

Поточные и Блочные шифры

Макаров Артём
МИФИ 2024

Выполнение лабораторных работ

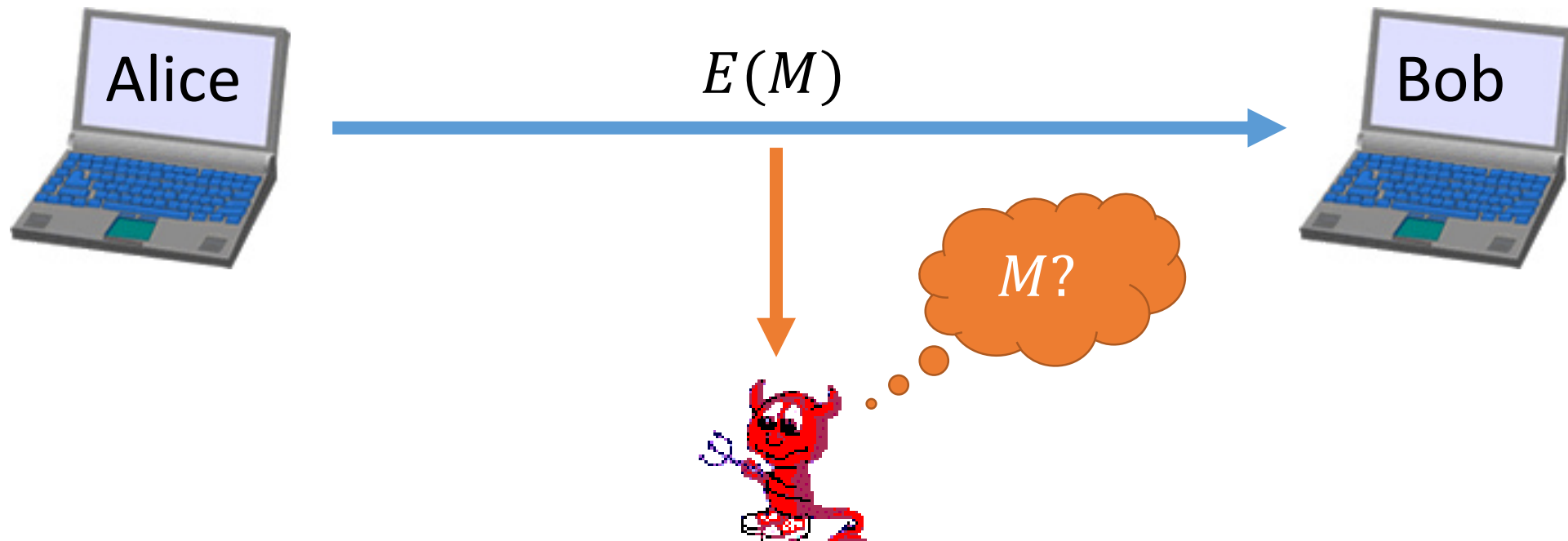
- Лабораторные работы выполняются самостоятельно
- Запрещается использовать части кода, полученные из открытых источников, включая генеративные сети
- Запрещается распространение исходных кодов выполненной или части выполненной лабораторной работы
- Лабораторная работа должна быть загружена в соответствующий репозиторий
- На выполнение каждой работы 3 недели, каждые 2 недели новая лабораторная работа
 - После истечения дедлайна сдавать лабораторную работу нельзя

Выполнение лабораторных работ

- Лабораторная работа должна быть загружена в соответствующий репозиторий
- После загрузки лабораторной работы необходимо написать мне для проверки работы
 - Результат проверки удовлетворительный – получаете оценку
 - Результат неудовлетворительный – получаете список исправлений для внесения, после чего загружаете изменения и вновь пишете мне для проверки
- Не загружать в репозиторий файлы, не относящиеся к исходным кодам выполненной работы
 - Не загружать бинарные файлы! Только файлы с исходным кодом!
 - Рекомендуется использование .gitignore
 - Если вы всё же загрузили бинарный файл – написать мне, не удалять самостоятельно (ибо нужно удалить из истории)!

Историческая задача криптографической защиты информации

- Передача зашифрованного сообщения по открытому каналу
- При перехвате зашифрованного сообщения открытый текст должен остаться неизвестным для злоумышленника



Шифр Шеннона

Шифр Шеннона - пара функций $E = (E, D)$, таких что:

- (1) Функция E (**функция зашифрования**) принимает на вход ключ k и сообщение m (называемой открытым текстом, РТ) и даёт на выходе шифртекст c (СТ), такой что

$$c = E(k, m).$$

Говорят, что c есть **зашифрование** m на ключе k .

