ЛЕКЦИЯ 12.
Хэш-функции

12.1. Требования к хэш−функциям.

12.2. Простые хэш−функции.

12.3. Парадокс дня рождения и атаки, на нем основанные.

12.4. Способы использования хэш−функций.

12.5. Криптоанализ хэш−функций.

Хэш-функцией называется **<u>односторонняя</u>** функция, предназначенная для получения *дайджеста* или "отпечатков пальцев" файла, сообщения или некоторого блока данных.

Хэш-код создается **функцией H**:

h = H (M),

где **M** является сообщением *произвольной* длины,

а **h** является *хэш-кодом* <u>фиксированной</u> длины.

Когда хэш-функция зависит от ключа, результат ее вычисления носит название **кода аутентификации сообщения (MAC– Message Authentication Code).**

Хэш-функция **H**, которая используется для аутентификации сообщений, должна обладать следующими свойствами:

1. *Хэш-функция* **H** должна применяться к блоку данных любой длины.
2. *Хэш-функция* **H** создает выход фиксированной длины.
3. **H (M)** относительно легко (за полиномиальное время) вычисляется для любого значения **M**.
4. Для любого данного значения *хэш-кода* **h** вычислительно невозможно найти **M** такое, что **H (M) = h.**
5. Для любого данного ***x*** <u>вычислительно невозможно</u> найти такое ***y*** $\neq$***x*,** что

$$H(y) = H(x).$$

Такое свойство называют **слабой сопротивляемостью коллизиям.**
Коллизией называется совпадение дайджестов для различных данных.
6. <u>Вычислительно невозможно</u> найти произвольную пару ***(x, y)*** такую, что

$$H(y) = H(x).$$

Это свойство называют **сильной сопротивляемостью коллизиям.**

Первые три свойства требуют, чтобы *хэш-функция* создавала *хэш-код* для любого сообщения.

Четвертое свойство определяет требование **односторонности** *хэш-функции*: **легко создать** *хэш-код* по данному сообщению, но **невозможно восстановить** сообщение по данному *хэш-коду*.

Пятое свойство гарантирует то, что не удастся найти другое сообщение, дающее в результате хэширования ***то же самое значение***, что и данное сообщение.

Если это свойство не выполнено, противник может действовать по следующей схеме:
- перехватить сообщение вместе с присоединенным к нему **шифрованным** хэш-кодом,
- вычислить *нешифрованный* хэш-код сообщения,
-создать *альтернативное* сообщение с **тем же хэш-кодом**.

Шестое свойство определяет **стойкость** функции хэширования к конкретному классу
атак, построенных на парадоксе «задачи о днях рождения».

Все функции хэширования построены на следующих *общих* принципах.
1. Вводимое значение (сообщение, файл и т.д.) рассматривается как
последовательность n-битовых блоков.
2. Вводимые данные обрабатываются
последовательно блок за блоком,
чтобы в результате получить
n-битовое значение функции хэширования.

Простейшая функция хэширования - связывание всех блоков операцией *поразрядного* **исключающего "ИЛИ"** (**XOR**): $C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$,

где

C_i – i-й бит хэш-кода, $1 \le i \le n$,

m – число n-битовых блоков ввода,

b_{ij} – i-й бит в j-м блоке,

$\oplus$ - операция **XOR**.

Эта процедура осуществляет *простой побитовый контроль четности* и называется ***продольным контролем чётности***.

Простая функция хэширования, выполняющая операцию XOR

	Бит 1	**Бит 2**	**...**	**Бит n**
Блок 1	b_{11}	b_{21}	…	b_{n1}
Блок 2	b_{12}	b_{22}	…	b_{n2}
...	…	…	…	…
Блок m	b_{1m}	b_{2m}	…	b_{nm}
Хэш-код	C_1	C_2	**…**	C_n

Возможно усовершенствовать такую схему выполнением *однобитового циклического сдвига* или *поворота* значения функции хэширования после завершения обработки каждого очередного блока.

