ХЭШ-ФУНКЦИИ

Отсутствие коллизий

Стойкость к коллизиям

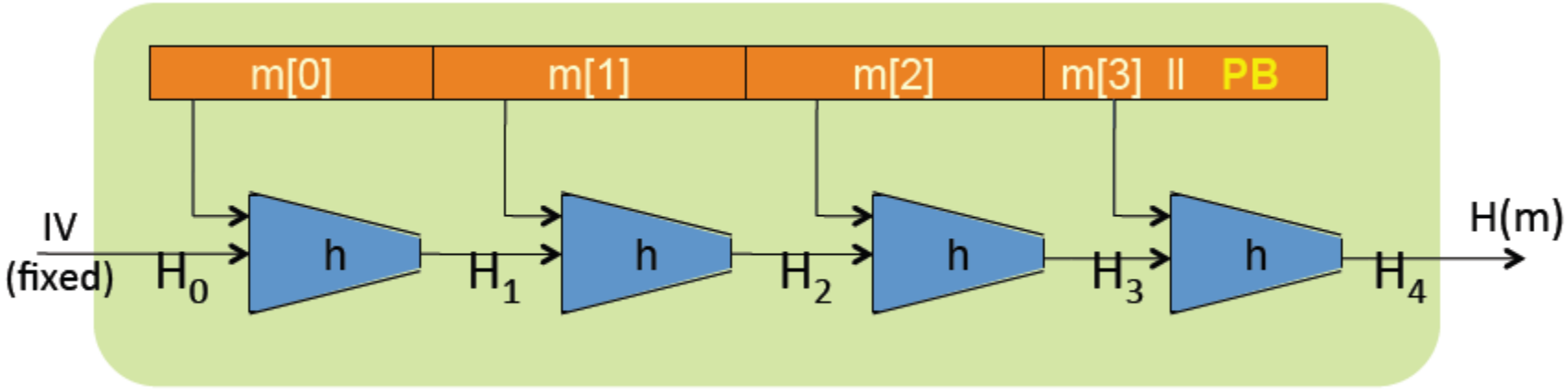
- $H\colon M \to T \ (|M| \gg |T|)$

- Коллизия H – это такая пара $m_0, m_1 \in M$:

$$H(m_0) = H(m_1) \text{ и } m_0 \neq m_1$$

- Функция H свободна от коллизий, если для $\forall$ алгоритма A:
- $Adv_{CR}[A, H] = \Pr\{A \text{ найдет коллизию для } H\}$ – стат. Незначима.

Конструкция Меркла-Дамгарда

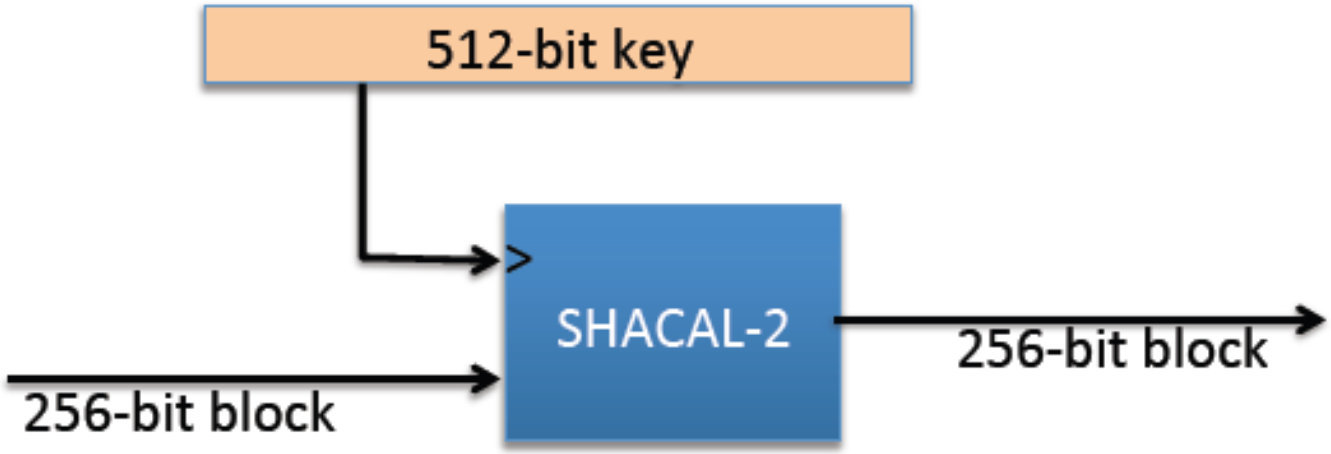


- h – короткая функция свободная от коллизии
- H – длинная функция, свободная от коллизий