- (2) Функция D (**функция расшифрования**) принимает на вход ключ k и шифртекст c и даёт на выходе сообщение m , такое что

$$m = D(k, c)$$

Говорят, что m это **расшифрование** c на ключе k .

Шифр Шеннона

- (3) Функция D обращает функцию E (**свойство корректности**):
$$\forall k, \forall m \ D(k, E(k, m)) = m.$$

Пусть K – **множество ключей**, M – **множество сообщений**, C – **множество шифртекстов**.

Тогда шифром Шеннона, определённым над (K, M, C) называют пару функций $E = (E, D)$:

$$\begin{aligned} E: K \times M &\rightarrow C, \\ D: K \times C &\rightarrow M, \end{aligned}$$

для которых выполняются свойства (1) – (3).

Пример: Одноразовый блокнот

Пусть $E = (E, D)$ – **шифр Шеннона**, для которого $K = M = C = \{0,1\}^L$, где L – фиксированный параметр.

Для ключа $k \in K$ и сообщения $m \in M$ функция **зашифрования** определена как:

$$E(k, m) = k \oplus m.$$

Для ключа $k \in K$ и шифртекста $c \in C$ функция **расшифрования** определена как:

$$D(k, c) = k \oplus c.$$

\oplus - побитное сложение по модулю 2 (XOR).

Корректность: $D(k, E(k, m)) = D(k, k \oplus m) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = 0^L \oplus m = m.$

Пример: Аддитивный одноразовый блокнот

Пусть $E = (E, D)$ – **шифр Шеннона**, для которого $K = M = C = \{0, \dots, n - 1\}^L$, где n, L – фиксированные параметры.

Для ключа $k \in K$ и сообщения $m \in M$ функция **зашифрования** определена как:

$$E(k, m) = (m + k) \bmod n, \text{ по координатам}$$

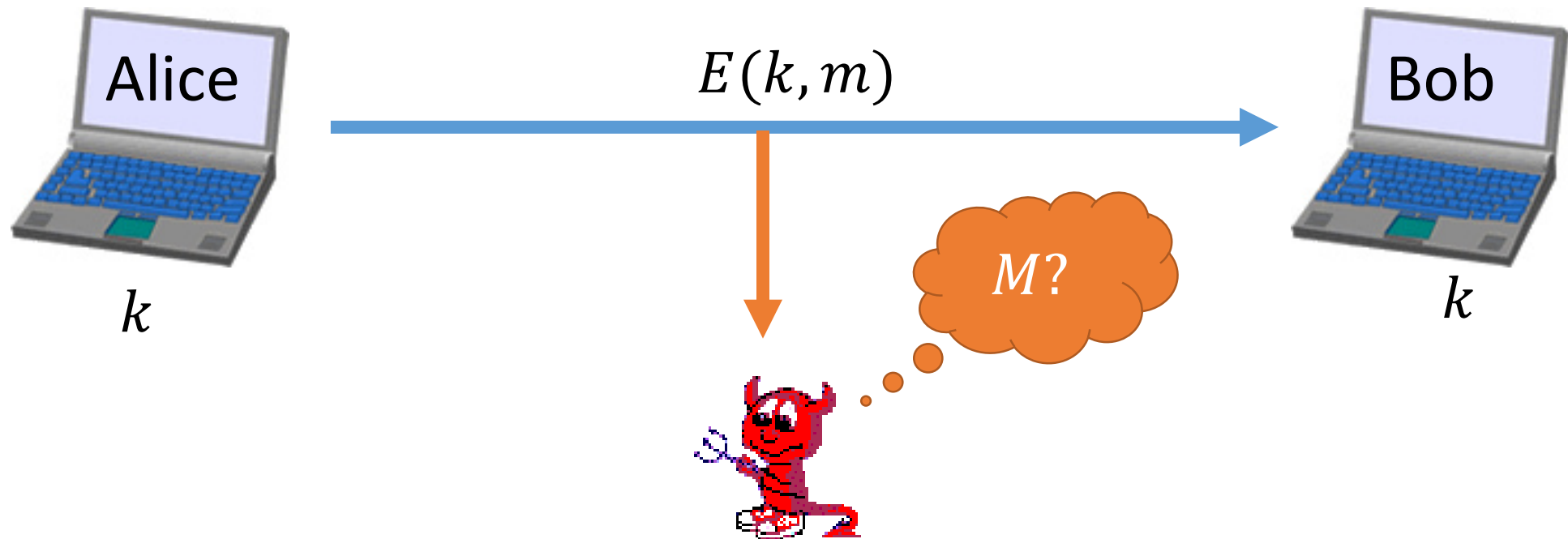
Для ключа $k \in K$ и шифртекста $c \in C$ функция **расшифрования** определена как:

$$D(k, c) = (c - k) \bmod n, \text{ по координатам}$$

Корректность: $D(k, E(k, m)) = D(k, m + k) = (m + k) - k = m.$

Цель шифра Шеннона

- Цель шифра Шеннона – обеспечение **секретности** передаваемых сообщений по открытому каналу
- Для обеспечения секретности необходим общий секретный ключ $k \in K$, неизвестный для злоумышленника



Одноразовый блокнот – абсолютно стойкий шифр

Теорема 1.2. Пусть $E = (E, D)$ - одноразовый блокнот при $K = M = C = \{0,1\}^L$ для параметра L . Тогда E – абсолютно стойкий шифр.

