

Лекция №3

Коды аутентификации сообщений.

Криптографическая хэш-функция.

Шифрование с аутентификацией

Елена Киршанова

Курс “Основы криптографии”

Конфиденциальность & целостность

- В предыдущих лекциях: **конфиденциальность** сообщений
- В этой лекции: **целостность**

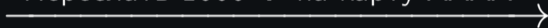
Криптопрититив: Код Аутентификации Сообщения (или Имитовставка)
Message Authentication Code (MAC)

Мотивация

Алиса



“Переслать 1000 ₺ на карту XXXX”



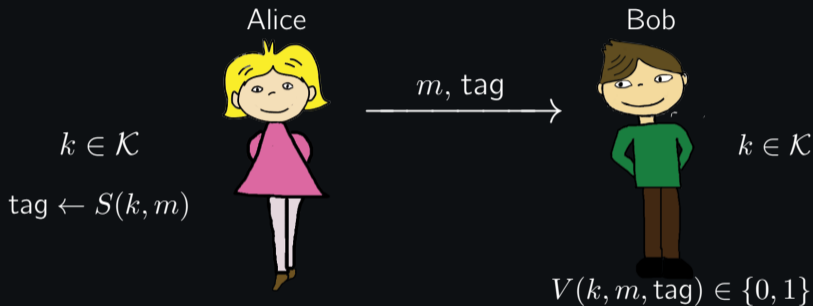
Боб



Ева

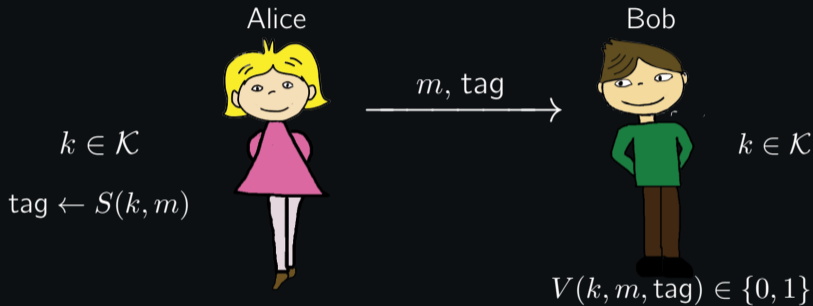
MAC: определение

Цель: отправить сообщение m от Алису к Бобу так, чтобы злоумышленник не смог модифицировать m , оставаясь незамеченным



MAC: определение

Цель: отправить сообщение m от Алису к Бобу так, чтобы злоумышленник не смог модифицировать m , оставаясь незамеченным



Код Аутентификации Сообщения состоит из 3-х ppt алгоритмов:

- Генерация ключа: $\text{KeyGen}(1^\lambda) : k \leftarrow \mathcal{K}$
- Генерация тага: $S(k, m) : \text{tag} \leftarrow \mathcal{T}$
- Верификация тага: $V(k, m, \text{tag}) : \{0, 1\}$

MAC: корректность и безопасность

Код Аутентификации Сообщения состоит из 3-х ррт алгоритмов:

- Генерация ключа: $\text{KeyGen}(1^\lambda) : k \leftarrow \mathcal{K}$
- Генерация тага: $S(k, m) : \text{tag} \leftarrow \mathcal{T}$
- Верификация тага: $V(k, m, \text{tag}) : \{0, 1\}$

Корректность: $V(k, m, S(k, m)) = 1 \forall m \in \mathcal{M}, k \in \mathcal{K}$

Безопасность: Для ррт атакующего \mathcal{A} , имеющего пары $\{(m_1, t_1), \dots, (m_N, t_N)\}$ для m_i , выбранных им самим, \mathcal{A} не может сгенерировать новую пару (m, t)

$$(m, t) \notin \{(m_1, t_1), \dots, (m_N, t_N)\}$$

Безопасная длина тага: 96, 128, 256 бит.

