Эссе по теме "Цифровые подписи"

Костиков Егор Вячеславович

Июнь 2024

Содержание

1. Введение

В данной работе приводится описание основных архитектур, используемых при построении цифровых подписей (конструкция Hash-and-Sign и преобразование Фиата-Шамира), а также приводятся описания реализации следующих схем цифровых подписей: DSA, ECDSA, ГОСТ Р 34.10-1994, ГОСТ 34.10-2018, схема подписи Фиата-Шамира, схема подписи Шнорра, схема подписи CFS (Courtois, Finiasz, Sendrier), подпись CRYSTALS–Dilithium. Также приводится более подробное описание и анализ схемы цифровой подписи Меркля-Лампорта.

2. Термины и определения

Под схемой электронной цифровой подписи будем понимать тройку алгоритмов $\Pi = (Gen, Sign, Vrfy)$, где:

1) Gen — полиномиальный вероятностный алгоритм генерации ключей, который принимает на вход некоторый параметр безопасности и выдает пару ключей (pk, sk) — открытый и секретный ключи соответственно;

2) $Sign$ — полиномиальный вероятностный алгоритм генерации подписи, который принимает сообщение m и закрытый ключ sk и выводит подпись σ сообщения m;

3) $Vrfy$ — полиномиальный детерминированный алгоритм проверки подписи, который принимает в качестве входных данных открытый ключ pk, сообщение m и подпись σ и выводит один бит, обозначающий принятие или отклонение подписи.

Криптографической функцией хэширования (хэш-функцией) называется отображение $H : V^* \to V_n$, где n — натуральное число, V^* — множество всех двоичных векторов конечной размерности (включая пустую строку), V_n — множество всех n-мерных двоичных векторов. Возвращаемое хэш-функцией значение называется хэш-значением.

Коллизионная атака на хеш-функцию — поиск двух различных входных блоков криптографической хеш-функции, производящих одинаковые значения хеш-функции.

Эллиптической кривой над полем P называется множество пар чисел (x, y), где x, y из P, которые удовлетворяют тождеству: $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, где коэффициенты a_1, a_2, a_3, a_4, a_6 также являются элементами поля P. Множество точек (x, y) данной кривой образуют группу точек эллиптической кривой.

Пусть имеется эллиптическая кривая $y^2 = x^3 + ax + b \mod p$ над конечным простым полем F_p. Тогда для точек $Q_1(x_1, y_1)$ и $Q_2(x_2, y_2)$ кривой следующим образом определена операция сложения $Q_1 + Q_2 = Q_3 = (x_3, y_3)$:

1) Если $x_1 \neq x_2$, то $x_3 = \lambda^2 - x_1 - x_2 \mod p$, $y_3 = \lambda(x_1 - x_3) - y_1 \mod p$, где $\lambda = \frac{y_2 - y_1}{x_2 - x1} \mod p$;

2) Если $x_1 = x_2, y_1 = y_2 \neq 0$, то $x_3 = \lambda^2 - 2x_1 \mod p$, $y_3 = \lambda(x_1 - x_3) - y_1 \mod p$, где $\lambda = \frac{3x_1^2 + a}{2y_1} \mod p$.

Естественным образом определяется операция умножения точки на число, как сложение точки самой с собой необходимое число раз: $Q = P + \ldots + P = kP$.

Далее в работе используются следующие обозначения:

- $x \parallel y$ — конкатенация строк x и y;
- F_q — конечное поле Галуа, содержащее q элементов.

3. Описания архитектур построения цифровых подписей

3.1. Конструкция Hash-and-Sign

Пусть имеется некоторая схема подписи Π, представляющая собой тройку алгоритмов $(Gen, Sign, Vrfy)$, и некоторая хэш-функция H. Сконструированная на основе подхода Hash-and-Sign электронная подпись $\Pi' = (Gen', Sign', Vrfy')$ представляет собой тройку из следующих алгоритмов:

1) $Gen' = Gen$;

2) $Sign'_{sk}(m) : \sigma := Sign_{sk}(H(m))$;

3) $Vrfy'_{pk}(m, \sigma)$: проверяется, равно ли значение $Vrfy_{pk}(H(m), \sigma)$ значению 1;

Известно, что если в условиях выше Π — схема подписи, стойкая к экзистенциальной подделке, а H — хэш-функция, стойкая к поиску коллизий, то построенная схема подписи Π' является также стойкой к экзистенциальной подделке.

3.2. Преобразование Фиата–Шамира

Подход заключается в создании определенной схемы идентификации, и последующего ее преобразования в схему цифровой подписи. Использование преобразования Фиата-Шамира было впервые предложено в работе [1]. В этой же работе было показано, как следует преобразовать интерактивную схему идентификации в схему цифровой подписи. Далее приводится описание схемы идентификации, при которой участник A пытается пройти идентификацию у участника B:

1) Выбираются простые числа p и q;

2) Вычисляется значение $n = pq$;

3) Выбираются целые числа k и t;

4) Выбирается односторонняя функция f, которая неотличима от истинно случайной функции за полиномиальное время;

5) Подготавливается строка I, которая содержит всю необходимую информацию о пользователе A (его имя, адрес, идентификационный номер, физическое описание, допуск и т.д.);

6) Вычисляются значения $v_i = f(I \parallel i)$ для $i = 1, \ldots, k$; вычисленные значения являются открытыми;