Следствия абсолютной стойкости – невозможны атаки, лучше чем атаки прямым перебором ключевого множества.

Сложность атаки - 2^L

Плохие новости

Теорема 1.7 (Шеннона). Пусть $E = (E, D)$ шифр Шеннона на (K, M, C) . Если E – абсолютно стойкий, то

- $|K| \geq |M|$
- $H(\mathbf{k}) \geq H(\mathbf{m}), \mathbf{k} \in_R K, \mathbf{m} \in_R M$

Простое объяснение – невозможно получить равномерно распределённую случайную величину длины m , используя детерминированный алгоритм над равномерно распределённой случайной величиной длины $n < m$.

Иными словами, для шифрования 1 Gb данных **любым** абсолютно стойким шифром потребуется ключ размера как минимум 1 Gb.

Идея одноразового блокнота

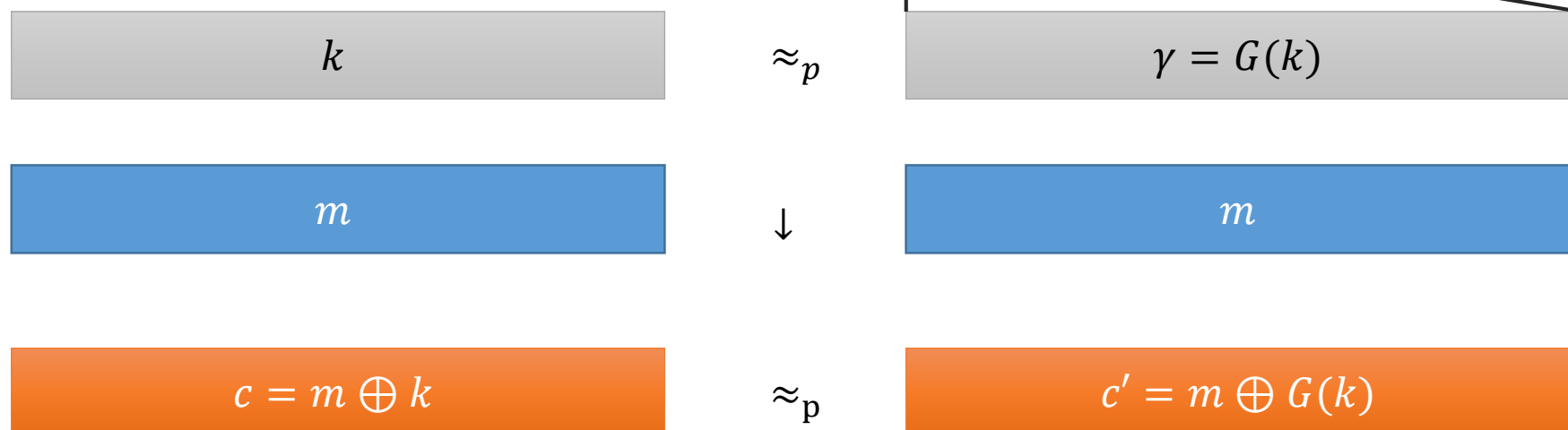
Одноразовый блокнот – сложение (побитное) случайного равновероятного вектора ключа с вектором открытого текста, для получения шифртекста.

Проблема (Теорема Шеннона) – длина (энтропия) ключа должна быть больше или равна длине сообщения.

Основная идея – заменить случайный длинный вектор ключа на «псевдослучайную» последовательность, называемую гаммой.

Идея одноразового блокнота

Заменяем использование случайного ключа k псевдослучайной последовательностью γ . Если последовательность «неотличима» от случайной равновероятной, то шифртекст c' неотличим от шифртекста в одноразовом блокноте.



Поточный шифр

Эффективно вычислимая функция $G: S \rightarrow R$ называется **псевдослучайным генератором** на (S, R) **PRG**.

Шифр $E = (E, D)$ с параметрами (l, L) на (K, M, C) : $K = \{0,1\}^l, M = C = \{0,1\}^L$, называется **поточным шифром**, если

$$E(k, m) = G(k) \oplus m,$$

где $G: \{0,1\}^l \rightarrow \{0,1\}^L$ - псевдослучайный генератор.

