**Министерство науки и высшего образования Российской Федерации**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Выпускная Квалификационная Работа**

**На тему:**

«Разработка интерактивной методики оценки эффективности систем цифровой подписи на основе библиотеки PyCryptodome»

**Выполнил:**

Гутник Дмитрий Вячеславович,

студент группы N34491

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Таранов Сергей Владимирович,

доцент факультета БИТ

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(отметка о выполнении)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

Санкт-Петербург

2023г.

Содержание

[Введение 3](#_Toc133011448)

[1 Анализ и 1юение на виды актуальных алгоритмов цифровой подписи 5](#_Toc133011449)

[1.1 PKCS#1 v1.5 5](#_Toc133011450)

[1.2 PKCS#1 PSS 11](#_Toc133011451)

[1.3 EdDSA, HashEdDSA, PureEdDSA 16](#_Toc133011452)

[1.4 DSA 21](#_Toc133011453)

[1.5 ECDSA 25](#_Toc133011454)

[2 Описание критериев к методике оценки цифровых подписей 28](#_Toc133011455)

[3 Описание методики оценки эффективности пороговых подписей 29](#_Toc133011456)

[4 Описание программной реализации методики 30](#_Toc133011457)

[5 Рекомендации к практическому использования методики 31](#_Toc133011458)

[Заключение 32](#_Toc133011459)

[Список использованных источников 33](#_Toc133011460)

Введение

Современная информационная эпоха требует надежных и безопасных методов передачи данных. Одним из таких методов является использование систем цифровой подписи, которые позволяют обеспечить аутентичность и целостность информации. Однако, эффективность таких систем может быть ограничена различными факторами, такими как выбор алгоритма подписи, длина ключа и многими другими. В данном дипломном проекте рассматривается разработка интерактивной методики оценки эффективности систем цифровой подписи на основе библиотеки PyCryptodome. Эта методика позволит улучшить качество и безопасность передачи данных в современном информационном мире.

**Актуальность** дипломной работы заключается в том, что согласно TIOBE [[1]](#TIOBE) на апрель 2023 года Python является самым популярным языком программирования. Рейтинг самых популярных языков программирования расположен на рисунке 1

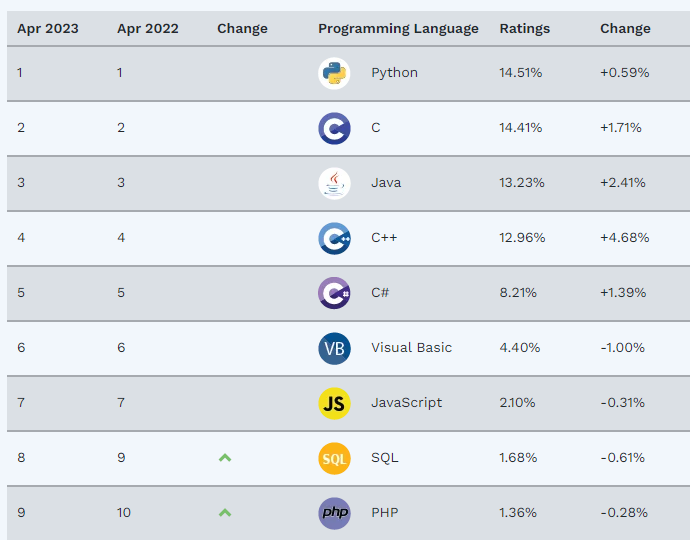


Рисунок 1 – Самые популярные языки программирования на апрель 2023 года [[1]](#TIOBE)

А PyCryptodome, в свою очередь, является одной из самых популярных библиотек [[2]](#tleapps) для имплементации криптографических протоколов и алгоритмов, эти факторы в купе с поддерживаемыми алгоритмами электронной подписи обусловливают выбор данной библиотеки в качестве базиса создание методики, а её пакета Crypto.Signature в качестве объекта исследования данной выпускной квалификационной работы .

**Цель работы** – Повышение эффективности применения алгоритмов цифровой подписи путём рекомендации подходящего вида ЦП в зависимости от условий применения

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Анализ и разделение на виды актуальных алгоритмов цифровой подписи

2. Разработка критериев к методике оценки цифровых подписей

3. Создание методики оценки эффективности пороговых подписей

4. Разработка программной реализации методики

5. Разработка рекомендаций к практическому использования методики

# Анализ и 1юение на виды актуальных алгоритмов цифровой подписи

Согласно официальной документации [[3]](#documentation) Библиотека PyCryptodome поддерживает 3 алгоритма электронной подписи, каждый из которых имеет две вариации.