Достаточно построить h – сжимающую ф-цию

Примеры SHA-256

- Функция объединения Меркля-Дамгарда
- Блоковый шифр: SHACAL-2

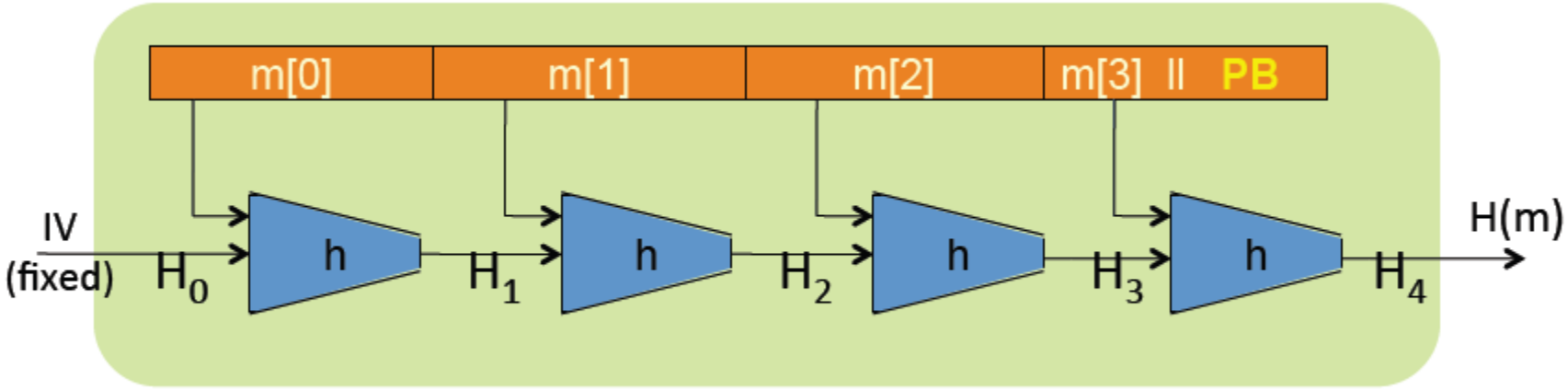


Доказано стойкая сжимающая функция

- Выберем p – простое число (2000 бит) и пару случайных чисел 1< u,v < p.
- Для пары входных значений m,H ∈ {0,1,…,p-1} определим $\mathrm{h}(\mathrm{H},\mathrm{m}) = u^H \cdot v^m \; mod \; p$
- Стойкость: найти коллизию для h(,) можно только решив задачу дискретного логарифма
- Проблема: медленные вычисления.

HMAC

- Как построить MAC, если мы знаем как построить стойкий к коллизиям хэш? Например возьмем CR Меркл-Дамгард хэш-функцию H(IV,m)



  - Вариант 1: S(k,m) =H(k||m)

Стандартный подход: HMAC

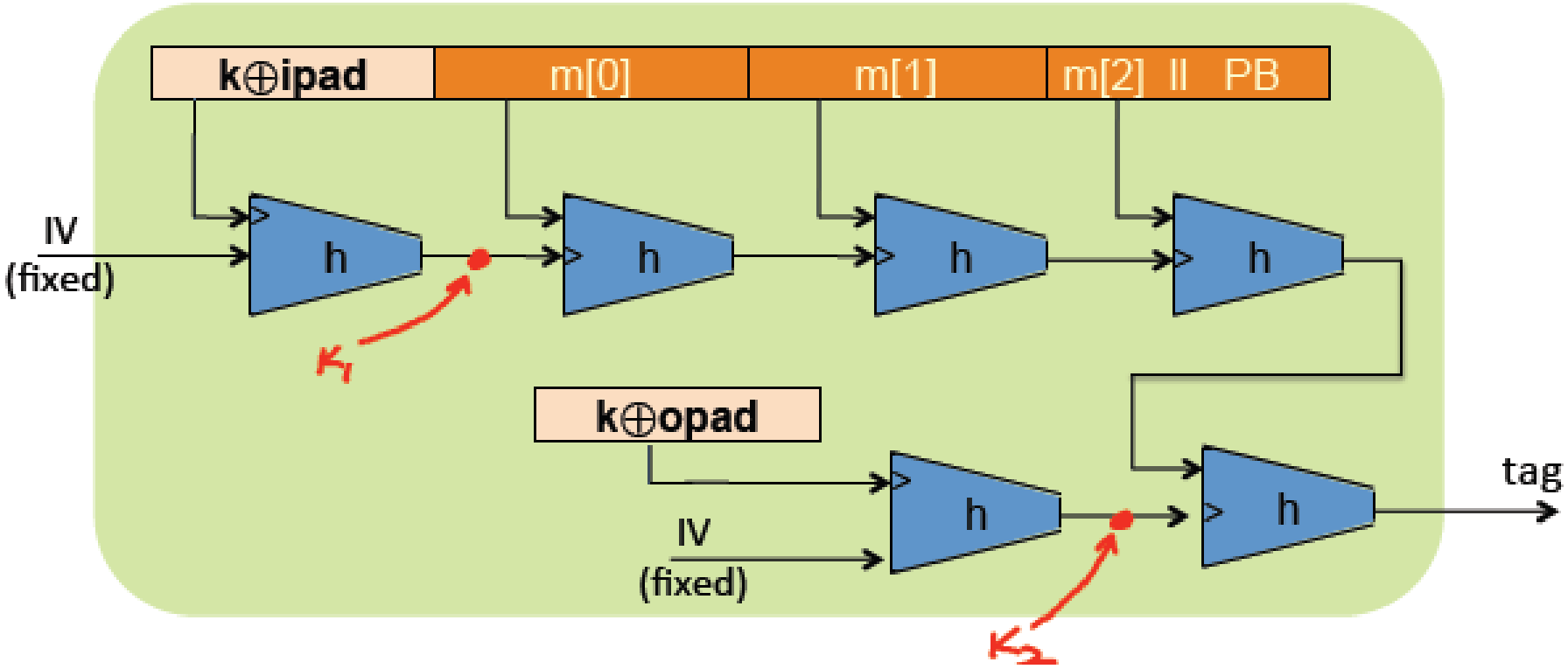
- Пусть есть любая надежная хэш-функция H
  - Например SHA-256 (длины выхода 256 бит)

- Как построить MAC на основе этой функции?

