ГОСУДАРСТВЕННЫЙ СТАНДАРТ
РЕСПУБЛИКИ БЕЛАРУСЬ

СТБ 34.101.45-2011

Информационные технологии и безопасность

АЛГОРИТМЫ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ НА ОСНОВЕ ЭЛЛИПТИЧЕСКИХ КРИВЫХ

Інфармацыйныя тэхналогіі і бяспека

АЛГАРЫТМЫ ЭЛЕКТРОННАГА ЛІЧБАВАГА ПОДПІСУ НА АСНОВЕ ЭЛІПТЫЧНЫХ КРЫВЫХ

Госстандарт
Минск

УДК МКС 35.240.40 КП 05

Ключевые слова: электронная цифровая подпись, криптографические алгоритмы на эллиптических кривых, транспорт ключа, генерация псевдослучайных чисел

Предисловие

Цели, основные принципы, положения по государственному регулированию и управлению в области технического нормирования и стандартизации установлены Законом Республики Беларусь «О техническом нормировании и стандартизации».

1 РАЗРАБОТАН учреждением Белорусского государственного университета «Научно-исследовательский институт прикладных проблем математики и информатики»

ВНЕСЕН Оперативно-аналитическим центром при Президенте Республики Беларусь (ОАЦ)

2 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ постановлением Госстандарта Республики Беларусь от 30 мая 2011 г. № 25 в качестве предварительного государственного стандарта Республики Беларусь со сроком действия с 01.01.2012 г. по 01.01.2014 г.

3 ВВЕДЕН ВПЕРВЫЕ

4 Срок представления разработчику предстандарта замечаний и предложений, предложений о целесообразности (нецелесообразности) перевода предстандарта в государственный стандарт — до 01.07.2013 г.

Адрес: 220030, г. Минск, пр. Независимости, 4
Факс: (017) 209-51-04
Телефон: (017) 209-50-71
E-mail: apmi@bsu.by

Издан на русском языке

Содержание

ПРЕДВАРИТЕЛЬНЫЙ ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ БЕЛАРУСЬ

Информационные технологии и безопасность

АЛГОРИТМЫ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ НА ОСНОВЕ ЭЛЛИПТИЧЕСКИХ КРИВЫХ

Інфармацыйныя тэхналогіі і бяспека

АЛГАРЫТМЫ ЭЛЕКТРОННАГА ЛІЧБАВАГА ПОДПІСУ НА АСНОВЕ ЭЛІПТЫЧНЫХ КРЫВЫХ

Information technology and security

Digital signature algorithms based on elliptic curves

Дата введения 2012-01-01

Дата окончания действия 2014-01-01

1 Область применения

Настоящий предварительный государственный стандарт (далее — предстандарт) устанавливает алгоритмы выработки и проверки электронной цифровой подписи (далее — ЭЦП), генерации и проверки параметров эллиптической кривой, генерации и проверки личных и открытых ключей подписи, а также вспомогательные алгоритмы транспорта ключа и генерации псевдослучайных чисел.

Настоящий предстандарт применяется при разработке средств криптографической защиты информации, в том числе средств ЭЦП.

2 Нормативные ссылки

В настоящем предстандарте использованы ссылки на следующие технические нормативные правовые акты в области технического нормирования и стандартизации (далее — ТНПА):

СТБ 1176.2-99 Информационная технология. Защита информации. Процедуры выработки и проверки электронной цифровой подписи

СТБ 34.101.19-2009 Информационные технологии. Форматы сертификатов и списков отозванных сертификатов инфраструктуры открытых ключей

СТБ 34.101.31-2011 Информационные технологии. Защита информации. Криптографические алгоритмы шифрования и контроля целостности

ГОСТ 34.973-91 (ИСО 8824-87) Информационная технология. Взаимосвязь открытых систем. Спецификация абстрактно-синтаксической нотации версии 1 (АСН.1)

ГОСТ 34.974-91 (ИСО 8825-87) Информационная технология. Взаимосвязь открытых систем. Описание базовых правил кодирования для абстрактно-синтаксической нотации версии 1 (АСН.1)

Примечание — При пользовании настоящим предстандартом целесообразно проверить действие ТНПА по каталогу, составленному по состоянию на 1 января текущего года, и по соответствующим информационным указателям, опубликованным в текущем году. Если ссылочные ТНПА заменены (изменены), то при пользовании настоящим предстандартом следует

руководствоваться замененными (измененными) ТНПА. Если ссылочные ТНПА отменены без замены, то положение, в котором дана ссылка на них, применяется в части, не затрагивающей эту ссылку.

3 Термины и определения

В настоящем предстандарте применяют следующие термины с соответствующими определениями:

3.1 ключ: Параметр, который управляет криптографическими операциями зашифрования и расшифрования, выработки и проверки ЭЦП, генерации псевдослучайных чисел и др.

3.2 личный ключ: Ключ, который связан с конкретной стороной, не является общедоступным и используется в настоящем предстандарте для выработки ЭЦП и для разбора токена ключа.

3.3 октет: Двоичное слово длины 8.

3.4 открытый ключ: Ключ, который строится по личному ключу, связан с конкретной стороной, может быть сделан общедоступным и используется в настоящем предстандарте для проверки ЭЦП и для создания токена ключа.

3.5 синхропосылка: Открытые входные данные криптографического алгоритма, которые обеспечивают уникальность результатов криптографического преобразования на фиксированном ключе.

3.6 сообщение: Двоичное слово конечной длины.

3.7 токен ключа: Сообщение, которое передается от одной стороны другой при транспорте ключа.

3.8 транспорт ключа: Конфиденциальная передача ключа от одной стороны другой.

3.9 хэш-значение: Двоичное слово фиксированной длины, которое определяется по сообщению без использования ключа и служит для контроля целостности сообщения и для представления сообщения в сжатой форме.

3.10 хэширование: Выработка хэш-значений.

3.11 электронная цифровая подпись; ЭЦП: Двоичное слово, которое служит для контроля целостности и подлинности сообщения, обеспечивает невозможность отказа от авторства, определяется с использованием личного ключа и проверяется с использованием открытого ключа.

4 Обозначения

4.1 Список обозначений

$\{0,1\}^n$	множество всех слов длины n в алфавите $\{0,1\}$;
$\{0,1\}^*$	множество всех слов конечной длины в алфавите $\{0,1\}$ (включая пустое слово длины 0);
$\|u\|$	длина слова $u \in \{0,1\}^*$;
$\{0,1\}^{n*}$	множество всех слов из $\{0,1\}^*$, длина которых кратна n;

α^n	слово длины n из одинаковых символов $\alpha \in \{0,1\}$;
$L_m(u)$	слово из первых m символов слова u, $m \leqslant \lvert u\rvert$;
$u \parallel v$	конкатенация $u_1u_2\dots u_nv_1v_2\dots v_m$ слов $u = u_1u_2\dots u_n$ и $v = v_1v_2\dots v_m$;
$\texttt{01234...}_{16}$	представление $u \in \{0,1\}^{4*}$ шестнадцатеричным словом, при котором последовательным четырем символам u соответствует один шестнадцатеричный символ (например, $10100010 = \texttt{A2}_{16}$);
$x \bmod m$	для целого x и натурального m остаток от деления x на m, т. е. число $r \in \{0,1,\dots,m-1\}$ такое, что m делит $x - r$;
$x \equiv y \pmod m$	x сравнимо с y по модулю m, т. е. $x \bmod m = y \bmod m$;
$u \oplus v$	для $u = u_1u_2\dots u_n \in \{0,1\}^n$ и $v = v_1v_2\dots v_n \in \{0,1\}^n$ слово $w = w_1w_2\dots w_n \in \{0,1\}^n$ из символов $w_i = (u_i + v_i) \bmod 2$;
$\bar{u}$	а) для $u = u_1u_2\dots u_8 \in \{0,1\}^8$ число $2^7u_1 + 2^6u_2 + \dots + u_8$ и б) для $u = u_1 \parallel u_2 \parallel \dots \parallel u_n$, $u_i \in \{0,1\}^8$, число $\bar{u}_1 + 2^8\bar{u}_2 + \dots + 2^{8(n-1)}\bar{u}_n$;
$\langle U\rangle_{8n}$	для целого U слово $u \in \{0,1\}^{8n}$ такое, что $\bar{u} = U \bmod 2^{8n}$;
$u \boxplus v$	для $u, v \in \{0,1\}^{8n}$ слово $\langle \bar{u} + \bar{v}\rangle_{8n}$;
$\mathbb{F}_p$	для простого p множество $\{0,1,\dots,p-1\}$ с операциями сложения и умножения по модулю p, конечное поле из p элементов;
$E^*_{a,b}(\mathbb{F}_p)$	для $a, b \in \mathbb{F}_p$ множество решений (x, y), $x, y \in \mathbb{F}_p$, уравнения $y^2 = x^3 + ax + b$, множество аффинных точек эллиптической кривой;
O	бесконечно удаленная точка;
$E_{a,b}(\mathbb{F}_p)$	множество $E^*_{a,b}(\mathbb{F}_p) \cup \{O\}$ с операцией сложения точек, группа точек эллиптической кривой;
kP	для $P \in E_{a,b}(\mathbb{F}_p)$ сумма k экземпляров P, кратная P точка;
l	уровень стойкости, число из множества $\{128, 192, 256\}$;
$c \leftarrow u$	присвоение переменной c значения u;
$c \xleftarrow{R} U$	случайный равновероятный выбор c из множества U;
hBelt	алгоритм хэширования, определенный в СТБ 34.101.31;
$\mathrm{OID}(D)$	кодовое представление идентификатора объекта D, полученное в соответствии с ГОСТ 34.973, ГОСТ 34.974.

