

Лекция №4

ХЕШ-ФУНКЦИИ

Дисциплина: Криптографическая защита информации

Преподаватель: Миронов Константин Валерьевич

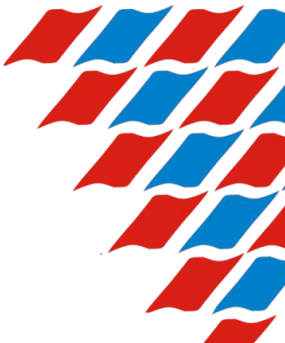
Поток: БПС-3

Учебный год: 2020/21



Содержание лекции

- **Функции сжатия**
 - **Общие сведения**
 - **Семейство MD/SHA**
- Криптографическая губка
- Имитовставки



Общие сведения

Криптографическая хеш-функция

- [Hash], дайджест или профиль сообщения [message digest], цифровой отпечаток [digital fingerprint]
- Односторонняя функция $y=f(x)$, где x – двоичный код произвольной длины, y – двоичный код строго заданной длины
- Должен обеспечиваться лавинный эффект - при незначительном изменении x , y меняется радикально

```
SHA3-256("The quick brown fox jumps over the lazy dog")=  
69070dda01975c8c120c3aada1b282394e7f032fa9cf32f4cb2259a0897dfc04  
SHA3-256("The quick brown fox jumps over the lazy dog.")=  
a80f839cd4f83f6c3dafc87feae470045e4eb0d366397d5c6ce34ba1739f734d
```



Общие сведения

Криптографическая хеш-функция

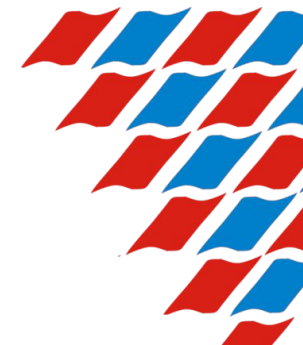
- **«Сильная» хэш-функция** - вычислительно невозможно подобрать два аргумента, дающих одно и то же значение
- **«Слабая» хэш-функция** – для любого заданного аргумента вычислительно невозможно подобрать другой, дающий то же значение
- **Хэш-функция с ключом (имитовставка, код аутентификации сообщения, Message Authentication Code, MAC)** – чтобы вычислить значение, нужно знать не только аргумент, но и дополнительную информацию – ключ



Общие сведения

Криптографическая хеш-функция

- Под взломом хеш-функции подразумевается нахождение коллизий
- Согласно парадоксу о днях рождения, если длина идеальной хеш-функции n и количество заданных пар аргумент-значение равно $2^{(n/2)}$, то вероятность наличия двух одинаковых значений равно 50%
- Безопасная длина хеш-функции начинается примерно от 160 бит
- **«Случайный оракул»** - идеальная хеш-функция
 - Таблица, содержащая пары аргумент-значения
 - При получении аргумента, система ищет его в таблице
 - Если он есть – выдается соответствующее значение
 - Если нет – оно генерируется случайным образом
 - Таблица должна быть единой для всех пользователей
 - Ни один из пользователей не имеет доступа к таблице



Общие сведения

Функция сжатия

Функция Меркла-Дамгарда

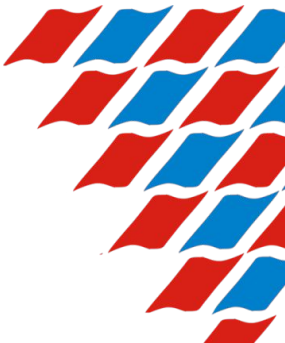
- Сообщение разбивается на блоки фиксированной длины $M_1 \dots M_n$
- Последний блок при необходимости дополняется
- Над первым блоком выполняется операция:

$$H_1 = f(M_1, IV)$$

- Над последующими:

$$H_i = f(M_i, H_{i-1})$$

- Значением хеш-функции является H_n
- В отличие от шифрования, при хешировании синхропосылка фиксированная