Аналогично можно ввести Поточный шифр по произвольному модулю.

Стойкость поточного шифра сводится к «качеству» псевдослучайной последовательности $\gamma = G(k)$

Блочный шифр

Блочный шифр – детерминированный шифр $E = (E, D)$ определённый на (K, X) : $E: K \times X \rightarrow X$.

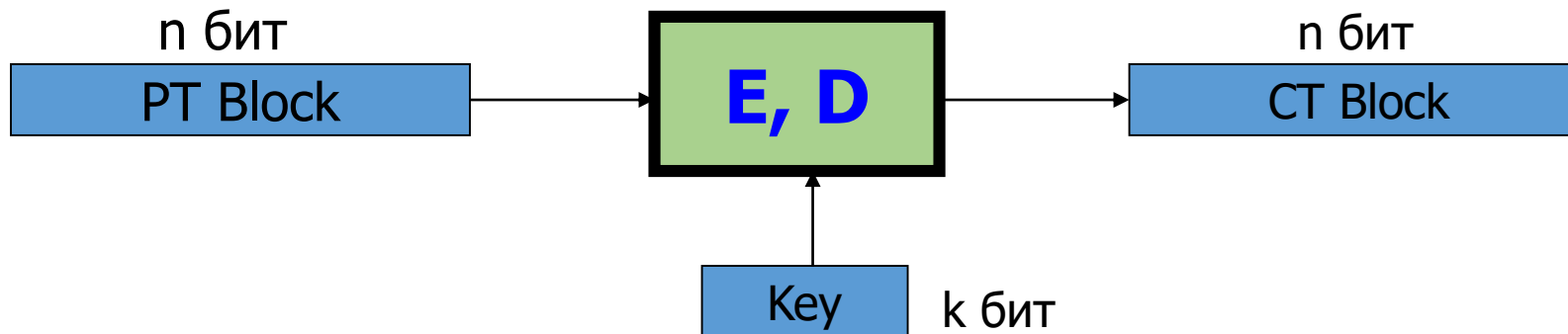
$x \in X$ – блок данных, X – множество блоков, K – множество ключей блочного шифра.

Для ключа $k \in K$ определим функцию $f_k: X \rightarrow X$: $f_k = E(k, *)$. $f_k^{-1}: X \rightarrow X$: $f_k^{-1} = D(k, *)$.

Из свойства корректности имеем f_k, f_k^{-1} – подстановки на множестве X , $f_k f_k^{-1} = e$, где e – тождественная подстановка на X .

Блочный шифр

- Блочные шифры являются основным криптографическим примитивом для построения симметричных криптосистем.
- Могут быть использованы для как схем шифрования (в схемах шифрования), так и для обеспечения аутентичности (в кодах аутентичности сообщений).



PRP

Пусть функция $E: K \times X \rightarrow X$ определена на (K, X) .

Тогда E – **псевдослучайная подстановка (PRP)**, если

- Существует эффективный алгоритм вычисляющий $E(k, x)$. $k \in K, x \in X$
- Функция $f_k = E(k, *)$ – подстановка.

PRP стойкая, если $k \in_R K, E(k, m) \approx_p r, r \in_R X$

(где \approx_p - вычислительная неотличимость, \in_R - выбор случайного элемента множества)

Модель стойкого блочного шифра

- Предполагается, что стойкий блочный шифр задаёт стойкую PRP
- Иными словами при случайном ключе, мы ожидаем, что выход зашифрования произвольного блока блочным шифром будет вычислительно неотличим от случайного блока, выбранного случайно равномерно.

Использование блочных шифров

Пусть $E = (E, D)$ – блочный шифр на (K, X) .

Можем ли мы использовать блочный шифр для построения шифров для сообщений произвольной длины?