Конструкции MAC

1. Блок-шифр (AES, ГОСТ), псевдослучайная ф-ия f – примеры конструкции MAC для 16-байтных сообщений

$$S(m) := f(k, m) \rightarrow t$$

$$V(k, m, t) := f(k, m) == t ? 1 : 0$$

2. Для более длинных сообщений:
 - CBC-MAC (целостность банковских транзакций)
Стандарты: ANSI, FIPS 186-3, ГОСТ
 - NMAC (Nested MAC)
 - HMAC (SSL, IPsec)

ECBC-MAC

$m = (m_1, m_2, m_3, \dots)$, F – блок-шифр (AES, ГОСТ), $|m_i|$ – 128 бит



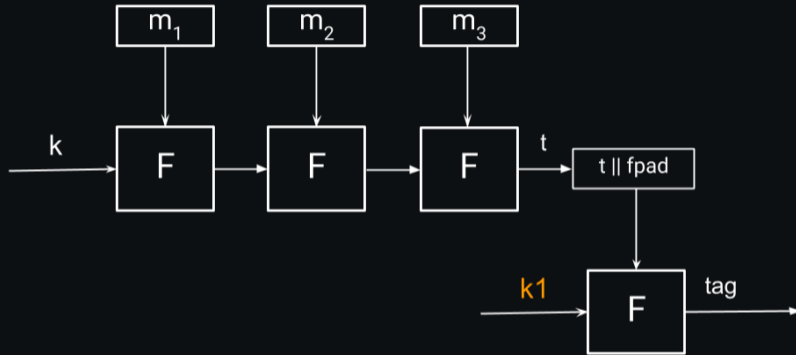
$V((k, k1), m)$ выполняет идентичные шаги.

Отличия от CBC-mode:

- $k, k1$ – два **разных** ключа для F
- Не используется IV
- Выходное значение **tag** может быть урезано (тем самым уменьшая безопасность схемы, стандарты ANSI X9.19, ISO/IEC 9797)

Вложенный (Nested) MAC

$m = (m_1, m_2, m_3, \dots)$, F – блок-шифр (AES, ГОСТ), $|m_i| = n$ бит, $k \in \{0, 1\}^\kappa$.



$\text{fpad} \in \{0, 1\}^{n-\kappa}$ – любое фиксированное значение

Набивка (padding)

Что если $\mathbf{m} = (m_1, m_2, \dots)$ не кратно длине блока?

Пример **плохой** набивки : добивать 0'-ми.

$$m = (\star \star \star), \quad |m| < 128 \text{бит}$$

Набивка такого m :

$$m_{\text{padded}} = (\star \star \star 0 \dots 0).$$

Тогда для сообщения $m' = (m || 0)$:

$$S(k, m_{\text{padded}}) = S(k, m')$$

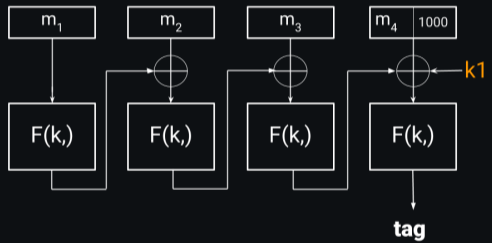
Для $\mathbf{m} \neq \mathbf{m}'$, должно выполняться $\mathbf{m} || \text{pad} \neq \mathbf{m}' || \text{pad}$.

Набивка должна быть функцией $1 \leftrightarrow 1$.

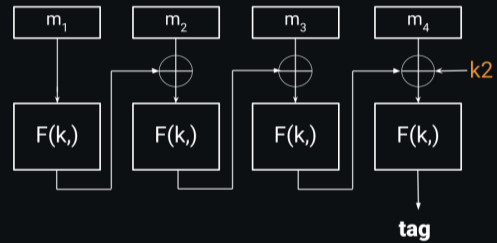
Набивка (padding)

1. ISO. Набивка: $10\dots0$. '1' означает начало набивки.
Добавляется фиктивный блок, если $|m| < \text{длина блока}$

2. NIST, GOST



$m_4 < \text{длина блока}$



$m_4 = \text{длина блока}$

Преимущества: нет фиктивного блока, нет дополнительного вызова F.

Часть II

Криптографическая хэш-функция

Криптографическая хэш-функция \neq Хэш-функция

Фильтры Блума, контрольные суммы (`sumXXX`, `fletcherXXX`), не являются криптографическими хэш-функциями.

