ЭЦП

Определение. ЭЦП (**Е**) – бинарная последовательность, которая добавляется к подписываемому документу **М** (также представленному в двоичной форме) и зависящая от **М** и от ключа (**К**); (**n>k**)

Е различна при неизменном **К**, но разных **М**

Функции и назначение ЭЦП

1. Подпись должна быть достоверна: подписавший документ человек сделал это осознано.
2. Подпись неподдельна. Подписавший — автор подписи.
3. Подпись невозможно использовать повторно, мошенник не должен иметь возможность переносить подпись без ведома подписавшегося.
4. Подписанный документ не может быть изменен, особенно, если сделано несколько копий.
5. От подписи нельзя отречься

Основные типы ЭЦП

1. На основе симметричных криптосистем
2. На основе симметричных криптосистем и посредника
3. На основе асимметричных криптосистем
4. На основе асимметричных криптосистем и однонаправленных хэш-функций

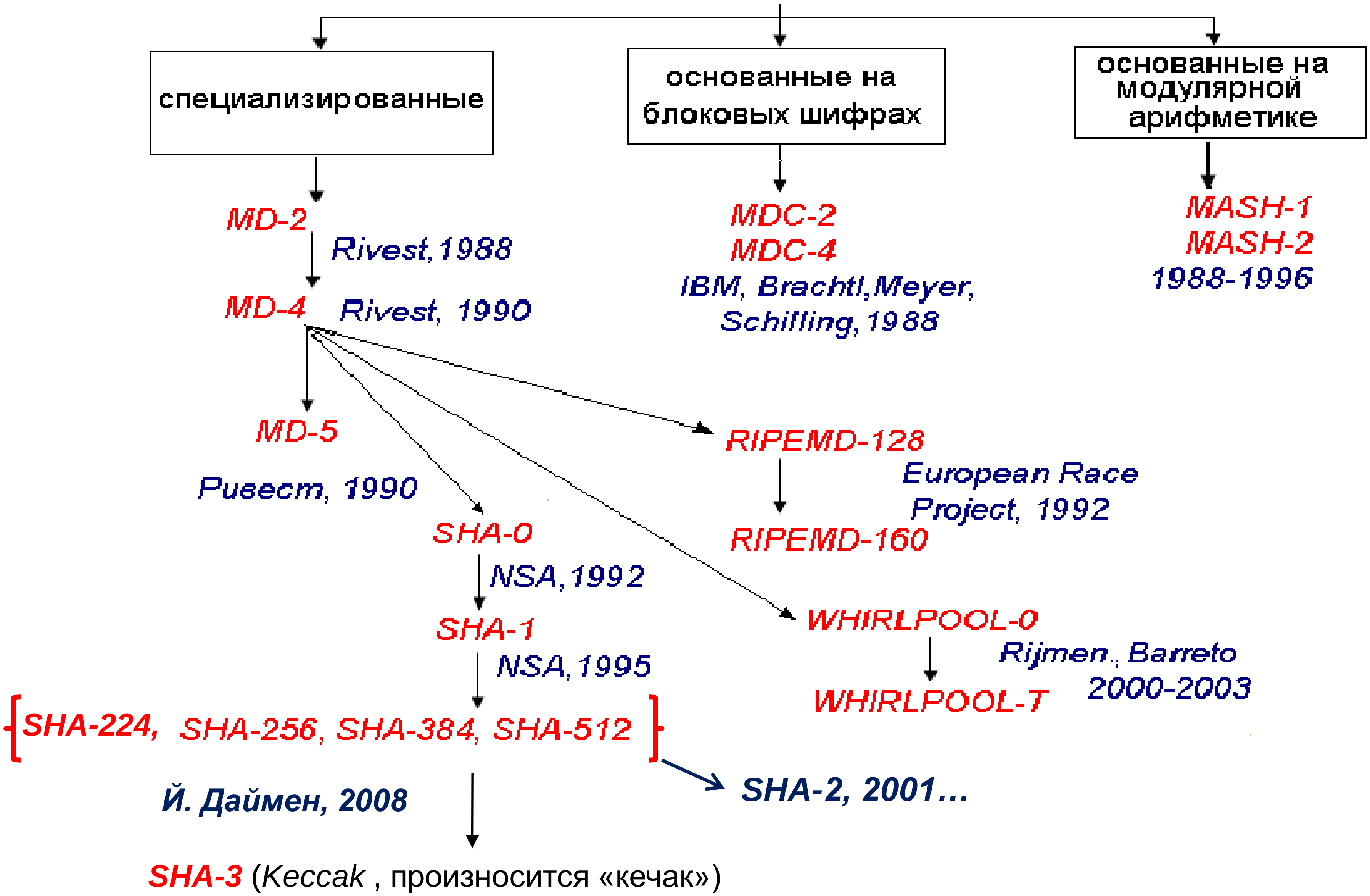
Понятие хеш-функции

Опр. ***Хеш-функция*** – математическая или иная ф., **h=H(M)** которая принимает на входе строку символов переменной (произвольной) длины и преобразует ее в выходную строку фиксированной (обычно – меньшей) длины,

Опр. Хеширование (или *хэширование*, англ. ***hashing***) – это преобразование входного массива данных определенного типа и произвольной длины в выходную битовую строку фиксированной длины.

Преобразования называются *хеш-функциями* или функциями свертки, а их результаты называют *хешем*, *хеш-кодом*, *хеш-таблицей* или *дайджестом* сообщения (анг. *message digest*).

Типы хеш-функций



Криптографические хеш-функции

Криптографическая хеш-функция — это специальный класс хеш-функций, который имеет различные свойства, необходимые для криптографии:

- аутентификация (хранение паролей),
- проверка целостности данных,
- защита файлов,
- обнаружение зловредного ПО,
- криптовалютные технологии

Свойства хеш-функций

Свойство 1: **Детерминированность:** независимо от того, сколько раз вычисляется **H(M),** M - const , при использовании одинакового **h** всегда получается тот же результат.

Свойство 2: **Быстрое вычисление H(M):** если процесс вычисления не достаточно быстрый, система просто не будет эффективна.

Свойство 3: **Сложность обратного вычисления:** для известного H (M) невозможно (практически) определить M, это свойство **односторонности** преобразования
(Пример. при длине хеша 128 бит в среднем нужно проверить – по методу «грубой силы» - (2^128)/2 = 2^127 вариантов) (!!!!)

Об односторонности хеш-функций на основе блочных шифров

Блочный шифр необратим по ключу шифрования, и, если в качестве ключа шифрования использовать выход предыдущего шага хеш-преобразования, а в качестве шифруемого сообщения - очередной блок сообщения (или наоборот), то можно получить хеш-функцию с хорошими криптографическими характеристиками с точки зрения односторонности.

Такой подход использовался, например, в российском стандарте хеширования – **ГОСТ Р 34.11-94**

Основным недостатком хеш-функций на основе блочных шифров является невысокая производительность.

Свойство 4: **Небольшие изменения в вводимых данных (М) изменяют хеш *h(M)***

Пример. Используеются алгоритмы хеширования **MD5** и **SHA1**

M1 = Hash+login2020

M2 = hash+login2020

получили H(M) для **MD5** :

H(M1) = c71b846d449901adb3b8308421ef203d

H(M2) = d228d48d152d929c8ff667dc1a2d663a

получили H(M) для **SHA1**:

H(M1) = a39ec24cf6a4ab6d15f51c39791211af5743963f

H(M2) = a31ec6b7710c0e7d4e0c23b261eab9a0ac37177c

Свойство 5: **Коллизионная устойчивость (стойкость)**

Зная **M**, трудно определить **M'** (**M≠ M'**), для которого **H(M)= H(M')** – **коллизия 1-го рода**

Трудно найти два случайных сообщения (**M** и **M'**), для которых **H(M)= H(M')** - **коллизия 2-го рода**

Для хеш-функций универсальным методом поиска коллизий является метод, основанный на известной статистической задаче – «<u>парадоксе дня рождения</u>».