7) Вычисляются значения $s_i = \sqrt{v_i^{-1}} \mod n$ для $i = 1, \ldots, k$; вычисленные значения передаются подписывающему сообщение;

8) A посылает строку I B;

9) B вычисляет значения $v_i = f(I, j)$ для $j = 1, \ldots, k$;

10) Далее в цикле по $i = 1, \ldots, t$:

    - A выбирает случайное значение $0 \leqslant r_i < n$;
    - A вычисляет и передает B значение $x_i = r_i^2 \mod n$;
    - B передает A случайный двоичный вектор $(e_{i,1}, \ldots, e_{i,k})$;
    - A вычисляет и передает B значение $y_i = r_i \prod\limits_{e_{i,j}=1} s_j \mod n$;
    - B проверяет, что $x_i = y_i^2 \prod\limits_{e_{i,j}=1} v_j \mod n$;

Дополнительное использование в данном алгоритме односторонней функции f по отношению к некоторому входному сообщению M может преобразовать данный алгоритм из процедуры интерактивной идентификации в схему цифровой подписи. Получившаяся таким образом схема подписи приводится далее в тексте работы.

4. Описания цифровых подписей

4.1. DSA

Описание алгоритма цифровой подписи DSA приводится на основе стандарта [2].

В алгоритме DSA используются следующие параметры:

1) Значение длины $512 \leqslant L \leqslant 1024$, при этом L кратно 64;

2) Простое число $2^{L-1} < p < 2^L$; значение p является открытым;

3) Простое число $2^{159} < q < 2^{160}$, при этом q является делителем $(p-1)$; значение q является открытым;

4) Число $g = h^{(p-1)/q} \mod p > 1$, где $1 < h < p-1$; значение g является открытым;

5) Закрытый ключ $0 < x < q$;

6) Открытый ключ $y = g^x \mod p$;

7) Случайно выбранное число $0 < k < q$.

Генерация подписи для сообщения M происходит по следующему алгоритму:

1) Вычисление $r = (g^k \mod p) \mod q$;

2) Вычисление $s = (k^{-1}(SHA(M) + x \cdot r)) \mod q$, где k^{-1} — обратное к k по модулю q, а $SHA(M)$ — 160-битный выход алгоритма вычисления хэш-значения SHA-1. Стоит отметить, что в оригинальном стандарте предлагается использовать алгоритм SHA-1, хотя в более новых версиях DSA допускается использовать самостоятельно выбранную хэш-функцию.

3) Если $r = 0$ или $s = 0$, то необходимо перевыбрать k и заново выполнить шаги 1 и 2;

4) Подписью сообщения M является пара значений (r, s).

Проверка подписи (r, s) для сообщения M происходит по следующему алгоритму:

1) Вычисление $w = s^{-1} \mod q$;

2) Вычисление $u_1 = SHA(M) \cdot w \mod q$;

3) Вычисление $u_2 = r \cdot w \mod q$;

4) Вычисление $v = ((g^{u_1} \cdot y^{u_2}) \mod p) \mod q$;

5) Подпись считается верной, если $v = r$.

4.2. ECDSA

Описание алгоритма цифровой подписи ECDSA приводится на основе работы [3] разработчиков данного алгоритма. Алгоритм ECDSA является вариантом алгоритма DSA, в котором используется криптография на эллиптических кривых.

В алгоритме ECDSA используются следующие параметры:

1) Порядок поля q, где q — простое нечетное число, либо равно 2^m для некоторого m;

2) Элементы a и b поля F_q, которые определяют уравнение эллиптической кривой E над полем F_q (либо $y^2 = x^3 + ax + b$ для $p > 3$, либо $y^2 + xy = x^3 + ax^2 + b$ для $p = 2$);

3) Элементы x_G и y_G поля F_q, которые определяют точку $G = (x_G, y_G)$ кривой E простого порядка;

4) Число n, являющееся порядком точки G; при этом $n > 2^{160}$ и $n > 4\sqrt{n}$.

Генерация ключевой пары для подписи происходит по следующему алгоритму:

1) Выбирается случайное или псевдослучайное число $1 \leqslant d \leqslant n - 1$;

2) Вычисляется $Q = dG$;

3) Значение d является закрытым ключом, а Q — открытым ключом.

Генерация подписи для сообщения M происходит по следующему алгоритму:

1) Выбирается случайное или псевдослучайное число $1 \leqslant k \leqslant n - 1$;

2) Вычисляется $kG = (x_1, y_1)$;

3) Вычисляется $r = x_1 \mod n$; если $r = 0$, то необходимо перейти к шагу 1;

4) Вычисляется $e = SHA(M)$, где $SHA(M)$ — 160-битный выход алгоритма вычисления хэш-значения SHA-1. Стоит отметить, что в оригинальном стандарте предлагается использовать алгоритм SHA-1, хотя в более новых версиях DSA допускается использовать самостоятельно выбранную хэш-функцию;

5) Вычисляется $s = k^{-1}(e + d \cdot r) \mod n$, где k^{-1} — обратное к k по модулю n; если $s = 0$, то необходимо перейти к шагу 1;

6) Подписью сообщения M является пара значений (r, s).

