Алгоритм цифровой подписи DSS

Алгоритм предложен NIST (Национальным институтом стандартов и технологий США) в 1991 и был запатентован, но НИСТ сделал патент доступным для использования без лицензий.

Алгоритм вместе с хеш-функцией SHA-1 является частью **DSS** (Digital Signature Standard), впервые опубликованного в 1994 г.

Алгоритм имеет теоретико-числовой характер и основан на криптосистеме Эль-Гамаля. Для хеширования документа перед его подписыванием рекомендуется алгоритм SHA-1 (позднее SHA-2).

По сути стандарт DSS – это схема подписи Эль-Гамаля, в которой конструируется конечное поле элементов F_p, содержащее меньшую подгруппу простого порядка q. Если в современных криптосистемах Эль-Гамаля выбирают $p \approx 2^{1024}$, то параметр q устанавливают равным $\approx 2^{160}$. Однако это не приводит к уменьшению сложности дискретного логарифмирования. Поэтому DSS позволяет уменьшить длину подписи по сравнению с подписью Эль-Гамаля.

Для подписи документа m подписывающий должен выбрать:

1. Криптографическую хеш-функцию $H(m)$.
2. Большое N-битовое простое число q (N совпадает с битовой длиной хэш-функции $H(m)$).
3. L-битовое простое число p такое, чтобы $(p-1)\vdots q$.
4. Число g, мультипликативный порядок которого по модулю p равен q. Для нахождения g выбирают случайное число B из интервала $(1; p-1)$ и проверяют неравенство $g \equiv B^{\frac{p-1}{q}} \pmod p \neq 1$. Если оно не выполнено, то переходят к другому случайному числу B, пока не получат $g \neq 1$. Когда такое число B найдено, то это гарантия того, что g – элемент цикличной группы порядка q (В большинстве случаев значение $B = 2$ удовлетворяет этому требованию).
5. ***Секретный ключ*** – целое число a из интервала $(0; q)$.

Открытый ключ вычисляется по формуле $y = g^a \pmod p$.

Открытые параметры – числа (p, q, g, y). Закрытый параметр один – число a. Числа (p, q, g) могут быть общими для группы пользователей, поэтому эти параметры называют ***доменными параметрами***. Числа a и y – закрытый и открытый ключи конкретного пользователя. Для подписи используют секретный ключ a

и ***эфемерный ключ*** k, причем число k должно выбираться случайно для подписи каждого нового сообщения.

Замечание. Рекомендуемые в стандарте пары значений чисел L и N в битах:

L = 1024, N = 160
L = 2048, N = 224
L = 2048, N = 256
L = 3072, N = 256

Пусть теперь пользователь **A** хочет подписать сообщение m и отправить его пользователю **B**.

Формирование подписи отправителем **A**:

1. Вычислить хэш сообщения $H = H(m)$.
2. Выбрать ***эфемерный ключ*** k – случайное число из интервала $(0; q)$.
3. Вычислить $R = (g^k (\mathrm{mod}\, p))(\mathrm{mod}\, q)$.
4. Найти $S = (H + aR)k^{-1} (\mathrm{mod}\, q)$.
5. Подпись сообщения образует пара (R, S), общая длина подписи $2N$ (порядка 320 битов). Она пересылается пользователю **B** вместе с сообщением. Если оказалось, что $R = 0$ или $S = 0$, то выбирается другое k.

Проверка подписи получателем **B**:

1. Вычислить хэш присланного сообщения $H = H(m)$.
2. Определить $H \cdot S^{-1} (\mathrm{mod}\, q)$ и $R \cdot S^{-1} (\mathrm{mod}\, q)$.
3. Вычислить $V = (g^{H \cdot S^{-1}} y^{R \cdot S^{-1}} \mathrm{mod}\, p)(\mathrm{mod}\, q)$ на открытом ключе y пользователя **A**.
4. Подпись считается законной, если $V = R$.