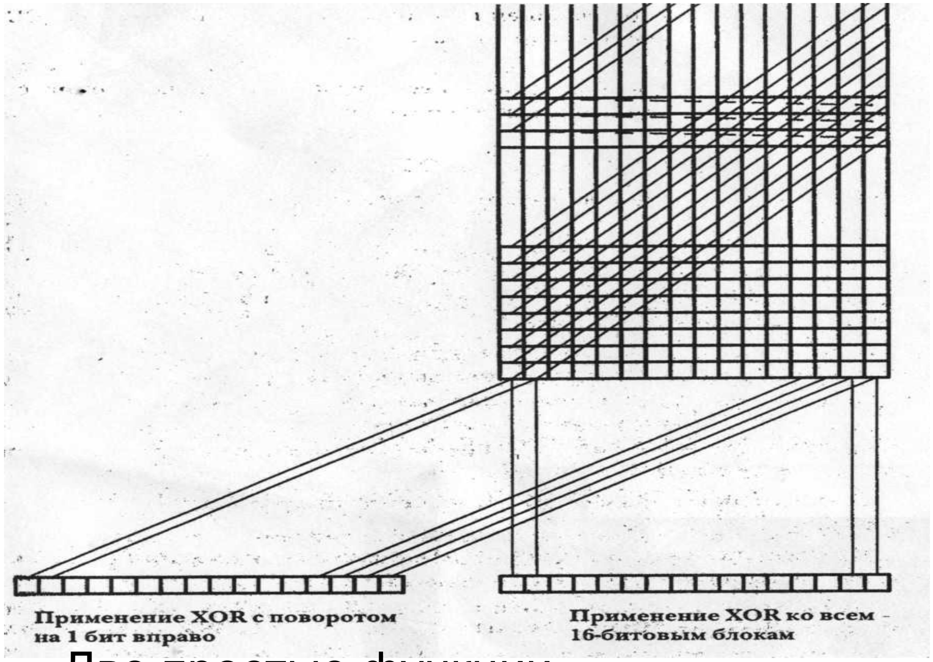
Эта процедура состоит из следующих этапов:

1. Начальная инициализация n-битового значения функции хэширования **нулевым** значением.
2. Последовательная обработка n-битовых блоков данных по следующему правилу:

*Выполнение циклического сдвига текущего значения функции хэширования влево на один бит.

*Добавление текущего блока к значению функции хэширования с помощью операции **XOR**.

Эта процедура демонстрирует эффект "***рандомизации***" вводимых данных и разрушения регулярностей, которые наблюдаются для вводимых данных.



Две простые функции хэширования

Когда шифруются и ***хэш-код,*** и ***сообщение*** требуется шифрование всего сообщения в **режиме сцепления блоков (CBC):**

1. Имея сообщение из последовательности ***64-битовых*** блоков $X_1, X_2, \ldots, X_n$, сначала следует вычислить **хэш-код C**, равный результату связывания всех блоков с помощью операции **XOR**,

$$C = X_1 \oplus X_2 \oplus \ldots \oplus X_n ,$$

2. а затем присоединить полученный **хэш-код C** к концу сообщения в качестве еще одного блока:

$$C = X_{n+1} .$$

3. После этого все сообщение вместе с присоединенным **хэш-кодом C** шифруется в режиме **CBC**, в результате чего получается шифрованное сообщение

$$Y_1, Y_2 \ldots Y_{n+1}.$$

Метод сцепления блоков

(техника сцепления шифрованных блоков, но без использования секретного ключа).

1. Сообщение **M** делится на блоки фиксированной длины $M_1, M_2, \ldots, M_n$
2. и используется любая система традиционного шифрования (например, ***DES***), чтобы вычислить хэш-код **G** следующим образом:

$$H_o = \textit{начальное значение},$$
$$H_i = E_{Mi}[H_{i-1}],$$
$$G = H_n\,.$$

Парадокс дня рождения

ПОСТАНОВКА ЗАДАЧИ (Ч.1)

Предположим, количество выходных значений *хэш-функции* **H** равно ***n***.