Проверка подписи (r, s) для сообщения M происходит по следующему алгоритму:

1) Необходимо проверить, что $1 \leqslant r \leqslant n - 1$ и $1 \leqslant s \leqslant n - 1$: если это не так, то подпись не верна;

2) Вычисляется $w = s^{-1} \mod n$;

3) Вычисляется $u_1 = SHA(M) \cdot w \mod n$;

4) Вычисляется $u_2 = r \cdot w \mod n$;

5) Вычисляется $X = u_1 G + u_2 Q = (x_1, y_1)$;

6) Если $X = \mathcal{O}$, то подпись не верна. В противном случае — вычислить $v = x_1 \mod n$;

7) Подпись считается верной, если $v = r$.

4.3. ГОСТ Р 34.10-1994

Описания работы алгоритмов выработки и проверки электронной подписи приводятся в соответствии со стандартом [4].

В работе алгоритмов генерации и проверки подписи используются следующие параметры:

1) Простое число $2^{509} < p < 2^{512}$, либо $2^{1020} < p < 2^{1024}$; значение p является открытым;

2) Простое число $2^{254} < q < 2^{256}$, при этом q является делителем $(p-1)$; значение q является открытым;

3) Целое число $1 < a < (p-1)$, при этом $a^q \mod p = 1$; значение a является открытым;

4) Закрытый ключ $0 < x < q$;

5) Открытый ключ $y = a^x \mod p$.

Генерация подписи для сообщения M происходит по следующему алгоритму:

1) С использованием хэш-функции h, определенной в стандарте [5], вычисляется 256-битное значение хэш-функции $h(M)$; при этом если десятичное значение $h(M) \mod q = 0$, то необходимо присвоить $h(M) = 1$;

2) Выбирается случайное число $0 < k < q$;

3) Вычисляются значения $r = a^k \mod p$ и $r' = r \mod q$; При этом если $r' = 0$, то необходимо перейти к шагу 2 и выбрать новое значение k;

4) Вычисляется значение $s = (x \cdot r' + k \cdot h(M)) \mod q$; при этом если $s = 0$, то необходимо перейти к шагу 2 и выбрать новое значение k;

5) Подписью сообщения M является пара значений (r', s).

Проверка подписи (r', s) для сообщения M происходит по следующему алгоритму:

1) Необходимо проверить, что $0 < s < q$ и $0 < r' < q$; если это не так, то подпись не верна;

2) С использованием хэш-функции h, определенной в стандарте [5], вычисляется 256-битное значение хэш-функции $h(M)$; при этом если десятичное значение $h(M) \mod q = 0$, то необходимо присвоить $h(M) = 1$;

3) Вычисляется значение $v = (h(M))^{q-2} \mod q$;

4) Вычисляются значения $z_1 = s \cdot v \mod q$ и $z_2 = (q - r') \cdot v \mod q$;

5) Вычисляется значение $u = (a^{z_1} \cdot y^{z_2} \mod p) \mod q$;

6) Подпись считается верной, если $r' = u$.

4.4. ГОСТ 34.10-2018

Описания работы алгоритмов выработки и проверки электронной подписи приводятся в соответствии со стандартом [6].

В работе алгоритмов генерации и проверки подписи используются следующие параметры:

1) Простое число $p > 3$;
2) Эллиптическая кривая E над конечным полем F_p: $y^2 = x^3 + ax + b \mod p$;
3) Целое число m, являющееся порядком группы точек эллиптической кривой E: $m \neq p$;
4) Простое число $2^{254} < q < 2^{256}$ или $2^{508} < q < 2^{512}$: $m = n \cdot q$, где $n \in \mathbb{Z}$ и $n \geqslant 1$;
5) Точка эллиптической кривой $P = (x_P, y_P) \neq \mathcal{O}$, такая что $qP = \mathcal{O}$;
6) Закрытый ключ $0 < d < q$;
7) Открытый ключ $Q = dP = (x_Q, y_Q)$;
8) Должно быть выполнено условие: $p^t \neq 1 \mod q$, для всех целых $t = 1, 2, \ldots, B$, где $B = 31$, если $2^{254} < q < 2^{256}$, и $B = 131$, если $2^{508} < q < 2^{512}$;
9) Должно быть выполнено условие для инварианта кривой $J(E) = 1728\frac{4a^3}{4a^3+27b^2} \mod p$: $J(E) \neq 0$ и $J(E) \neq 1728$.

Генерация подписи для сообщения M происходит по следующему алгоритму:

1) С использованием хэш-функции h, определенной в стандарте [7], вычисляется значение хэш-функции $h(M)$; при этом если десятичное значение $h(M) \mod q = 0$, то необходимо присвоить $h(M) = 1$;
2) Выбирается случайное число $0 < k < q$;
3) Вычисляется точка эллиптической кривой $C = kP = (x_C, y_C)$;
4) Вычисляется значение $r = x_C \mod q$; при этом если $r = 0$, то необходимо перейти к шагу 2 и выбрать новое значение k;
5) Вычисляется значение $s = (d \cdot r + k \cdot h(M)) \mod q$; при этом если $s = 0$, то необходимо перейти к шагу 2 и выбрать новое значение k;
6) Подписью сообщения M является пара значений (r, s).