Парадокс дней рождения - это кажущееся парадоксальным утверждение, что вероятность совпадения дней рождения (даты) **хотя бы у двух членов группы из 23** и более человек, превышает **0,5**.

Для **60** и более человек вероятность такого совпадения превышает **0,99 (**1,0 она достигает, только когда в группе не менее 367 чел.)

Такое утверждение может показаться неочевидным, так как вероятность совпадения дней рождения двух человек в любой день года (1/365 = 0,0027), помноженная на число человек в группе из 23, даёт лишь 23/365 = **0,063**.

Это рассуждение неверно, так как число возможных пар (253) значительно превышает число человек в группе.

Логического противоречия в этом нет, а парадокс заключается лишь в различиях между интуитивным восприятием и математическим расчётом.

Математическая оценка парадокса «дней рождения»

Вероятность **$P(n)$** того, что в группе из **n** человек дни рождения всех людей будут различными, если **$n > 365$**, будет нулевой: **$P(n) = 0$**.

При **$n \leq 365$** выберем наугад одного человека из группы и запомним его день рождения.

Затем выберем наугад второго человека, при этом вероятность того, что у него день рождения <u>не совпадёт </u>с днем рождения первого человека, равна **$P_{1\text{-}2} = 1\text{-}1/365$**.

Затем возьмём третьего человека, при этом вероятность того, что его день рождения <u>не совпадёт </u>с днями рождения первых двух, равна **$P_{1\text{-}3} = 1\text{-}2/365$**.

Рассуждая по аналогии, мы дойдём до последнего человека, для которого вероятность <u>несовпадения</u> его дня рождения со всеми предыдущими будет равна **$P_{1\text{-}n} = 1\text{-}(n\text{-}1)/365$**

Перемножая все эти вероятности, получаем вероятность того, что все дни рождения в группе будут различными:

$$\mathbf{P(n)} = 1\cdot\left(1-\frac{1}{365}\right)\cdot\left(1-\frac{2}{365}\right)\cdot\ldots\cdot\left(1-\frac{n-1}{365}\right) =$$

$$= \frac{365\cdot 364\cdot\ldots\cdot(365-n+1)}{365^n} = \frac{365!}{365^n(365-n)!}.$$

Тогда вероятность того, что хотя бы у двух человек из ***n*** дни рождения совпадут, равна

P(2,n) = 1 − P(n) .

Интересно, что при **n > 23** эта вероятность **P(2,n) > 0,5**

Используя разложение экспоненты (***exp x***) в ряд Тейлора, получим:

$$\mathbf{P(n)} \approx 1 \cdot e^{-\frac{1}{365}} \cdot e^{-\frac{2}{365}} \cdot \ldots \cdot e^{-\frac{n-1}{365}} = e^{-\frac{1+2+\ldots+(n-1)}{365}} = e^{-(n(n-1))/2\cdot 365}$$

Тогда

$$\mathbf{P(2,n)} \approx 1 - e^{-n^2/2\cdot 365}.$$

Поиск коллизии для хеш-функции
на основе парадокса «дней рождения»

Задача поиска коллизии: сколько сообщений надо просмотреть, чтобы найти сообщения с двумя одинаковыми хешами?

Вероятность P встретить одинаковые хеши для сообщений из двух разных наборов, содержащих n_1 и n_2 текстов, равна

$$P \approx 1 - e^{-\frac{n_1 n_2}{2^l}}$$

Если $n_1 = n_2 = 2^{(l/2)}$, то вероятность P обнаружения коллизии (успешной атаки) составляет $P = 1 - \exp(-1) = 0{,}63$

Коллизионная стойкость некоторых хеш-функций

MD 5: Длина хеша – 128 бит.
Коллизионная стойкость была взломана после ~2^21 хешей.
SHA 1: Длина хеша – 160 бит.
Коллизионная стойкость была взломана после ~2^61 хешей.

Кстати:
SHA 256: создает 256-битный хеш,
в настоящее время используется в Биткоине.

Keccak-256: создает 256-битный хеш,
в настоящее время используется Эфириуме.

Свойство 6: **Головоломка** (для майнеров):
для каждого известного ***h***, если ***k*** выбран из распределения с высокой мин-энтропией, невозможно найти вводные данные ***x*** такие, что ***H (k|x) = h***. (**|** - конкатенация)

Опр. Если некоторое число выбирается из диапазона от 1 до бесконечности, это - высокое распределение мин-энтропии.

Весь процесс **майнинга** работает на этом свойстве.

Можно ли использовать обычный метод сжатия без потерь , например ZIP, как криптографическую хеш--функцию?

Можно ли использовать функцию контрольной суммы (CRC) как криптографическую хеш-функцию ?

ЭЦП на основе симметр. криптографии

Генерация ЭЦП: $\mathbf{E = F\ (M,K)} \xrightarrow{\text{Осуществляется простое шифрование}} \mathbf{C = F\ (M,K)}$,

т.е. **E ≡ C**; физически ЭЦП нет

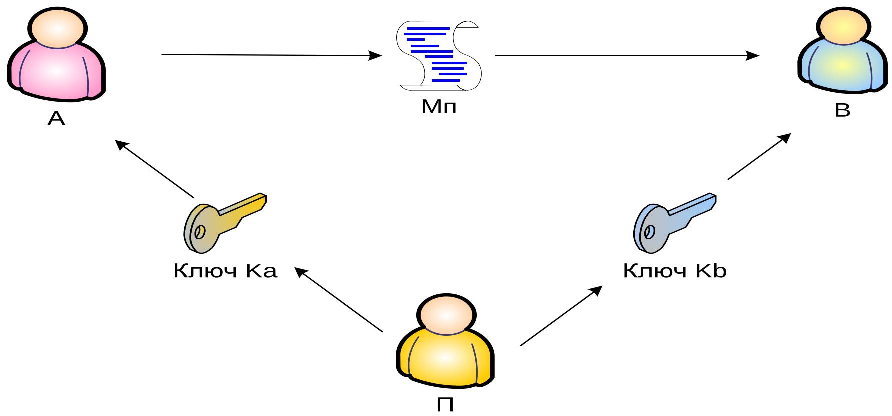
Проверка подлинности сообщения (верификация подписи):

$\mathbf{M = F\ (C,K)} \xrightarrow{\text{Осуществляется обычное расшифр.}} \mathbf{ЭЦП = F\ (C,K)}$

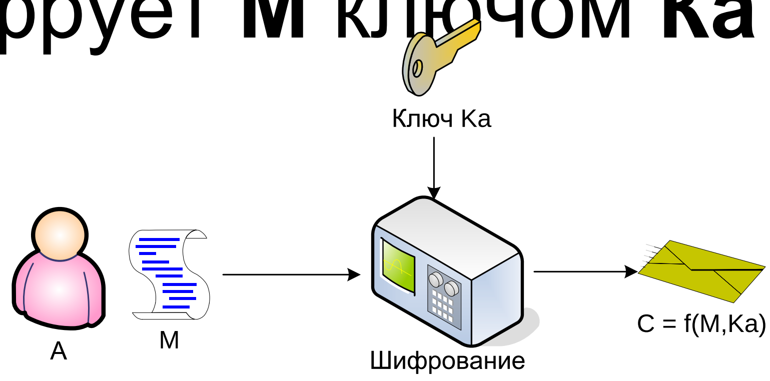
Если сообщ. расшифровано тайным ключом (**К**), известным только **А** и **В**, то это подтверждает аутентичность документа