Алгоритмы:

1. RSA со следующими вариациями:

* PKCS#1 v1.5
* PKCS#1 PSS

1. EdDSA со следующими вариациями:

* EdDSA
* Pure EdDSA

1. DSA со следующими вариациями:

* DSA
* ECDSA

Ниже рассмотрены вышеперечисленные алгоритмы и их вариации в соответствии с их документацией.

## PKCS#1 v1.5

Согласно RFC 8017 [[4]](#Rfc8017) PKCS#1 v1.5, далее RSASSA-PKCS1-v15 объединяет примитивы RSASP1 и RSAVP1 с методом кодирования EMSA-PKCS1-v15. Он совместим со схемой IFSSA, определенной в IEEE [[5]](#ieee1363) 13639, где примитивы подписи и проверки являются IFSP-RSA1 и IFVP-RSA1, а метод кодирования сообщения - EMSA-PKCS1-v15

Длина сообщений, на которых может работать RSASSA-PKCS1-v15, либо не ограничена, либо ограничена очень большим числом, в зависимости от хэш-функции, лежащей в основе метода кодирования EMSA-PKCS1-v15.

Предполагая, что вычисление e-ых корней по модулю n является невыполнимым и хэш-функция в EMSA-PKCS1-v15 обладает соответствующими свойствами, предполагается, что RSASSA-PKCS1-v15 обеспечивает безопасную подпись. Более точно, подделка подписей без знания закрытого ключа RSA предполагается вычислительно невозможной. Кроме того, в методе кодирования EMSA-PKCS1-v15 идентификатор хэш-функции встроен в кодирование. Из-за этой особенности злоумышленник, пытающийся найти сообщение с той же подписью, что и ранее подписанное сообщение, должен найти коллизии конкретной используемой хэш-функции; атака на другую хэш-функцию, отличную от выбранной подписантом, не является полезной для злоумышленника.

Примечание: как отмечено в PKCS #1 v1.5, метод кодирования EMSA-PKCS1-v15 имеет свойство того, что закодированное сообщение, преобразованное в целочисленное представление сообщения, гарантированно большое и по крайней мере отчасти "случайное". Это предотвращает атаки, предложенные Десмедтом и Одлизко, где мультипликативные отношения между представителями сообщений разрабатываются путем факторизации представителей сообщений на набор малых значений (например, набор малых простых чисел). Корон, Накаш и Стерн показали, что более сильная форма этого типа атаки может быть довольно эффективной против некоторых случаев схемы подписи ISO/IEC 9796-2. Они также проанализировали сложность этого типа атаки на метод кодирования EMSA-PKCS1-v15 и пришли к выводу, что атака будет непрактичной, требуя больше операций, чем поиск коллизий в основной хэш-функции (т.е. более 2^80 операций). Coppersmith, Halevi и Jutla впоследствии расширили атаку Корон и др., Чтобы сломать схему подписи ISO/IEC 9796-1 с восстановлением сообщения. Различные атаки иллюстрируют важность тщательного построения входных данных для примитива RSA-подписи, особенно в схеме подписи с восстановлением сообщения. Следовательно, метод кодирования EMSA-PKCS-v15 явно включает хэш-операцию и не предназначен для схем подписи с восстановлением сообщения. Кроме того, хотя атака на метод кодирования EMSA-PKCS-v15 неизвестна, рекомендуется постепенный переход к EMSA-PSS в качестве предосторожности против будущих разработок.

Операция генерации подписи

(1)

Вход:

K - закрытый ключ RSA подписанта

M - сообщение для подписи, строка октетов

Выход:

S подпись, строка октетов длиной k, где k - длина в октетах модуля RSA n

Ошибки: "сообщение слишком длинное"; "модуль RSA слишком короткий"

Шаги:

1. Кодирование EMSA-PKCS1-v15: примените операцию кодирования EMSA-PKCS1-v15 к сообщению M, чтобы получить закодированное сообщение EM длиной k октетов:

Если операция кодирования выводит "сообщение слишком длинное", выведите "сообщение слишком длинное" и остановитесь. Если операция кодирования выводит "намеренная длина закодированного сообщения слишком короткая", выведите "модуль RSA слишком короткий" и остановитесь.