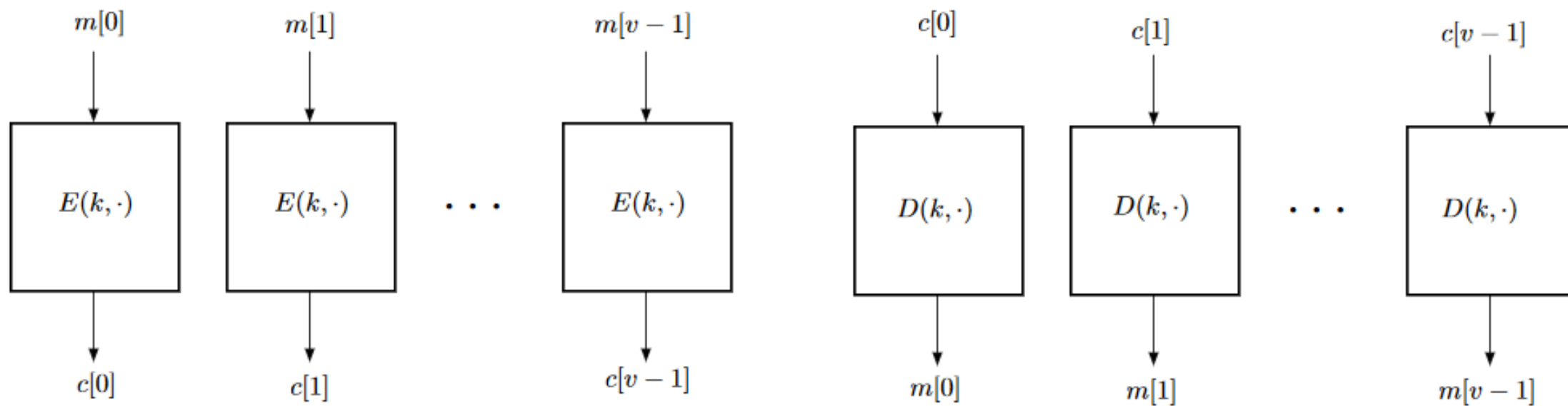
Сможем, будем использовать режимы шифрования блочных шифров, определяющие шифрование сообщений произвольной длины, на основе блочных шифров.

ЕСВ

Пусть $E = (E, D)$ – блочный шифр на (K, X) . Для полиномиально ограниченной величины $l \geq 1$ определим шифр $E' = (E', D')$ на $(K, X^{\leq l}, X^{\leq l})$ следующим образом:

- Для $k \in K, m \in X^{\leq l}, v = |m|$ определим
$$E'(k, m) = (E(k, m[0]), \dots, E(k, m[v - 1])).$$
- Для $k \in K, c \in X^{\leq l}, v = |c|$ определим
$$D'(k, c) = (D(k, c[0]), \dots, D(k, c[v - 1])).$$

ECB



Зашифрование

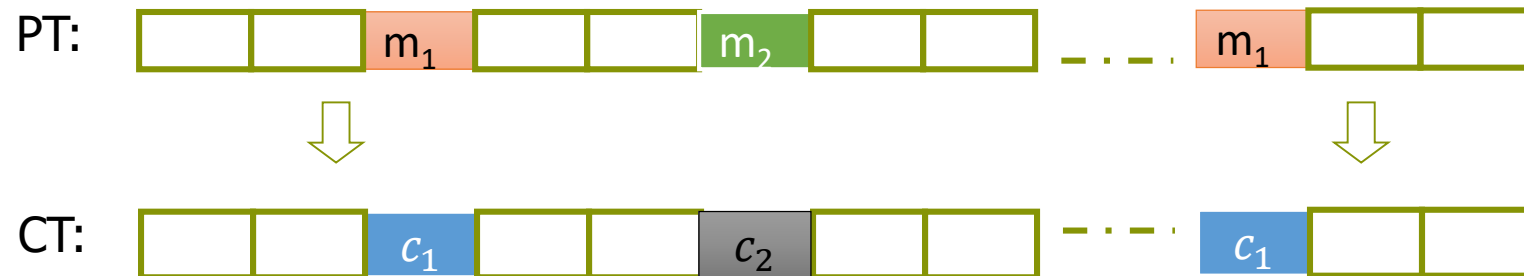
Расшифрование

Стойкость ЕСВ

- Стойкий блочный шифр в режиме ЕСВ –стойкий для
 - Сообщений, состоящих из уникальных, **попарно различных блоков** (например есть открытый текст – случайных ключ), не повторяющихся во время жизни ключа шифрования
 - Любых коротких, уникальных сообщений, длиной в один блок, не повторяющихся во время жизни ключа
- Что для произвольных сообщений произвольной длины?