Криптографическая хэш-функция: определение

Криптографическая хэш-функция – двойка полиномиальных алгоритмов $(\text{Gen}, \mathcal{H})$:

1. Вероятностный $\text{Gen} : s \leftarrow \text{Gen}(1^\lambda)$

2. Детерминированный $\mathcal{H}_s : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$,

где \mathcal{H}_s является стойкой к коллизиям:

для заданного s , не существует ppt алгоритма, который находит $x, x' (x \neq x')$,

$$\mathcal{H}_s(x) = \mathcal{H}_s(x')$$

Криптографическая хэш-функция обязана быть стойкой к коллизиям.

Существует много коллизий для \mathcal{H}_s , но должно быть трудно найти любую коллизию.

Свойство криптографической хэш-функции

$$\mathcal{H}_s : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$$

I. Стойкость к нахождению прообраза (или односторонность)

Дано: $(s, y \in \{0, 1\}^\ell)$

Найти: x , такой что $\mathcal{H}_s(x) = y$

Стойкая к коллизиям хэш-функция является стойкой к нахождению прообраза

II. Стойкость к нахождению 2-го прообраза

Дано: (s, x)

Найти: $x' \neq x$, такой что $\mathcal{H}_s(x) = \mathcal{H}_s(x')$

Стойкая к коллизиям хэш-функция является стойкой к нахождению 2-го прообраза

Стойкость к коллизиям \implies II. \implies I.

Экзотические свойства хэш-функций

В мире Bitcoin три свойства хэш-функции могут называться иначе:

нахождение прообраза → “hiding”

нахождение 2-го прообраза → “puzzle-friendliness”

стойкость к коллизиям → “collision resistance”

То есть для реализации Bitcoin подойдет криптографическая хэш-функция.

Атака на любую хэш-функцию: парадокс Дней рождений

Положим $h_1, h_2, \dots, h_n \in \{0, 1\}^\ell$ независимо случайные выбранные строки. Парадокс Дней рождений

$$\text{Для } n = \mathcal{O}\left(\sqrt{|\{0, 1\}^\ell|}\right) = \mathcal{O}\left(2^{\ell/2}\right) \quad \Pr[\exists(i \neq j) : h_i = h_j] > 1/2.$$

Алгоритм перебора находит коллизию после $\mathcal{O}(2^{\ell/2})$ вычисленных хэшей:

1. Выбрать $2^{\ell/2}$ случайных строк $m_1, \dots, m_{2^{\ell/2}}$
2. Для каждой m_i вычислить $h_i = \mathcal{H}_s(m_i)$, отсортировать пары (h_i, m_i) по значению h_i
3. Найти в упорядоченном списке $h_i = h_j$. Коллизия: (m_i, m_j) .

Алгоритм успешен с константной вероятностью по парадоксу ДР.

Вывод: Требуем $\ell \geq 160$.

Хэш-функции: исторический дайджест

1. 1980е: MD4 (Message Digest) предложено R. Rivest. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд
2. 1990: MD5. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд

Хэш-функции: исторический дайджест

1. 1980е: MD4 (Message Digest) предложено R. Rivest. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд
2. 1990: MD5. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд
3. 1995: SHA-1 (Secure Hash Algorithm 1) $\ell = 160$
Статус: **Взломана**. См. <https://shattered.io/> два PDF файла с одинаковым значением SHA-1.
!: всё еще используется некоторыми системами (GIT).

Хэш-функции: исторический дайджест

1. 1980e: MD4 (Message Digest) предложено R. Rivest. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд
2. 1990: MD5. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд
3. 1995: SHA-1 (Secure Hash Algorithm 1) $\ell = 160$
Статус: **Взломана**. См. <https://shattered.io/> два PDF файла с одинаковым значением SHA-1.
!: всё еще используется некоторыми системами (GIT).
4. 2001: SHA-2 (SHA-256, SHA-384, SHA-512). $\ell = 256, 384, 512$
Статус: **Считается безопасной**.

Хэш-функции: исторический дайджест

1. 1980е: MD4 (Message Digest) предложено R. Rivest. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд
2. 1990: MD5. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд
3. 1995: SHA-1 (Secure Hash Algorithm 1) $\ell = 160$
Статус: **Взломана**. См. <https://shattered.io/> два PDF файла с одинаковым значением SHA-1.
!: всё еще используется некоторыми системами (GIT).
4. 2001: SHA-2 (SHA-256, SHA-384, SHA-512). $\ell = 256, 384, 512$
Статус: **Считается безопасной**.
5. 2012: SHA-3 (Кессак). SHA-3 $\ell = 224/256/384/512$.
Статус: **Считается безопасной**.