Проверка подписи (r, s) для сообщения M происходит по следующему алгоритму:

1) Необходимо проверить, что $0 < s < q$ и $0 < r < q$; если это не так, то подпись не верна;
2) С использованием хэш-функции h, определенной в стандарте [7], вычисляется значение хэш-функции $h(M)$; при этом если десятичное значение $h(M) \mod q = 0$, то необходимо присвоить $h(M) = 1$;
3) Вычисляется значение $v = (h(M))^{-1} \mod q$;
4) Вычисляются значения $z_1 = s \cdot v \mod q$ и $z_2 = -r \cdot v \mod q$;
5) Вычисляется точка эллиптической кривой $C = z_1P + z_2Q = (x_C, y_C)$;
6) Вычисляется значение $R = x_C \mod q$;
7) Подпись считается верной, если $R = r$.

4.5. Схема подписи Фиата-Шамира

Описание схемы подписи Фиата-Шамира приводится на основе работы [1] авторов данной схемы подписи. В данной схеме подписи используются следующие параметры:

1) Выбираются простые числа p и q; значения p и q являются закрытыми;
2) Вычисляется значение $n = pq$; значение n является открытым;
3) Выбираются целые числа k и t;
4) Подготавливается строка I, которая содержит всю необходимую информацию о подписавшем сообщение пользователе (его имя, адрес, идентификационный номер, физическое описание, допуск и т.д.);
5) Односторонняя функция f, которая неотличима от истинно случайной функции за полиномиальное время;
6) Вычисляются значения $v_i = f(I \parallel i)$ для $i = 1, \ldots, k$; вычисленные значения являются открытыми;
7) Вычисляются значения $s_i = \sqrt{v_i^{-1}} \mod n$ для $i = 1, \ldots, k$; вычисленные значения передаются подписывающему сообщение;

Генерация подписи для сообщения M происходит по следующему алгоритму:

1) Выбираются случайные значения $0 \leqslant r_1, \ldots, r_t < n$;
2) Вычисляются значения $x_1 = r_1^2 \mod n; \ldots; x_t = r_t^2 \mod n$;
3) С использованием некоторой односторонней функции $f(m \| x_1 \| \ldots \| x_t)$ вычисляются значения $e_{i,j}$, где $1 \leqslant i \leqslant t$, $1 \leqslant j \leqslant k$;
4) Вычисляются значения $y_i = r_i \prod\limits_{e_{i,j}=1} s_j \mod n$, для $i = 1, \ldots, t$;
5) Подписью сообщения M является набор значений $(e_{i,j}, y_i)$, где $1 \leqslant i \leqslant t$, $1 \leqslant j \leqslant k$.

Проверка подписи $(e_{i,j}, y_i)$, где $1 \leqslant i \leqslant t$, $1 \leqslant j \leqslant k$, для сообщения M происходит по следующему алгоритму:

1) Вычисляются значения $z_i = y_i^2 \prod\limits_{e_{i,j}=1} v_j \mod n$, для $i = 1, \ldots, t$;
2) С использованием некоторой односторонней функции $f(m \| z_1 \| \ldots \| z_t)$ вычисляются значения $e'_{i,j}$, где $1 \leqslant i \leqslant t$, $1 \leqslant j \leqslant k$;
3) Подпись считается верной, если $e'_{i,j} = e_{i,j}$, где $1 \leqslant i \leqslant t$, $1 \leqslant j \leqslant k$.

4.6. Схема подписи Шнорра

Описание работы схемы подписи Шнорра приводится на основе работы [8] разработчика данной схемы подписи. В схеме подписи Шнорра определены следующие параметры:

1) Простое число $p \geqslant 2^{512}$; значение p является открытым;
2) Простое число $q \geqslant 2^{140}$, при этом q является делителем $(p - 1)$; значение q является открытым;
3) Число $a \neq 1$ из поля $\mathbb{Z}_p$, для которого верно условие: $a^q = 1 \mod p$; значение a является открытым;
4) Закрытый ключ $1 \leqslant s \leqslant q$;
5) Открытый ключ $v = a^{-s} \mod p$.

Генерация подписи для сообщения M происходит по следующему алгоритму:

1) Выбирается случайное целое значение $1 \leqslant r \leqslant q$;

2) Вычисляется значение $x = a^r \mod p$;

3) С использованием некоторой хэш-функции h вычисляется значение хэш-функции $e = h(x \parallel M)$;

4) Вычисляется значение $y = r + s \cdot e \mod q$;

5) Подписью сообщения M является пара значений (e, y).

Проверка подписи (e, y) для сообщения M происходит по следующему алгоритму:

1) Вычисляется значение $x = a^y \cdot v^e \mod p$;

2) С использованием хэш-функции h вычисляется значение хэш-функции $E = h(x \parallel M)$;

3) Подпись считается верной, если $E = e$.

4.7. Схема подписи Меркля-Лампорта

4.7.1. Описание схемы подписи

Схема подписи Меркля-Лампорта описывается на основе оригинальной работы [9]. Схема состоит из двух алгоритмов подписи: подпись Лампорта — схема цифровой подписи с открытым ключом; подпись Меркля — схема подписи, основанная на использовании дерева Меркля и какой-либо одноразовой цифровой подписи, например, подписи Лампорта.