Стойкость ЕСВ

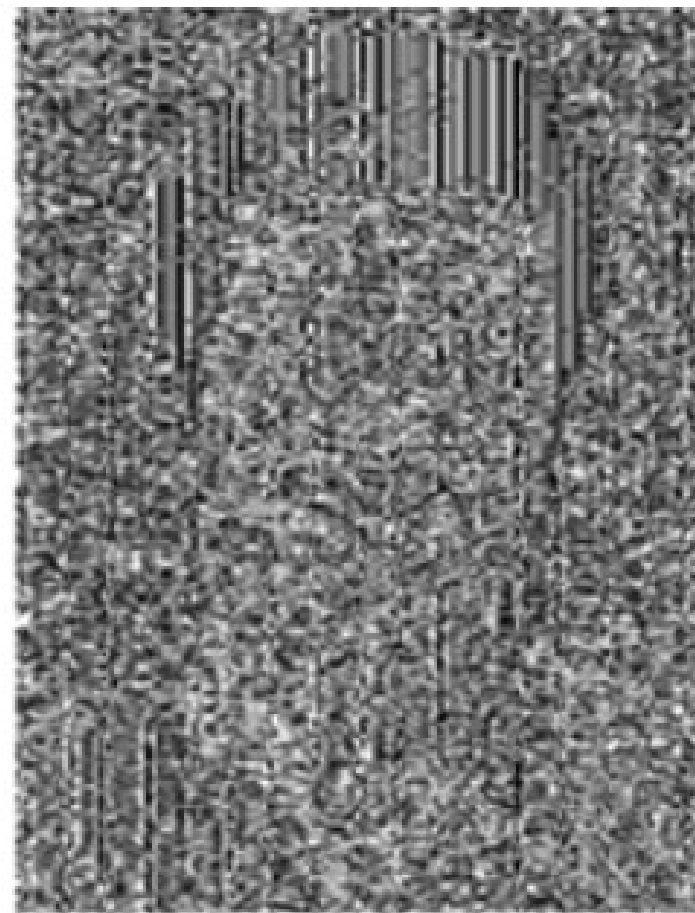
Зашифрование в режиме ЕСВ происходит детерминировано и поблочно, как следствие одинаковые блоки имеют одинаковый шифртекст.