ЭЦП на основе симметричных криптосистем и посредника

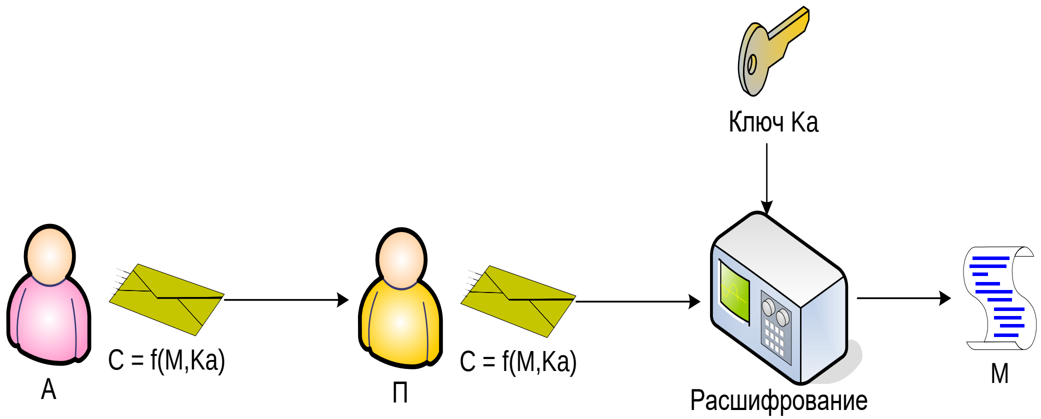
1. **П** (посредник, СЦ) вырабатывает для **А** и **В** разные ключи



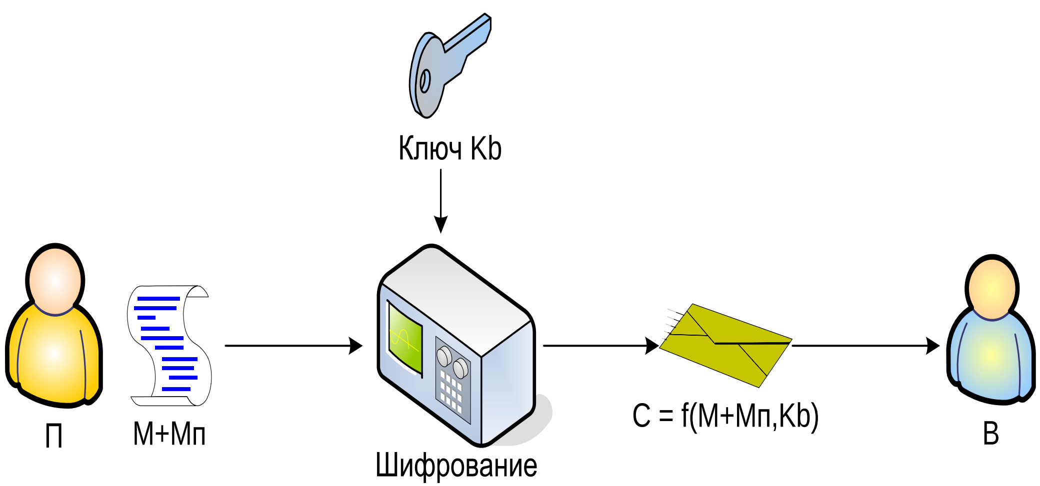
2. **А** шифрует **М** ключом **Ка** и отсылает его **П**



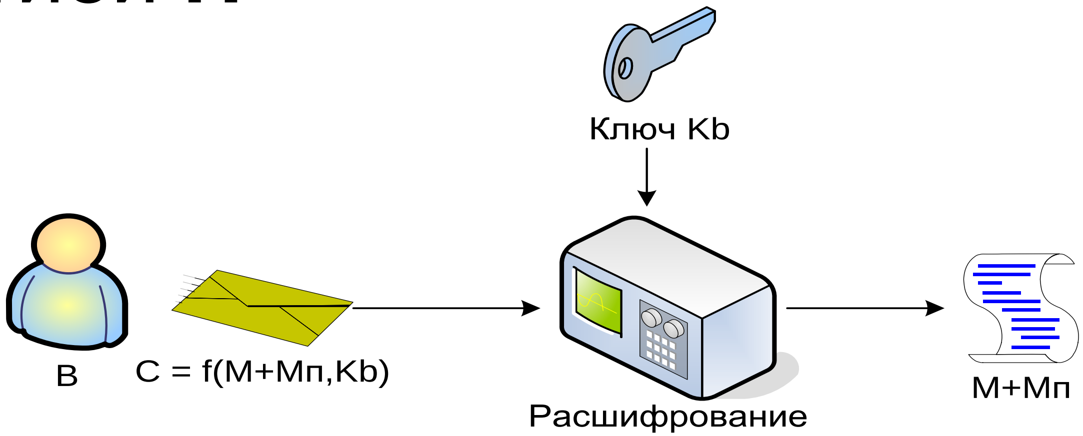
3. **П** расшир-т **С** ключом **Ка**, включает в него инф-ю, что оно получено от **А (М')**



4. **П** шифрует **М'** ключом **Кв** и отправляет его **В**



5. **В** расшифр-т ключом **Кв**, и читает **М’** с гарантией **П**



Таким образом:

1. Подпись достоверна (П – гарант)
2. Подпись неподдельна (только А и В знают кл: П – абсолютное доверие)
3. Подпись невозможно использовать повтор.
4. Подпис-й док-т нельзя изменить
5. Подпись нельзя отрицать

ЭЦП с открытым ключом

а) На основе алгоритма **RSA:**

- **A** шифрует документ **своим закрытым ключом:** $C = F(M; d_a, n_a)$
- **B** расш-т **C открытым ключом A:** $M = F(C; e_a, n_a)$

б) на основе алгоритма **Эль-Гамаля**

- Генерация ключа: **p** – простое число; выбираем два случайных числа: **g** и **x** (**g, x <p**); вычисляем:

$y = g^x \bmod p$; открытый ключ – **p, y, g**, закрытый ключ – **x**.

2. **A** шифрует документ **M своим закрытым ключом:**

выбирается случайное число **k** – взаимно простое с (**p-1**) генерируется подпись, состоящая из двух чисел: **a** и **b**:

$a=g^k$ mod p; **b** такое, что **M = (x*a+k*b) mod (p-1)**

3. **A** отправляет **B** подписанное сообщение: **M,a,b**

4. **B** получает **M,a,b** и сверяет подпись**:** подпись принадлежит **A,** если **$(b^a * a^b)$ mod p = g^M mod p**

Пример. **p=11, g=2, x=8,** вычисляем **$y=g^x$ mod p = = 2^8 mod 11 =3;** откр кл - **p=11, g=2, y=3;** закр кл **x=8**

Подпись сообщения **M=5:** выбираем **k=9** взаимно простое с **p-1=10;** вычисляем **$a=g^k$ mod p= 2^9mod11= =6; на основе M = (x*a+k*b) mod (p-1)** вычисляем **b:**

5=(8*6+9*b)mod 10, решение **b =3.**

Подписанное сообщение: M=5, a=6, b =3

Проверка подписи: **$(3^6 * 6^3)$ mod 11 = 2^5 mod 11= 10**

ЭЦП на основе асимметричных криптосистем и однонаправленных хэш-функций

Типы хэш-функций **- h(M)**

1.MD4 (message digest 4); Р.Ривест

Длина **h(M) – 128 бит,** длина **M – k** (произвольна)

Начальное преобразование **M**: дополняется дв символами так, чтобы **(k + r + 64) mod 512 =0;**

64 бита **-** двоичное представле-ние числа **k;** 100 … 00

получаем **m** блоков преобразованного **M** длины **512** бит;

каждый из этих блоков разбивается на **16** подбл дл **32** бита (16 * 32 = 512)

Весь алгоритм состоит из 3-х раундов.