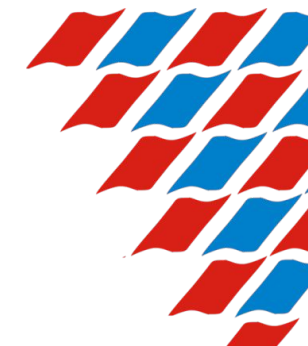


Общие сведения

Функция сжатия

Особенности:

- Итеративное вычисление легче реализовать
- При обработке большого потока данных хеш-функцию можно вычислять на лету
- Потенциальная уязвимость:
$$\text{hash}(\text{IV}, \text{text_1} || \text{text_2}) = \text{hash}(\text{hash}(\text{text_1}), \text{text_2})$$
- Защита: вместо $h = \text{hash}(\text{OT})$ вычисляется $h = \text{hash}(\text{hash}(\text{OT}))$



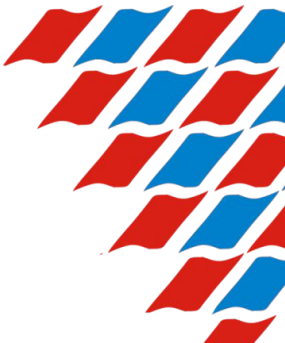
Линейка MD/SHA

[Message Digest] – хеш-функции от Рональда Райвеста

- **MD2** (1989) – взломана и является устаревшей
- **MD4** (1990) – обнаружены уязвимости
- **MD5** (1991) – исправленная версия MD4, теоретически взломана, но все еще применяется
- **MD6** (2008) – на самостоятельное изучение

[Secure Hash Algorithm] – национальные стандарты в США

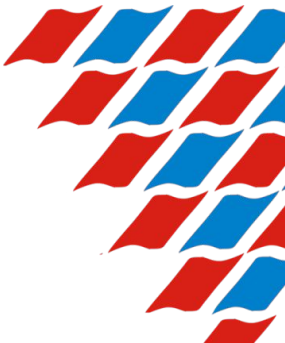
- **SHA-1** (1995) – разработана АНБ на основе MD4, взломана в 2014-2017
- **SHA-2** (2002) – расширение SHA-1 на другие длины
- **SHA-3/Кескак** (2012/2007) – выбрана на основе конкурса SHA-3



Линейка MD/SHA

MD4, MD5, SHA-1, SHA-2 основаны на одинаковых принципах и различаются особенностями и длиной

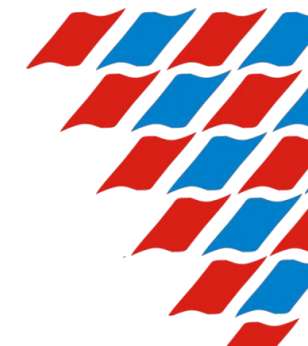
	Длина хеш-функции	Длина блока	Взломан а
MD4	128 бит	512 бит	1991-96
MD5	128 бит	512 бит	2004-06
SHA-1	160 бит	512 бит	2014-17
SHA2-256	256 бит	512 бит	???
SHA2-512	512 бит	1024 бит	???



SHA256

- SHA-2 - по сути семейство хэш-функций, к которому относится и SHA-1
- Наиболее популярна SHA-256, т. е. 256-битная
- Дополнение последнего блока
 - Если до конца блока осталось <66 бит, создается еще один блок
 - К сообщению дописывается единица
 - В последние 64 бита последнего блока записывается число бит в исходном сообщении
 - Промежуток заполняется нулями
- 512-битный блок разбивается на 16 32-битных слов
- Они расширяются до 80 32-битных слов
- Каждое следующее слово определяется по формуле:

$$W(i) = \text{XOR}(W(i-3), W(i-8), W(i-14), W(i-16)) \lll 1$$



SHA256

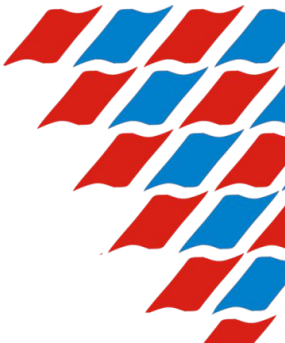
- 256-битный IV разбивается на 8 32-битных слов $\{a,b,c,d,e,f,g,h\}$
- Каждое слово IV заполняется первыми 32 битами от дробной части квадратного корня одного из первых 8 простых чисел
- После обработки всех блоков из этих слов составляется хеш-функция
- Вычисление функции сжатия от одного блока состоит из 64 итераций
- На каждой i -й итерации выполняются следующие преобразования:
$$a(i) = e(i) \oplus d(i-1) \oplus MA(a(i-1), b(i-1), c(i-1));$$
 // MA() – мажоритарная функция, i – номер итерации
$$b(i) = a(i-1); c(i) = b(i-1); d(i) = c(i-1);$$
$$e(i) = h(i-1) \oplus W(t) \oplus K(t) \oplus (e(i-1) \& f(i-1) \text{ OR } !e(i-1) \& g(i-1)) \oplus d(i-1) \oplus \text{XOR}(e(i-1) \ggg 6, e(i-1) \ggg 11, e(i-1) \ggg 25);$$
$$f(i) = e(i-1); h(i) = f(i-1);$$

*мажоритарная функция – логическая функция,
принимаящая то же значение, что и большинство ее
аргументов



Содержание лекции

- Функции сжатия
- **Криптографическая губка**
 - **Общие сведения**
 - **SHA-3 (Кессак)**
- Имитовставки



Общие сведения

Криптографическая губка

Задана функция $f(A)$ – псевдослучайное преобразование строки A состоящей из двух фрагментов R и C , длиной $a=r+c$

r – битовая скорость; c – битовая мощность

шаг 1. Хешируемый текст дополняется до целого числа блоков длиной r . Количество блоков обозначим как n , сами блоки как $M_1 \dots M_n$

шаг 2. Массив A заполняется нулями.

шаг 3. Заполнение губки [absorbing]. Для $i=1:n$

шаг 3.1 $R = \text{xor}(R, M_i)$

шаг 3.2 $A = f(A)$

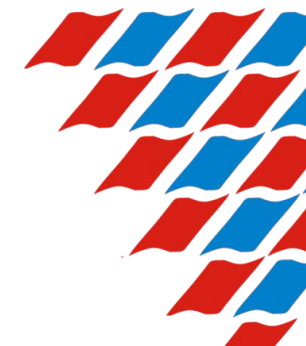
шаг 4. Выжимание губки [squeezing]. Для $i=1:m$

шаг 4.1 $A = f(A)$

шаг 4.2 $H(i) = R$

Значением хеш-функции является
 $H(1) \dots H(m)$

Условие криптостойкости: мощность должна быть хотя бы вдвое больше, чем длина хеш-функции



SHA-3 (Кессак)

Конкурс SHA-3 (2007-2012) завершился победой алгоритма Кессак

- Версии функции часто обозначают Кессак-256, Кессак-512 и т.д., потому что "SHA" ассоциируется с SHA-2
- A - массив 5×5 64-битных слов; изначально заполняется нулями
- r и s определяются требуемой длиной хеша и необходимым условием криптостойкости
- Для небольших длин хеш-функций отжатие можно не выполнять, достаточно взять нужное число бит из начала A после заполнения
- Применяемые операции считаются стойкими к анализу побочных излучений шифратора
- Порядок дополнения последнего блока:

если дополнить нужно 1 байт

то это байт 0x81

иначе

шаг 1. дописывается байт, состоящий из единиц.

шаг 2. в конец блока дописывается байт 0x80.