Стойкость ЕСВ



(a) plaintext

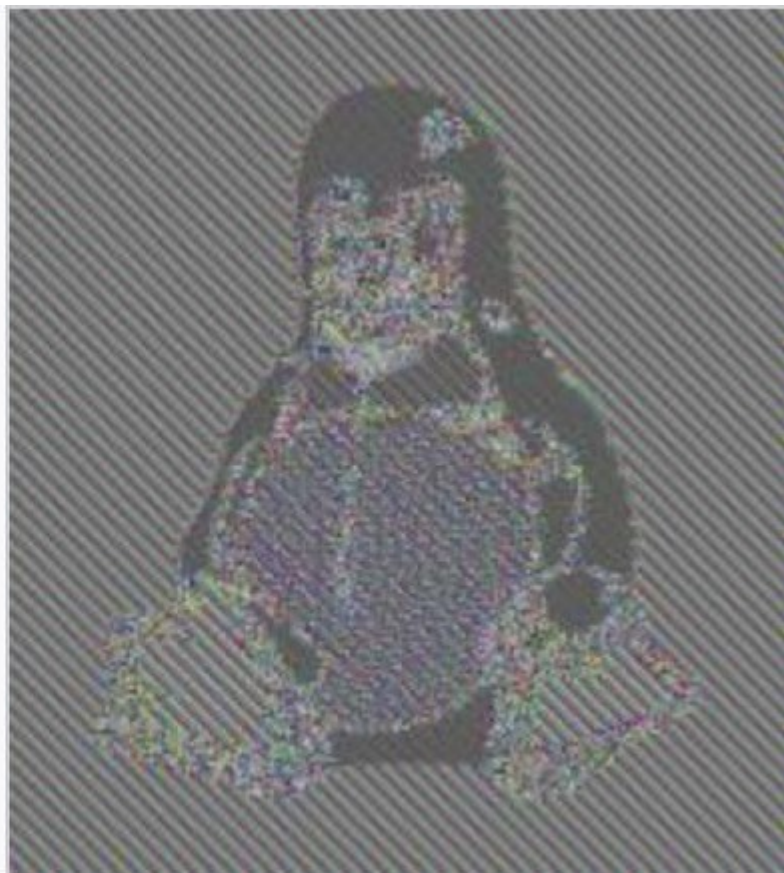


(b) plaintext encrypted in ECB mode
using AES

Стойкость ECB



Original image



Encrypted using ECB mode

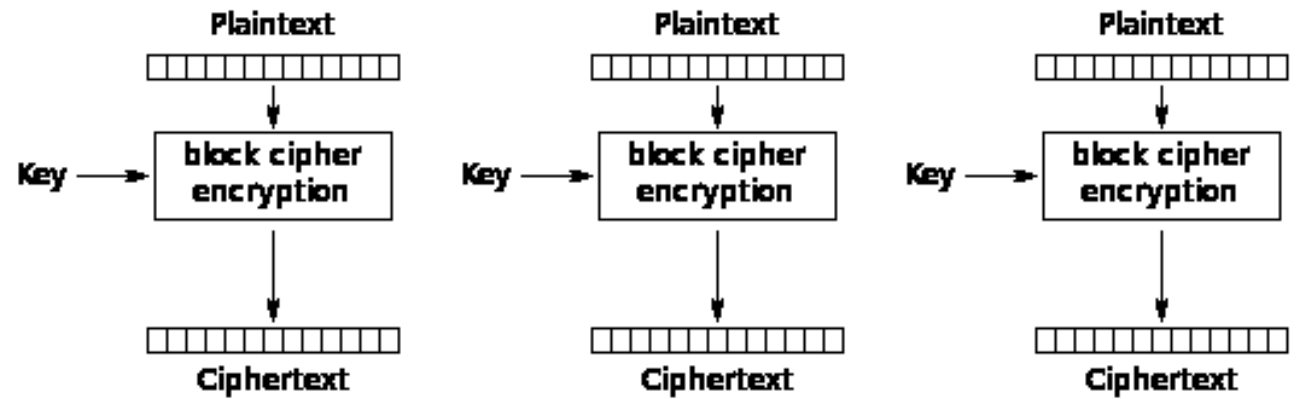
Вопросы для достижения дзена в режимах шифрования

На сколько битов, в каких блоках и каким образом влияет

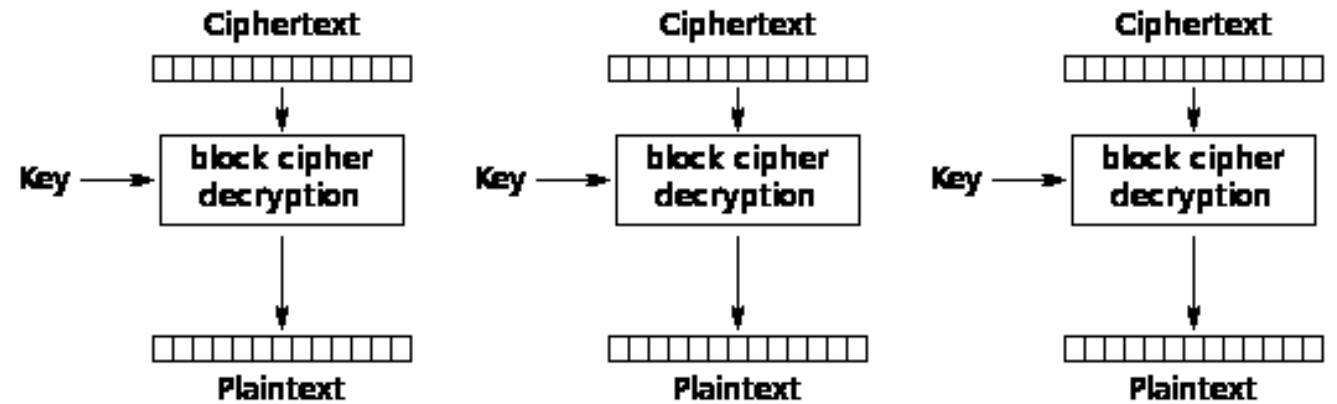
- Изменение одного бита открытого текста на шифртекст
- Изменение одного бита шифртекста на расшифрованный открытый текст

Можно ли контролируемо изменить определённый бит расшифрованного открытого текста, изменив биты шифртекста, как?