Опишем схему подписи Лампорта (схема Лампорта описывается на основе работы [10]):

1) Пусть необходимо подписать s-битовое сообщение $m = m_1, \ldots, m_s$;

2) Случайно выбирается набор из $2s$ битовых значений: $(y_{1,0}, y_{1,1}), (y_{2,0}, y_{2,1}), \ldots, (y_{s,0}, y_{s,1})$;

3) С использованием односторонней функции F вычисляются значения: $z_{i,j} = F(y_{i,j}), 1 \leqslant i \leqslant s, 0 \leqslant j \leqslant 1$;

4) Набор из $2s$ значений $y_{i,j}$ считается секретным ключом, а набор из $2s$ значений $z_{i,j}$ считается открытым ключом;

5) Подписью сообщения m является набор $(c_1, \ldots, c_s) = (y_{1,m_1}, \ldots, y_{s,m_s})$;

6) Подпись $(c_1, \ldots, c_s)$ сообщения m считается верной, если $F(c_i) = z_{i,m_i}$ для всех $1 \leqslant i \leqslant s$.

Опишем схему подписи Меркля:

1) Необходимо с помощью какого-либо алгоритма одноразовой подписи (далее рассматривается только подпись Лампорта) сформировать наборы открытых и секретных ключей длины $N = 2^n$ для некоторого значения n, являющегося глубиной будущего дерева; пусть были сформированы наборы (X_i, Y_i), где X_i — секретный ключ, а Y_i — открытый ключ;

2) Для каждого Y_i необходимо вычислить результат применения некоторой публичной хэш-функции h; результат обозначим через $H(i, i, Y)$, где $i = 1, 2, \ldots, 2^n$; данные значения являются листами дерева;

3) Каждый не листовой элемент дерева имеет двух потомков — пусть это элементы $H(a, b, Y)$ и $H(c, d, Y)$; тогда рассматриваемому элементу дерева приписывается значение $H(a, d, Y) = h(H(a, b, Y) \parallel H(c, d, Y))$;

4) Корневой элемент дерева $H(1, 2^n, Y)$ является открытым ключом схемы;

5) Генерация подписи происходит следующим образом:

   а) Определяется некоторая пара ключей (X_i, Y_i), с помощью которой для сообщения M создается цифровая подпись H;

   б) Для листа $H(i, i, Y)$ строится путь до вершины дерева; например, при $i = 5$ и $n = 3$ (см. Рисунок 2) путь до вершины будет состоять из вершин $H(5, 5, Y), H(5, 6, Y), H(5, 8, Y), H(1, 8, Y)$;

   в) Для каждого не корневого элемента в дереве существует еще один элемент с тем же родительским элементом; например, элементы $H(6, 6, Y), H(7, 8, Y), H(1, 4, Y)$; обозначим такие элементы как $H_0, H_1, \ldots, H_{n-1}$;

    г) Тогда цифровой подписью сообщения M будет являться набор $S = (H, Y_i, H_0, H_1, ..., H_{n-1})$.

6) Проверка подписи $S = (H, Y_i, H_0, H_1, ..., H_{n-1})$ сообщения M происходит следующим образом:

    а) Проверяется одноразовая подпись H сообщения M; если подпись не верна, то проверка завершается;
    б) Вычисляется значение $c_0 = h(Y_i)$;
    в) Для каждого $j = 1, ..., n$ вычисляются значения $C_j = h(C_{j-1} \| H_{j-1})$;
    г) Подпись считается верной, если C_n равен корневому элементу $H(1, 2^n, Y)$.