4.2 Пояснения к обозначениям

4.2.1 Слова

Двоичные слова представляют собой последовательности символов из алфавита $\{0,1\}$. Символы нумеруются слева направо от единицы. В настоящем подразделе в качестве примера рассматривается слово

$$w = 10110001100101001011101011001000.$$

В этом слове первый символ — 1, второй — 0, ..., последний — 0.

Слова разбиваются на тетрады из четверок последовательных двоичных символов. Тетрады кодируются шестнадцатеричными символами по следующим правилам (см. таблицу 1):

Таблица 1

тетрада	символ	тетрада	символ	тетрада	символ	тетрада	символ
0000	$\mathtt{0}_{16}$	0001	$\mathtt{1}_{16}$	0010	$\mathtt{2}_{16}$	0011	$\mathtt{3}_{16}$
0100	$\mathtt{4}_{16}$	0101	$\mathtt{5}_{16}$	0110	$\mathtt{6}_{16}$	0111	$\mathtt{7}_{16}$
1000	$\mathtt{8}_{16}$	1001	$\mathtt{9}_{16}$	1010	$\mathtt{A}_{16}$	1011	$\mathtt{B}_{16}$
1100	$\mathtt{C}_{16}$	1101	$\mathtt{D}_{16}$	1110	$\mathtt{E}_{16}$	1111	$\mathtt{F}_{16}$

Пары последовательных тетрад образуют октеты. Последовательные октеты слова w имеют вид:

$$10110001 = \mathtt{B1}_{16},\ 10010100 = \mathtt{94}_{16},\ 10111010 = \mathtt{BA}_{16},\ 11001000 = \mathtt{C8}_{16}.$$

4.2.2 Слова как числа

Октету $u = u_1u_2\dots u_8$ ставится в соответствие байт — число $\bar{u} = 2^7u_1 + 2^6u_2 + \dots + u_8$. Например, октетам w соответствуют байты

$$177 = 2^7 + 2^5 + 2^4 + 1,\ 148 = 2^7 + 2^4 + 2^2,\ 186 = 2^7 + 2^5 + 2^4 + 2^3 + 2^1,\ 200 = 2^7 + 2^6 + 2^3.$$

Число ставится в соответствие не только октетам, но и любому другому двоичному слову, длина которого кратна 8. При этом используется распространенное для многих современных процессоров соглашение «от младших к старшим» (little-endian): считается, что первый байт является младшим, последний — старшим. Например, слову w соответствует число

$$\bar{w} = 177 + 2^8 \cdot 148 + 2^{16} \cdot 186 + 2^{24} \cdot 200 = 3367670961.$$

4.2.3 Конечные поля

Элементы $\mathbb{F}_p$ складываются и умножаются как целые числа с заменой результата на остаток от его деления на p. Множество $\mathbb{F}_p$ с такими операциями является конечным простым полем. Нулевым элементом поля является число 0, а мультипликативной единицей — число 1 (подробнее см. [1]).

Кроме сложения и умножения, в поле $\mathbb{F}_p$ можно выполнять вычитание и деление. Вычитание u состоит в сложении с $p-u$. Деление на $u \in \{1, 2, \dots, p-1\}$ состоит в умножении на число $v \in \{1, 2, \dots, p-1\}$ такое, что $uv \equiv 1 \pmod p$.

Например, в поле $\mathbb{F}_7$ выполняется:

$$4 + 5 = 2, \quad 4 \cdot 5 = 6, \quad 4 - 5 = 4 + (7 - 5) = 6, \quad 4/5 = 4 \cdot 3 = 5.$$

4.2.4 Эллиптические кривые

Множество $E^*_{a,b}(\mathbb{F}_p)$ состоит из решений уравнения $y^2 = x^3 + ax + b$ относительно $x, y \in \mathbb{F}_p$. Уравнение такого вида определяет эллиптическую кривую над полем $\mathbb{F}_p$, его решения (x, y) называются аффинными точками кривой. К аффинным точкам добавляется

специальная бесконечно удаленная точка O и образуется множество $E_{a,b}(\mathbb{F}_p)$. Например,

$$E_{4,1}(\mathbb{F}_7) = \{O, (0,1), (0,6), (4,2), (4,5)\}.$$

Пусть $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$. Тогда множество $E_{a,b}(\mathbb{F}_p)$ является аддитивной группой при следующих правилах сложения:

1 $O + P = P + O = P$ для всех $P \in E_{a,b}(\mathbb{F}_p)$.

2 Если $P = (x, y) \in E^*_{a,b}(\mathbb{F}_p)$, то $-P = (x, p - y)$ и $P + (-P) = O$.

3 Если $P_1 = (x_1, y_1) \in E^*_{a,b}(\mathbb{F}_p)$, $P_2 = (x_2, y_2) \in E^*_{a,b}(\mathbb{F}_p)$ и $P_2 \neq -P_1$, то $P_1 + P_2 = (x_3, y_3)$, где $x_3 = \lambda^2 - x_1 - x_2$, $\quad y_3 = \lambda(x_1 - x_3) - y_1$, $\quad \lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & P_1 \neq P_2, \\ \dfrac{3x_1^2 + a}{2y_1}, & P_1 = P_2 \end{cases}$

(вычисления ведутся в $\mathbb{F}_p$).

Сумма k экземпляров точки P называется k-кратной ей точкой и обозначается через kP. Например, для $P = (4, 2) \in E_{4,1}(\mathbb{F}_7)$ ее кратные имеют вид:

$$2P = (4,2) + (4,2) = (0,1), \qquad 3P = (0,1) + (4,2) = (0,6),$$
$$4P = 2(0,1) = (4,5), \qquad 5P = (0,1) + (0,6) = O.$$

Считается, что $0P = O$.

4.2.5 Идентификаторы объектов

В ГОСТ 34.973 определены правила абстрактно-синтаксической нотации версии 1 для описания различных информационных объектов. Эти правила регламентируют, в том числе присвоение объектам уникальных идентификаторов.

Идентификатор объекта представляет собой последовательность целых чисел. При записи идентификатора числа разделяются пробелами. Вся последовательность окаймляется фигурными скобками. Например, идентификатор алгоритма хэширования *h*BELT определен в СТБ 34.101.31 как {1 2 112 0 2 0 34 101 31 81}.

Идентификатор объекта кодируется двоичным словом по правилам, заданным в ГОСТ 34.974 и кратко изложенным в приложении А. Например, OID(*h*BELT) = $\texttt{06092A7000020022651F51}_{16}$.

5 Общие положения

5.1 Назначение

Настоящий предстандарт определяет алгоритмы ЭЦП, которые предназначены для контроля целостности и подлинности сообщений. Автор сообщения использует свой личный ключ для выработки ЭЦП, а связанный с личным ключом открытый ключ используется другими сторонами для проверки ЭЦП. При правильном управлении ключами корректность проверяемой подписи означает, что она была выработана владельцем личного ключа и после этого сообщение не изменялось. Только владелец личного ключа может выработать корректную ЭЦП, что не позволяет ему отказаться от авторства сообщения и может быть использовано другими сторонами для доказательства такого авторства.

Примечание — Алгоритмы ЭЦП установлены также в СТБ 1176.2. Переход от алгоритмов СТБ 1176.2 к алгоритмам предстандарта позволит уменьшить время выработки и проверки ЭЦП, сократить длины параметров и ключей при сохранении уровня криптографической стойкости.

Алгоритмы выработки и проверки ЭЦП построены по схеме Шнорра [2]. При выполнении алгоритмов используются вычисления в группе точек эллиптической кривой над конечным простым полем. В предстандарте определяются алгоритмы генерации и проверки параметров, описывающих искомую группу. Определены также алгоритм генерации пары ключей (личного и открытого) и алгоритм проверки открытого ключа.

Алгоритмы проверки параметров эллиптической кривой и открытого ключа следует применять в тех случаях, когда отсутствует гарантия их математической корректности. Такая гарантия обеспечивает достоверность выводов о стойкости алгоритмов ЭЦП. Вместе с тем алгоритм проверки открытого ключа не гарантирует, что ключ действительно принадлежит определенной стороне или что сторона знает соответствующий личный ключ. Проверка знания личного ключа, удостоверение принадлежности открытого ключа и проверка такой принадлежности реализуются с помощью дополнительных методов и средств, в совокупности называемых инфраструктурой открытых ключей. Например, в СТБ 34.101.19 определяются элементы инфраструктуры на основе сертификатов открытых ключей.