?

$$S(k, m) = H(k \quad opad \,||\, H(k \oplus ipad \,||\, m))$$

Схема HMAC



Похоже по построению на NMAC

Отличие: Зависимые ключи

Свойства HMAC

- Конструкция не зависит от используемой ф-ции H()
- HMAC – криптостойкая PRF
  - Это может быть доказано при ряде допущений, о свойствах h() – PRF
  - Требования для обеспечения стойкости аналогичны NMAC (q << |T|^1/2)
- В Transport Layer Security (TSL): поддерживается старая версия HMAC-SHA1-96.

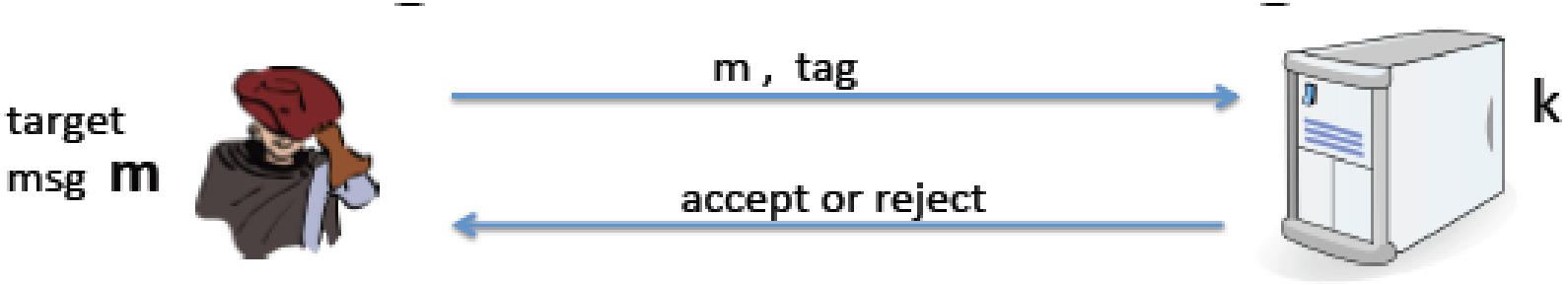
Специфические атаки

- Атака по времени верификации

```
def Verify(key, msg, sig_bytes):
        return HMAC(key, msg) == sig_bytes
```

  - Проблема в выполнении сравнения:
  - Чем раньше найдется несовпадающий байт – тем быстрее будет получен отказ

Описание атаки



- Шаг 1: Отправить случайный tag
- Шаг 2: Перебирать 1 байт и остановиться, когда задержка станет больше, чем в первый раз
- Шаг 3: Повторить процедуру для всех байт tag

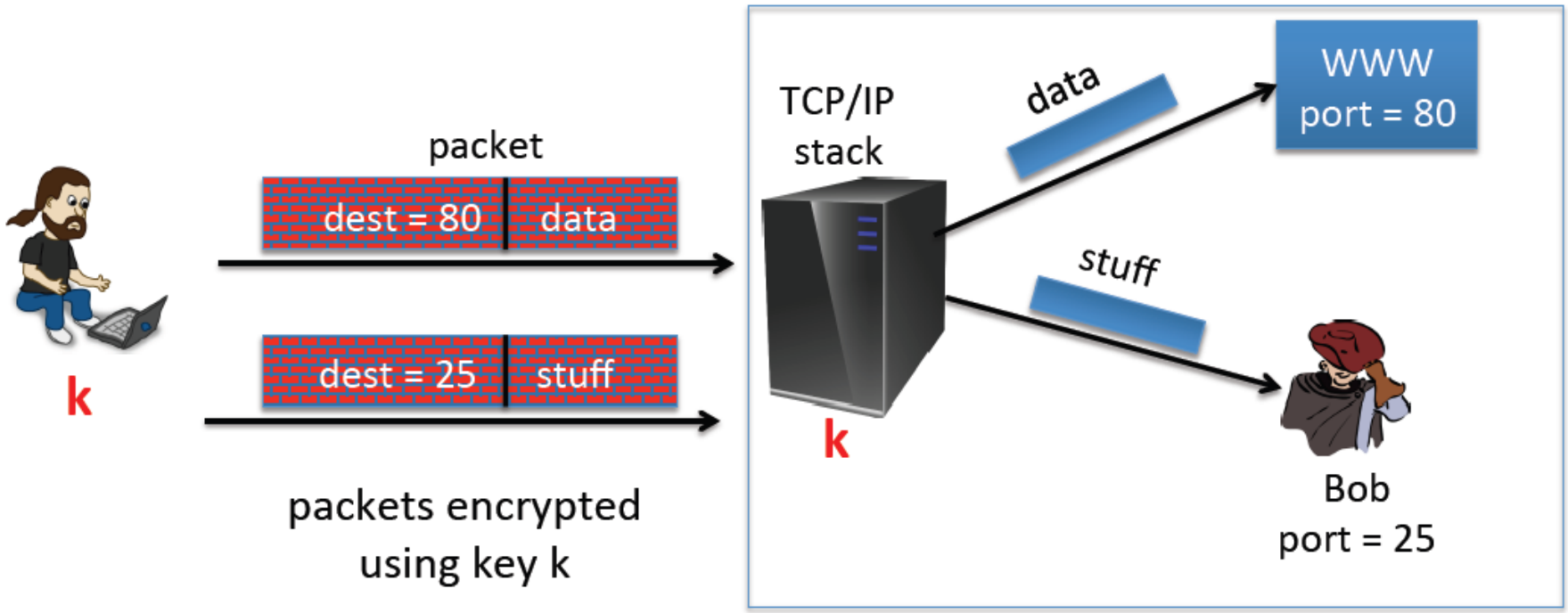
Возможные решения

- Всегда просматривать сравниваемые величины до конца
  - Сложность: оптимизаторы кода
- Внесение псевдослучайных задержек при проверке
  - Сложность: увеличивает время работы
- Увеличение времени проверки для каждого след. запроса.
  - Сложность: нужно контролировать запросы