шаг 3. пространство между ними заполняется нулями



SHA-3 (Кескак)

Преобразование $f(A)$ – 24 раунда

На каждом выполняются операции:

\\ $C(5 \times 1)$, $D(5 \times 1)$ и $B(5 \times 5)$ – массивы переменных

\\ $r(5 \times 5)$ и $RC(24 \times 1)$ – массивы констант

\\ k – номер раунда

шаг 1. Для $i=0 \dots 4$ выполнить $C[i] = \text{xor}(A[i,0], \dots, A[i,4])$

шаг 2. Для $i=0 \dots 4$ выполнить $D[i] = C[i-1] \text{ xor } (C[i+1] \gg \gg 1)$

шаг 3. Для $i=0 \dots 4$ для $j=0 \dots 4$ выполнить $A[i,j] = A[i,j] \text{ xor } D[i]$

шаг 4. Для $i=0 \dots 4$ для $j=0 \dots 4$ выполнить $B[j,2i+3j] = A[i,j] \gg \gg r[i,j]$

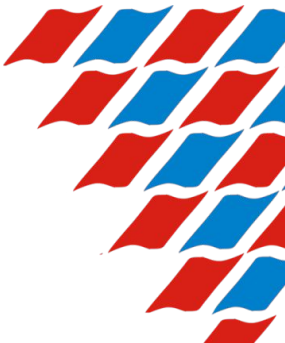
шаг 5. Для $i=0 \dots 4$ для $j=0 \dots 4$ выполнить $A[i,j] = B[i,j] \text{ xor } (\sim B[i+1,j] \text{ and } B[i+2,j])$

шаг 6. $A[0,0] = A[0,0] \text{ xor } RC[k-1]$

	$x = 3$	$x = 4$	$x = 0$	$x = 1$	$x = 2$
$y = 2$	25	39	3	10	43
$y = 1$	55	20	36	44	6
$y = 0$	28	27	0	1	62
$y = 4$	56	14	18	2	61
$y = 3$	21	8	41	45	15

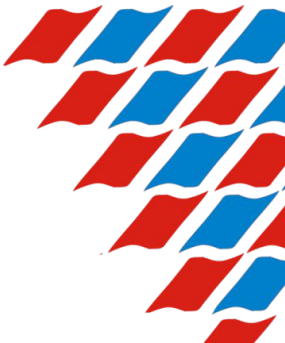
Значения
 $r[x][y]$

RC[0]	0x0000000000000001	RC[12]	0x000000008000808B
RC[1]	0x0000000000008082	RC[13]	0x800000000000008B
RC[2]	0x800000000000808A	RC[14]	0x8000000000008089
RC[3]	0x8000000080008000	RC[15]	0x8000000000008003
RC[4]	0x000000000000808B	RC[16]	0x8000000000008002
RC[5]	0x0000000080000001	RC[17]	0x8000000000000080
RC[6]	0x8000000080008081	RC[18]	0x000000000000800A
RC[7]	0x8000000000008009	RC[19]	0x800000008000000A
RC[8]	0x000000000000008A	RC[20]	0x8000000080008081
RC[9]	0x0000000000000088	RC[21]	0x8000000000008080
RC[10]	0x0000000080008009	RC[22]	0x0000000080000001
RC[11]	0x000000008000000A	RC[23]	0x8000000080008008



Содержание лекции

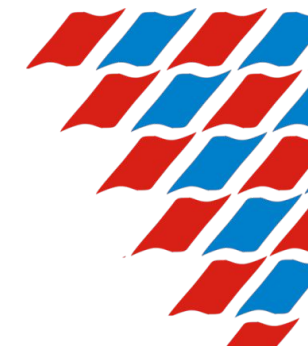
- Функции сжатия
- Криптографическая губка
- **Имитовставки**
 - **Имитовставки на основе блочных симметричных шифров**
 - **Имитовставка на основе бесключевых хеш-функций**
 - **Особенности применения**