2. RSA-подпись:

a. Преобразование закодированного сообщения EM в целочисленное представление сообщения m:

(3)

b. Применение примитива подписи RSASP1 к закрытому ключу RSA K и целочисленному представлению сообщения m для получения целочисленного представителя подписи s:

(4)

c. Преобразование целочисленного представителя подписи s в подпись S длиной k октетов:

(5)

3. Вывод подписи S.

Операция проверки подписи

(6)

Вход:

(n, e) - открытый ключ RSA подписанта

M - сообщение, подпись которого требуется проверить, октетовая строка

S - подпись, которую необходимо проверить, октетовая строка длиной k,

где k - длина в октетах модуля RSA n

Выход: "верная подпись" или "неверная подпись"

Ошибки: "слишком длинное сообщение"; "слишком короткий модуль RSA".

Шаги:

1. Проверка длины: если длина подписи S не равна k октетам, вывести "неверная подпись" и остановиться.

2. Проверка RSA:

a. Преобразование подписи S в целочисленный представитель подписи s:

(7)

b. Применение примитива проверки RSAVP1 к открытому ключу RSA (n, e) и целочисленному представителю подписи s для получения целочисленного представителя сообщения m:

(8)

Если RSAVP1 выдает "целочисленный представитель подписи вне диапазона", вывести "неверная подпись" и остановиться.

c. Преобразование целочисленного представителя сообщения m в закодированное сообщение EM длиной k октетов:

(9)

Если I2OSP выдает "слишком большое целое число", вывести "неверная подпись" и остановиться.

3. Кодирование EMSA-PKCS1-v1\_5: Применение операции кодирования EMSA-PKCS1-v1\_5 к сообщению M для получения второго закодированного сообщения EM' длиной k октетов:

(10)

Если операция кодирования выдает "слишком длинное сообщение", вывести "слишком длинное сообщение" и остановиться. Если операция кодирования выдает "недостаточная длина закодированного сообщения", вывести "слишком короткий модуль RSA" и остановиться.

4. Сравнение закодированных сообщений EM и EM'. Если они совпадают, вывести "верная подпись"; в противном случае, вывести "неверная подпись".

Примечание: Другой способ реализации операции проверки подписи - применение "декодирования" (не указанного в данном документе) к закодированному сообщению для восстановления базового хеш-значения, а затем сравнение его с вновь вычисленным хеш-значением. Это имеет преимущество в том, что требуется меньше промежуточного хранения (два хеш-значения вместо двух закодированных сообщений), но недостаток в том, что требуется дополнительный код.

## PKCS#1 PSS

Согласно RFC 8017 [[4]](#Rfc8017), PKCS#1 PSS, далее RSASSA-PSS объединяет примитивы RSASP1 и RSAVP1 с методом кодирования EMSA-PSS. Он совместим с схемой подписи с приложением Integer Factorization Signature Scheme with Appendix (IFSSA), как это определено в IEEE [[5]](#ieee1363) 1363, где примитивы подписи и проверки - IFSP-RSA1 и IFVP-RSA1, а метод кодирования сообщения - EMSA4. EMSA4 немного более общий, чем EMSA-PSS, так как он действует на битовые строки, а не на октетные строки. EMSA-PSS эквивалентен EMSA4, ограниченному случаем, когда операнды, а также значения хеша и соли являются октетными строками.

Длина сообщений, на которых может работать RSASSA-PSS, либо неограничена, либо ограничена очень большим числом, в зависимости от хеш-функции, лежащей в основе метода кодирования EMSA-PSS.

Предполагая, что вычисление e-й корней по модулю n невозможно и функции генерации хеша и маски в EMSA-PSS имеют соответствующие свойства, RSASSA-PSS обеспечивает безопасные подписи. Это заверение может быть доказано в том смысле, что сложность подделки подписей может быть непосредственно связана со сложностью инвертирования функции RSA, при условии, что функции генерации хеша и маски рассматриваются как черные ящики или случайные оракулы. Границы в доказательстве безопасности являются в основном "жесткими", что означает, что вероятность успеха и время выполнения для лучшего фальсификатора против RSASSA-PSS очень близки к соответствующим параметрам для лучшего алгоритма инвертирования RSA.