Параметры эллиптической кривой, личный и открытый ключи могут быть использованы не только для контроля целостности и подлинности, но и для обеспечения конфиденциальности. В предстандарте определяются алгоритмы транспорта ключа, предназначенные для защищенной передачи ключей и других секретных данных между двумя сторонами. С помощью транспортируемого ключа стороны могут выполнять шифрование или другие криптографические операции.

Для реализации транспорта отправитель вызывает алгоритм создания токена ключа. Токен представляет собой сообщение, которое включает транспортируемый ключ в защищенной форме, а также данные, необходимые получателю для снятия защиты. Получатель вызывает алгоритм разбора токена и восстанавливает транспортируемый ключ. При создании токена отправитель использует открытый ключ получателя. При разборе токена получатель использует свой личный ключ.

Предстандарт определяет вспомогательный алгоритм генерации псевдослучайных чисел с секретным параметром, который может быть использован для создания ключей.

В приложении Б приводятся стандартные наборы параметров эллиптической кривой, которые были получены с помощью соответствующего алгоритма генерации и могут быть использованы напрямую, без повторного построения.

В приложении В приводятся примеры выполнения алгоритмов предстандарта. Примеры можно использовать для проверки корректности реализаций алгоритмов.

В приложении Г приводится модуль абстрактно-синтаксической нотации версии 1 (АСН.1), определенной в ГОСТ 34.973. Модуль задает идентификаторы алгоритмов и других объектов предстандарта, описывает структуры данных для хранения ключей и параметров. Рекомендуется использовать модуль при встраивании алгоритмов предстан-

дарта в информационные системы, в которых также используется АСН.1. В частности, модуль может быть использован для уточнения описаний сертификатов открытых ключей и списков отозванных сертификатов, определенных в СТБ 34.101.19.

5.2 Уровень стойкости

Алгоритмы ЭЦП построены так, что злоумышленнику вычислительно трудно решить задачу подделки ЭЦП. В этой задаче злоумышленник получает параметры эллиптической кривой и открытый ключ ЭЦП. Злоумышленник не знает личный ключ, но может передавать для подписи на нем произвольные сообщения, получать и анализировать результаты. Ему требуется построить корректную ЭЦП к любому сообщению, отличному от ранее подписанных.

Стойкость алгоритмов ЭЦП определяется уровнем $l \in \{128, 192, 256\}$. На уровне l для подделки ЭЦП злоумышленнику требуется выполнить порядка 2^l операций. Стойкость основывается на сложности дискретного логарифмирования в группе точек эллиптической кривой и на стойкости используемых функций хэширования.

Уровень l определяет длины параметров, ключей, подписей и, соответственно, быстродействие алгоритмов ЭЦП. Следует учитывать, что с ростом l, кроме повышения стойкости, снижается быстродействие алгоритмов.

Для алгоритмов транспорта ключа вводятся аналогичные уровни стойкости $l \in \{128, 192, 256\}$. На уровне l для определения транспортируемого ключа по токену и открытому ключу получателя злоумышленнику требуется выполнить порядка 2^l операций.

5.3 Параметры эллиптической кривой

Модуль p**.** Используется простое число p, которое удовлетворяет условиям: $2^{2l-1} < p < 2^{2l}$, $p \equiv 3 \pmod 4$. Модуль определяет поле $\mathbb{F}_p$, над которым строится эллиптическая кривая. Можно использовать произвольное допустимое p, в том числе простое специального вида.

Коэффициенты a**,** b**.** Используются числа $a, b \in \mathbb{F}_p$, которые удовлетворяют условиям: $a, b \neq 0$, $b^{(p-1)/2} \equiv 1 \pmod p$, $4a^3 + 27b^2 \not\equiv 0 \pmod p$. Коэффициенты a, b вместе с модулем p определяют группу точек эллиптической кривой $E_{a,b}(\mathbb{F}_p)$.

Параметр *seed***.** Числа p и a выбираются, а b строится по ним. При построении b используется дополнительный параметр $seed \in \{0,1\}^{64}$, который может быть выбран произвольным образом.

Порядок q**.** После построения группы $E_{a,b}(\mathbb{F}_p)$ рассчитывается ее порядок $q = |E_{a,b}(\mathbb{F}_p)|$. Выбирается группа, порядок которой удовлетворяет следующим ограничениям: q — простое, $2^{2l-1} < q < 2^{2l}$, $q \neq p$, q не делит числа вида $p^m - 1$ для $m = 1, 2, \ldots, 50$.

Базовая точка G**.** Используется базовая точка $G \in E^*_{a,b}(\mathbb{F}_p)$ вида $G = (0, y_G)$, где $y_G = b^{(p+1)/4} \bmod p$. Кратные G, $2G$,..., $(q-1)G$ базовой точки пробегают все элементы $E^*_{a,b}(\mathbb{F}_p)$, а $qG = O$.

Алгоритм генерации параметров эллиптической кривой определен в 6.1.3. Алгоритм проверки параметров определен в 6.1.4.

Алгоритм генерации параметров имеет высокую вычислительную сложность. Основные издержки связаны с расчетом порядка q. Алгоритм проверки параметров имеет значительно меньшую сложность, поскольку требуется проверять, а не определять q.

5.4 Ключи

Личным ключом является число $d \in \{1, 2, \ldots, q-1\}$. По личному ключу определяется открытый ключ $Q = dG$. Алгоритм генерации личного и открытого ключей определен в 6.2.2. Алгоритм проверки открытого ключа определен в 6.2.3.

Личный ключ должен вырабатываться без возможности предсказания. При хранении и распространении должны обеспечиваться конфиденциальность и контроль целостности личного ключа, контроль целостности открытого ключа. Применяемые методы управления ключами должны гарантировать принадлежность открытого ключа стороне, подпись которой проверяется, и знание данной стороной соответствующего личного ключа.

Одни и те же ключи d, Q могут использоваться как в алгоритмах ЭЦП, так и в алгоритмах транспорта ключа. Использование данных ключей в других алгоритмах запрещено.

Кроме личного ключа, в алгоритме выработки ЭЦП используется одноразовый ключ, который сохраняется в переменной k. Одноразовый ключ используется также в алгоритме создания токена ключа. Одноразовые ключи должны вырабатываться без возможности предсказания и уничтожаться сразу после использования.

Для создания личных и одноразовых ключей может быть использован физический генератор случайных чисел, удовлетворяющий ТНПА, или алгоритм генерации псевдослучайных чисел с секретным параметром, определенный в 7.2 или в другом ТНПА.

В информационных системах ключи представляются двоичными словами. Для обеспечения совместимости рекомендуется представлять личный ключ d словом $\langle d \rangle_{2l}$, а открытый ключ $Q = (x_Q, y_Q)$ — словом $\langle x_Q \rangle_{2l} \,\|\, \langle y_Q \rangle_{2l}$.

5.5 Функция хэширования

В алгоритмах выработки и проверки ЭЦП используется функция хэширования h, которая ставит в соответствие подписываемому или проверяемому сообщению X его хэш-значение $h(X)$.

На уровне стойкости l должна использоваться функция h, значениями которой являются двоичные слова длины $2l$. Например, при $l = 128$ в качестве h может использоваться функция hBelt.

Функция h должна быть определена в ТНПА. Алгоритму хэширования в ТНПА должен быть назначен уникальный идентификатор. Кодовое представление $\mathrm{OID}(h)$ этого идентификатора используется в алгоритмах ЭЦП.

5.6 Транспорт ключа

В алгоритмах транспорта ключа используется заголовок $I \in \{0,1\}^{128}$. Заголовок содержит открытые атрибуты транспортируемого ключа, включая данные об отправителе или получателе, и может передаваться вместе с токеном ключа. Если необходимости в

передаче атрибутов ключей нет, то могут использоваться постоянные заголовки, которые не требуется передавать. По умолчанию $I = 0^{128}$.

Один и тот же ключ может транспортироваться одновременно нескольким сторонам. В этом случае отправитель должен создать токены ключа для каждой из сторон. Если стороны-получатели используют одинаковые параметры эллиптической кривой, то отправитель может использовать при создании токенов один и тот же одноразовый ключ k.

6 Алгоритмы электронной цифровой подписи

6.1 Генерация и проверка параметров эллиптической кривой

6.1.1 Входные и выходные данные

Входными данными алгоритма генерации параметров эллиптической кривой являются уровень стойкости $l \in \{128, 192, 256\}$, простой модуль p и целый коэффициент a. Должны выполняться следующие условия: $2^{2l-1} < p < 2^{2l}$, $p \equiv 3 \pmod 4$, $0 < a < p$.