В каждом из раундов выполняется 16 шагов (по числу подблоков). Каждый шаг вычисляет нелинейную функцию над 3-мя переменными из {a, b, c, d}.

Начальные значения переменных:

a= *0x*01234567, b=*0x*89abcdef, c=*0x*fedcba98, d=*0x*7654321

В каждом раунде – своя нелин функция:

1 р-д: **$F(x,y,z)=x\cdot y + x\cdot\overrightarrow{z}$**; на **j**-ом шаге:

$F(a,b,c,d,M_j,s) \longrightarrow a=b \oplus ((a \oplus F(b,c,d) \oplus M_j) \ll s$;

M_j - **j**-ый (0 <= **j** >= 15) подблок сообщения;

<<s – сдвиг на **s** разрядов влево

2 р-д: **$G(x,y,z)=((x\cdot y) + (x\cdot z) + (y\cdot z))$**

3 р-д : $H(x,y,z)=x \oplus y \oplus z$.

X – соответствует блоку сообщ-я на соответст-м шаге;

Y = 1 р-д: 0 (32 раза повторяющийся ноль), 2 р-д: **Y** представляется в 16-ой форме: 5A827999, где 5 → 0101 (4 разряда), A → 1010 и так далее; 3 р-д: **Y** = 6ED9EBEF; **Z** - очередность каждого из 16-ти подбл: 1р-д: 0,1,2…15 ; 2р-д: 0, 4,8,2,1,5,9,13,… ; 3р-д:0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15

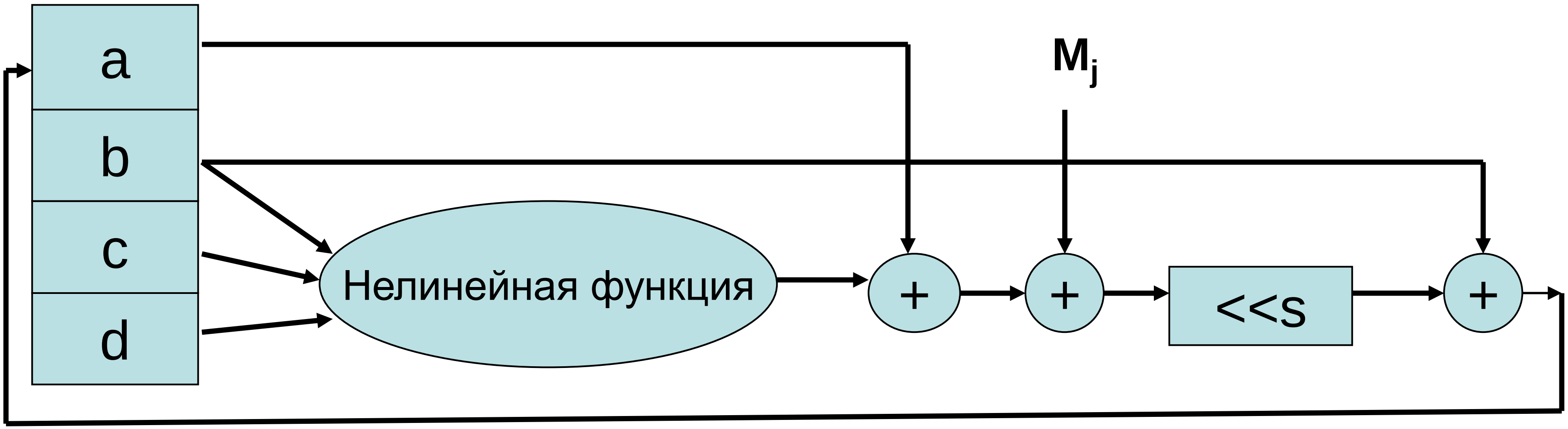


Рис.1. Пример одной операции алгоритма MD4

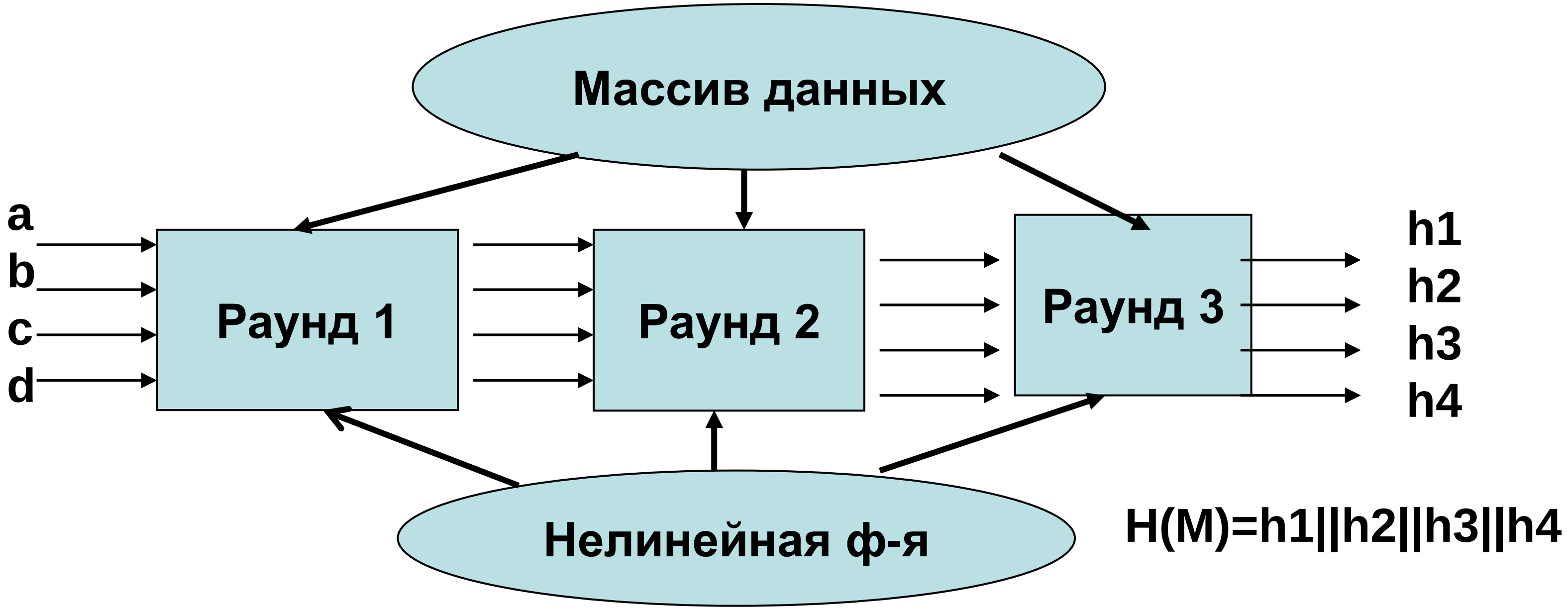


Рис.2.Общая схема алгоритма MD4

Хеш-функция

- При создании хэш-функции используют следующие операции:
- операция логического суммирования (\/ — “OR”, дизъюнкция, (+));
- операция логического умножения (/\ — “AND”, конъюнкция, (.));
- операция отрицания (“NOT”);
- операция логического сдвига на S разрядов;
- операция суммирования по модулю 2 ($\oplus$).