В отличие от схемы подписи RSASSA-PKCS1-v1\_5, идентификатор хеш-функции не встроен в закодированное сообщение EMSA-PSS, поэтому в теории возможно, чтобы злоумышленник заменил выбранную подписантом хеш-функцию на другую (и потенциально более слабую). Поэтому рекомендуется, чтобы функция генерации маски EMSA-PSS была основана на той же хеш-функции. Таким образом, вся закодированная сообщение будет зависеть от хеш-функции, и для злоумышленника будет трудно заменить другую хеш-функцию, чем ту, которую задумал подписант. Это соответствие хеш-функций необходимо только для предотвращения замены хеш-функций и не является необходимым, если замена хеш-функций решается другими средствами (например, проверяющий принимает только определенную хеш-функцию). См. для дальнейшего обсуждения этих вопросов. Доказуемая безопасность RSASSA-PSS не зависит от того, является ли функция генерации маски хеш-функцией, применяемой к сообщению.

RSASSA-PSS отличается от других схем подписи на основе RSA тем, что он вероятностный, а не детерминированный, включая случайно генерируемое значение соли. Значение соли повышает безопасность схемы, предоставляя "более жесткое" доказательство безопасности, чем детерминированные альтернативы, такие как Full Domain Hashing (FDH). Однако случайность не является критической для безопасности. В ситуациях, когда случайная генерация невозможна, можно использовать фиксированное значение или номер последовательности, и результативная доказуемая безопасность будет аналогична FDH [FDH].

Операция генерации подписи

(11)

Входные данные:

K - закрытый ключ RSA подписчика

M - сообщение, которое должно быть подписано, октетовая строка

Выходные данные:

S - подпись, октетовая строка длиной k, где k - длина в октетах модуля RSA n

Ошибки: "сообщение слишком длинное"; "ошибка кодирования"

Шаги:

1. Кодирование EMSA-PSS: Применить операцию кодирования EMSA-PSS к сообщению M, чтобы получить закодированное сообщение EM длиной ceil ((modBits - 1)/8) октетов так, чтобы длина битового представления целого числа OS2IP (EM) была не больше modBits - 1, где modBits - длина в битах модуля RSA n:

(12)

Обратите внимание, что длина октетового представления EM будет на один меньше k, если modBits - 1 кратно 8, и равна k в противном случае.

Если операция кодирования выдает "сообщение слишком длинное", выдать "сообщение слишком длинное" и остановиться.

Если операция кодирования выдает "ошибка кодирования", выдать "ошибка кодирования" и остановиться.

2. RSA подпись:

a. Преобразование закодированного сообщения EM в представление целого числа m

(13)

b. Применение примитива подписи RSASP1 к закрытому ключу RSA K

и представлению целого числа сообщения m, чтобы получить представление целого числа подписи s:

(14)

c. Преобразование представления целого числа подписи s в подпись S длиной k октетов

(15)

3. Вывод подписи S.

Операция проверки подписи

(16)

Входные данные:

(n, e) - открытый ключ RSA подписчика

M - сообщение, подпись которого должна быть проверена, октетовая строка

S - подпись, которую нужно проверить, октетовая строка длиной k, где k - длина в октетах модуля RSA n

Выходные данные: "верная подпись" или "неверная подпись"

Шаги:

1. Проверка длины: если длина подписи S не равна k октетам, выдать "неверная подпись" и остановиться.

2. Проверка RSA подписи:

a. Преобразование подписи S в представление целого числа подписи s:

(17)

b. Применение примитива верификации RSAVP1 к открытому ключу RSA (n, e) и представлению целого числа подписи s, чтобы получить представление целого числа сообщения m:

(18)

Если RSAVP1 выдает "signature representative out of range", выдать "invalid signature"и остановиться.

c. Преобразование представления целого числа сообщения m в закодированное сообщение EM длиной emLen = \ceil ((modBits - 1)/8) октетов, где modBits - длина в битах модуля RSA n:

(19)

Обратите внимание, что если modBits - 1 кратно 8, то emLen будет на один меньше k,а в противном случае emLen равен k. Если I2OSP выдает "integer too large", выдать "invalid signature" и остановиться.

3. Проверка EMSA-PSS: Применить операцию проверки EMSA-PSS к сообщению M и закодированному сообщению EM, чтобы определить, соответствуют ли они друг другу:

(20)

4. Если Result = "consistent", выдать "valid signature". В противном случае выдать "invalid signature".

# EdDSA, HashEdDSA, PureEdDSA

Согласно RFC8032 [[7]](#RFC8032) EdDSA— это система цифровой подписи с 11 параметрами.