Выходными данными алгоритма генерации параметров являются параметр $seed \in \{0,1\}^{64}$, коэффициент b ($0 < b < p$), порядок q ($2^{2l-1} < q < 2^{2l}$) и базовая точка $G \in E^*_{a,b}(\mathbb{F}_p)$.

Входными данными алгоритма проверки параметров эллиптической кривой являются модуль p, коэффициенты a и b, параметр $seed$, порядок q и базовая точка G. Параметры p, a, b, q являются целыми числами, $seed \in \{0,1\}^{64}$, точка G задается двумя целыми координатами.

Выходными данными алгоритма проверки параметров является ответ `ДА` или `НЕТ`. Ответ `ДА` означает, что переданные параметры описывают допустимую группу точек эллиптической кривой и были сгенерированы надлежащим образом. Ответ `НЕТ` означает обратное.

6.1.2 Вспомогательные алгоритмы и переменные

Вычисление порядка группы точек. На шаге 5 алгоритма генерации параметров определяется порядок группы точек эллиптической кривой. Для вычисления порядка может быть использован алгоритм Шуфа или его модернизации, например алгоритм Шуфа — Элкиса — Аткина (см. [3], пункт 4.2.3).

Проверка простоты. В перечислении 2) на шаге 6 алгоритма генерации параметров и в перечислении 3) на шаге 2 алгоритма проверки параметров контролируется простота чисел. Для проверки простоты рекомендуется использовать алгоритм Миллера — Рабина, описанный в приложении Д.

Переменная B. Используется переменная B со значениями из $\{0,1\}^{512}$.

6.1.3 Алгоритм генерации параметров эллиптической кривой

Генерация параметров эллиптической кривой состоит в выполнении следующих шагов:

1 Выбрать произвольным образом $seed$.

2 Установить $B \leftarrow h\textsc{Belt}(\langle p\rangle_{2l} \,\|\, \langle a\rangle_{2l} \,\|\, seed) \,\|\, h\textsc{Belt}(\langle p\rangle_{2l} \,\|\, \langle a\rangle_{2l} \,\|\, seed \boxplus \langle 1\rangle_{64})$.

3 Установить $b \leftarrow \overline{B} \mod p$.

4 Если нарушается одно из условий:

1) $b \neq 0$;
2) $b^{(p-1)/2} \equiv 1 \pmod p$;
3) $4a^3 + 27b^2 \not\equiv 0 \pmod p$,

то вернуться к шагу 1.

5 Установить $q \leftarrow |E_{a,b}(\mathbb{F}_p)|$.

6 Если нарушается одно из условий:
1) $2^{2l-1} < q < 2^{2l}$;
2) q — простое;
3) $p \neq q$;
4) $p^m \not\equiv 1 \pmod q$ для $m = 1, 2, \ldots, 50$,

то вернуться к шагу 1.

7 Установить $G \leftarrow (0, b^{(p+1)/4} \bmod p)$.

8 Возвратить $(seed, b, q, G)$.

6.1.4 Алгоритм проверки параметров эллиптической кривой

Проверка параметров эллиптической кривой состоит в выполнении следующих шагов:

1 Определить уровень стойкости l как минимальное натуральное, для которого $p < 2^{2l}$.

2 Если нарушается одно из условий:
1) $l \in \{128, 192, 256\}$;
2) $2^{2l-1} < p, q < 2^{2l}$;
3) p, q — простые;
4) $p \equiv 3 \pmod 4$;
5) $q \neq p$;
6) $p^m \not\equiv 1 \pmod q$ для $m = 1, 2, \ldots, 50$,

то возвратить `НЕТ`.

3 Установить $B \leftarrow h\textsc{Belt}(\langle p\rangle_{2l} \| \langle a\rangle_{2l} \| seed) \| h\textsc{Belt}(\langle p\rangle_{2l} \| \langle a\rangle_{2l} \| seed \boxplus \langle 1\rangle_{64})$.

4 Если нарушается одно из условий:
1) $0 < a, b < p$;
2) $b \equiv \overline{B} \pmod p$;
3) $4a^3 + 27b^2 \not\equiv 0 \pmod p$;
4) $b^{(p-1)/2} \equiv 1 \pmod p$;
5) $G = (0, b^{(p+1)/4} \bmod p)$;
6) $qG = O$,

то возвратить `НЕТ`.

5 Возвратить `ДА`.

6.2 Генерация и проверка ключей

6.2.1 Входные и выходные данные

Входными данными алгоритма генерации пары ключей являются параметры p, a, b, q, G, которые описывают группу точек эллиптической кривой. Параметры должны удовлетворять условиям алгоритма 6.1.4.

Выходными данными алгоритма генерации пары ключей являются личный ключ $d \in \{1, 2, \ldots, q-1\}$ и соответствующий открытый ключ $Q \in E^*_{a,b}(\mathbb{F}_p)$.

Входными данными алгоритма проверки открытого ключа являются параметры p, a, b, которые описывают группу точек эллиптической кривой, и открытый ключ $Q = (x_Q, y_Q)$, где x_Q, y_Q — целые числа. Параметры эллиптической кривой должны удовлетворять условиям алгоритма 6.1.4.

Выходными данными алгоритма проверки открытого ключа является ответ `ДА` или `НЕТ`. Ответ `ДА` означает, что Q является допустимым открытым ключом. Ответ `НЕТ` означает обратное.

6.2.2 Алгоритм генерации пары ключей

Генерация пары ключей состоит в выполнении следующих шагов:

1 Выработать $d \xleftarrow{R} \{1, 2, \ldots, q-1\}$.

2 Установить $Q \leftarrow dG$.

3 Возвратить (d, Q).

6.2.3 Алгоритм проверки открытого ключа

Проверка открытого ключа состоит в выполнении следующих шагов:

1 Если нарушается одно из условий:

1) $0 \leqslant x_Q, y_Q < p$;

2) $y_Q^2 \equiv x_Q^3 + ax_Q + b \pmod p$,

то возвратить `НЕТ`.

2 Возвратить `ДА`.

6.3 Выработка и проверка электронной цифровой подписи

6.3.1 Входные и выходные данные

Входными данными алгоритмов ЭЦП являются параметры p, a, b, q, G, которые описывают группу точек эллиптической кривой. Параметры должны удовлетворять условиям алгоритма 6.1.4. По модулю p определяется уровень стойкости l как минимальное натуральное, для которого $p < 2^{2l}$.

Кроме параметров эллиптической кривой, входными данными алгоритма выработки ЭЦП являются сообщение $X \in \{0,1\}^*$ и личный ключ $d \in \{1, 2, \ldots, q-1\}$.

Выходными данными алгоритма выработки ЭЦП является слово $S \in \{0,1\}^{3l}$ — электронная цифровая подпись X.

Кроме параметров эллиптической кривой, входными данными алгоритма проверки ЭЦП являются сообщение $X \in \{0,1\}^*$, открытый ключ $Q \in E^*_{a,b}(\mathbb{F}_p)$ и электронная цифровая подпись $S \in \{0,1\}^*$. Открытый ключ Q должен удовлетворять условиям алгоритма 6.2.3.

Выходными данными алгоритма проверки ЭЦП является ответ `ДА` или `НЕТ`. Ответ `ДА` означает, что S является корректной подписью X. Ответ `НЕТ` означает обратное.

6.3.2 Вспомогательные преобразования и переменные

Функция хэширования $h\text{BELT}_l$**.** Функция $h\text{BELT}_l$ действует из $\{0,1\}^*$ в $\{0,1\}^l$ по правилу: $h\text{BELT}_l(u) = L_l(h\text{BELT}(u))$.

Функция хэширования h**.** Функция хэширования h действует из $\{0,1\}^*$ в $\{0,1\}^{2l}$. Алгоритм хэширования, который определяет действие h, должен быть задан в ТНПА. В ТНПА должен быть указан идентификатор алгоритма хэширования, который кодируется словом $\text{OID}(h) \in \{0,1\}^{8*}$.

Функция π_m **(**$m \in \{256, 384, 512\}$**).** Функция π_m действует из $E^*_{a,b}(\mathbb{F}_p)$ в $\{0,1\}^m$ и ставит в соответствие точке (x, y) слово $\langle x \rangle_m$.

Переменная k**.** При выработке ЭЦП используется переменная k со значениями из множества $\{1, 2, \ldots, q-1\}$. Значения k должны вырабатываться без возможности предсказания и уничтожаться сразу после использования.

Переменная R**.** Используется переменная R со значениями из $E_{a,b}(\mathbb{F}_p)$.

6.3.3 Алгоритм выработки электронной цифровой подписи

ЭЦП составляется из частей $S_0 \in \{0,1\}^l$ и $S_1 \in \{0,1\}^{2l}$. Выработка ЭЦП состоит в выполнении следующих шагов:

1 Выработать $k \xleftarrow{R} \{1, 2, \ldots, q-1\}$.
2 Установить $R \leftarrow kG$.
3 Установить $S_0 \leftarrow h\text{BELT}_l(\text{OID}(h) \parallel \pi_{2l}(R) \parallel h(X))$.
4 Установить $S_1 \leftarrow \Big\langle (k - \overline{h(X)} - (\overline{S_0} + 2^l)d) \bmod q \Big\rangle_{2l}$.
5 Установить $S \leftarrow S_0 \parallel S_1$.
6 Возвратить S.