Хеш-функция MD5

В сравнении с **MD4:**

- добавлен 4-ый р-д с нелинейной функцией:

$I(x,y,z) = y + (x + \overrightarrow{z})$;

- на каждом раунде и шаге используется уникальная константа t:

$F(a, b, c, d, M_j, t_i, S)$;

$a = b \oplus (a \oplus F(b,c,d) \oplus M_j \oplus t_i \ll S_j)$

- функция **G** во втором раунде заменена на менее симметричную функцию:

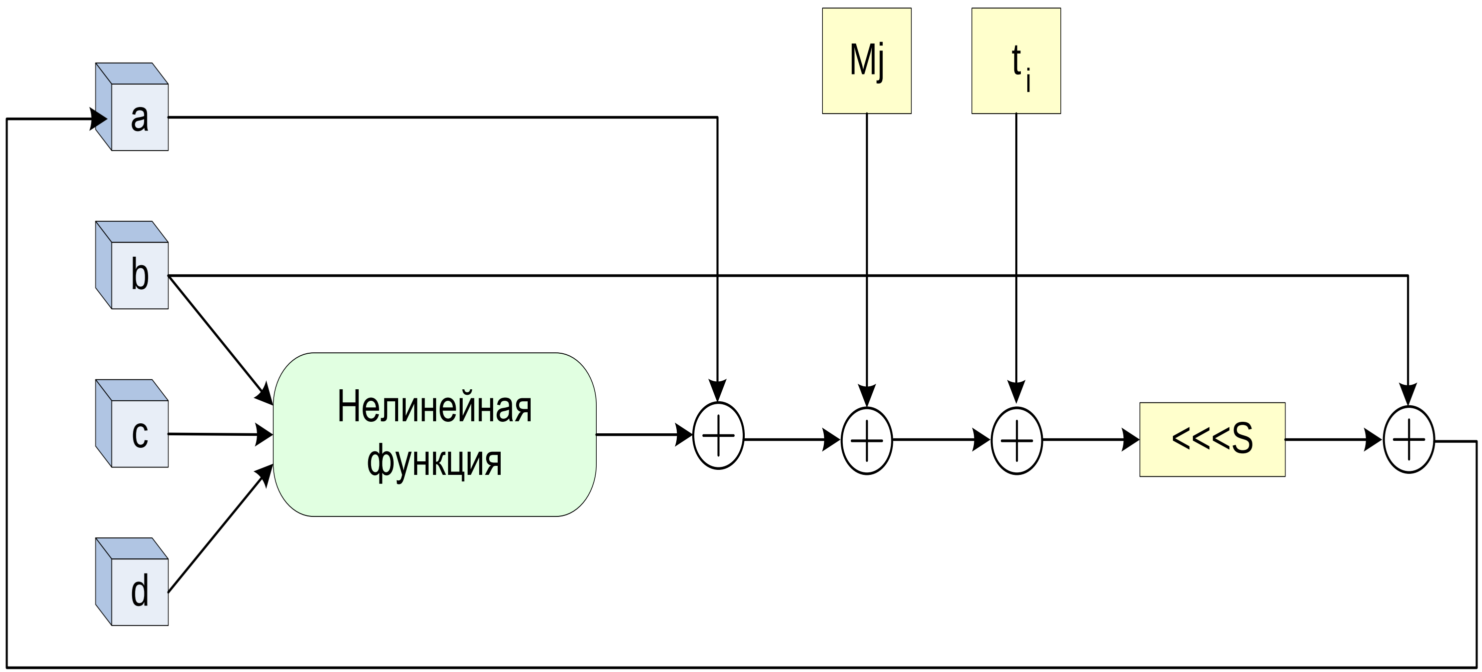
$G(x,y,z) = ((x \cdot z) + (y \cdot \overrightarrow{z}))$

Хеш-функция MD5

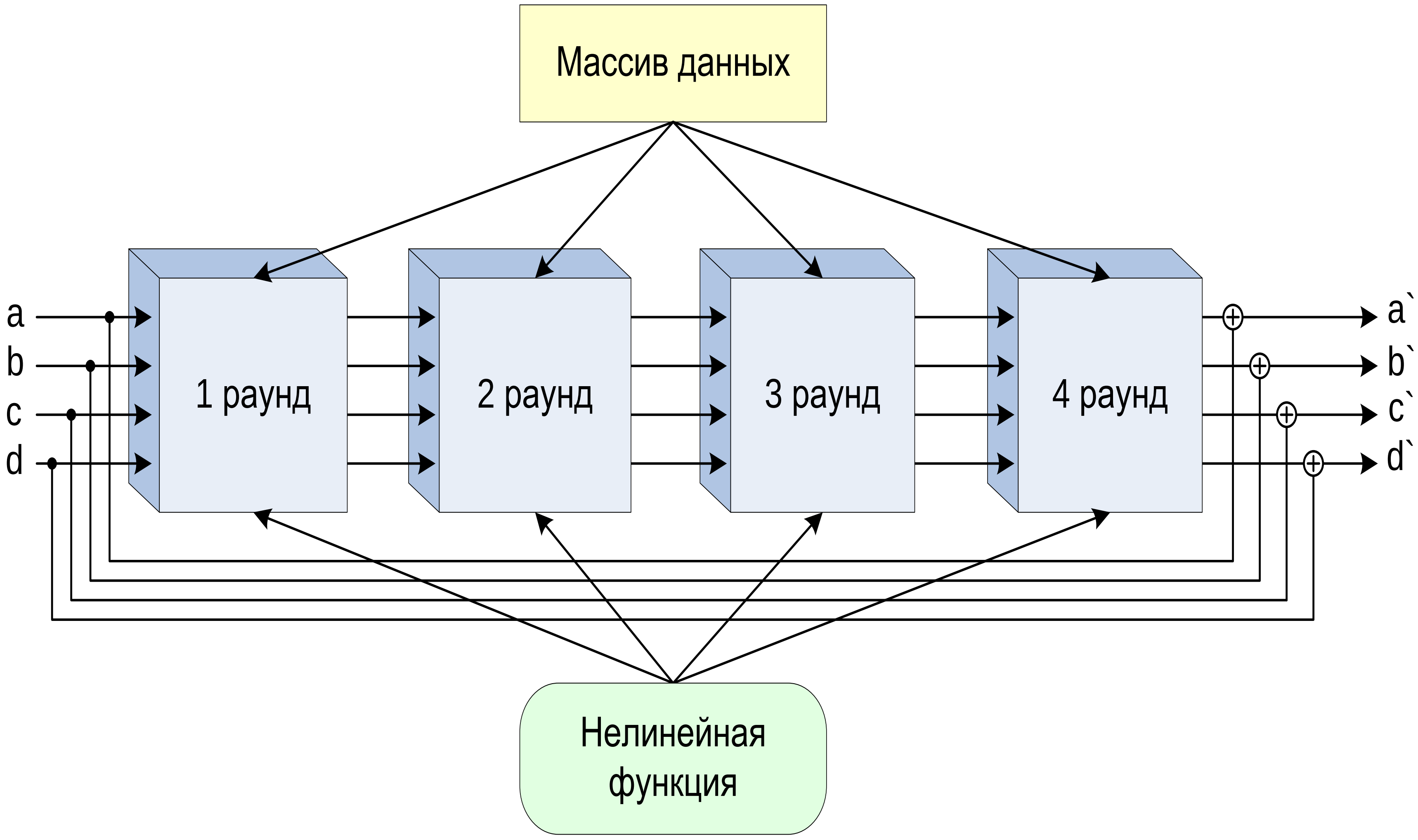
- результат каждого раунда добавляется к предыдущему;
- порядок следования подблоков изменяется во 2-м и в 3-м раундах;
- Пример: переменная **t** на 1-м шаге 1 раунда: $\mathbf{t_{11}}$ = d76aa178; на 2-м шаге первого раунда: $\mathbf{t_{12}}$ = e8c7в756;