Общая система цифровой подписи EdDSA с ее 11 входными параметрами не предназначена для прямой реализации. Выбор параметров критичен для безопасной и эффективной работы. Вместо этого вы должны реализовать конкретный выбор параметров для EdDSA (например, Ed25519 или Ed448), иногда немного обобщенный для повторного использования кода для охвата Ed25519 и Ed448.

Поэтому точное объяснение общего EdDSA не особенно полезно для реализаторов. Для контекста и полноты здесь приводится краткое описание общего алгоритма EdDSA.

Определение некоторых параметров, таких как n и c, может помочь объяснить некоторые шаги алгоритма, которые неочевидны.

EdDSA имеет 11 параметров:

1. Нечетная простая степень p. EdDSA использует эллиптическую кривую над конечным полем GF(p).

2. Целое число b с 2^(b-1)> p. У EdDSA открытые ключи имеют ровно b бит, а подписи EdDSA имеют ровно 2b бит. Рекомендуется, чтобы b было кратно 8, так чтобы длины открытого ключа и подписи были целым числом октетов.

3. (b-1)-битное кодирование элементов конечного поля GF(p).

4. Криптографическая хеш-функция H, производящая выходной бит 2b. Рекомендуются консервативные хеш-функции (то есть хеш-функции, где создание коллизий невозможно) и они не оказывают большого влияния на общую стоимость EdDSA.

5. Целое число c, которое равно 2 или 3. Секретные скаляры EdDSA являются кратными 2^c. Целое число c является двоичным логарифмом так называемого кофактора.

6. Целое число n с c <= n < b. Секретные скаляры EdDSA имеют ровно n + 1 бит, с наивысшим битом (позиция 2^n) всегда установленным и нижними c битами всегда очищенными.

7. Не квадратичный элемент d из GF(p). Обычное рекомендуемое значение — это ближайшее к нулю значение, которое дает приемлемую кривую.

8. Ненулевой квадратичный элемент a из GF(p). Обычная рекомендация для лучшей производительности — это a = -1, если p mod 4 = 1, и a = 1, если p mod 4 = 3.

9. Элемент B!= (0,1) множества E = { (x,y) является членом GF(p) x GF(p), таким что a x^2 + y^2 = 1 + d x^2 y^2 }.

10. Нечетное простое число L такое, что [L]B = 0 и 2^c L = #E. Число #E (количество точек на кривой) является частью стандартных данных, предоставленных для эллиптической кривой E, или его можно вычислить как кофактор порядок.

11. Функция "прехэширования" PH. PureEdDSA означает EdDSA, где PH является идентификационной функцией, то есть PH(M) = M. HashEdDSA означает EdDSA, где PH генерирует короткий вывод, независимо от того, насколько длинным является сообщение; например, PH(M) = SHA-512(M).

Точки на кривой образуют группу при сложении, (x3, y3) = (x1, y1) + (x2, y2), с формулами [21](#формула21) и [22](#формула22)

(21)

(22)

Нейтральный элемент в группе - (0,1).

В отличие от многих других кривых, используемых для криптографических приложений, эти формулы являются "полными"; они действительны для всех точек на кривой без исключений. В частности, знаменатели ненулевые для всех входных точек.

Есть более эффективные формулы, которые все еще являются полными и используют однородные координаты, чтобы избежать дорогостоящих инверсий по модулю p.

**Кодирование**

Целое число 0 < S < L - 1 кодируется в форме little-endian как строка из b битов ENC(S).

Элемент (x,y) E кодируется как строка из b битов, называемая ENC(x,y), которая является кодированием (b-1)-битов y, объединенных с одним битом, который равен 1, если x отрицательный, и 0, если x не отрицательный.

Кодирование GF(p) используется для определения "отрицательных" элементов GF(p): конкретно, x является отрицательным, если кодирование (b-1)-битов x лексикографически больше кодирования (b-1)-битов -x.

**Ключи**

Приватный ключ EdDSA представляет собой строку из b битов k. Пусть хэш H(k) = (h0, h1, ..., h(2b-1)) определяет целое число s, которое равно 2^n плюс сумме m = 2^i \* hi для всех целых i, c <= i < n. Пусть s определяет кратное A = sB. Публичный ключ EdDSA — это ENC(A). Биты hb, ..., h(2b-1) используются при подписании.

**Подпись**

Подпись EdDSA сообщения M под приватным ключом k определяется как подпись PureEdDSA PH(M). Другими словами, EdDSA просто использует PureEdDSA для подписи PH(M).