Хэш-функции: исторический дайджест

1. 1980е: MD4 (Message Digest) предложено R. Rivest. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд
2. 1990: MD5. $\ell = 128$
Статус: **Взломана**. Коллизию можно найти в течение секунд
3. 1995: SHA-1 (Secure Hash Algorithm 1) $\ell = 160$
Статус: **Взломана**. См. <https://shattered.io/> два PDF файла с одинаковым значением SHA-1.
!: всё еще используется некоторыми системами (GIT).
4. 2001: SHA-2 (SHA-256, SHA-384, SHA-512). $\ell = 256, 384, 512$
Статус: **Считается безопасной**.
5. 2012: SHA-3 (Кессак). SHA-3 $\ell = 224/256/384/512$.
Статус: **Считается безопасной**.

В России:

1. GOST R 34.11-94 and GOST 34.311-95. $\ell = 256$
Статус: **Считается устаревшей**. Коллизия за 2^{105} операций,
2. GOST R 34.11-2012. Стрибог $\ell = 256, 512$
Статус: **Считается безопасной**.

Часть III

Конструкция Меркла-Дамгора

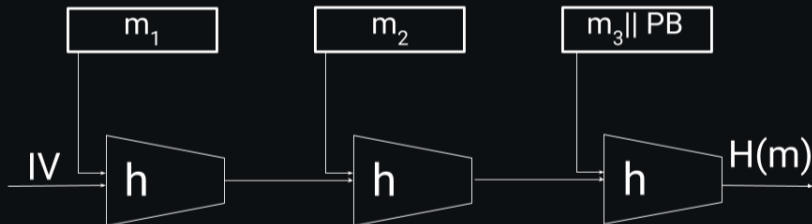
Конструкция хэш-функций: парадигма Merkle-Damgård

Из функции компрессии (определим позже)

$$h : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{K}$$

построим $\mathcal{H} : \mathcal{M}^* \rightarrow \mathcal{K}$.

Пусть $m = (m_1, m_2, m_3)$ произвольной длины



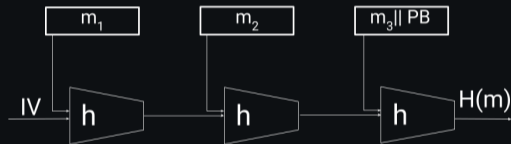
IV – Начальное значение, фиксировано для конкретной хэш-функции

PB – Блок добивки $[100 \dots 0 || |m|]$.

Если **PB** не влезает, добавляем новый блок.

Безопасность конструкции Merkle-Damgård

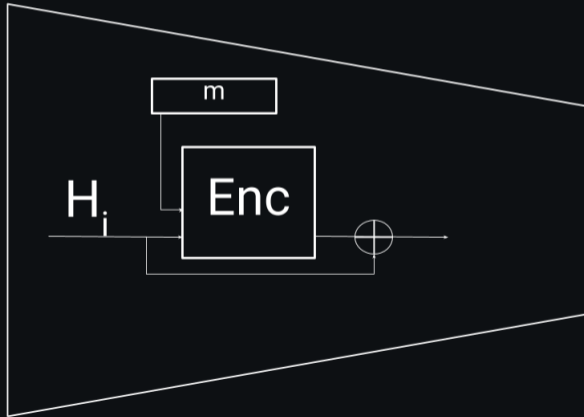
Теорема: Если h стойкая к коллизиям, то и H стойкая к коллизиям.



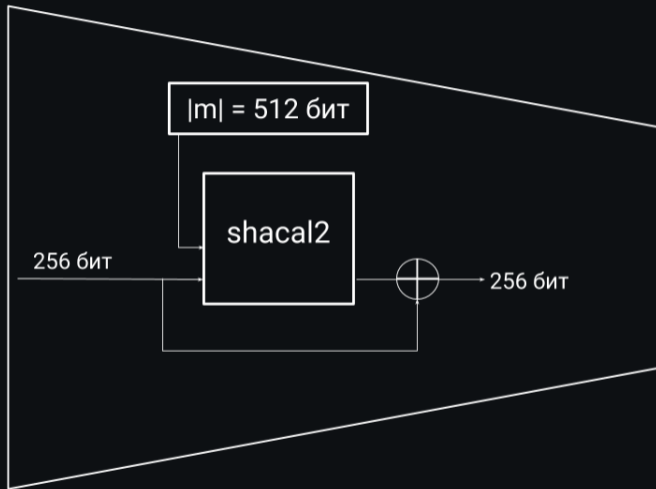
Конструкции функции компрессии h

$\text{Enc} : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ – блок-шифр.