6.3.4 Алгоритм проверки электронной цифровой подписи

Проверка ЭЦП состоит в выполнении следующих шагов:

1 Если $|S| \neq 3l$, то возвратить `НЕТ`.
2 Представить S в виде $S = S_0 \parallel S_1$, где $S_0 \in \{0,1\}^l$, $S_1 \in \{0,1\}^{2l}$.
3 Если $\overline{S_1} \geqslant q$, то возвратить `НЕТ`.
4 Установить $R \leftarrow \Big((\overline{S_1} + \overline{h(X)}) \bmod q\Big) G + (\overline{S_0} + 2^l)Q$.
5 Если $R = O$, то возвратить `НЕТ`.
6 Если $h\text{BELT}_l(\text{OID}(h) \parallel \pi_{2l}(R) \parallel h(X)) \neq S_0$, то возвратить `НЕТ`.
7 Возвратить `ДА`.

7 Вспомогательные алгоритмы

7.1 Транспорт ключа

7.1.1 Входные и выходные данные

Входными данными алгоритмов транспорта ключа являются параметры p, a, b, q, G, которые описывают группу точек эллиптической кривой. Параметры должны удовлетворять условиям алгоритма 6.1.4. По модулю p определяется уровень стойкости l как минимальное натуральное, для которого $p < 2^{2l}$.

Кроме параметров эллиптической кривой, входными данными алгоритма создания токена являются транспортируемый ключ $X \in \{0,1\}^{8*}$, его заголовок $I \in \{0,1\}^{128}$ и открытый ключ $Q \in E^*_{a,b}(\mathbb{F}_p)$ получателя X. Длина X должна быть не меньше 128. Открытый ключ Q должен удовлетворять условиям алгоритма 6.2.3.

Выходными данными алгоритма создания токена является слово $Y \in \{0,1\}^{2l+|X|+128}$ — токен ключа X.

Кроме параметров эллиптической кривой, входными данными алгоритма разбора токена являются токен $X \in \{0,1\}^*$, заголовок $I \in \{0,1\}^{128}$ транспортируемого в нем ключа и личный ключ $d \in \{1,2,\ldots,q-1\}$ получателя токена.

Выходными данными алгоритма разбора токена является признак `ОШИБКА` либо слово $Y \in \{0,1\}^{|X|-2l-128}$ — ключ, который транспортируется в токене X. Возврат признака `ОШИБКА` означает некорректность токена.

7.1.2 Вспомогательные алгоритмы и преобразования, переменные

Алгоритм `KeyWrap`**.** Алгоритм `KeyWrap` берет на вход транспортируемый ключ $X \in \{0,1\}^{8*}$, заголовок $I \in \{0,1\}^{128}$, ключ защиты $\theta \in \{0,1\}^{256}$ и возвращает защищенный ключ $Y \in \{0,1\}^{|X|+128}$. Алгоритм определен в СТБ 34.101.31 (пункт 6.8.3).

Алгоритм `KeyUnwrap`**.** Алгоритм `KeyUnwrap` берет на вход защищенный ключ $X \in \{0,1\}^{8*}$, заголовок $I \in \{0,1\}^{128}$, ключ защиты $\theta \in \{0,1\}^{256}$ и возвращает либо признак `ОШИБКА`, либо транспортируемый ключ $Y \in \{0,1\}^{|X|-128}$. Возврат признака `ОШИБКА` означает нарушение целостности транспортируемого ключа. Алгоритм определен в СТБ 34.101.31 (пункт 6.8.4).

Функция π_m**.** Используется функция π_m, определенная в 6.3.2.

Переменная k**.** При создании токена используется переменная k со значениями из множества $\{1,2,\ldots,q-1\}$. Значения k должны вырабатываться без возможности предсказания и уничтожаться сразу после использования.

Переменная θ**.** Используется переменная θ со значениями из $\{0,1\}^{256}$. Значения θ должны уничтожаться сразу после использования.

Переменная R**.** Используется переменная $R = (x_R, y_R)$ со значениями из $E^*_{a,b}(\mathbb{F}_p)$.

7.1.3 Алгоритм создания токена ключа

Алгоритм создания токена ключа состоит в выполнении следующих шагов:

1 Выработать $k \xleftarrow{R} \{1,2,\ldots,q-1\}$.

2 Установить $R \leftarrow kG$.

3 Установить $\theta \leftarrow \pi_{256}(kQ)$.

4 Установить $Y \leftarrow \pi_{2l}(R) \parallel \mathtt{KeyWrap}(X, I, \theta)$.

5 Возвратить Y.

7.1.4 Алгоритм разбора токена ключа

Алгоритм разбора токена ключа состоит в выполнении следующих шагов:

1 Если длина X не кратна 8 или $|X| < 2l + 256$, то возвратить `ОШИБКА`.

2 Представить X в виде $X_0 \parallel X_1$, где $X_0 \in \{0,1\}^{2l}$, $X_1 \in \{0,1\}^{|X|-2l}$.

3 Установить $x_R \leftarrow \overline{X_0}$.

4 Если $x_R \geqslant p$, то возвратить ОШИБКА.
5 Установить $y_R \leftarrow (x_R^3 + ax_R + b)^{(p+1)/4} \bmod p$.
6 Если $y_R^2 \not\equiv x_R^3 + ax_R + b \pmod p$, то возвратить ОШИБКА.
7 Построить $R = (x_R, y_R)$.
8 Установить $\theta \leftarrow \pi_{256}(dR)$.
9 Если $\mathtt{KeyUnwrap}(X_1, I, \theta) = \mathtt{ОШИБКА}$, то возвратить ОШИБКА.
10 Установить $Y \leftarrow \mathtt{KeyUnwrap}(X_1, I, \theta)$.
11 Возвратить Y.

7.2 Генерация псевдослучайных чисел

7.2.1 Входные и выходные данные

Входными данными алгоритма генерации псевдослучайных чисел являются натуральное n, ключ $\theta \in \{0,1\}^{256}$, синхропосылка $S \in \{0,1\}^{256}$. Число n определяет объем псевдослучайных чисел. Синхропосылка S обеспечивает вариабельность данных, вырабатываемых на одном и том же ключе. При заданном θ должны использоваться различные синхропосылки.

Используется дополнительное входное слово $X \in \{0,1\}^{256n}$. Слово X записывается в виде $X = X_1 \parallel X_2 \parallel \ldots \parallel X_n$, где $X_i \in \{0,1\}^{256}$ — дополнительные данные, которые используются на i-й итерации алгоритма. Слова X_i могут выбираться произвольным образом, в том числе случайным или псевдослучайным методом. По умолчанию $X_i = 0^{256}$.

Выходными данными алгоритма является слово $Y \in \{0,1\}^{256n}$ — псевдослучайные числа, полученные на ключе θ при использовании синхропосылки S и дополнительных данных X. Слово Y записывается в виде $Y = Y_1 \parallel Y_2 \parallel \ldots \parallel Y_n$, где $Y_i \in \{0,1\}^{256}$.

7.2.2 Переменные

Используются переменные s, r со значениями из $\{0,1\}^{256}$. Значение r должно быть уничтожено сразу после выработки Y.

7.2.3 Алгоритм генерации псевдослучайных чисел

Генерация псевдослучайных чисел состоит в выполнении следующих шагов:
1 Установить $s \leftarrow S$.
2 Установить $r \leftarrow S \oplus 1^{256}$.
3 Для $i = 1, 2, \ldots, n$ выполнить:
  1) $Y_i \leftarrow h\textsc{Belt}(\theta \parallel s \parallel X_i \parallel r)$;
  2) $s \leftarrow s \boxplus \langle 1 \rangle_{256}$.
  3) $r \leftarrow r \oplus Y_i$.
4 Установить $Y \leftarrow Y_1 \parallel Y_2 \parallel \ldots \parallel Y_n$.
5 Возвратить Y.

Приложение А

(справочное)

Кодирование идентификаторов объектов

Пусть D — некоторый объект, снабженный идентификатором $\{d_1\ d_2\ \ldots d_n\}$ в соответствии с ГОСТ 34.973. Допустимый идентификатор должен удовлетворять следующим ограничениям: $d_1, d_2, \ldots, d_n$ — неотрицательные целые числа; $n \geqslant 2$; $d_1 \in \{0, 1, 2\}$; если $d_1 \in \{0, 1\}$, то $d_2 < 40$.