Аутентифицированное шифрование

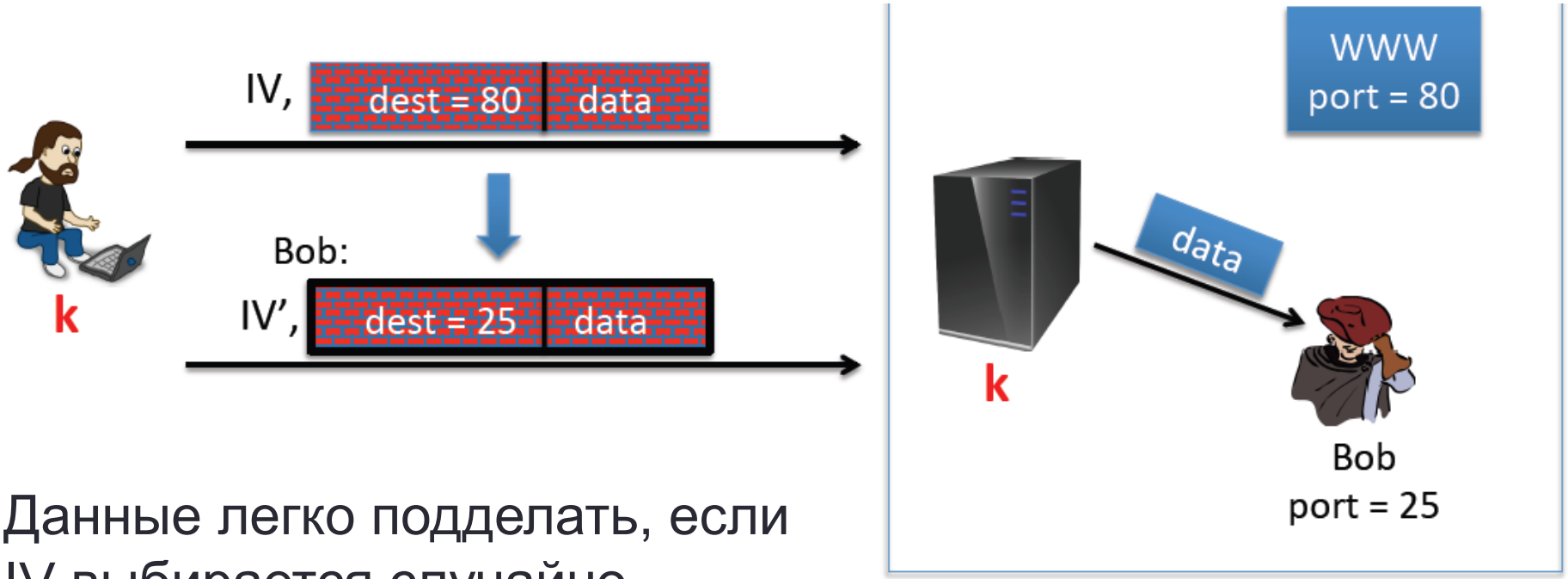
- Криптография должна обеспечивать:
  - Конфиденциальность: Семантическая стойкость против CPA
    - Шифрование
  - Целостность: Неподделываемость при CPA
    - MAC(CBC-MAC, NMAC, HMAC, CW-MAC, PMAC)
  - Шифрование защищенное от подделки:
    - Обеспечение конфиденциальности и целостности.

Описание примера активной атаки



Как получить доступ к чужим данным?

- Атакующий получит расшифрованные данные, если у них Dest = 25



Данные легко подделать, если IV выбирается случайно (достаточно изменить IV)