Подпись PureEdDSA сообщения M под приватным ключом k - это строка из 2b битов ENC(R) || ENC(S). R и S определяются следующим образом. Сначала определим r = H(h\_b || ... || h\_(2b-1) || M), интерпретируя строки из 2b битов в форме little-endian как целые числа в {0, 1, ..., 2^(2b) - 1}. Пусть R = [r]B и S = (r + H(ENC(R) || ENC(A) || PH(M)) s) mod L.

**Проверка**

Чтобы проверить подпись PureEdDSA ENC(R) || ENC(S) сообщения M под публичным ключом ENC(A), выполните следующие действия. Разберите входные данные так, чтобы A и R были элементами E, а S был членом множества {0, 1, ..., L-1}. Вычислите h = H(ENC(R)|| ENC(A) || M) и проверьте групповое уравнение 2^c \* S B = 2^c R + [2^c \* h] A в E. Подпись отклоняется, если разбор не удался (включая S, находящийся вне диапазона) или если групповое уравнение не выполняется.

Проверка EdDSA для сообщения M определяется как проверка PureEdDSA для PH(M).

PureEdDSA, HashEdDSA и именование

Один из параметров алгоритма EdDSA — это "prehash" функция. Она может быть идентичной функцией, что приведет к алгоритму с названием PureEdDSA, или функцией хэширования, такой как SHA-512, что приведет к алгоритму с названием HashEdDSA.

Выбор варианта зависит от того, какое свойство считается более важным между 1) устойчивостью к коллизиям и 2) интерфейсом однопроходной подписи. Свойство устойчивости к коллизиям означает, что EdDSA защищен даже в случае возможности вычисления коллизий для хэш-функции. Свойство интерфейса однопроходной подписи означает, что для создания подписи требуется только один проход по входному сообщению. PureEdDSA требует двух проходов по входу. Многие существующие API, протоколы и среды предполагают, что цифровые алгоритмы подписи требуют только один проход по входу и могут иметь проблемы с API или пропускной способностью при использовании других вариантов.

Обратите внимание, что однопроходная проверка невозможна с большинством использований подписей, независимо от выбранного алгоритма подписи. Это связано с тем, что большую часть времени сообщение нельзя обработать до того, как подпись будет проверена, что требует прохода по всему сообщению.

# DSA

Ниже представлен анализ алгоритма DSA в соответствии с FIPS PUB 186-4 [[6]](#FIPSPUB186_4)

**Параметры DSA**

Цифровая подпись DSA вычисляется с помощью набора параметров области, закрытого ключа x, секретного числа k для каждого сообщения, данных для подписи и хэш-функции. Цифровая подпись проверяется с помощью тех же параметров области, открытого ключа y, который математически связан с закрытым ключом x, используемым для генерации цифровой подписи, данных для проверки и той же хэш-функции, которая использовалась при генерации подписи. Эти параметры определяются следующим образом:

p - простое модуль, где 2L-1 < p < 2L, а L - длина битов p.

q - простой делитель (p-1), где 2N-1 < q < 2 N, а N - длина битов q.

g - генератор подгруппы порядка q в мультипликативной группе GF(p), такой что 1 < g < p.

x - закрытый ключ, который должен оставаться секретным; x является случайно или псевдослучайно сгенерированным целым числом, таким что 0 < x < q, т.е. x находится в диапазоне 1, q-1.

y - открытый ключ, где y = gx mod p.

k - секретное число, уникальное для каждого сообщения; k является случайно или псевдослучайно сгенерированным целым числом, таким что 0 < k < q, т.е. k находится в диапазоне 1, q-1.

**Выбор размеров параметров и хэш-функций для DSA**

Этот стандарт определяет следующие выборы для пары L и N (длины битов p и q соответственно):

L = 1024, N = 160

L = 2048, N = 224

L = 2048, N = 256

L = 3072, N = 256

**DSA уникальный Секретный Номер**

Перед генерацией каждой цифровой подписи должен быть сгенерирован новый секретный случайный номер k для использования в процессе генерации подписи. Этот секретный номер должен быть защищен от несанкционированного раскрытия и модификации.

является мультипликативным обратным к k по умножению по модулю q; т.е. 0 < < q и 1 = (k) mod q. Этот обратный элемент требуется для процесса генерации подписи.

k и могут быть предварительно вычислены, так как для вычислений не требуется знание сообщения, которое будет подписано. Когда k и предварительно вычислены, их конфиденциальность и целостность должны быть защищены.