Каким должно быть число **k**, чтобы для конкретного значения **X** и значений $Y_1, \ldots, Y_k$ вероятность того, что хотя бы для одного Y_i выполнялось равенство

$$H(X) = H(Y),$$

была бы ***больше 0,5?***

РЕШЕНИЕ

1. Для одного **Y** вероятность того, что **H (X) = H (Y),** равна $1/n$.
2. Соответственно, вероятность того, что **H(X)** $\neq$ **H(Y),** равна $1 - 1/n$.
3. Если создать ***k*** значений, то вероятность того, что ни для одного из них не будет совпадений, равна произведению вероятностей, соответствующих одному значению, т.е.

$$(1 - 1/n)^k.$$

Следовательно, вероятность, по крайней мере, одного совпадения равна

$$1 - (1 - 1/n)^k.$$

По формуле бинома Ньютона

$$(1 - a)^k = 1 - ka + (k(k-1)/2!)a^2 - \ldots \approx 1 - ka.$$

Т.е.

$$1 - (1 - k/n) = k/n = 0{,}5,$$

откуда

$$k = n/2.$$

Для ***m***-битового *хэш-кода* достаточно выбрать 2^{m-1} сообщений, чтобы вероятность совпадения *хэш-кодов* была больше **0,5**.

ПОСТАНОВКА ЗАДАЧИ (Ч.2)

Обозначим **P(n, k)** вероятность того, что во множестве из ***k*** элементов, каждый из которых может принимать ***n*** значений, есть хотя бы два с одинаковыми значениями.

Чему должно быть равно ***k***, чтобы **P(n, k)** была бы больше **0,5**?

РЕШЕНИЕ

Число различных способов выбора элементов таким образом, чтобы при этом не было дублей, равно

$$n(n-1) \ldots (n-k+1)n!/(n-k)!.$$

Всего число возможных способов выбора элементов равно

$$n^k.$$

Вероятность того, что дублей нет, равна

$$n!/(n-k)!n^k.$$

Вероятность того, что есть дубли, соответственно равна:

$$1 - n!/(n-k)!n^k.$$

$$P(n, k) = 1 - n! / ((n-k)! \times n^k) =$$

$$1 - (n \times (n-1) \times \ldots \times (n-k-1)) / n^k =$$

$$1 - [\, (n-1)/n \times (n-2)/n \times \ldots \times (n-k+1)/n] =$$

$$1 - [(1- 1/n) \times (1 - 2/n) \times \ldots \times (1 - (k-1)/n)].$$

Известно, что $1 - x \leq e^{-x}$.

По условию задачи

$$P(n, k) > 1 - [e^{-1/n} \times e^{-2}/n \times \ldots \times e^{-k}/n],$$ т.е.

$$P(n, k) > 1 - e^{-k(k-1)/n}.$$

Следовательно,

$$1/2 = 1 - e^{-k(k-1)/n},$$ и

$$2 = e^{k(k-1)/n}.$$

Отсюда

$$\ln 2 = k\,(k-1) / 2n$$ и

$$k\,(k-1) \approx k^2.$$

Окончательно имеем

$$k = (2n \times \ln 2)^{1/2} = 1{,}17\, n^{1/2} \approx n^{1/2}.$$

Таким образом, если *хэш-код* имеет длину ***m*** бит, т.е. принимает 2^m значений, то

$$k = \sqrt{2^m} = 2^{m/2}.$$

"Парадокс дня рождения" **-** для того, чтобы вероятность совпадения дней рождения у двух человек была больше 0,5, в группе должно быть всего **23** человека.

Атаки, основанные на парадоксе дня рождения

Возможна следующая стратегия:

1. ***Противник*** (*атакующий*) создает $2^{m/2}$ вариантов сообщения, каждое из которых имеет некоторый определенный смысл.

Противник подготавливает <u>такое же</u> количество сообщений, каждое из которых является *поддельным* и предназначено для замены настоящего сообщения.

Два набора сообщений сравниваются в поисках пары сообщений, имеющих **одинаковый** *хэш-код*. Вероятность успеха в соответствии с "парадоксом дня рождения" больше, чем 0,5.

Если соответствующая пара не найдена, то создаются дополнительные исходные и поддельные сообщения *до тех пор, пока не будет найдена пара.*