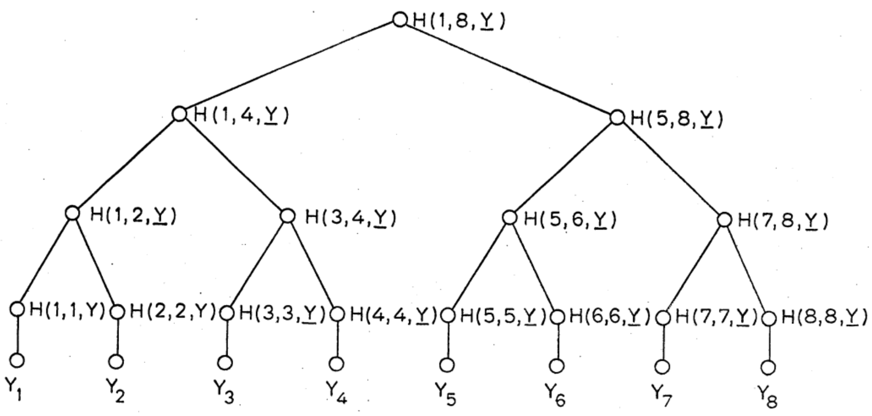

Рис. 1. Пример дерева Меркле при $n = 3$

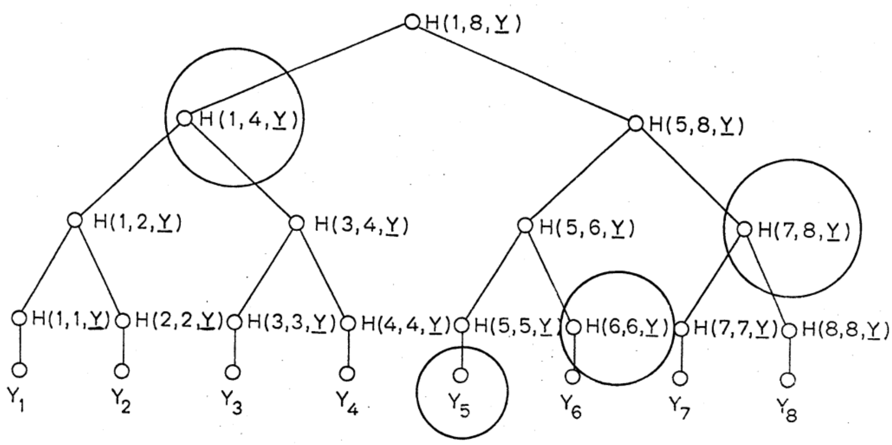

Рис. 2. Путь подписания в дереве Меркле при $n = 3$

4.7.2. Анализ схемы подписи Меркля-Лампорта

В работе [11] рассматриваются возможности проведения злоумышленником следующих атак:

1) Злоумышленник может полностью вычислить секретный ключ (FB);

2) Злоумышленник может подделать подпись для любого сообщения (UU);

3) Злоумышленник может подделать подпись для некоторых сообщений по своему выбору (SU);

4) Злоумышленник может подделать подпись для одного произвольного сообщения (EU).

При этом рассматриваются злоумышленники со следующими возможностями:

1) Злоумышленник может изучать только открытый ключ и цифровые подписи набора случайных сообщений (RMA);

2) Злоумышленник может изучать только открытый ключ и цифровые подписи набора выбранных им сообщений (CMA).

Для введенных атак и моделей злоумышленников были получены следующие оценки сложности проведения атак и вероятности успеха (далее через m обозначена длина входного сообщения M, используемого для выбора элементов секретного ключа):

1) Атака EU-CMA — сложность $O((\frac{4}{3})^{\frac{m}{3}})$ — вероятность $\frac{1}{2}$;

2) Атака SU-CMA — сложность $O((\frac{4}{3})^{\frac{m}{3}})$ — вероятность $\frac{1}{2}$;

3) Атака UU-CMA — сложность $O(2^{\frac{m}{2}})$ — вероятность $\frac{1}{2}$;

4) Атака FB-CMA — сложность $O(2^{\frac{m}{2}})$ — вероятность $\frac{1}{2}$;

5) Атака EU-RMA — сложность $O((\frac{4}{3})^{m})$ — вероятность $\frac{1}{2}$;

6) Атака SU-RMA — вероятность $(\frac{3}{4})^{m}$;

7) Атака UU-RMA — вероятность $(\frac{3}{4})^{m}$;

8) Атака FB-RMA — вероятность $(\frac{1}{2})^{\frac{m}{2}}$;

В работе [12] авторами было показано, что схему подписи Меркла практически невозможно подделать при атаке с использованием выбора сообщений (CMA). В работе отмечается, что стойкость самой схемы основывается на использовании хэш-функция, стойкой к коллизиям. Также с использованием хэш-функции RIPEMD-160 на основе одноразовой подписи Лампорта была построена схема подписи, которая является безопасной относительно вычислений с использованием квантовых компьютеров.

В работе [13] была представлена реализацию схемы подписи Меркля на 8-битном микропроцессоре смарт-карты. Реализация предназначена для 8-битных микроконтроллеров AVR, семейства 8-битных RISC-микроконтроллеров. Для реализованной схемы подписи было получено, что время создания подписи быстрее, чем у RSA, и сравнимо с ECDSA.

Как отмечается в работе [10] схема подписи Лампорта не самая практичная схема цифровой электронной подписи, так как для каждого сообщения приходится генерировать новую пару ключей. При этом также отмечается, что неизвестно никаких алгоритмов с полиномиальным временем, которые могли бы подделать фальшивую подпись алгоритма Лампорта со сколь угодно значимой вероятностью, даже при использовании квантового компьютера и квантовых вычислений.

4.8. Схема подписи CFS(Courtois, Finiasz, Sendrier)

Описание данной схемы подписи приводится на основе оригинальной статьи [14] разработчиков данной схемы. В схеме подписи используются следующие параметры:

1) Целые числа n и k, являющиеся параметрами безопасности;

2) n-мерные битовые векторы z, s, s_i, где $i = 0, 1, 2, \ldots$;

3) Хэш-функция h, выдающая $n - k$ бит на выходе;

4) Случайная невырожденная битовая матрица S из $(n - k)$ строк и $(n - k)$ столбцов;

5) Случайная перестановочная битовая матрица P из n строк и n столбцов;

6) Секретный ключ H_0 — битовая матрица из $(n - k)$ строк и n столбцов;

7) Открытый ключ $H = S \cdot H_0 \cdot P$;

Генерация подписи для сообщения M происходит по следующему алгоритму:

1) С использованием хэш-функции H вычисляется значение $s = h(M)$;
2) Вычисляются значения $s_i = h(s \parallel i)$ для каждого $i = 0, 1, 2, \ldots$;
3) Необходимо найти значение i_0 — наименьшее значение i такое, что s_i можно декодировать;
4) Декодировать s_i — то есть найти такое z, что $H \cdot z^T = s_{i_0}$;
5) Для вектора z необходимо вычислить его индекс I_z: $I_z = 1 + \binom{i_1}{1} + \ldots \binom{i_t}{t}$, где $i_1 < \ldots < i_t$ — позиции ненулевых битов в z (стоит отметить, что данное преобразование выполняется для уменьшения длины подписи);
6) Подписью сообщения M является пара (I_z, i_0).