Математические обоснование законности проверки подписи:

Если $g \equiv B^{\frac{p-1}{q}} (\mathrm{mod}\, p)$, то из этого по малой теореме Ферма следует $g^q = B^{p-1} = 1 (\mathrm{mod}\, p)$. Поскольку первообразный корень g является простым числом, то его порядок равен q. Из подписи сообщения $S \equiv (H + aR)k^{-1} (\mathrm{mod}\, q) \Rightarrow S\,k \equiv (H + aR)(\mathrm{mod}\, q) \Rightarrow$

$$k \equiv HS^{-1} + aRS^{-1} (\mathrm{mod}\, q).$$

Так как g имеет порядок q, получим

$$g^k \equiv g^{(H \cdot S^{-1} + aRS^{-1}) \mathrm{mod}\, q} \equiv g^{H \cdot S^{-1} \mathrm{mod}\, q} (\underbrace{g^a}_{y})^{RS^{-1} \mathrm{mod}\, q} \mathrm{mod}\, p \equiv$$

$$\equiv g^{H\cdot S^{-1}} y^{RS^{-1}} \bmod p.$$

Наконец, корректность схемы DSA следует из

$$R = (g^k \bmod p) \bmod q = (g^{H\cdot S^{-1}} y^{R\cdot S^{-1}} \bmod p) \bmod q = V.$$

Пример. $q = 13$, $p = 4q + 1 = 53$ и $g = 16$ – открытые параметры домена. Ключевая пара пользователя имеет вид $a = 3$ и $y = g^a (\bmod p) \equiv 16^3 (\bmod 53) \equiv 15$. Подписать сообщение M с хэшем $H = 5$. Выполнить верификацию подписи.

Р е ш е н и е. Пусть эфемерный ключ $k = 2$.

$$R = (g^k (\bmod p))(\bmod q) \equiv (16^2 (\bmod 53)(\bmod 13) = 5;$$

$$k^{-1} (\bmod q) = 2^{-1} (\bmod 13) = 7$$

$$S = (H + aR) k^{-1} (\bmod q) = (5 + 3\cdot 5)\cdot 7 (\bmod 13) = 10.$$

Подпись сообщения $\langle M, 5, 10 \rangle$. Выполним проверку:

$$S^{-1} (\bmod q) = 10^{-1} (\bmod 13) = 4,$$

$$A = H \cdot S^{-1} (\bmod q) = 5\cdot 4 (\bmod 13) \equiv 7,$$

$$B = R \cdot S^{-1} (\bmod q) = 5\cdot 4 (\bmod 13) \equiv 7.$$

$$V = (g^A y^B (\bmod p))(\bmod q) = (16^7 \cdot 15^7 (\bmod 53))(\bmod 13) \equiv 5 \equiv R.$$

Подпись законна.

НИСТ сделал алгоритм свободным к использованию, и он может свободно реализовываться программным или аппаратным образом. Кроме того, НИСТ'ом разработана программа проверки соответствия реализации алгоритма стандарту.

По сравнению с алгоритмом цифровой подписи Эль Гамаля алгоритм DSA имеет следующие основные преимущества:

1. При любом допустимом уровне стойкости, т.е. при любой паре чисел G и P (от 512 до 1024 бит), числа q, X, r, s имеют длину по 160 бит, сокращая длину подписи до 320 бит.

2. Большинство операций с числами K, r, s, X при вычислении подписи производится по модулю числа q длиной 160 бит, что сокращает время вычисления подписи.

3. При проверке подписи большинство операций с числами u_1, u_2, v, w также производится по модулю числа q длиной 160 бит, что сокращает объем памяти и время вычисления.

Недостатком алгоритма DSA является то, что при подписывании и при проверке подписи приходится выполнять сложные операции деления по модулю q:

$$s = \frac{m+rX}{K} \pmod q, \quad w = \frac{1}{s} \pmod q,$$

что не позволяет получать максимальное быстродействие.