Хеш-функция MD5

Одна из операций на шаге выглядит следующим образом:



Хеш-функция MD5



$H(M) = h_1 \| h_2 \| h_3 \| h_4;$

Применение MD5

•Поиск дублирующихся файлов на компьютере или в интернете (сравнивая MD5 файлов)

Пример. Графическая программа dupliFinder под Windows и Linux;   это утилита для поиска дубликатов фотографий в папках на диске компьютера путем сравнения их контрольных сумм MD5

• Проверка целостности скачанных файлов — некоторые программы идут вместе со значением хеша.

Пример. Диски для инсталляции.

•Хеширование паролей.

Пример. ("md5") = 1bc29b36f623ba82aaf6724fd3b16718

("") = d41d8cd98f00b204e9800998ecf8427e (нулевая строка)

Алгоритм хеширования SHA1

Secure Hash Algorithm 1

Длина входного сообщения - максимум **2^{64} -1** бит, (2 эксабайта – **$2*10^{18}$ байт**)

Алгоритм генерирует **160-битное** хеш-значение

Сходства SHA1 и MD5 :

- Четыре этапа.
- Каждое действие прибавляется к ранее полученному результату.
- Размер блока обработки равный 512 бит.
- Оба алгоритма выполняют сложение по модулю 2^{32}: рассчитаны на 32-битную архитектуру.

Различия SHA1 и MD5 (основные):

- В MD5 длина дайджеста составляет **128** б, в SHA1 — **160** б.
- В MD5 четыре различных элементарных логических функции, в SHA1 — три.
- SHA1 содержит больше раундов (80 вместо 64) и выполняется на 160-битном буфере по сравнению со 128-битным буфером MD5. SHA-1 приблизительно на 25 % медленнее, чем MD5
- В SHA1 добавлена пятая переменная.
- SHA1 использует циклический код исправления ошибок.

Применение SHA1

- **ЭЦП**
- **Системы управления версиями** (*Version Control System, VCS* — ПО для облегчения работы с изменяющейся информацией. СУВ позволяет хранить несколько версий одного и того же документа (коды программ), возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение.
- Для построения кодов аутентификации (процедура проверки подлинности: путем сравнения введенного пароля с паролем в БД пользователей;

SHA1 используется в следующих приложениях:

- S/MIME — дайджесты сообщений.
- SSL — дайджесты сообщений.
- IPSec — для алгоритма проверки целостности в соединении «точка-точка».
- SSH — для проверки целостности переданных данных.
- PGP — для создания электронной цифровой подписи.
- Git — для идентификации каждого объекта по SHA1-хешу от хранимой в объекте информации.
- BitTorrent — для проверки целостности загружаемых данных.

Для обнаружения коллизий нужно выполнить 2^{52} операций

"sha" = d8f45903 20e1343a 915b6394 170650a8 f35d6926

"Sha" = ba79baeb 9f10896a 46ae7471 5271b7f5 86e74640

Криптоанализ хеш-функций SHA1

Направлен на исследование уязвимости к различного вида атакам.

Основные атаки:

- нахождение коллизий —двум различным исходным сообщениям соответствует одно и то же хеш-значение.

При решении методом «грубой силы» требует в среднем $\mathbf{2^{160/2} = 2^{80}}$ операций

- нахождение прообраза — исходного сообщения — по его хешу.

При решении методом «грубой силы» требует $\mathbf{2^{160}}$ операций.

Сравнительная таблица:

Алгоритм вычисления	Длина (бит)	Скорость хеширования (Кбит/с)
MD4	128	236
MD5	128	174
SHA1	160	75
ГОСТ	256	11

ЭЦП **DSA**

1991г. – алгоритм ЭЦП **DSA** (*Digital Signature Algorithm*)

Предложен Национальным Институтом Стандартов и Технологий (США) (U.S. Patent 5231668),

Основан на сложности вычисления логарифмов в конечных полях.

Секретное создание хеш-значения и возможность его публичной проверки означает, что только один субъект может создать хеш-значение сообщения, но любой может проверить её корректность.

Для подписания сообщений необходима пара ключей — открытый и закрытый: *закрытый ключ известен тому, кто подписывает сообщения, а открытый — проверяющему подлинность сообщения*.

Общедоступными являются параметры самого алгоритма.

Необходимо, чтобы **подписываемое сообщение являлось числом. Хеш-функция должна преобразовать любое сообщение в число**

Используемые обозначения: **p** – простое число длиной **L** бит: **L** mod 64=0;

Размерность **p** задаёт криптостойкость системы. Ранее рекомендовалась длина **L =** 1024 бита. В последнее время рекомендуется **L =2048 (3072)** бита.

q – простой множитель **(p-1)**; размерность **q** (в битах - **N**) совпадает с размерностью в битах значений хэш-функции **h(x) –** для **SHA1- N =160 бит (сейчас – SHA2)**

Генерация ключа:

1. Выбирается число **v** такое, что:

1<= v <= p-1 ;

q≠ 1;

2. Вычисляется $\mathbf{g = v^{(p-1)/q} \bmod p}$**;**

p, q ,**v** являются открытыми и могут применяться группой пользоват-й;

Закрытый ключ – **x** **(x<q)**, открытый ключ - **y** (**$y = g^x \bmod p$**)**;**

x – любое не менее чем **160**-битовое число**, y** – **p**-битовое число

Генерация ЭЦП сообщ-я **M (*A* *для* *B*)**.