Проверка подписи (I_z, i_0) для сообщения M происходит по следующему алгоритму:

1) Необходимо восстановить по индексу I_z значение z;
2) Вычисляется значение $s_1 = H \cdot z^T$;
3) Вычисляется значение $s_2 = h(h(M) \parallel i_0)$;
4) Подпись считается верной, если $s_1 = s_2$.

4.9. CRYSTALS–Dilithium

Описание схемы электронной подписи CRYSTALS–Dilithium приводится на основе работы [15] авторов данной схемы.

В алгоритмах схемы подписи CRYSTALS–Dilithium в зависимости от необходимого уровня стойкости могут использоваться следующие наборы параметров:

	weak	medium	recommended	very high
q	8380417	8380417	8380417	8380417
d	14	14	14	14
weight of c	60	60	60	60
$\gamma_1 = (q-1)/16$	523776	523776	523776	523776
$\gamma_2 = \gamma_1/2$	261888	261888	261888	261888
(k, ℓ)	$(3, 2)$	$(4, 3)$	$(5, 4)$	$(6, 5)$
η	7	6	5	3
β	375	325	275	175
ω	64	80	96	120

Рис. 3. Наборы параметров CRYSTALS–Dilithium

Опишем алгоритм генерации ключевой пары:

1) Выбираются 256-битные значения ρ и K;
2) Выбираются случайные векторы s_1 и s_2 размеров l и k соответственно; элементами векторов являются многочлены из кольца $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ с коэффициентами не больше η;
3) С использованием значения ρ по некоторому алгоритму строится матрица A из k строк и l столбцов с элементами из R_q;
4) Вычисляется вектор $t = (t_1, t_0) = As_1 + s_2$;
5) Вычисляется 384-битный вектор $tr = CRH(\rho \parallel t_1)$, где CRH — некоторая устойчивая к коллизиям хэш-функция;
6) Публичным ключом является пара (ρ, t_1);
7) Секретным ключом является шестерка $(\rho, K, tr, s_1, s_2, t_0)$.

Генерация подписи для сообщения M происходит по следующему алгоритму:

1) С использованием значения ρ по некоторому алгоритму строится матрица A из k строк и l столбцов с элементами из R_q;
2) Вычисляется 386-битное значение $\mu = CRH(tr \parallel M)$;
3) С использованием значений K, μ, κ (κ — итерационный коэффициент) вычисляется l-мерный вектор y, состоящий из многочленов из кольца R_q с коэффициентами не больше η;
4) Вычисляется значение Ay, старшие биты сохраняются в w_1;
5) С использованием w_1 и μ создается многочлен c из кольца R_q, имеющий ровно 60 ненулевых коэффициентов, которые равны 1 или -1; данное преобразование реализуется открытой функцией $H = H(\mu \parallel w_1)$;
6) Вычисляется значение $z = y + cs_1$, являющееся потенциальной подписью;
7) Выделяются младшие биты выражения $Ay - cs_2$ и сохраняются в r_0;
8) Анализируются коэффициенты в z и r_0: если коэффициенты не удовлетворяют определяемым параметрами схемы значениям, то необходимо перейти на новую итерацию к шагу 3;
9) Создается подсказка hдля подписи z; анализируются коэффициенты в ct_0 и число единиц в h: если коэффициенты и число единиц не удовлетворяют определяемым параметрами схемы значениям, то необходимо перейти на новую итерацию к шагу 3;
10) Подписью сообщения M считается набор значений $\sigma = (z, h, c)$.

Проверка подписи $\sigma = (z, h, c)$ для сообщения M происходит по следующему алгоритму:

1) С использованием значения ρ по некоторому алгоритму строится матрица A из k строк и l столбцов с элементами из R_q;
2) Вычисляется 386-битное значение $\mu = CRH(CRH(\rho \parallel t_1) \parallel M)$;
3) С использованием подсказки h определяется значение w_1';
4) Подпись считается верной, если:
    - все коэффициенты z меньше, чем $\gamma_1 - \beta$;
    - $c = H(\mu \parallel w_1')$;
    - число единиц в векторе-подсказке h не больше, чем w.

Подробная схема подписи CRYSTALS–Dilithium приведена на Рисунке 4.

5. Заключение

В работе были описаны основные архитектуры, используемые при построении цифровых подписей: конструкция Hash-and-Sign и преобразование Фиата-Шамира, а также были рассмотрены следующие схемы цифровых подписей: DSA, ECDSA, ГОСТ Р 34.10-1994, ГОСТ 34.10-2018, схема подписи Фиата-Шамира, схема подписи Шнорра, схема подписи CFS (Courtois, Finiasz, Sendrier), подпись CRYSTALS–Dilithium. Также более подробно была рассмотрена схема цифровой подписи Меркля-Лампорта. Схема подписи Меркля-Лампорта может считаться одной из самых стойких и безопасных криптографических схем цифровой подписи: данная схема является устойчивой как к классическим атакам, так и к атакам с использованием квантовых вычислений, а, в частности, стойкость самой схемы основывается на использовании хэш-функция, стойкой к коллизиям.

