codes //IMA International Conference on Cryptography and Coding. – Berlin, Heidelberg : Springer Berlin Heidelberg, 1995. – С. 205-212.

[11] Barg A. Complexity issues in coding theory //Handbook of Coding theory. – 1998. – Т. 1. – С. 649-754.

[12] Loidreau P., Sendrier N. Weak keys in the McEliece public-key cryptosystem //IEEE Transactions on Information Theory. – 2001. – Т. 47. – №. 3. – С. 1207-1211.