Для определения $\mathrm{OID}(D)$ числа $40d_1 + d_2, d_3, \ldots, d_n$ кодируются двоичными словами, которые последовательно конкатенируются и образуют составное слово V. Каждое кодируемое число d записывается в виде

$$d = \sum_{j=0}^{r} a_j 128^j, \quad 0 \leqslant a_j < 128,$$

где $a_r \neq 0$, если $d \neq 0$, и $r = a_0 = 0$ при $d = 0$. Затем число d кодируется словом

$$\langle 128 + a_r \rangle_8 \,\|\, \langle 128 + a_{r-1} \rangle_8 \,\|\, \ldots \,\|\, \langle 128 + a_1 \rangle_8 \,\|\, \langle a_0 \rangle_8.$$

После определения V вычисляется его длина $l = |V|/8$ в октетах. Если $l < 128$, то длина кодируется словом $L = \langle l \rangle_8$. Если $l \geqslant 128$, то длина представляется в виде

$$l = \sum_{j=0}^{r} b_j 256^j, \quad 0 \leqslant b_j < 256, \quad b_r \neq 0,$$

и кодируется словом

$$L = \langle 128 + r \rangle_8 \,\|\, \langle b_r \rangle_8 \,\|\, \langle b_{r-1} \rangle_8 \,\|\, \ldots \,\|\, \langle b_0 \rangle_8.$$

Окончательно $\mathrm{OID}(D)$ определяется как

$$\mathrm{OID}(D) = \mathtt{06}_{16} \,\|\, L \,\|\, V.$$

Например, идентификатору {1 2 112 0 2 0 34 101 31 81} соответствуют числа 42, 112, 0, 2, 0, 34, 101, 31 и 81. Данные числа кодируются словами $\mathtt{2A}_{16}$, $\mathtt{70}_{16}$, $\mathtt{00}_{16}$, $\mathtt{02}_{16}$, $\mathtt{00}_{16}$, $\mathtt{22}_{16}$, $\mathtt{65}_{16}$, $\mathtt{1F}_{16}$ и $\mathtt{51}_{16}$, которые образуют слово $V = \mathtt{2A7000020022651F51}_{16}$. Длина V кодируется словом $L = \mathtt{09}_{16}$. Окончательно кодовое представление исходного идентификатора имеет следующий вид: $\mathtt{06092A7000020022651F51}_{16}$.

Приложение Б

(рекомендуемое)

Стандартные параметры эллиптической кривой

В таблицах Б.1 — Б.3 представлены стандартные параметры эллиптической кривой для различных уровней стойкости.

Таблица Б.1 — Стандартные параметры ($l = 128$)

p	$2^{256} - 189$
$\langle p \rangle_{256}$	43FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$_{16}$
a	$2^{256} - 192$
$\langle a \rangle_{256}$	40FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$_{16}$
$\langle b \rangle_{256}$	F1039CD6 6B7D2EB2 53928B97 6950F54C BEFBD8E4 AB3AC1D2 EDA8F315 156CCE77$_{16}$
$seed$	5E380100 00000000$_{16}$
q	$2^{256} - 51\ 359303463\ 308904523\ 350978545\ 619999225$
$\langle q \rangle_{256}$	07663D26 99BF5A7E FC4DFB0D D68E5CD9 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$_{16}$
$\langle y_G \rangle_{256}$	936A5104 18CF291E 52F608C4 66399178 5D83D651 A3C9E45C 9FD616FB 3CFCF76B$_{16}$

Таблица Б.2 — Стандартные параметры ($l = 192$)

p	$2^{384} - 317$
$\langle p \rangle_{384}$	C3FEFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$_{16}$
a	$2^{384} - 320$
$\langle a \rangle_{384}$	C0FEFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$_{16}$
$\langle b \rangle_{384}$	64BF7368 23FCA7BC 7CBDCEF3 F0E2BD14 3A2E71E9 F96A21A6 96B1FB0F BB482771 D2345D65 AB5A0733 20EF9C95 E1DF753C$_{16}$
$seed$	23AF0000 00000000$_{16}$
q	$2^{384} - 9886\ 438520659\ 958522437\ 788006980\ 660965037\ 549058207\ 958390857$
$\langle q \rangle_{384}$	B7A70CF3 3FDCB73D 0AFFA4A6 E7DA4680 BB7BAF73 03C4CC6C FEFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$_{16}$
$\langle y_G \rangle_{384}$	51C433F7 31CB5EEA F9422A6B 273E4084 55D3B166 9EE74905 A0FF86DC 119A723A 89BF2D43 7E113063 9E9E2EA8 2482435D$_{16}$

Таблица Б.3 — Стандартные параметры ($l = 256$)

p	$2^{512} - 569$
$\langle p \rangle_{512}$	C7FDFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$_{16}$
a	$2^{512} - 572$
$\langle a \rangle_{512}$	C4FDFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$_{16}$
$\langle b \rangle_{512}$	909C13D6 98693409 7AA2493A 272286EA 43A2AC87 8C003329 955E24C4 B5DC1127 88B0ADDA E313CE17 51255DDD EEA9C65B 8958FD60 6A5D8CD8 438C3B93 4459B46C$_{16}$
$seed$	AE170200 00000000$_{16}$
$\langle q \rangle_{512}$	F18E060D 49ADFFDC 32DF5695 E5CA1B36 F413212E B0EB6BF2 4E009801 2C09C0B2 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$_{16}$
$\langle y_G \rangle_{512}$	BDEDEFCE 6FAE92B7 040D4CC9 B983AA67 6122E8EE 957377FF D26FFA0E E2DD7369 DACACC00 1BF8EDD2 E2BC61B3 B341ABB0 AB8FD1A0 F7E682B1 817603E4 7AFF26A8$_{16}$

Приложение В
(справочное)
Тестовые примеры

В.1 Генерация личного ключа

В таблице В.1 представлен пример генерации личного ключа. Используются параметры эллиптической кривой, заданные в таблице Б.1.

Таблица В.1 — Генерация личного ключа

$\langle d \rangle_{256}$	1F66B5B8 4B733967 4533F032 9C74F218 34281FED 0732429E 0C79235F C273E269$_{16}$
$\langle x_Q \rangle_{256}$	BD1A5650 179D79E0 3FCEE49D 4C2BD5DD F54CE46D 0CF11E4F F87BF7A8 90857FD0$_{16}$
$\langle y_Q \rangle_{256}$	7AC6A603 61E8C817 3491686D 461B2826 190C2EDA 5909054A 9AB84D2A B9D99A90$_{16}$

В.2 Выработка электронной цифровой подписи

В таблице В.2 представлен пример выработки ЭЦП. Используются параметры эллиптической кривой, заданные в таблице Б.1, и ключи, заданные в В.1. В качестве h используется функция хэширования hBelt.

Таблица В.2 — Выработка электронной цифровой подписи

OID(h)	06092A70 00020022 651F51$_{16}$
X	B194BAC8 0A08F53B 366D008E 58$_{16}$
$h(X)$	ABEF9725 D4C5A835 97A367D1 4494CC25 42F20F65 9DDFECC9 61A3EC55 0CBA8C75$_{16}$
$\langle k \rangle_{256}$	4C0E74B2 CD5811AD 21F23DE7 E0FA742C 3ED6EC48 3C461CE1 5C33A77A A308B7D2$_{16}$
$\langle x_R \rangle_{256}$	CCEEF1A3 13A40664 9D15DA0A 851D486A 695B641B 20611776 252FFDCE 39C71060$_{16}$
$\langle y_R \rangle_{256}$	7C9EA1F3 3C23D20D FCB8485A 88BE6523 A28ECC32 15B47FA2 89D6C9BE 1CE837C0$_{16}$
S_0	E36B7F03 77AE4C52 4027C387 FADF1B20$_{16}$
S_1	CE72F153 0B71F2B5 FD3A8C58 4FE2E1AE D20082E3 0C8AF650 11F4FB54 649DFD3D$_{16}$
S	E36B7F03 77AE4C52 4027C387 FADF1B20 CE72F153 0B71F2B5 FD3A8C58 4FE2E1AE D20082E3 0C8AF650 11F4FB54 649DFD3D$_{16}$

В.3 Проверка электронной цифровой подписи

В таблице В.3 представлен пример проверки ЭЦП. Используются параметры эллиптической кривой, заданные в таблице Б.1, и ключи, заданные в В.1. В качестве h используется функция хэширования hBelt.

Таблица В.3 — Проверка ЭЦП

$\mathrm{OID}(h)$	06092A70 00020022 651F51$_{16}$
X	B194BAC8 0A08F53B 366D008E 584A5DE4 8504FA9D 1BB6C7AC 252E72C2 02FDCE0D 5BE3D612 17B96181 FE6786AD 716B890B$_{16}$
S	47A63C8B 9C936E94 B5FAB3D9 CBD78366 290F3210 E163EEC8 DB4E921E 8479D413 8F112CC2 3E6DCE65 EC5FF21D F4231C28$_{16}$
$h(X)$	9D02EE44 6FB6A29F E5C982D4 B13AF9D3 E90861BC 4CEF27CF 306BFB0B 174A154A$_{16}$
S_0	47A63C8B 9C936E94 B5FAB3D9 CBD78366$_{16}$
S_1	290F3210 E163EEC8 DB4E921E 8479D413 8F112CC2 3E6DCE65 EC5FF21D F4231C28$_{16}$
$\langle x_R \rangle_{256}$	1D5A382B 962D4ED0 6193258C A6DE535D 8FD7FACB 853171E9 32EF93B5 EE800120$_{16}$
$\langle y_R \rangle_{256}$	03DBB7B5 BD070363 80BAFA47 FCA7E6CA 3F179EDD D1AE5086 64790918 3628EDDC$_{16}$

Значение $h\textsc{Belt}_{128}(\mathrm{OID}(h) \parallel \langle x_R \rangle_{256} \parallel h(X))$, вычисляемое на шаге 6, совпадает с S_0 и алгоритм возвращает ДА.