1. **A** выбир-т случ число **k** **(k<q);**
2. Вычисление **h(M) –** хэш-функция (SHA) сообщ-я **M**;
3. подпись – числа **r** и **s**:

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

$s=(k^{-1} * (h(M) + x*r)) \bmod q$;

Выбор другого k, если оказалось, что r=0 или s=0

4. **A** персыл-т **B**: **M, r**, **s**

Проверка подписи:

1. B вычис-т: $w=s^{-1} \bmod q$, $u_1=(h(M)*w) \bmod q$,

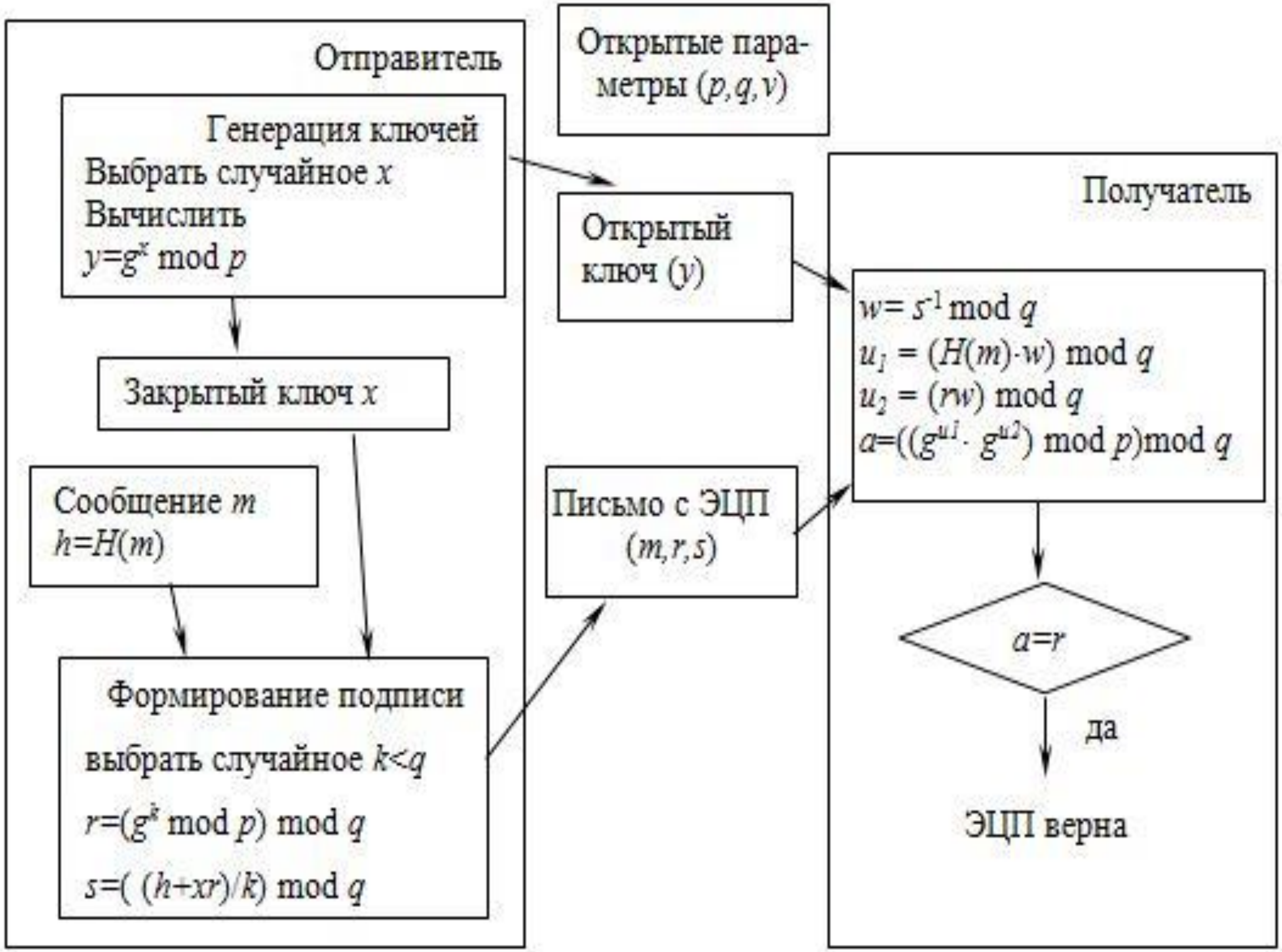
$u_2 = (r*w) \bmod q$, $a = ((g^{u1} * g^{u2}) \bmod p) \bmod q$;

Подпись достоверна, если **a = r**

Фактически подписывается не само сообщение, а его хеш

Рос.стандарт (ГОСТ Р 34.10-94) основан на DSA;

в наст время – на эллиптических кривых (2001 г.)


Отправитель
Генерация ключей
Выбрать случайное x
Вычислить
y=g^x mod p
Закрытый ключ x
Сообщение m
h=H(m)
Формирование подписи
выбрать случайное k<q
r=(g^k mod p) mod q
s=((h+xr)/k) mod q
Открытые пара-метры (p,q,v)
Открытый ключ (y)
Письмо с ЭЦП (m,r,s)
Получатель
w= s^-1 mod q
u1 = (H(m)·w) mod q
u2 = (rw) mod q
a=((g^u1 · g^u2) mod p)mod q
a=r
да
ЭЦП верна

ЭЦП на основе хэш-ф и RSA

A: e_A, d_A, n_A; B: e_B, d_B, n_B; $A \xrightarrow{M} B$

1. <u>Шифруется только</u> **h(M):**

<u>**A**</u>: $M \xrightarrow{H} h(M)$; $h(M) \xrightarrow{d_A, n_A} C(h(M))$; **A** $\longrightarrow$ **B:** $M||C(h(M))$;

<u>**B**</u>: *получает* $M||C(h(M))$; $M \xrightarrow{H} h'(M)$; $C(h(M)) \xrightarrow{e_A,\ n_A} h''(M)$;

Проверка подписи: если , **$h'(M) \equiv h''(M)$,** то подпись принадлежит **A** (**$h'(M) \equiv h''(M) \equiv h(M)$**), и док-т **M** не изменялся

!!! Используются ключи отправителя (О)

2. <u>Шифруются все данные</u>

<u>**A**</u>: $M \xrightarrow{H} h(M)$; $h(M) \xrightarrow{d_A, n_A} C(h(M))$; $M||C(h(M)) \longrightarrow M'$; $M' \xrightarrow{e_B, n_B} C(M')$; **A** $\longrightarrow$ **B:** $C(M')$;

<u>**B**</u>: *получает* $C(M')$; $C(M') \xrightarrow{d_B, n_B} M'$; $M' = M||C(h(M))$; $M \xrightarrow{H} h'(M)$; $C(h(M)) \xrightarrow{e_A, n_A} h(M)$;

Проверка подписи: если **$h'(M) \equiv h(M)$,** то подпись принадлежит **A,** и док-т **M** не изменялся; !!! Используются ключи П и О

Алгоритм ЭЦП Шнорра

Основа стандарта ЭЦП РБ (Claus Schnorr)

являeтся модификацией схем Эль-Гамаля (1985) и Фиата-Шамира (1986)

p – простое число длиной примерно **512** или **1024** бит,

q – примерно **140**-битный простой множитель **(p-1)**; т.е **$p - 1 \equiv 0 \pmod q$**

выбирается любое число **z** (**z≠1)** такое, что **$z^q = 1 \bmod p$**;

p, z и **q** являются откр-ми и могут прим-ся группой пользоват-й;

Выбирается число **s<q**; вычисл-ся **$v = z^{-s} \bmod p$; или $vz^s = 1 \bmod p$**

s – тайн ключ, **v** – откр кл

Генерация ЭЦП сообщ-я **M** ***(A для B)***. **A** выбир-т случ число **k** **(k<q)** и вычисляет **$x = z^k \bmod p$**;

подпись – **числа e и y**: **$e = h(M||x)$; $y = (k + s*e) \bmod q$**

A персыл-т **B**: **M, e, y**

Проверка подписи: **B** вычис-т: **$x' = z^y * v^e \bmod p$**; затем вычисляет **$e' = h'(M||x')$**

- **Подпись достоверна, если $e = e'$**

Пример

- Генерация ключей:

$p = \mathbf{11}$, $q = \mathbf{5}$; причем $p = 2q + 1$, т.е. $\mathbf{p - 1 \equiv 0 \ (\mathrm{mod}\ q)}$

Выбирается $z = 3$, д.б.: $\mathbf{z \neq 1}$, и $\mathbf{z^q = 1 \ \mathrm{mod}\ p : 3^5 = 1 \ \mathrm{mod}\ 11}$

Тайный ключ $\mathbf{s} = 3$, тогда $\mathbf{v = z^{-s} \ \mathrm{mod}\ p = 3^{-3} \ \mathrm{mod}\ 11}$

Или $\mathbf{vz^s = 1 \mathrm{mod}\ p , v = 9}$

Открытый ключ: $p = \mathbf{11}$, $q = \mathbf{5}$, $z = \mathbf{3}$, $\mathbf{v = 9}$

Тайный ключ: $\mathbf{s} = \mathbf{3}$

- Генерация подписи: **M = 1000**

Выбирыется случайное $\mathbf{k = 2}$ $\mathbf{(k<q)}$

вычисляется $\mathbf{x = z^k \ \mathrm{mod}\ p = 3^2 \ \mathrm{mod}\ 11 = 9}$

конкатенация **M||x : 10009;** предположим $\mathbf{e = h(M||x) = 15}$

вычисляется $\mathbf{y = (k + s*e) \ \mathrm{mod}\ q = (2+3*15) \ \mathrm{mod}\ 5 = 2}$

Получателю высылается: 1000, 15, 2

Безопасность – на трудности вычисления ДИСКР Логар

Использование сертификатов

- **Обеспечивает одновременно аутентичность и целостность** при распределении открытых ключей.
- Заключается в использовании сертификатов.
- Имеется центральный орган (ЦО, сертификационный центр), как и в случае распределения секретных ключей.
- Каждый пользователь может осуществлять безопасное взаимодействие с ЦО. Для этого требуется, чтобы у каждого пользователя был открытый ключ ЦО – $E_{ЦО}$.
- Каждый пользователь **A** может зарегистрировать в ЦО свой открытый ключ E_A. Поскольку $E_{ЦО}$ является открытым, это можно сделать по почте, по открытому каналу электросвязи и т.п.
- При регистрации в ЦО **A** будет следовать определенной аутентификационной процедуре.

A получает сертификат, подписанный ЦО и содержащий E_A, ЦО формирует сообщение **M**, содержащее E_A, идентификационную информацию для **A** : (I_A), период действия сертификата и т.п.

- ЦО вычисляет $CERT_A = D_{ЦО}(M)$, который и становится сертификатом **A**.

$CERT_A$ делается общедоступным документом, который содержит E_A и аутентифицирует его, поскольку сертификат подписан ЦО.

Алгоритмы обмена ключами

- Алгоритм Диффи-Хеллмана: генерация секретного ключа (который нельзя исп-ть при зашифр/расшифр)

A и **B** выбирают совместно большие простые ч: **n** и **g**

Прот-л обмена:

1.**A** выб-т случ число **x**, выч-т $\mathbf{X = g^x \bmod n}$ и отсылает **X** → **B**

2.**B** выб-т случ число **y**, выч-т $\mathbf{У = g^y \bmod n}$ и отсылает **У** → **A**

3. **A** выч-т $\mathbf{k1 = Y^x \bmod n}$

4. **B** выч-т $\mathbf{k2 = X^y \bmod n}$

Получается: $\mathbf{k1 = k2 = g^{xy} \bmod n = k}$ **–** *секретный ключ*

Зная n, g, X, Y, невозможно выч-ть **k – проблема дискр логар**

- Алгоритм на основе нейросетевых технологий

Совместное исп-ние методов преобр-я инф-и

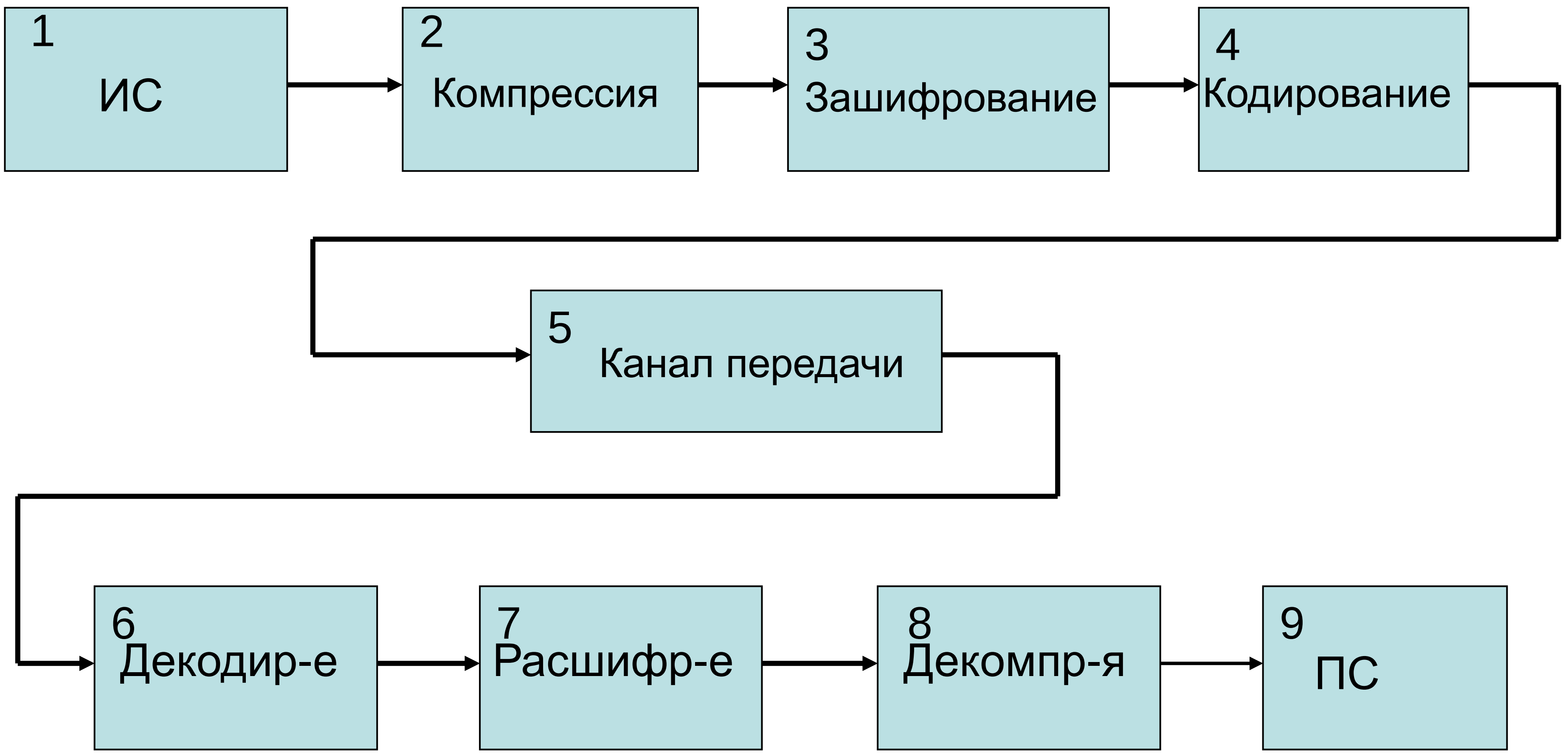


Рис.1 Информационная система передачи данных