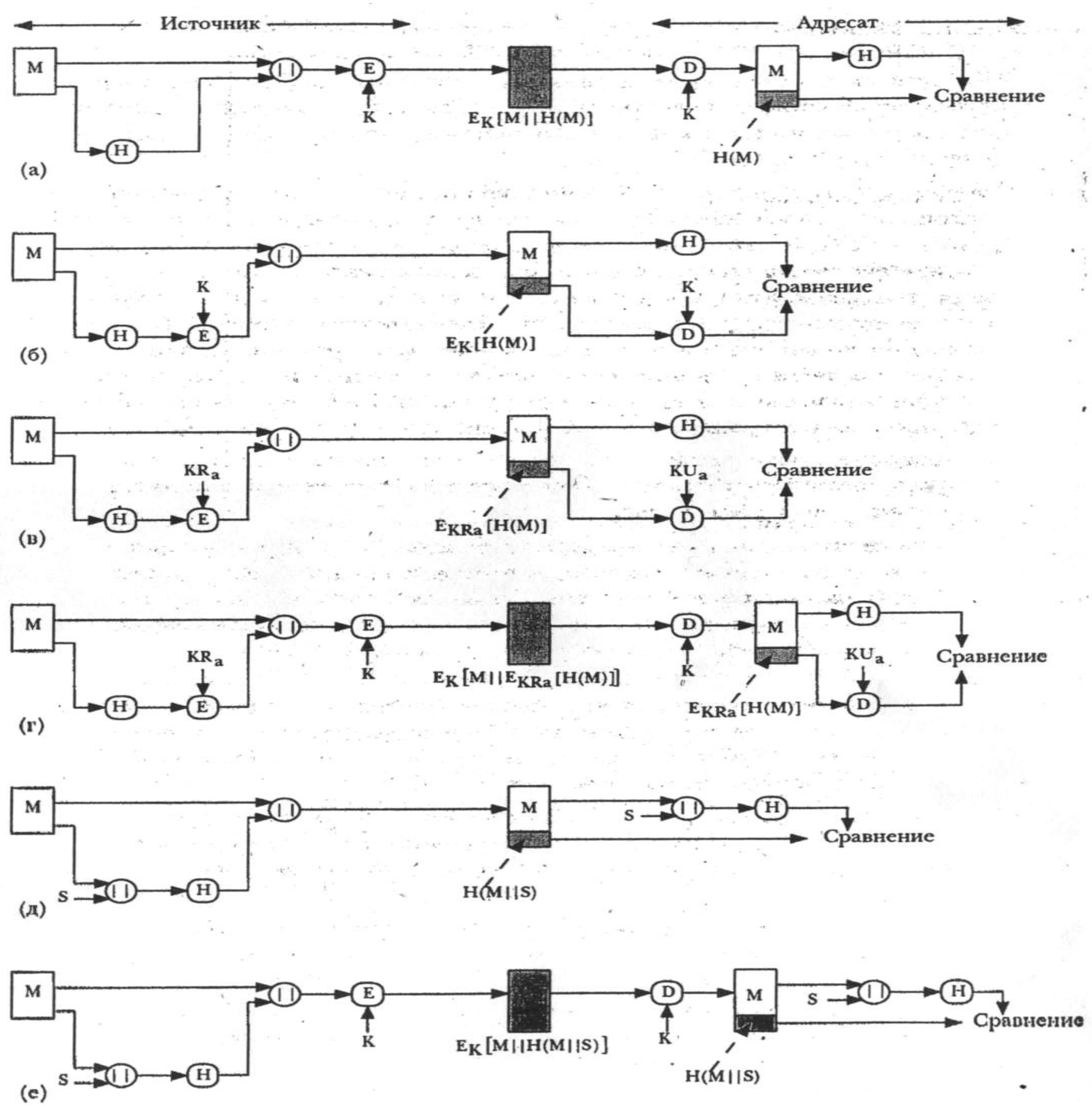
2. Атакующий предлагает **отправителю** исходный вариант сообщения для подписи.

Эта подпись может быть затем присоединена к поддельному варианту для передачи получателю. Так как оба варианта имеют **один и тот же** *хэш-код*, будет создана одинаковая подпись.

Противник будет уверен в успехе, даже не зная ключа шифрования.