Общие сведения

Имитовставки

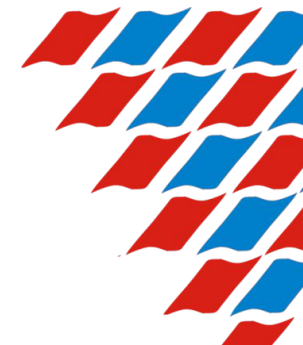
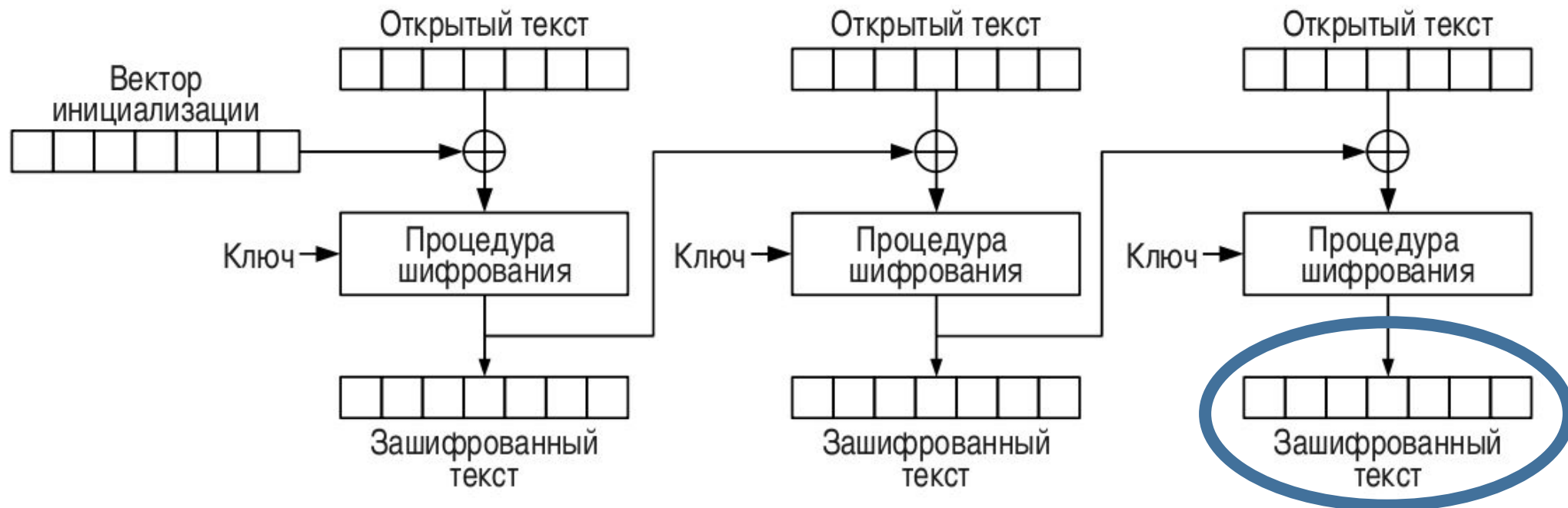
- Имитовставка – хеш-функция, для вычисления которой необходимо знание секретного элемента – ключа
- Требование устойчивости к коллизиям на имитовставку не распространяется
- Имитовставка позволяет провести **аутентификацию данных**, т.е. убедиться, что:
 - Сообщение отправлено конкретным пользователем
 - Сообщение отправлено именно в том виде, в котором пришло
- Простой и безопасный способ построения имитовставки – вычислить хеш-функцию, а потом зашифровать ее на ключе
 - С точки зрения реализации это сложный способ – нужно реализовать и операцию хеширования, и операцию шифрования
 - Алгоритмы реализуют как правило что-то одно



Имитовставки на основе шифров

СВС-МАС: Имитовставкой является последний блок ШТ при шифровании в режиме СВС:

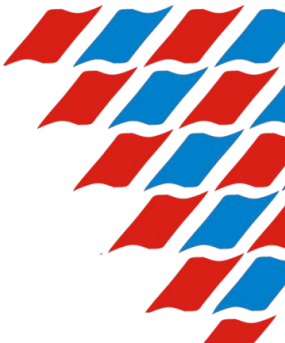
$$\begin{aligned} \text{ШТ}_1 &= \text{Ш}(\text{ОТ}_1 \oplus IV, K) \\ \text{ШТ}_i &= \text{Ш}(\text{ОТ}_i \oplus \text{ШТ}_{i-1}, K) \\ \text{МАС} &= \text{ШТ}_n \end{aligned}$$



Имитовставки на основе шифров

CBC-MAC:

- Если размер блока больше, чем нужная длина MAC, можно использовать часть блока
- Единая операция для шифрования и аутентификации позволяет сэкономить программный код или аппаратные ресурсы
- Боб может генерировать сообщения с таким же MAC, как у присланного Алисой, подставляя нужные значения при расшифровке от последнего блока к первому
- Если есть 2 сообщения $A[1]..A[n]$ и $B[1]..B[m]$ и $CBC-MAC(A[1]..A[n])=T$ то $CBC-MAC(\text{xor}(T,B[1]) || B[2]..B[m]) = CBC-MAC(A[1]..A[n] || B[1]B[m])$
 - В первый блок сообщения записывается его длина и/или порядковый номер
- Если Ева может подменить IV, то она может изменить первый блок без изменения CBC-MAC
 - Значение IV в стандарте CBC-MAC должно быть =0
 - В первый блок сообщения записывается его длина



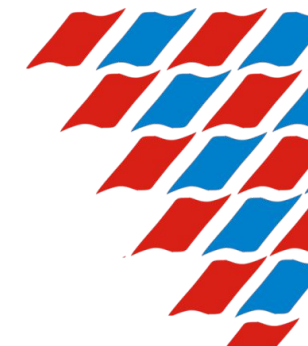
Имитовставки на основе шифров

C-MAC - Модификация CBC-MAC

- Последний блок шифруется другим ключом, генерируемым на основе данных и исходного ключа в зависимости от конкретного протокола
- В качестве ХФ берется не весь последний блок шифротекста, а только некоторое количество левых бит

Помимо режима CBC можно аналогичным образом использовать режим CFB

- Пример – имитовставка ГОСТ28147-89



Имитовставки на основе хеш-функций

H-MAC:

$$\text{HMAC}(M) = \text{Hash}(K \text{ xor } O \parallel \text{Hash}(K \text{ xor } I \parallel M))$$

- O , I и K – имеют ту же длину, что и хеш-функция
- O и I – константы, в стандарте каждый их байт = $0x5c$ и = $0x36$ соответственно
- Если исходный ключ больше, чем K , то K вычисляется как хэш-функция от него
- Если исходный ключ меньше, чем K , то он дополняется нулями

Имитовставки

Особенности применения (задаются конкретным протоколом)

- Имитовставка по возможности должна включать порядковый номер и направление передачи сообщения (от Алисы к Бобу, или от Боба к Алисе)
- Имитовставка должна аутентифицировать не только содержание сообщения, но и его смысл
 - Не «20 80 753», а «температура 20 влажность 80 давление 753»
 - Принцип введен, чтобы избежать ошибок при переменной длине полей
- Часто один и тот же текст нужно и шифровать, и аутентифицировать:
 - Можно добавить имитовставку, затем все вместе зашифровать
 - Можно зашифровать текст, затем вычислить имитовставку от ШТ
 - Если имитовставка не верна, можно не расшифровывать (актуально при DoS-атаках)
 - Злоумышленник видит пары сообщение-имитовставка
 - Ключи шифрования и аутентификации должны быть различны
 - Можно применить специальный режим шифрования с аутентификацией
 - Такие режимы часто запатентованы, либо не получили широкого применения



Темы докладов

- **Хеш-функция MD6**
- **Хеш-функция Blake**
- **Хеш-функция Luffa**
- **Имитовставка U-MAC**
- **Режим шифрования ОСВ**