Следует отметить, что реальное исполнение алгоритма DSA может быть ускорено с помощью выполнения предварительных вычислений. Заметим, что значение r не зависит от сообщения M и его хэш-значения m. Можно заранее создать строку случайных значений K и затем для каждого из этих значений вычислить значения r. Можно также заранее вычислить обратные значения K^{-1} для каждого из значений K. Затем, при поступлении сообщения M, можно вычислить значение s для данных значений r и K^{-1}. Эти предварительные вычисления значительно ускоряют работу алгоритма DSA.

§ 8. Цифровая подпись на эллиптической кривой (ECDSA)

Американский алгоритм ECDSA (Elliptic Curve Digital Signature Algorithm) для цифровой подписи аналогичен по своему строению DSS, но работает в отличие от него не над полем целых чисел, а в группе точек эллиптической кривой. Принят в качестве международного стандарта. В этом стандарте используются эллиптические кривые над полем $GF(2)$ характеристики 2. Но криптографически стойких кривых над $GF(2)$ сравнительно мало, поэтому рассмотрим ЭЦП на эллиптических кривых, заданных над полем $GF(p)$ большей характеристики (именно такого типа стандарт ЭЦП ГОСТ 34.10-2001 принят в России).

Пусть $p > 2$ – простое число, а эллиптическая кривая $E_p(a,b)$ задана уравнением $y^2 = x^3 + ax + b (\mathrm{mod}\, p)$, N – число точек кривой (порядок кривой). Кривую следует выбрать так, чтобы выполнялись такие условия:

1. $$\begin{cases} p+1-2\sqrt{p} \le N \le p+1+2\sqrt{p}, & (*) \\ G \in E_p(a,b) \Rightarrow N \cdot G = O. & (**) \end{cases}$$

Чисел, удовлетворяющих условию (*) обычно много, поэтому чтобы отсеять лишние числа из интервала $(p+1-2\sqrt{p};\ p+1+2\sqrt{p})$ можно проверять условие (**) для разных точек G. Единственное оставшееся число и будет искомым порядком кривой (кроме того,

определить N можно и специальными методами оптимизации – методом Шуфа, методом больших-малых шагов).

2. $N \neq p$ и $N \neq p+1$;

http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1029_report.pdf

Implication 1. *For elliptic curve cryptography we select an elliptic curve such that*

$$\#E(K) = N = h \cdot l$$

where l is a large prime and h is very small. Usually one chooses $h = 1, 2$ or 4.

The security criteria for elliptic curves, so that the discrete logarithm problem is difficult, is that one should choose curves E over finite fields $K = \mathbb{F}_q$ such that if

$$\#E(K) = h \cdot l$$

where l is prime, then

- $l > 2^{160}$.
- l should not divide $q^k - 1$ for $k \leq 30$.
- $l \neq q$.
- q should be equal to a large prime, or a prime power of two.

3. $p^k \neq 1 (\mathrm{mod}\, N)$ для всех $k = 1, 2, ..., C$, где C настолько велико, что вычислить дискретный логарифм в $GF(p^C)$ за приемлемое время невозможно. В настоящее время считается достаточным значение $C = 30$;

4. Порядок N кривой должен иметь большой простой делитель (больше 2^{160}). Если такого делителя не существует, то процедуру выбора кривой повторяют.

Порядок n базовой точки G, используемой в системе ЭЦП, должен быть простым числом, причем $n > \max\left\{2^{160}; 4\sqrt{p}\right\}$. Такую базовую точку можно выбрать следующим образом. Найти случайную точку G_1 на кривой и вычислить точку $G = \frac{N}{n} G_1$. Если $G \neq O$, то требуемая точка найдена, если же $G = O$, то выбираем другую точку G_1.

Замечание. Кривая $E_p(a,b)$ и точка G могут быть общими для всех пользователей.

Индивидуальные параметры пользователей – это секретный ключ открытый ключи. ***Закрытый ключ*** – случайное число d из интервала $(0; n)$, ***открытый ключ*** – точка эллиптической кривой $Q = d \cdot G$. Алгоритм ЕЦП также использует хеш-функцию $H(m)$.