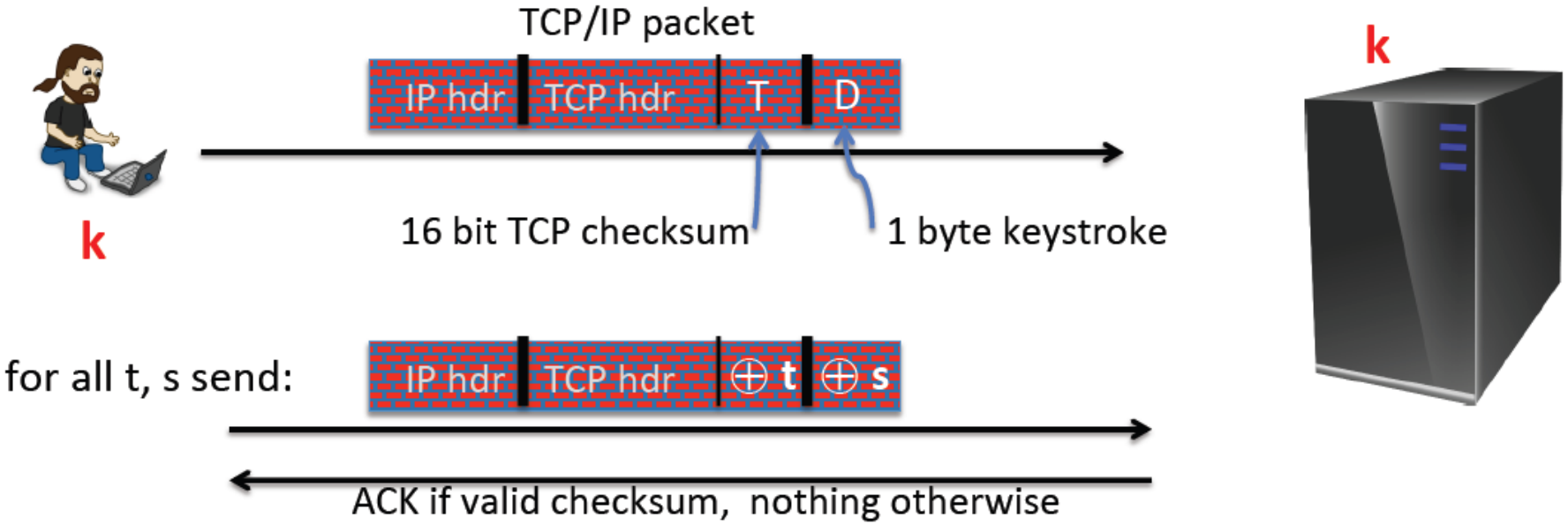
- Как получить новый IV' ?

- Если известно, что текст зашифрован в режиме CBC со случайным вектором IV.

$$m[0] = D(k, c[0]) \oplus IV = \text{“dest=80…”}$$

Атака на терминальную сессию

- Каждое нажатие клавиши шифруется в CTR режиме



$\{ \quad checksum(hdr, D) = t \oplus checksum(hdr, D \oplus s) \quad \} \Rightarrow$

D -- ?

Вывод

- Даже доказанно CPA стойкие функции не гарантируют защиту от активных атак

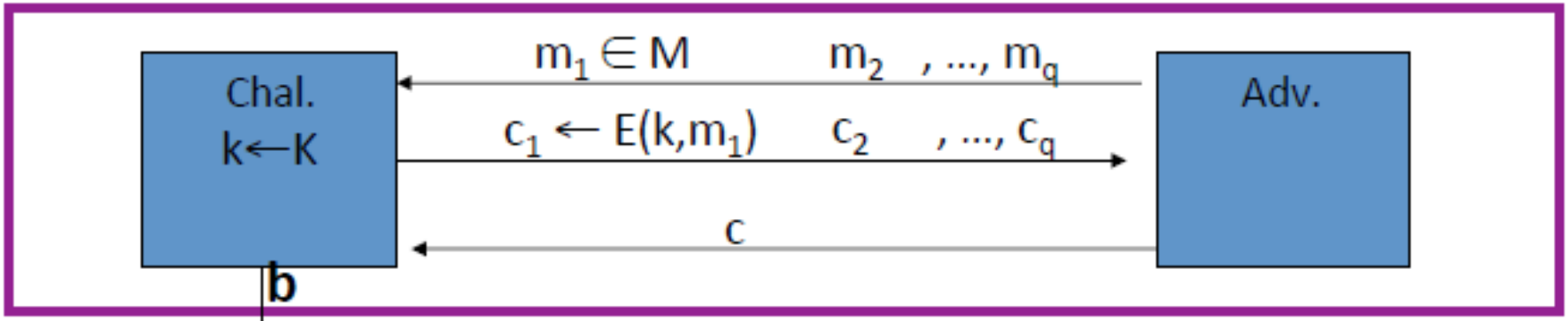
- Для защиты от активных атак 2 подхода
  - MAC
  - Аутентифицированное шифрование

Определение

- Аутентифицированное шифрование – это алгоритм шифрования (E,D), у которого:
  - Как обычно E: K x M x N -> C
  - но D: K x C x N -> M $\cup$ {$\perp$}
- Стойкость включает:
  - Семантическая стойкость против CPA
  - Целостность шифротекста

Целостность шифротекста

- Пусть пара (E,D) шифр с пространством сообщений M



$$\begin{cases} \mathbf{b}=1 \quad \text{if } D(k,c) \neq \perp \quad \text{and } c \notin \{c_1, \ldots, c_q\} \\ \mathbf{b}=0 \quad \text{otherwise} \end{cases}$$

- Def: (E,D) обеспечивают целостность шифротекста, если для любого алгоритма A:

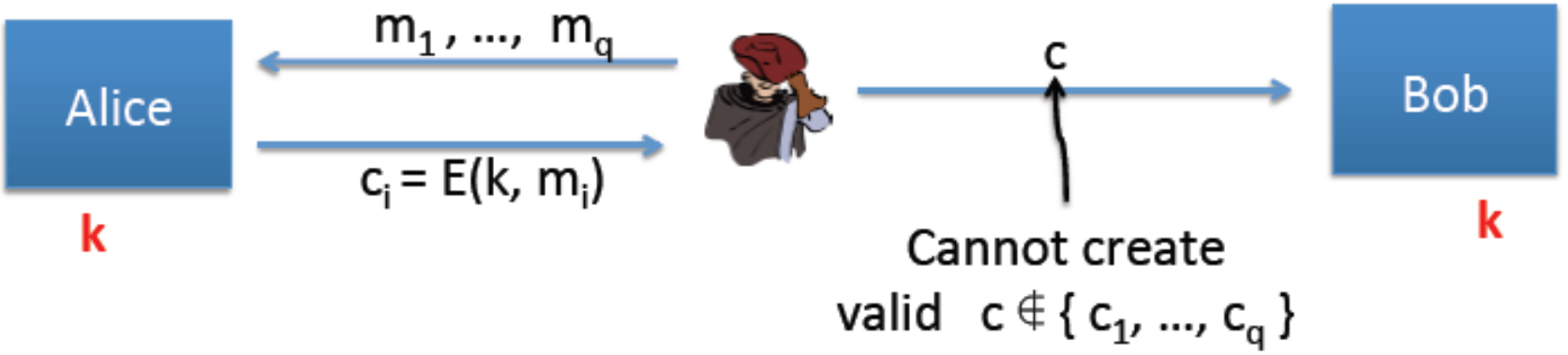
$$Adv_{CI}[A,E] = \Pr[\mathbf{b} = 1] \text{ мало}$$

Аутентифицированное шифрование

- Def: Шифр (E,D) обеспечивает аутентифицированное шифрование (**authenticated encryption(AE)**), если он
  - Семантически стойкий против CPA
  - обладает свойством целостности шифротекста
- Неудачный пример: CBC со случайным IV не является (AE)
  - D не может вернуть отказ, значит атакующий выиграет игру

Применение AE 1

- Атакующий не может обмануть Боба, и подделать сообщение, отправленное Алисой



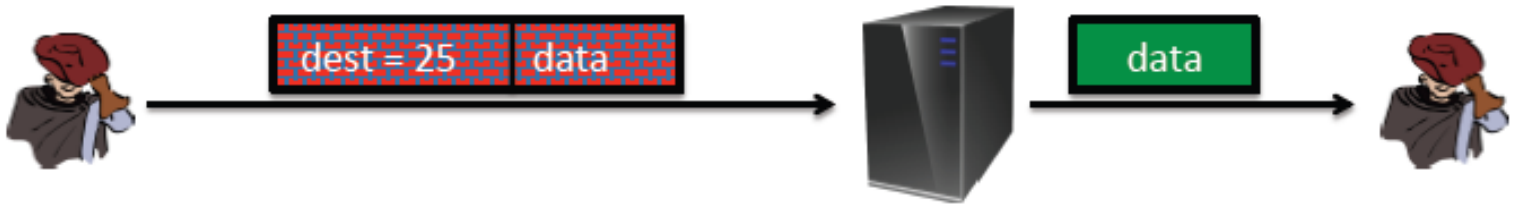
- Если $D(k, c) \neq \perp$ Боб понимает, что сообщение создано кем-то, кто знает ключ

Применение AE 2

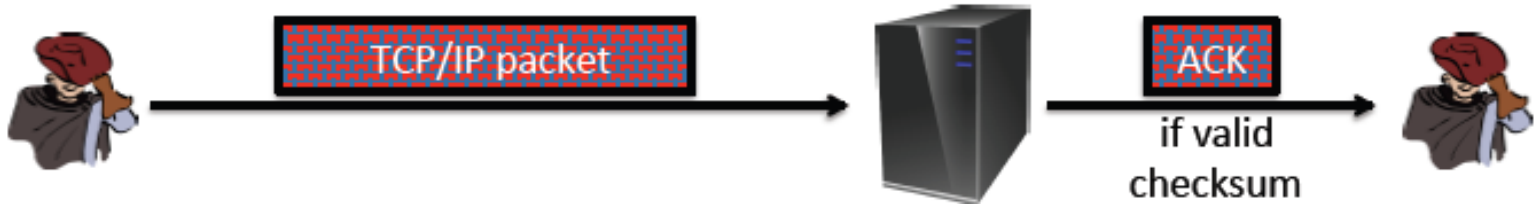
- Аутентифицированное шифрование –
- защита от атаки по выбранному шифротексту (CCA)

Примеры атак по выбранному шифротексту

- У атакующего есть шифротекст (c) , который он хочет расшифровать
  - Он может обмануть сервер и расшифровать некоторое сообщение (не c)