Конструкция Davies-Meyer: $h(H_i, m) = \text{Enc}(H_i, m) \oplus H_i$.



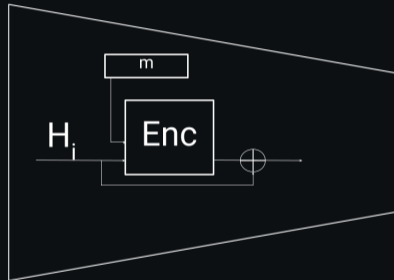
Пример: Функция компрессии в SHA-256



Безопасность Davies-Meyer

Теорема (неформально):

$\text{Enc}(k, \cdot) : \{0, 1\}^n \rightarrow \{0, 1\}^n$ неотличима от случайной перестановки, то нахождение коллизии $h(H, m) = h(H', m')$ требует $\mathcal{O}(2^{n/2})$ вычислений $(\text{Enc}, \text{Enc}^{-1})$.



Альтернативные конструкции h

Конструкция Davies-Meyer:

$$h(H, m) = \text{Enc}(H, m) \oplus H$$

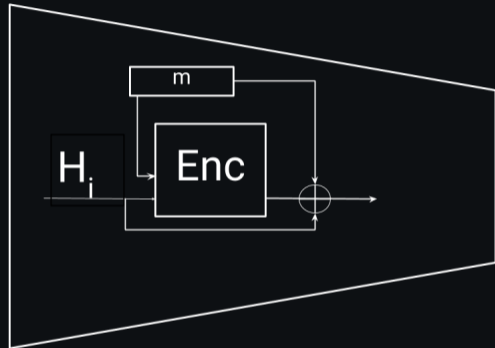
Конструкция Miyaguchi-Preneel:

$$h(H, m) = \text{Enc}(H, m) \oplus H \oplus m$$

ГОСТ Р 34.11-2012 (Стрибог)
использует Miyaguchi-Preneel.

Другие комбинации Enc , H , m
возможны, см. B. Preneel, R. Govaerts,
J. Vandewalle. "Hash functions based on
block ciphers: a synthetic approach."

Не все комбинации безопасны!



Часть IV

Где используются хэш-функции

- Построение MACa
- Протокол идентификации
- Доказательство работы (proof of work)

Построение MACa

Задача: из хэш-функции

$$H : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$$

построить функцию генерации MACa

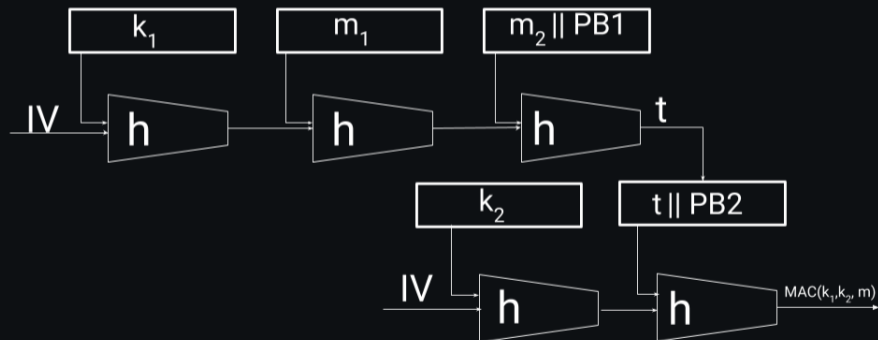
$$S : \mathcal{K} \times \{0, 1\}^* \rightarrow \{0, 1\}^\ell.$$

Основная сложность: хэш-функция – бесключевой примитив.