Формирование подписи отправителем **A**:

1. Вычислить хэш сообщения $H = H(m)$.
2. Выбрать случайное число k из интервала $[1, n-1]$.
3. Найти точку $(x, y) = k \cdot G$.

4. Вычислить $R \equiv x(\mathrm{mod}\, n)$.

5. Если $R = 0$, то вернуться к шагу 2.

6. Вычислить $S = k^{-1}(H + d \cdot R)(\mathrm{mod}\, n)$.

7. Если $S = 0$, то вернуться к шагу 2.

Подпись сообщения – пара (R, S). Она пересылается пользователю **B** вместе с сообщением.

Замечание: если $R = 0$, то результат вычисления S не зависит от секретного ключа d, а если $S = 0$, то необходимого для проверки числа $S^{-1}(\mathrm{mod}\, n)$ не существует.

Проверка подписи получателем **B**:

1. Если хотя бы одно из условий $1 \le R \le n-1,\ 1 \le S \le n-1$ нарушено, то подпись фальшивая.

2. Вычислить хэш сообщения $H = H(m)$.

3. Вычислить $U_1 = H \cdot S^{-1}(\mathrm{mod}\, n)$.

4. Вычислить $U_2 = R \cdot S^{-1}(\mathrm{mod}\, n)$.

5. Вычислить точку $(x; y) = U_1 \cdot G + U_2 \cdot Q$.

6. Если $R \equiv x(\mathrm{mod}\, n)$, то подпись действительная, иначе она фальшивая.

Таким образом, эта схема является переложением стандарта DSA на эллиптические кривые. *В одной из работ высказано предположение о том, что этот переход связан с большими успехами специальных служб России и США в решении задач дискретного логарифмирования.*

Преимущества ECDSA над DSA

- ECDSA позволяет работать над полями значительно меньших размеров, чем поле $GF(p)$. Предполагается, что битовый размер открытого ключа в ESDSA равен двойному размеру секретного ключа в битах. Для сравнения, при уровне безопасности в 80 бит атакующему необходимо проверить 2^{80} версий подписи для нахождения секретного ключа. При этом размер открытого ключа DSA равен 1024 бит, тогда как для открытого ключа ECDSA — 160 бит. С другой стороны размер подписи одинаков и для DSA, и для ECDSA: 4*t* бит, где *t* — уровень безопасности, измеренный в битах, то есть при уровне безопасности в 80 бит примерно 320 бит для DSA и 80 бит для ECDSA.

Некоторые проблемы и трудности в использовании систем на основе эллиптических кривых.

1) ***Реальная безопасность эллиптических систем все еще недостаточно осознана***. Главная проблема состоит в том, что

истинная сложность дискретного логарифмирования на эллиптической кривой ещё не осознана полностью. Недавнее исследование показало, что некоторые использовавшиеся для отработки алгоритмов шифрования эллиптические кривые, фактически не подходят для таких операций.

2) ***Трудность генерации подходящих кривых***. При определении системы эллиптической кривой требуются кривая и базовая точка, создать которые трудно. Главная проблема – подсчитать количество точек на кривой. Для этого необходимо выбрать подходящую базовую точку, координаты которой должны иметь достаточно большое значение, чтобы гарантировать трудность взлома. Но координаты должны делиться на количество точек на кривой (помните, что точки на кривой вместе с бесконечно удаленной точкой образуют конечную группу). И весьма вероятно, что, найдя число точек на кривой, мы не сможем найти базовую точку.

Подводя итог вышеизложенному, можно утверждать, что создание кривых – непростая задача. Пользователи могут использовать «стандартные» кривые, используя специальное программное обеспечение (типа THALES “Elliptic Curve Generation Bureau”), либо создавать собственные кривые, что занимает много времени.

3) ***Относительно медленная проверка цифровой подписи***.

Как уже было упомянуто, системы на основе эллиптической кривой используют ключи малых размеров. Это снижает требования к вычислительным мощностям по сравнению с требованиями систем на основе RSA. Как это влияет на скорость обработки? Следующая таблица показывает сравнительные характеристики алгоритмов RSA и ECDSA при создании и проверки подписей (проверка подписи RSA с экспонентой e = 65537).