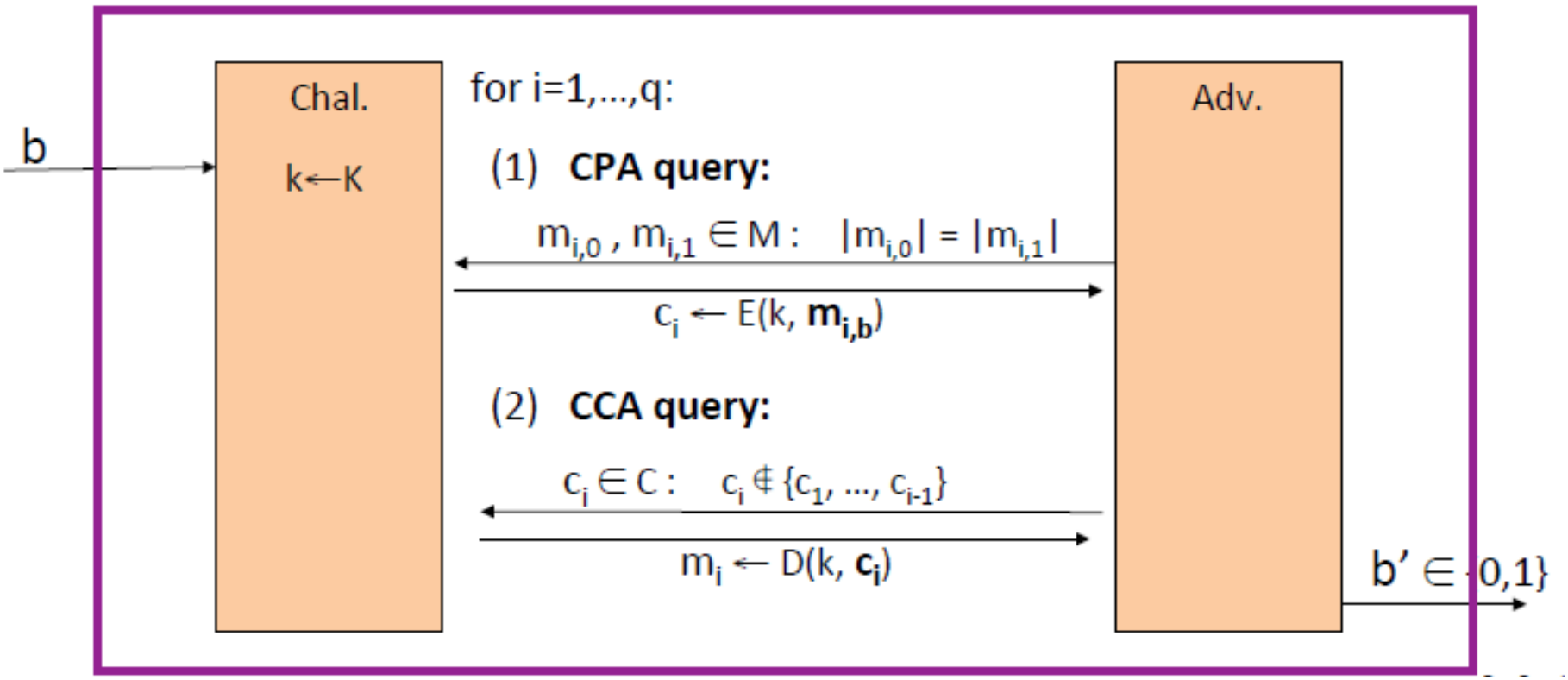
  - Он может получить частичную информацию об открытом тексте



Стойкость к атаке по выбранному шифротексту

- Возможности атакующего: CPA и CCA атаки доступны
  - Может получать зашифрованные сообщения по своему выбору
  - Может получать расшифрованное сообщение по своему выбору, кроме искомого с
- Цель атакующего: Нарушить семантическую стойкость

Определение CCA security



$Adv_{CCA}[A, E] = |\Pr\{b' = 1 \mid b = 0\} - \Pr\{b' = 1 \mid b = 1\}|$ мало

Аут. шифрование –> CCA security

- Теорема: Если (E,D) – шифр, обеспечивающий аут. шифрование, тогда (E,D) -- CCA secure.

- Было доказано, что для любого алгоритма A с q запросами существуют алгоритмы B_1 и B_2, такие что

$$Adv_{CCA}[A, E] \leq 2q\, Adv_{CI}[B_1, E] + Adv_{CPA}[B_2, E]$$

Ограничения аут. шифрования

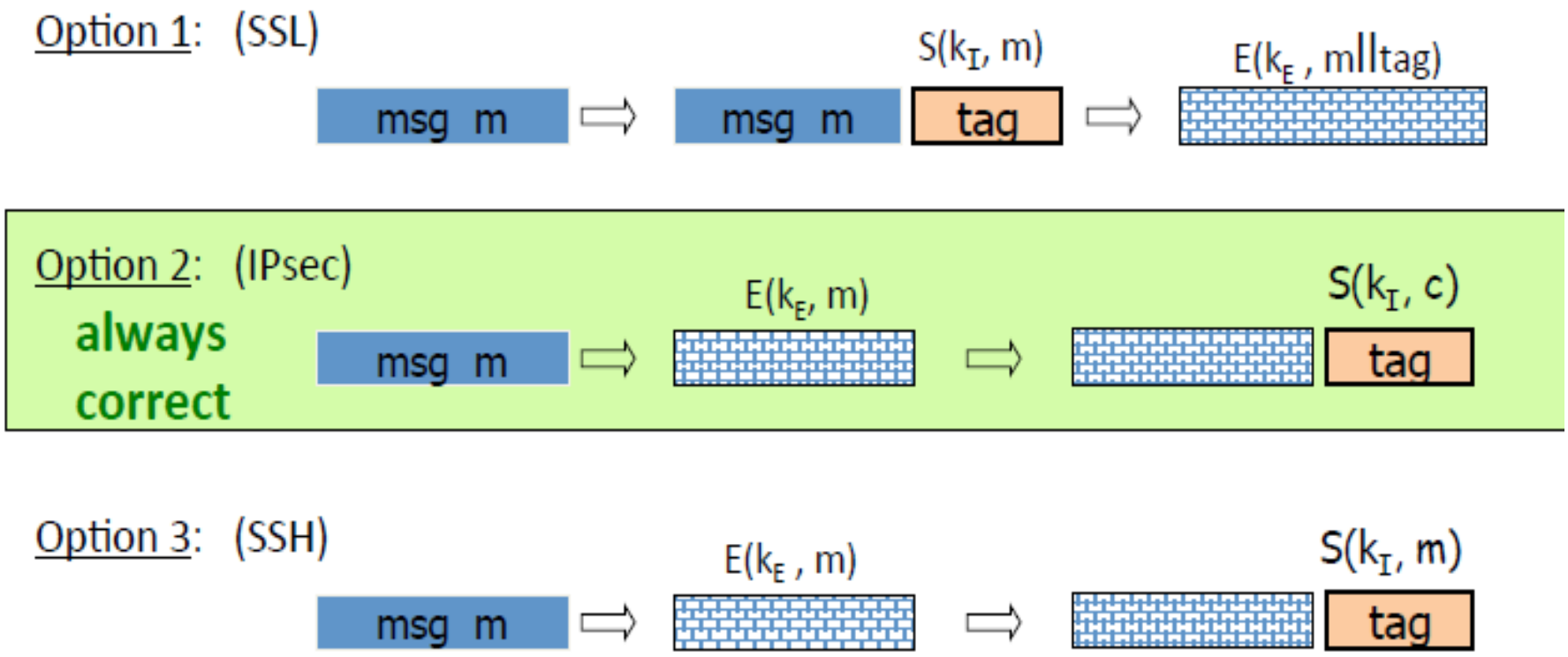
- Не может противостоять атакам с повторением сообщений
- Не принимает во внимание вторичные каналы утечки информации (timing)