Конструкция "Two-key Nest"

Для сообщения $M = (m_1, m_2)$ и $h : \{0, 1\}^\ell \times \mathcal{M} \rightarrow \{0, 1\}^\ell$ – функции компрессии

$$S((k_1, k_2), M) = \mathcal{H}_{\text{NMAC}}(k_2 \parallel \mathcal{H}_{\text{NMAC}}(k_1 \parallel M))$$



Теорема: Если $h(\cdot, \cdot)$ и $h(\cdot, \cdot)$ – псевдослучайная функция, то Two-key Nest – безопасный MAC.

Протокол идентификации: определение

$\text{Id} = (\text{KeyGen}, P, V)$ – интерактивный протокол, состоящий из

- $\text{KeyGen}(1^\lambda) \rightarrow (\text{vk}, \text{sk})$
- $P(\text{sk})$ – Доказывающий (Prover)
- $V(\text{vk}) \rightarrow \{0, 1\}$ – Проверяющий (Verifier)

Корректность: $V(\text{vk}) \rightarrow 1$ при общении с sk , где $\text{KeyGen}(1^\lambda) = (\text{vk}, \text{sk})$.

Безопасность:

	Челленджер \mathcal{C}		\mathcal{A}
	$\text{KeyGen}(1^\lambda) \rightarrow (\text{vk}, \text{sk})$	$\xrightarrow{\text{vk}}$	
		\leftarrow	
		\rightarrow	

Выигрыш $\mathcal{A} - \mathcal{C}$ возвращает “1” (Ассепт).

Id – безопасный Id -протокол относительно прямых атак, если $\forall \text{ppt } \mathcal{A}$ вероятность выигрыша $\text{negl}(\lambda)$.

Протокол парольной идентификации

$\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ — хэш-функция

KeyGen \rightarrow (sk = pwd , vk = $\mathcal{H}(pwd)$)

P(pwd)

V(vk)

\xrightarrow{pwd}

$\mathcal{H}(pwd) == vk ? 1 : 0$

\mathcal{H} — криптографическая хэш-функция \implies протокол
Id = (KeyGen, P, V) безопасен относительно прямых атак.

Протокол парольной идентификации

$\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ – хэш-функция

KeyGen \rightarrow ($sk = pwd, vk = [\mathcal{H}(pwd, salt), salt \xleftarrow{\$} S]$)

$P(pwd)$

$V(vk = (h, salt))$

\xrightarrow{pwd}

$\mathcal{H}(pwd, salt) == h ? 1 : 0$

\mathcal{H} – криптографическая хэш-функция \implies протокол
 $\text{Id} = (\text{KeyGen}, P, V)$ безопасен относительно прямых атак.

Хэш-функция в BitCoin

Важный примитив в BitCoin: **Proof of Work (PoW) / Доказательство работы**

Интуиция: вычислительная мощность пользователя \implies пользователь должен доказать это результатом вычислений

- PoW предложен Dwork & Naor (1992) как противодействие спаму
- **Идея:** заставить пользователя решить пазл “средней сложности” (решение должно быть легко верифицировать)

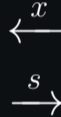
Хэш-функция в Bitcoin: конструкция PoW

Основной примитив: криптографическая хэш-функция $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$, сложность вычисления которой $T(\mathcal{H})$.

Алиса
Доказывающий



Боб
Проверяющий



2. Выбрать $s \in \{0, 1\}^*$
т.ч. $\mathcal{H}(s||x)$
начинается с n 0'ей
Время: $2^n T(\mathcal{H})$

1. $x \in \{0, 1\}^*$
3. Проверить:
 $\mathcal{H}(s||x)$ имеет n 0'ей?
Время: $T(\mathcal{H})$

Для криптографической хэш-функции \mathcal{H} Алиса не может найти s быстрее, чем перебором. Это атака на прообраз.

Часть IV

Шифрование с аутентификацией

Шифрование с аутентификацией: определение

Шифрование с аутентификацией (AE) состоит из трех ppt алгоритмов

- Генерация ключа: $\text{KeyGen}(1^\lambda) : k \leftarrow \mathcal{K}$
- Шифрование: $\text{Enc} : \mathcal{M} \times \mathcal{N} \rightarrow \mathcal{C}$
- Дешифрование: $\text{Dec} : \mathcal{K} \times \mathcal{C} \times \mathcal{N} \rightarrow \mathcal{M} \cup \{\perp\}$

\mathcal{K} - мн-во ключей, \mathcal{M} - мн-во открытых текстов, \mathcal{C} - мн-во шифр-текстов, \mathcal{N} - мн-во **нонсов**.