	Создание подписи	Проверка подписи
RSA (1024 бита)	25 ms	< 2 ms
ECDSA (160 бит)	32 ms	33 ms
RSA (2048 битов)	120 ms	5ms
ECDSA (216 битов)	68 ms	70 ms

Таким образом, при увеличении размеров ключа создание подписей с помощью ECDSA производится значительно быстрее, чем в аналогичных RSA системах. Это различие в ещё большей степени проявляется для однопроцессорных систем. С другой стороны проверка подписи с помощью ECDSA производится намного медленнее, чем эта же процедура в системах RSA и опять же это различие усиливается для систем с одним процессором. Мощность

процессора, затраченная на проверку подписи при использовании, скажем, ECDSA, может замедлить выполнение других приложений в системе. Множество систем имеют большое количество удаленных устройств, соединенных с центральным сервером и время, затраченное удаленным устройством для создания подписи – несколько секунд – не влияет на производительность системы в целом, но сервер должен также и подтверждать подписи причем очень быстро и в некоторых случаях системы RSA (даже использующие большие ключи) возможно, будут более приемлемы для использования, чем криптосистемы на основе эллиптической кривой.

http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf вібор кривіх и рекомендации

CERTICOM'S ECC CHALLENGE. Certicom initiated an ECC challenge [18] in November 1997 in order to encourage and stimulate research on the ECDLP. Their challenges consist of instances of the ECDLP on a selection of elliptic curves. The challenge curves are divided into three categories listed below. In the following, ECCp-k denotes a random curve over a field $\mathbb{F}_p$, ECC2-k denotes a random curve over a field $\mathbb{F}_{2^m}$, and ECC2K-k denotes a Koblitz curve (see §5.3) over $\mathbb{F}_{2^m}$; k is the bitlength of n. In all cases, the bitsize of the order of the underlying finite field is equal or slightly greater than k (so curves have either prime order or almost prime order).

1. Randomly generated curves over $\mathbb{F}_p$, where p is prime: ECCp-79, ECCp-89, ECCp-97, ECCp-109, ECCp-131, ECCp-163, ECCp-191, ECCp-239, and ECCp-359.
2. Randomly generated curves over $\mathbb{F}_{2^m}$, where m is prime: ECC2-79, ECC2-89, ECC2-97, ECC2-109, ECC2-131, ECC2-163, ECC2-191, ECC2-238, and ECC2-353.
3. Koblitz curves over $\mathbb{F}_{2^m}$, where m is prime: ECC2K-95, ECC2-108, ECC2-130, ECC2-163, ECC2-238, and ECC2-358.

RESULTS OF THE CHALLENGE. Escott et al. [25] report on their 1998 implementation of the parallelized Pollard's rho algorithm which incorporates some improvements of Teske [107]. The hardest instance of the ECDLP they solved was the Certicom ECCp-97 challenge. For this task they utilized over 1200 machines from at least 16 countries, and found the answer in 53 days. The total number of steps executed was about 2×10^{14} elliptic curve additions which is close to the expected time $((\sqrt{\pi n})/2 \approx 3.5 \times 10^{14}$, where $n \approx 2^{97})$. Escott et al. [25] conclude that the running time of Pollard's rho algorithm in practice fits well with the theoretical predictions. They estimate that the ECCp-109 challenge could be solved by a network of 50,000 Pentium Pro 200MHz machines in about 3 months.

10.2 NIST Recommended Curves

This subsection presents the 15 elliptic curves that were recommended (but not mandated) by NIST for U.S. Federal Government use [74].

RECOMMENDED FINITE FIELDS. There are 10 recommended finite fields:

1. The prime fields $\mathbb{F}_p$ for $p = 2^{192} - 2^{64} - 1$, $p = 2^{224} - 2^{96} + 1$, $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$, and $p = 2^{521} - 1$.
2. The binary fields $\mathbb{F}_{2^{163}}$, $\mathbb{F}_{2^{233}}$, $\mathbb{F}_{2^{283}}$, $\mathbb{F}_{2^{409}}$, and $\mathbb{F}_{2^{571}}$.