Способы построения AE

- Понятие аутентифицирующего шифрования введено в 2000 году

- AE =
  - CPA-secure шифрование +
  - MAC

Способы комбинирования MAC и ENC



Обеспечение AE

- Если (E,D) – CPA secure шифр, а (S,V) – стойкий MAC. Тогда:

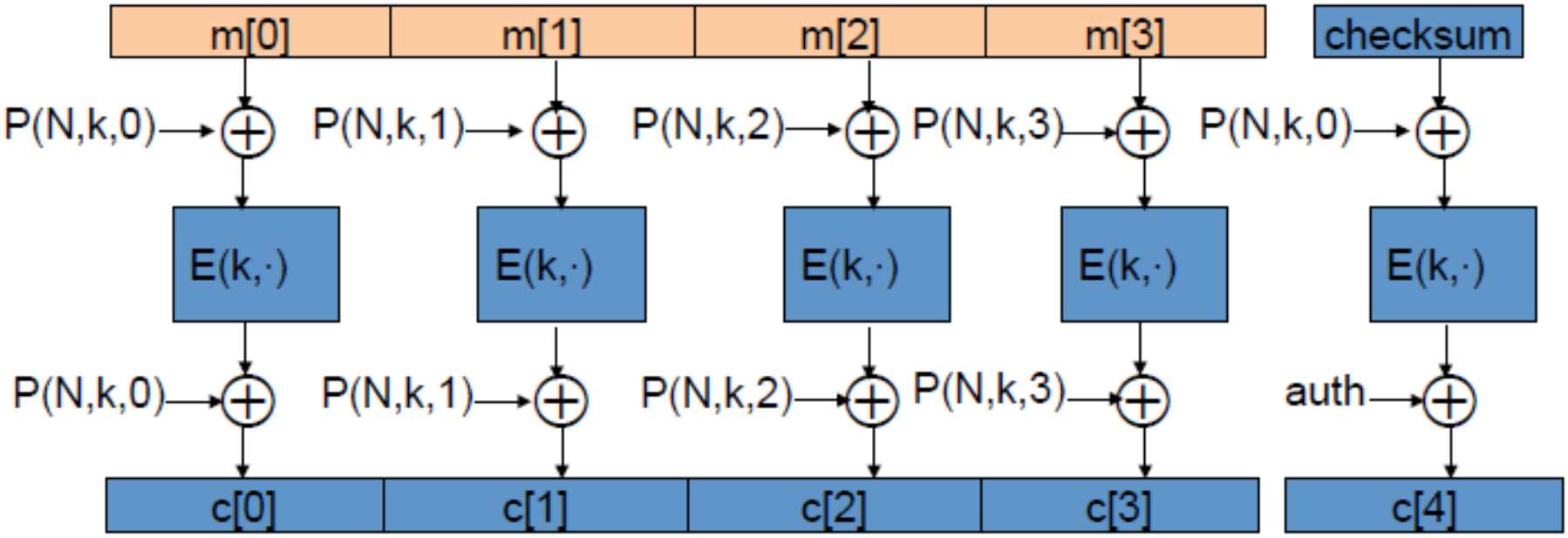
1. Шифрование потом MAC: ВСЕГДА обеспечивает AE
2. MAC потом шифрование: может оказаться нестойким к CCA атаке.
   - При этом известно, что если (E,D) работает в rand-CTR или rand-CBC режиме, то схема MAC потом шифрование обеспечивает AE.
   - В других случаях возможны варианты.

Стандарты

- GSM: щифрование в режиме CTR + CW-MAC
- CCM: CBC-MAC потом шифрование в режиме CTR
- AEX: щифрование в режиме CTR + CMAC
- Все могут работать в режиме AEAD.

Режим шифрования OCB

- Offset Codebook Mode (ocb)



Производительность

AMD Opteron, 2.2 GHz (Linux)

Cipher	code size	Speed (MB/sec)		
AES/GCM	large**	108	AES/CTR	139
AES/CCM	smaller	61	AES/CBC	109
AES/EAX	smaller	61		
			AES/CMAC	109
AES/OCB		129*	HMAC/SHA1	147