Пусть N - это длина q в битах. Пусть min(N, outlen) обозначает минимум из положительных целых чисел N и outlen, где outlen - это длина блока вывода хэш-функции в битах.

Подпись сообщения M состоит из пары чисел r и s, которые вычисляются в соответствии с следующими уравнениями:

(23)

z - это первые min(N, outlen) бит хэша (Hash) M.

(24)

При вычислении s строка z, полученная из Hash(M), должна быть преобразована в целое число.

Обратите внимание, что r может быть вычислен, когда k, p, q и g доступны, например, когда известны параметры области p, q и g, и k был предварительно вычислен, r также может быть предварительно вычислен, так как для вычисления r не требуется знание сообщения, которое будет подписано. Значения предварительно вычисленных k, k-1 и r должны быть защищены так же, как и закрытый ключ x до тех пор, пока не будет вычислено s.

Значения r и s должны быть проверены для определения, равно ли r = 0 или s = 0. Если r = 0 или s = 0, должно быть сгенерировано новое значение k, и подпись должна быть пересчитана. Если подписи генерируются правильно, то крайне маловероятно, что r = 0 или s = 0.

Подпись (r, s) может быть передана вместе с сообщением верификатору.

**Проверка и валидация подписи DSA**

Проверка подписи может быть выполнена любой стороной (т.е. подписантом, предполагаемым получателем или любой другой стороной) с использованием открытого ключа подписанта. Подписант может захотеть проверить правильность вычисленной подписи, возможно, перед отправкой подписанного сообщения предполагаемому получателю. Предполагаемый получатель (или любая другая сторона) проверяет подпись, чтобы определить ее подлинность.

Перед проверкой подписи подписанного сообщения параметры области и утвержденный открытый ключ, и идентификатор предполагаемого подписанта должны быть предоставлены верификатору в аутентифицированном виде. Например, открытый ключ может быть получен в виде сертификата, подписанного доверенной стороной (например, ЦС) или на личной встрече с владельцем открытого ключа.

Пусть M ', r' и s 'будут полученными версиями M, r и s соответственно; пусть y будет открытым ключом утвержденного подписанта; и пусть N - это длина q в битах. Также пусть min(N, outlen) обозначает минимум из положительных целых чисел N и outlen, где outlen - это длина блока вывода хэш-функции в битах.

Процесс проверки подписи выглядит следующим образом:

1. Верификатор должен проверить, что 0 < r ' < q и 0 < s' < q; если одно из условий не выполняется, подпись должна быть отклонена как недействительная.

2. Если два условия в шаге 1 выполнены, верификатор вычисляет следующее:

(25)

z = первые min(N, outlen) бит хэша (Hash) M'.

(26)

(27)

(28)

# ECDSA

Ниже представлен анализ алгоритма EcDSA в соответствии с FIPS PUB 186-4 [[6]](#FIPSPUB186_4)

**Алгоритм цифровой подписи на эллиптических кривых (ECDSA)**

Стандарт цифровой подписи на эллиптических кривых (ECDSA), был разработан для Американского национального института стандартов Аккредитованным комитетом по стандартам финансовых услуг X9.

ANS X9.62 определяет методы генерации и проверки цифровой подписи с использованием алгоритма цифровой подписи на эллиптических кривых (ECDSA). Также в ANS X9.62 включены спецификации для генерации параметров области, используемых при генерации и проверке цифровых подписей. ECDSA является эллиптической кривой аналогом DSA. Ключи ECDSA не должны использоваться для каких-либо других целей (например, установки ключа).

**Генерация параметров области**

Данный стандарт определяет пять диапазонов для n (см. [таблицу 1](#Таблица1)). Для каждого диапазона также указывается максимальный размер кофактора. Следует отметить, что спецификация кофактора h в наборе параметров области является необязательной в ANS X9.62, тогда как реализации, соответствующие FIPS 186-4, должны указывать кофактор h в наборе параметров области. [Таблица 1](#Таблица1) предоставляет максимальные размеры для кофактора h.

|  |  |
| --- | --- |
| Длинна n в битах | Максимальный кофактор (h) |
| 160-223 |  |
| 224-255 |  |
| 256-383 |  |
| 384-511 |  |
| ≥ 512 |  |

Таблица 1 – Параметры безопасности ECDSA

ECDSA определен для двух арифметических полей: конечного поля GFp и конечного поля GF m2. Для поля GFp требуется, чтобы p было нечетным простым числом.