Если используется 64-битный *хэш-код*, то необходимая сложность вычислений

2^{32}



Основные способы использования функции хэширования

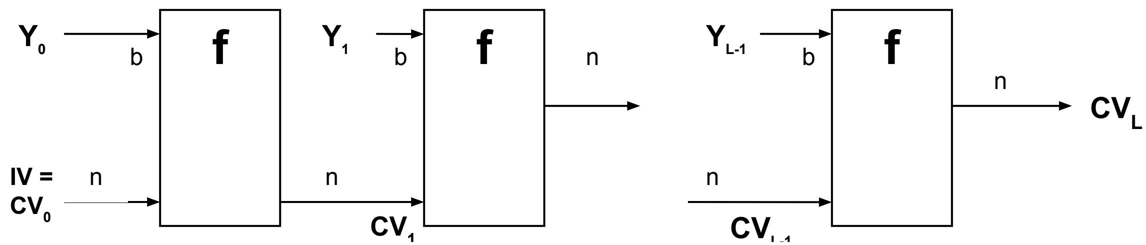
Основные возможности использования функции хэширования

(а) A→B: E_K **[M || H(M)]** •Обеспечивает конфиденциальность - Только стороны **A** и **B** знают ***К*** •Обеспечивает аутентификацию - **H(M) криптографически** защищено	**(г) A→B:** E_K **[M ||** E_{KRa}**[H(M)]]** •Обеспечивает аутентификацию и **цифровую подпись** •Обеспечивает конфиденциальность - Только стороны **A** и **B** знают **К**
(б) A→B: **M ||** E_K **[H(M)]** •Обеспечивает аутентификацию - **H(M) криптографически** защищено	**(д) A→B:** **M || H(M || S)** •Обеспечивает аутентификацию - Только стороны **A** и **B** знают **S**
(в) A→B: **M ||** E_{KRa}**[H(M)]** •Обеспечивает аутентификацию и **цифровую подпись** - **H(M) криптографически** защищено - Только сторона **A** может создать E_{KRa}**[H(M)]**	**(е) A→B:** E_K **[M || H(M || S)]** • Обеспечивает аутентификацию - Только стороны **A** и **B** знают **S** • Обеспечивает конфиденциальность - Только стороны **A** и **B** знают **К**

Причины интереса к методам,
позволяющим избежать шифрования:

- Программное обеспечение, выполняющее шифрование, работает довольно **медленно**.
- Цены на аппаратные средства шифрования довольно **высокие**.
- Аппаратные средства шифрования оптимизируются для работы с большими объемами данных. При малых блоках данных значительная часть времени тратится **непроизводительно** на *инициализацию/вызов.*
- Алгоритмы шифрования могут быть защищены патентами, что тоже выливается в дополнительные **расходы**.
- Алгоритмы шифрования являются одним из вопросов **экспортного государственного регулирования** .

Криптоанализ хэш−функций.



Общая структура защищенного хэш-кода

(итерированной функцией хэширования)

На рис. использованы обозначения:

IV – начальное значение,

CV – переменная сцепления,

$\mathbf{Y_i}$ – i-й вводимый блок,

f – алгоритм сжатия,

L – число вводимых блоков,

n – длина хэш-кода,

b – длина вводимого блока.

Функция хэширования может быть описана следующим образом:

$$\mathbf{CVo = IV = начальное\ n\text{-}битовое\ значение,}$$

$$\mathbf{CV_1 = f(CV_{i-1}, Y_{i-1}), \quad 1 \le i \le L,}$$

$$\mathbf{H(M) = CV_L,}$$

где вводимыми данными функции хэширования является сообщение **M**, складывающееся из блоков $\boldsymbol{Y_0, Y_1, \ldots, Y_{L-1}}$.

Если **функция сжатия** обладает ***сопротивляемостью коллизиям***, то такой же будет и *итерированная функция хэширования.*

Противник *должен найти:*

1. либо два сообщения равной длины, имеющие одинаковые значения **функции хэширования,**
2. либо два сообщения разной длины, которые вместе с соответствующими им значениями длины будут иметь одинаковые значения **функции хэширования.**

Проблема создания защищенной функции хэширования сводится к проблеме поиска ***функции сжатия***, обладающей **сопротивляемостью коллизиям** и работающей с вводимыми данными некоторой фиксированной длины.

Криптоанализ функций хэширования обычно сосредоточен на исследовании

внутренней структуры f

и опирается на попытки найти эффективные

методы обнаружения коллизий при

однократном выполнении f.

Следует при этом иметь в виду, что

коллизии должны существовать в ***любой*** функции хэширования,

поскольку последняя отображает как минимум

блок длины ***b*** в хэш-код длины ***n***, где **b > n**.

Требуется лишь **вычислительная невозможность** обнаружить такие коллизии.