В.4 Создание токена ключа

В таблице В.4 представлен пример создания токена ключа. Используются параметры эллиптической кривой, заданные в таблице Б.1, и ключи, заданные в В.1.

Таблица В.4 — Создание токена ключа

X	B194BAC8 0A08F53B 366D008E 584A5DE4 8504$_{16}$
I	5BE3D612 17B96181 FE6786AD 716B890B$_{16}$
k	0F51D913 47617C20 BD4AB07A EF4F26A1 AD1362A8 F9A3D42F BE1B8E6F 1C88AAD5$_{16}$
$\langle x_R \rangle_{256}$	9B4EA669 DABDF100 A7D4B6E6 EB76EE52 51912531 F426750A AC8A9DBB 51C54D8D$_{16}$
$\langle y_R \rangle_{256}$	6AB7DBF1 5FCBD768 EE68A173 F7B236EF C15A01E2 AA6CD1FE 98B947DA 7B38A2A0$_{16}$
θ	11B3A639 83BCCB6D 32C5943F 66F01D4C EA8CEE35 E4A6AE98 B1407C53 674317AC$_{16}$
Y	9B4EA669 DABDF100 A7D4B6E6 EB76EE52 51912531 F426750A AC8A9DBB 51C54D8D EB9289B5 0A46952D 0531861E 45A8814B 008FDC65 DE9FF1FA 2A1F16B6 A280E957 A814$_{16}$

В.5 Разбор токена ключа

В таблице В.5 представлен пример разбора токена ключа. Используются параметры эллиптической кривой, заданные в таблице Б.1, и ключи, заданные в В.1.

Таблица В.5 — Разбор токена ключа

X	4856093A 0F6C1301 5FC8E15F 1B23A762 02D2F4BA 6E5EC52B 78658477 F6486DE6 87AFAEEA 0EF7BC13 26A7DCE7 A10BA10E 3F91C012 6044B222 67BF30BD 6F1DA29E 0647CF39 C1D59A56 BB0194E0 F4F8A2BB$_{16}$
I	E12BDC1A E28257EC 703FCCF0 95EE8DF1$_{16}$
$\langle x_R \rangle_{256}$	4856093A 0F6C1301 5FC8E15F 1B23A762 02D2F4BA 6E5EC52B 78658477 F6486DE6$_{16}$
X_1	87AFAEEA 0EF7BC13 26A7DCE7 A10BA10E 3F91C012 6044B222 67BF30BD 6F1DA29E 0647CF39 C1D59A56 BB0194E0 F4F8A2BB$_{16}$
$\langle y_R \rangle_{256}$	7BD78A8A 5C49E878 7EFD2D70 B935CC10 0734283D B2E0A741 2ACBB0CA 5FD67493$_{16}$
θ	3E2D4915 38A58FA5 108CF809 85222670 661794AB 2423E410 9E785A22 D1529BC6$_{16}$
Y	B194BAC8 0A08F53B 366D008E 584A5DE4 8504FA9D 1BB6C7AC 252E72C2 02FDCE0D$_{16}$

В.6 Генерация псевдослучайных чисел

В таблице В.6 представлен пример генерации псевдослучайных чисел.

Таблица В.6 — Генерация псевдослучайных чисел

n	3
θ	E9DEE72C 8F0C0FA6 2DDB49F4 6F739647 06075316 ED247A37 39CBA383 03A98BF6$_{16}$
S	BE329713 43FC9A48 A02A885F 194B09A1 7ECDA4D0 1544AF8C A58450BF 66D2E88A$_{16}$
X	B194BAC8 0A08F53B 366D008E 584A5DE4 8504FA9D 1BB6C7AC 252E72C2 02FDCE0D 5BE3D612 17B96181 FE6786AD 716B890B 5CB0C0FF 33C356B8 35C405AE D8E07F99 E12BDC1A E28257EC 703FCCF0 95EE8DF1 C1AB7638 9FE678CA F7C6F860 D5BB9C4F$_{16}$
Y	1F66B5B8 4B733967 4533F032 9C74F218 34281FED 0732429E 0C79235F C273E269 4C0E74B2 CD5811AD 21F23DE7 E0FA742C 3ED6EC48 3C461CE1 5C33A77A A308B7D2 0F51D913 47617C20 BD4AB07A EF4F26A1 AD1362A8 F9A3D42F BE1B8E6F 1C88AAD5$_{16}$

Приложение Г
(рекомендуемое)
Модуль ACH.1

Г.1 Идентификаторы

Алгоритмам предстандарта присваиваются следующие идентификаторы:

`bign-with-hspec`	алгоритмы ЭЦП (6.3) с функцией хэширования, определяемой долговременными параметрами;
`bign-with-hbelt`	алгоритмы ЭЦП (6.3) с функцией хэширования *h*Belt;
`bign-genec`	алгоритм генерации параметров эллиптической кривой (6.1.3);
`bign-valec`	алгоритм проверки параметров эллиптической кривой (6.1.4);
`bign-genkeypair`	алгоритм генерации пары ключей (6.2.2);
`bign-valpubkey`	алгоритм проверки открытого ключа (6.2.3);
`bign-keytransport`	алгоритмы транспорта ключа (7.1);
`bign-prng`	алгоритм генерации псевдослучайных чисел (7.2).

В алгоритмах ЭЦП используется функция хэширования h. Идентификатор алгоритмов ЭЦП может либо явно определять h, либо указывать, что h задается ссылочно, через долговременные параметры алгоритмов. Явно определено использование функции хэширования *h*Belt. При задании h долговременными параметрами может использоваться компонент `hash` типа `DomainParameters` (тип определен ниже).

Уровень стойкости алгоритмов ЭЦП и транспорта ключа не указывается в их идентификаторах и определяется по размерностям параметров используемой эллиптической кривой. Размерности параметров и длина значений используемой функции хэширования должны соответствовать друг другу.

Открытому ключу, который вырабатывается по алгоритму из 6.2.2, присваивается идентификатор `bign-pubkey`. Открытый ключ может использоваться в алгоритмах ЭЦП и (или) транспорта ключа.

Стандартным параметрам эллиптической кривой, заданным в приложении Б, присваиваются следующие идентификаторы:

`bign-curve256v1`	параметры, определенные в таблице Б.1;
`bign-curve384v1`	параметры, определенные в таблице Б.2;
`bign-curve512v1`	параметры, определенные в таблице Б.3.

Г.2 Описание конечного поля и его элементов

Для описания конечного поля, над которым строится эллиптическая кривая, используется тип

```
FieldID ::= SEQUENCE {
  fieldType   OBJECT IDENTIFIER (bign-primefield),
  parameters  INTEGER
}
```

Компонент `fieldType` этого типа определяет вид поля. Примененный синтаксис обязывает использовать только простые конечные поля, которым назначен идентификатор `bign-primefield`. Компонент `parameters` описывает модуль p поля $\mathbb{F}_p$.

Элемент u поля $\mathbb{F}_p$ должен представляться значением типа `OCTET STRING`. На уровне стойкости l искомое значение должно быть строкой из $l/4$ последовательных октетов двоичного слова $\langle u \rangle_{2l}$.

Г.3 Описание долговременных параметров

Долговременные параметры алгоритмов ЭЦП и транспорта ключа могут представляться значениями типа

```
DomainParameters ::= SEQUENCE {
  ecp   ECParameters,
  hash  OBJECT IDENTIFIER OPTIONAL
}
```

Компонент `ecp` этого типа описывает параметры используемой эллиптической кривой. Компонент `hash` описывает идентификатор алгоритма хэширования, который применяется в алгоритмах `bign-with-hspec`. В других алгоритмах компонент `hash` не используется и поэтому может опускаться.

Параметры эллиптической кривой могут определяться тремя способами:

```
ECParameters ::= CHOICE {
  specified  SpecifiedECParameters,
  named      OBJECT IDENTIFIER,
  implicit   NULL
}
```

Выбор компонента `specified` означает явное задание параметров. Компонент `named` используется для ссылки на именованные параметры, заданные в приложении А или в другом документе. Компонент `implicit` используется для указания на то, что наследуются параметры внешнего источника, например удостоверяющего центра.