Предоставляются три типа кривых:

1. Кривые над простыми полями, которые идентифицируются как P-xxx,

2. Кривые над бинарными полями, которые идентифицируются как B-xxx

3. Кривые Коблица, которые идентифицируются как K-xxx,

где xxx указывает битовую длину размера поля.

**Приватные/публичные ключи**

Пара ключей ECDSA состоит из приватного ключа d и публичного ключа Q, который связан с определенным набором параметров области ECDSA; d, Q и параметры области математически связаны друг с другом. Приватный ключ обычно используется в течение определенного периода времени (т.е. криптопериода); публичный ключ может продолжать использоваться так долго, как требуется проверка цифровых подписей, сгенерированных с использованием связанного приватного ключа (т.е. публичный ключ может продолжать использоваться после криптопериода связанного приватного ключа). Дополнительные руководства можно найти в SP 800-57.

Ключи ECDSA могут использоваться только для генерации и проверки цифровых подписей ECDSA.

**Генерация пары ключей**

Генерируется пара ключей цифровой подписи d и Q для набора параметров области (q, FR, a, b {, domain\_paramete\_rseed}, G, n, h).

**Генерация секретного числа**

Перед генерацией каждой цифровой подписи должно быть сгенерировано новое секретное случайное число k для использования в процессе генерации подписи. Это секретное число должно быть защищено от несанкционированного раскрытия и модификации.

является мультипликативным обратным к k по умножению по модулю n; т.е. 0 < < n и 1 = ( k) mod n. Этот обратный элемент необходим для процесса генерации подписи.

k и могут быть предварительно вычислены, поскольку знание сообщения, которое будет подписано, не требуется для вычислений. При предварительном вычислении k и их конфиденциальность и целостность должны быть защищены.

**Генерация и проверка цифровой подписи ECDSA**

Цифровая подпись ECDSA (r, s) должна быть сгенерирована в соответствии с ANS X9.62, используя:

1. Параметры области, правила генерации которых описаны выше

2. Личный ключ, правила генерации которого описаны выше

3. Секретное число для каждого сообщения, правила генерации которого описаны выше

4. Утвержденную хэш-функцию

5. Утвержденный генератор случайных битов.

Цифровая подпись ECDSA должна быть проверена в соответствии с ANS X9.62, используя те же параметры области и хэш-функцию, которые использовались при генерации подписи.

# Описание критериев к методике оценки цифровых подписей

# Описание методики оценки эффективности пороговых подписей

# Описание программной реализации методики

# Рекомендации к практическому использования методики

Заключение

Было выполнено то-се пятое-десятое…

Это позволили получить / закрепить навыки и т.п…

Мы все сделали, мы молодцы, поставьте нам **отлично автоматом**.

Список использованных источников

1. TIOBE Index for April 2023 – URL:

<https://www.tiobe.com/tiobe-index/>

1. 3 Best Python Encryption Libraries in 2023 – Url:

<https://tleapps.com/best-python-encryption-libraries/>

1. PyCryptodome’s documentation – URL:

<https://www.pycryptodome.org>

1. RFC 8017 – URL:

<https://datatracker.ietf.org/doc/html/rfc8017>

1. 1363-2000 - IEEE Standard Specifications for Public-Key Cryptography – URL:

<https://ieeexplore.ieee.org/document/891000>

1. FIPS PUB 186-4 – URL:

<https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>

1. RFC8032 – URL:

<https://datatracker.ietf.org/doc/html/rfc8032>

1. Васильева И. Н. Криптографические методы защиты информации: учебник и практикум для академического бакалавриата / Москва: Юрайт, 2016. – 349 с.
2. Крамаров С. О., Криптографическая защита информации: Учебное пособие / Крамаров С. О., Митясова О.Ю., Соколов С. В., Тищенко Е.Н., Шевчук П.С., Под ред. Крамарова С.О - Минск: РИОР, 2018. – 324 с.
3. A. Menezes, P. van Oorschot and S. Vanstone. Handbook of Applied Cryptography. CRC Press 1996. – 816с.
4. R. Gennaro, S. Goldfeder. Fast Multiparty Threshold ECDSA with Fast Trustless Setup. Conference on Computer and Communications Security 2018. – 31c.