НОВОЕ: $\{\perp\}$ – шифр-текст отклонен

Нонсе (nonce) = “number that can only be used once”

Нонс может быть предсказуем, но он не должен быть использован **дважды** для одного ключа.

Корректность, безопасность шифрования с аутентификацией

Шифрование с аутентификацией (АЕ) состоит из трех рпт алгоритмов

- Генерация ключа: $\text{KeyGen}(1^\lambda) : k \leftarrow \mathcal{K}$
- Шифрование: $\text{Enc} : \mathcal{K} \times \mathcal{M} \times \mathcal{N} \rightarrow \mathcal{C}$
- Дешифрование: $\text{Dec} : \mathcal{K} \times \mathcal{C} \times \mathcal{N} \rightarrow \mathcal{M} \cup \{\perp\}$

Корректность: $\forall m, \forall k, \forall n : \text{Dec}(k, \text{Enc}(k, m, n), n) = m$

Безопасность (неформально):

- $\text{Enc}(k, m_0, n)$ неотличимо от $\text{Enc}(k, m_1, n) \forall m_0 \neq m_1$ (без знания k)
- Эффективный злоумышленник не в состоянии сформировать шифр-текст, которые не дешифруется в $\{\perp\}$.

Атака на выбранный шифр-текст

В реальных протоколах атакующий может получить доступ к (частичной) функции **дешифрования**.

Пример: атакующий может знать ответ на вопрос “корректно ли сформирован шифр-текст” (атака Bleichenbacher на RSA)

Симметрическое шифрование “из коробки” (CBC, CTR) **не являются** стойкими к таким атакам. **Причина:** измененный шифр-текст является **корректным** шифр-текстом.

Решение: шифрование с аутентификацией

Конструкции шифрований с аутентификацией

AE = Безопасное шифрование + Криптографический MAC

Два ключа: Ключ шифрования k_E , ключ MACа K_M

Две основные парадигмы:

I. Encrypt-then-MAC

1. $c = \text{Enc}(k_E, m)$
2. $t = \text{MAC}(k_M, c)$
3. return (c, t)

Пример: IPSec

II. MAC-then-Encrypt

1. $t = \text{MAC}(k_M, n)$
2. $c = \text{Enc}(k_E, m||t)$
3. return c

Пример: SSL

Конструкции шифрований с аутентификацией

AE = Безопасное шифрование + Криптографический MAC

Два ключа: Ключ шифрования k_E , ключ MACа k_M

Две основные парадигмы:

I. Encrypt-then-MAC

1. $c = \text{Enc}(k_E, m)$
2. $t = \text{MAC}(k_M, c)$
3. return (c, t)

Пример: IPSec

II. MAC-then-Encrypt

1. $t = \text{MAC}(k_M, n)$
2. $c = \text{Enc}(k_E, m||t)$
3. return c

Пример: SSL

- Encrypt-then-MAC всегда даёт AE
- MAC-then-Encrypt даёт AE, если в Enc используются режимы шифрования CTR/CBC
- Другие комбинации MAC / Enc обычно не дают безопасное AE

AE стандарты

1. GCM (Galois Counter Mode). Encrypt-then-MAC

Шифрование: CTR mode + быстрый Mac (Carter-Wegman Mac).

Применение: TLS

Преимущество: скорость

AE стандарты

1. GCM (Galois Counter Mode). Encrypt-then-MAC

Шифрование: CTR mode + быстрый Mac (Carter-Wegman Mac).

Применение: TLS

Преимущество: скорость

2. CCM. MAC-then-Encrypt

Шифрование: CBC MAC (AES)+ CTR mode (AES)

Применение: 802.11i

Преимущество: компактный код

AE стандарты

1. GCM (Galois Counter Mode). Encrypt-then-MAC

Шифрование: CTR mode + быстрый Mac (Carter-Wegman Mac).

Применение: TLS

Преимущество: скорость

2. CCM. MAC-then-Encrypt

Шифрование: CBC MAC (AES)+ CTR mode (AES)

Применение: 802.11i

Преимущество: компактный код

3. ChaCha20-Poly1305. Encrypt-then-MAC

Шифрование: ChaCha20 (Enc) + Poly1305 MAC

Применение: TLS

Преимущество: скорость