<u>Gen</u>

$\rho \leftarrow \{0,1\}^{256}$
$K \leftarrow \{0,1\}^{256}$
$(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k$
$\mathbf{A} \in R_q^{k\times\ell} := \mathsf{ExpandA}(\rho)$ // $\mathbf{A}$ is stored in NTT Domain Representation
$\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
$(\mathbf{t}_1, \mathbf{t}_0) := \mathsf{Power2Round}_q(\mathbf{t}, d)$
$tr \in \{0,1\}^{384} := \mathsf{CRH}(\rho \parallel \mathbf{t}_1)$
return $(pk = (\rho, \mathbf{t}_1), sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0))$

<u>$\mathsf{Sign}(sk, M)$</u>

$\mathbf{A} \in R_q^{k\times\ell} := \mathsf{ExpandA}(\rho)$ // $\mathbf{A}$ is stored in NTT Domain Representation
$\mu \in \{0,1\}^{384} := \mathsf{CRH}(tr \parallel M)$
$\kappa := 0$, $(\mathbf{z}, \mathbf{h}) := \perp$
while $(\mathbf{z}, \mathbf{h}) = \perp$ do
  $\mathbf{y} \in S_{\gamma_1 - 1}^\ell := \mathsf{ExpandMask}(K \parallel \mu \parallel \kappa)$
  $\mathbf{w} := \mathbf{A}\mathbf{y}$
  $\mathbf{w}_1 := \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
  $c \in B_{60} := \mathsf{H}(\mu \parallel \mathbf{w}_1)$
  $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$
  $(\mathbf{r}_1, \mathbf{r}_0) := \mathsf{Decompose}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ or $\mathbf{r}_1 \neq \mathbf{w}_1$, then $(\mathbf{z}, \mathbf{h}) := \perp$
  else
    $\mathbf{h} := \mathsf{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$
    if $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$ **or** the # of 1's in $\mathbf{h}$ is greater than ω, then $(\mathbf{z}, \mathbf{h}) := \perp$
  $\kappa := \kappa + 1$
return $\sigma = (\mathbf{z}, \mathbf{h}, c)$

<u>$\mathsf{Verify}(pk, M, \sigma = (\mathbf{z}, \mathbf{h}, c))$</u>

$\mathbf{A} \in R_q^{k\times\ell} := \mathsf{ExpandA}(\rho)$ // $\mathbf{A}$ is stored in NTT Domain Representation
$\mu \in \{0,1\}^{384} := \mathsf{CRH}(\mathsf{CRH}(\rho \parallel \mathbf{t}_1) \parallel M)$
$\mathbf{w}_1' := \mathsf{UseHint}_q(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$
return $[\![\|\mathbf{z}\|_\infty < \gamma_1 - \beta]\!]$ **and** $[\![c = \mathsf{H}(\mu \parallel \mathbf{w}_1')]\!]$ **and** $[\![$# of 1's in $\mathbf{h}$ is $\leq \omega]\!]$

Рис. 4. Схема алгоритма CRYSTALS–Dilithium

Список литературы

1. *Fiat A.*, *Shamir A.* How To Prove Yourself: Practical Solutions to Identification and Signature Problems // Advances in Cryptology — CRYPTO' 86. CRYPTO 1986. Lecture Notes in Computer Science, vol 263. — 1987.
2. FIPS PUB 186: Digital Signature Standard (DSS) // Federal Information Processing Standards Publication 186. — 1994.
3. *Johnson D.*, *Menezes A.*, *Vanstone S.* The Elliptic Curve Digital Signature Algorithm (ECDSA) // International Journal of Information Security. — 2001.
4. ГОСТ Р 34.10-94 Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма. — 1994.
5. ГОСТ Р 34.11—94 Информационная технология. Криптографическая защита информации. Функция хэширования. — 1994.
6. ГОСТ 34.10-2018 Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. — 2018.
7. ГОСТ 34.11—2018 Информационная технология. Криптографическая защита информации. Функция хэширования. — 2018.
8. *Schnorr C.* Efficient signature generation by smart cards // Cryptology 4, 161–174. — 1991.
9. *Merkle R. C.* Secrecy, authentication, and public key systems. — 1979.
10. *Zentai D.* On the Efficiency of the Lamport Signature Scheme // Land Forces Academy Review. — 2020. — Т. 25. — С. 275—280.

11. *Groot Bruinderink L.*, *Hülsing A.* “Oops, I did it again” – Security of One-Time Signatures under Two-Message Attacks // Selected Areas in Cryptography – SAC 2017. SAC 2017. Lecture Notes in Computer Science(), vol 10719. — 2017.

12. *Coronado C.* On the security and the efficiency of the Merkle signature scheme // IACR Cryptol. ePrint Arch. — 2005. — Т. 2005. — С. 192.

13. Fast Hash-Based Signatures on Constrained Devices / S. Rohde [и др.] // Smart Card Research and Advanced Applications. CARDIS 2008. Lecture Notes in Computer Science, vol 5189. — 2008.

14. *Courtois N.*, *Finiasz M.*, *Sendrier N.* How to Achieve a McEliece-Based Digital Signature Scheme // Advances in Cryptology — ASIACRYPT 2001. ASIACRYPT 2001. Lecture Notes in Computer Science, vol 2248. — 2001.

15. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme / L. Ducas [и др.] // IACR Transactions on Cryptographic Hardware and Embedded Systems, 238–268. — 2018.