Явно задаваемые параметры должны представляются значениями типа

```
SpecifiedECParameters ::= SEQUENCE {
  version  INTEGER {ecpVer1(1)} (ecpVer1),
  fieldID  FieldID,
  curve    Curve,
  order    INTEGER,
  base     OCTET STRING (SIZE(32|48|64))
}
```

Компонент `version` указывает на версию данного типа ACH.1. Примененный синтаксис обязывает использовать версию 1, которая обозначена через `ecpVer1`. Компонент `fieldID` описывает поле $\mathbb{F}_p$, над которым строится эллиптическая кривая. Компонент `curve` описывает уравнение эллиптической кривой. Компонент `order` описывает по-

рядок q группы точек эллиптической кривой. Компонент `base` описывает базовую точку эллиптической кривой.

Для описания уравнения эллиптической кривой используется тип

```
Curve ::= SEQUENCE {
  a     OCTET STRING (SIZE(32|48|64)),
  b     OCTET STRING (SIZE(32|48|64)),
  seed  BIT STRING (SIZE(64))
}
```

Последовательные компоненты этого типа определяют коэффициенты a, b эллиптической кривой и параметр *seed*, использованный для построения b при заданных p и a.

Коэффициенты a, b и базовая точка $G = (0, y_G)$ задаются строками октетов. Эти строки строятся по a, b и y_G как элементам поля $\mathbb{F}_p$ по правилам, определенным в Г.3.

Г.4 Описание открытого ключа

Открытый ключ алгоритмов ЭЦП и транспорта ключа может быть описан с помощью АСН.1 различными способами. Если открытый ключ используется в сертификатах и списках отозванных сертификатов СТБ 34.101.19, то он может представляться значениями типа

```
SubjectPublicKeyInfo ::= SEQUENCE {
  algorithm         AlgorithmID,
  subjectPublicKey  BIT STRING (SIZE(512|768|1024))
}
```

Компонент `algorithm` этого типа описывает свойства открытого ключа. Компонент `subjectPublicKey` описывает значение открытого ключа.

Для описания свойств открытого ключа используется тип

```
AlgorithmID ::= SEQUENCE {
  algorithm   OBJECT IDENTIFIER (bign-pubkey),
  parameters  DomainParameters
}
```

Компонент `algorithm` этого типа определяет идентификатор открытого ключа. Примененный синтаксис обязывает использовать только идентификатор `bign-pubkey`. Компонент `parameters` описывает долговременные параметры алгоритмов, с которыми используется открытый ключ.

На уровне стойкости l открытому ключу $Q = (x_Q, y_Q)$ ставится в соответствие двоичное слово $\langle x_Q \rangle_{2l} \,\|\, \langle y_Q \rangle_{2l}$. Это слово должно быть значением `subjectPublicKey`.

Г.5 Описание личного ключа

Личный ключ алгоритмов ЭЦП и транспорта ключа может представляться значениями типа

```
PrivateKey ::= SEQUENCE {
  privateKey  OCTET STRING (SIZE(32|48|64)),
  parameters  DomainParameters OPTIONAL,
  publicKey   BIT STRING (SIZE(512|768|1024)) OPTIONAL
}
```

Компонент `privateKey` этого типа описывает значение личного ключа. Необязательные компоненты `parameters`, `publicKey` определяются так же, как в Г.5.

На уровне стойкости l личному ключу d ставится в соответствие двоичное слово $\langle d \rangle_{2l}$, которое разбивается на $l/4$ последовательных октетов. Полученная строка октетов определяет значение компонента `privateKey`.

Г.6 Описание ЭЦП

На уровне стойкости l ЭЦП является двоичным словом длины $3l$. Рекомендуется задавать ЭЦП значением типа

```
Signature ::= BIT STRING (SIZE(384|576|768))
```

Г.7 Модуль ACH.1

```
Bign-module-v1 {iso(1) member-body(2) by(112) 0 2 0 34 101 45 module(1) ver1(1)}
DEFINITIONS ::=
BEGIN
  bign OBJECT IDENTIFIER ::= {iso(1) member-body(2) by(112) 0 2 0 34 101 45}

  bign-with-hspec OBJECT IDENTIFIER ::= {bign 11}
  bign-with-hbelt OBJECT IDENTIFIER ::= {bign 12}
  bign-genec OBJECT IDENTIFIER ::= {bign 21}
  bign-valec OBJECT IDENTIFIER ::= {bign 22}
  bign-genkeypair OBJECT IDENTIFIER ::= {bign 31}
  bign-valpubkey OBJECT IDENTIFIER ::= {bign 32}
  bign-keytransport OBJECT IDENTIFIER ::= {bign 41}
  bign-prng OBJECT IDENTIFIER ::= {bign 51}

  bign-keys OBJECT IDENTIFIER ::= {bign keys(2)}
  bign-pubkey OBJECT IDENTIFIER ::= {bign-keys 1}

  bign-curves OBJECT IDENTIFIER ::= {bign curves(3)}
  bign-curve256 OBJECT IDENTIFIER ::= {bign-curves 1}
  bign-curve384 OBJECT IDENTIFIER ::= {bign-curves 2}
  bign-curve512 OBJECT IDENTIFIER ::= {bign-curves 3}

  bign-fields OBJECT IDENTIFIER ::= {bign fields(4)}
  bign-primefield OBJECT IDENTIFIER ::= {bign-fields prime(1)}
```

```
DomainParameters ::= SEQUENCE {
  ecp   ECParameters,
  hash  OBJECT IDENTIFIER OPTIONAL
}

ECParameters ::= CHOICE {
  specified  SpecifiedECParameters,
  named      OBJECT IDENTIFIER,
  implicit   NULL
}

SpecifiedECParameters ::= SEQUENCE {
  version  INTEGER {ecpVer1(1)} (ecpVer1),
  fieldID  FieldID,
  curve    Curve,
  order    INTEGER,
  base     OCTET STRING (SIZE(32|48|64))
}

FieldID ::= SEQUENCE {
  fieldType   OBJECT IDENTIFIER (bign-primefield),
  parameters  INTEGER
}

Curve ::= SEQUENCE {
  a     OCTET STRING (SIZE(32|48|64)),
  b     OCTET STRING (SIZE(32|48|64)),
  seed  BIT STRING (SIZE(64))
}

SubjectPublicKeyInfo ::= SEQUENCE {
  algorithm         AlgorithmID,
  subjectPublicKey  BIT STRING (SIZE(512|768|1024))
}

AlgorithmID ::= SEQUENCE {
  algorithm   OBJECT IDENTIFIER (bign-pubkey),
  parameters  DomainParameters
}

PrivateKey ::= SEQUENCE {
  privateKey  OCTET STRING (SIZE(32|48|64)),
  parameters  DomainParameters OPTIONAL,
```

```
    publicKey   BIT STRING (SIZE(512|768|1024)) OPTIONAL
  }

  Signature ::= BIT STRING (SIZE(384|576|768))
END
```

Приложение Д
(рекомендуемое)
Алгоритм Миллера — Рабина

Д.1 Входные и выходные данные

Входными данными алгоритма Миллера — Рабина являются натуральное нечетное число n, простота которого проверяется, и число итераций T.

Выходными данными алгоритма является ответ ДА или НЕТ. Ответ ДА означает, что n вероятно простое. Ответ НЕТ означает, что n — составное.

Для простых n алгоритм всегда выдает верный ответ ДА. Для составных n может быть получен как верный ответ НЕТ, так и ошибочный ответ ДА. Вероятность ошибочного ответа уменьшается с ростом числа итераций и не превосходит 2^{-2T}. Если l — длина двоичного представления n, т. е. $2^{l-1} \leqslant n < 2^l$, то рекомендуется выбирать $T \geqslant l/4$.

Д.2 Вспомогательные переменные

Используются переменные a, b со значениями из $\{1, 2, \ldots, n-1\}$.

Д.3 Алгоритм

Проверка простоты n состоит в выполнении следующих шагов:

1 Представить n в виде $2^s r + 1$, где s — натуральное, r — натуральное нечетное.

2 Для $t = 1, 2, \ldots, T$ выполнить:
- 1) $a \xleftarrow{R} \{2, 3, \ldots, n-2\}$;
- 2) $b \leftarrow a^r \bmod n$;
- 3) если $b = 1$ или $b = n - 1$, то перейти к шагу 2.6;
- 4) для $i = 1, 2, \ldots, s-1$ выполнить:
  - (a) $b \leftarrow b^2 \bmod n$;
  - (b) если $b = 1$, то возвратить НЕТ;
  - (c) если $b = n - 1$, то перейти к шагу 2.6;
- 5) возвратить НЕТ;
- 6) продолжить.

3 Возвратить ДА.

Библиография

[1] Лидл Р., Нидеррайтер Г. Конечные поля
М.: Мир, 1988

[2] Schnorr C. P. Efficient Signature Generation by Smart Cards
J. Cryptology, 4(3): 161–174, 1991

[3] Hankerson D., Menezes A., Vanstone S. Guide to Elliptic Curve Cryptography
N. Y.: Springer, 2004