The factors which influenced the choices of fields were:

(i) The fields were selected so that the bitlengths of their orders are twice the key lengths of common symmetric-key block ciphers — this is because exhaustive key search of a k-bit block cipher is expected to take roughly the same time

as the solution of an instance of the elliptic curve discrete logarithm problem using Pollard's rho algorithm for an appropriately-selected elliptic curve over a finite field whose order has bitlength $2k$. The correspondence between symmetric cipher key lengths and field sizes is given in Table 1.

Symmetric cipher key length	Example algorithm	Bitlength of p in prime field F_p	Dimension m of binary field F_{2^m}
80	SKIPJACK [77]	192	163
112	Triple-DES	224	233
128	AES Small [75]	256	283
192	AES Medium [75]	384	409
256	AES Large [75]	521	571

Table 1. Recommended field sizes for U.S. Federal Government use.

(ii) For prime fields $\mathbb{F}_p$, the prime moduli p are of a special type (called *generalized Mersenne numbers*) for which modular multiplication can be carried out more efficiently than in general; see [74] and [101].

(iii) For binary fields $\mathbb{F}_{2^m}$, m was chosen so that there exists a Koblitz curve of almost prime order over $\mathbb{F}_{2^m}$. Since $\#E(\mathbb{F}_{2^l})$ divides $\#E(\mathbb{F}_{2^m})$ whenever l divides m, this requirement imposes the condition that m be prime.

RECOMMENDED ELLIPTIC CURVES. There are three types of elliptic curves:

1. Random elliptic curves over $\mathbb{F}_p$.
2. Koblitz elliptic curves over $\mathbb{F}_{2^m}$.
3. Random elliptic curves over $\mathbb{F}_{2^m}$.

- DSS permits using any of three digital signature algorithms:
  - RSA
  - Digital Signature Algorithm (DSA) -- a variation of El Gamal signature algorithm
  - Elliptic Curve Digital Signature Algorithm (ECDSA)
- DSS permits using the SHA-1 or SHA-2 hash functions
- DSS with RSA
  - Must use a 1024-, 2048-, or 3072-bit modulus n
  - Security based on hardness of integer factorization
- DSS with DSA
  - Uses a subgroup of Z_n^*, subgroup order is q
  - Allowed sizes:
    - p = 1024 bits, q = 160 bits
    - p = 2048 bits, q = 224 bits
    - p = 2048 bits, q = 256 bits
    - p = 3072 bits, q = 256 bits
  - Security based on hardness of discrete logarithm problem in Z_n^*
- DSS with ECDSA
  - Uses an elliptic curve group (mod p)
  - Recommended curves (bit sizes q): P-192, P-224, P-256, P-384, P-521
  - Security based on hardness of discrete logarithm problem in elliptic curve group
- Signature generation computations
  - RSA: One exponentiation modulo n, ($\log_2$ n)-bit exponent
  - DSA: One exponentiation modulo p, q-bit exponent, plus some small stuff
  - ECDSA: One point multiplication, q-bit multiplier

- Signature size
  - RSA: One ($\log_2 n$)-bit number
  - DSA: Two q-bit numbers
  - ECDSA: Two q-bit numbers
- Signature verification computations
  - RSA: One exponentiation modulo n, small exponent (if we use e = 3 or 65537)
  - DSA: Two exponentiations modulo p, q-bit exponents, plus some small stuff
  - ECDSA: Two point multiplications, q-bit multipliers, one point addition, plus some small stuff
- RSA vs. DSA or ECDSA
  - RSA signature generation takes more time than DSA or ECDSA
  - RSA signatures are larger than DSA or ECDSA signatures
  - RSA signature verification (with small e) takes much less time than DSA or ECDSA signature verification
- RSA is the most popular digital signature algorithm
  - Faster signature verification
  - Simpler implementation