ECB

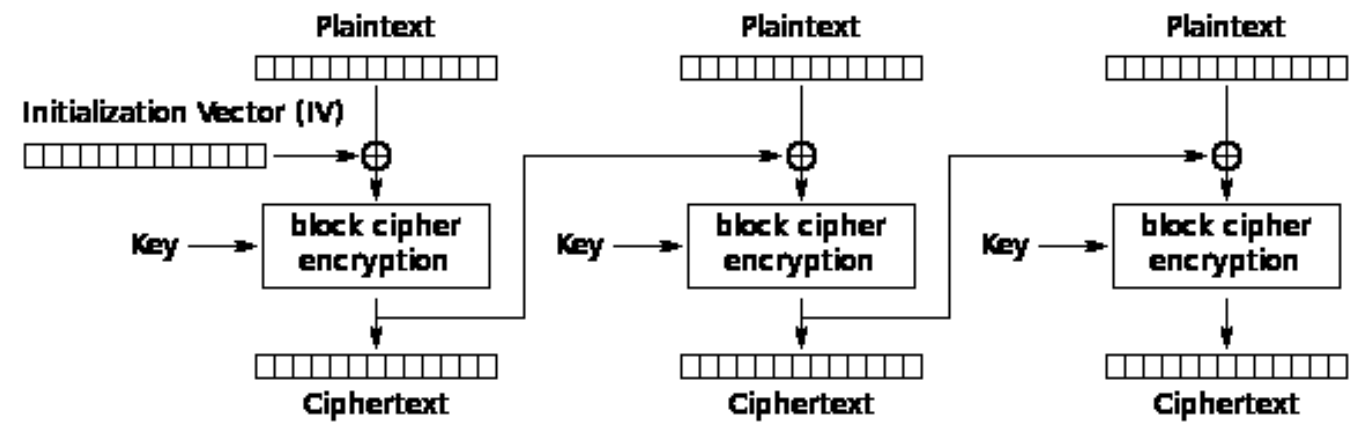


Electronic Codebook (ECB) mode encryption

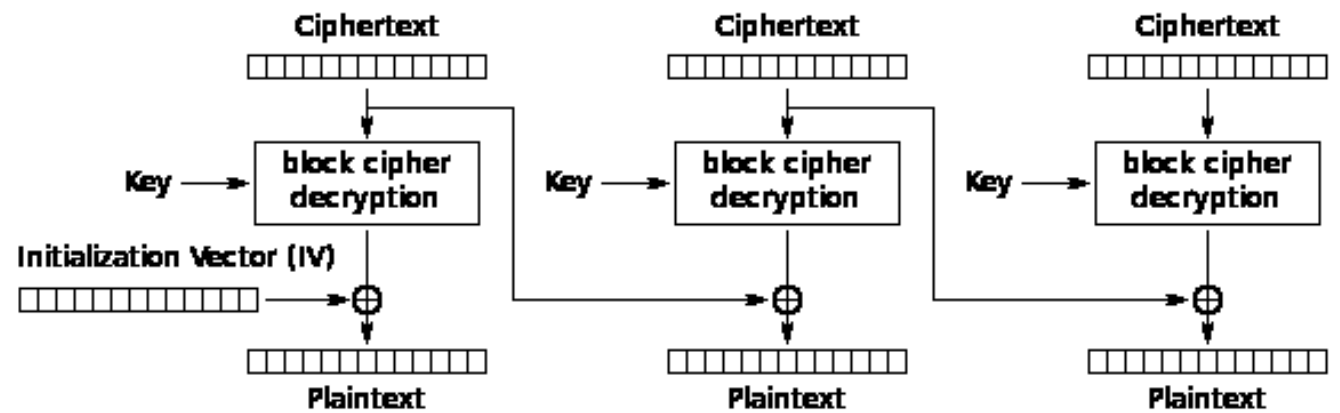


Electronic Codebook (ECB) mode decryption

CBC

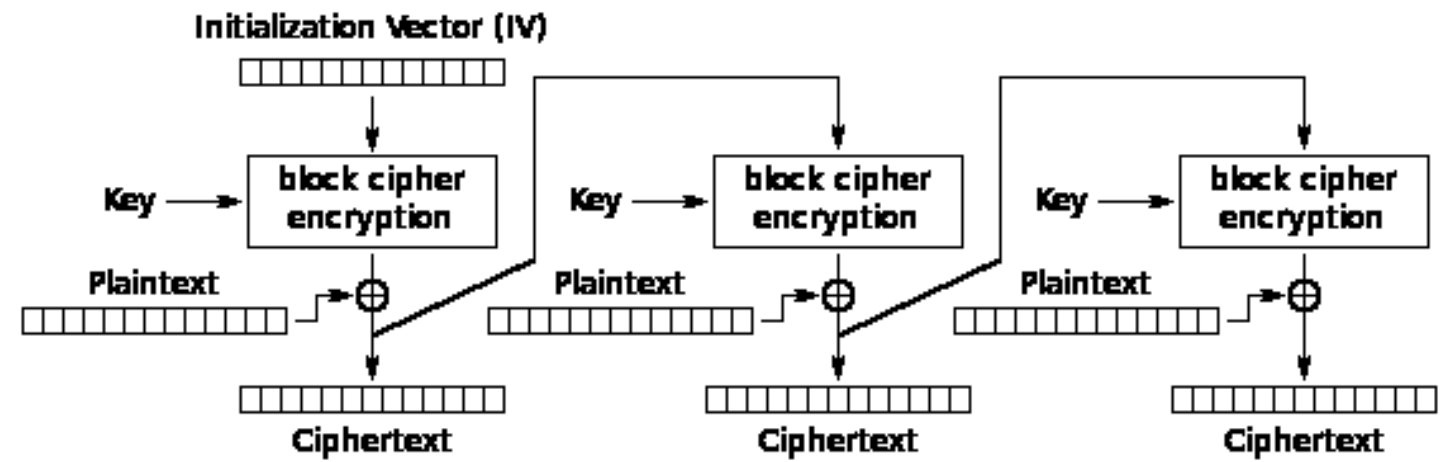


Cipher Block Chaining (CBC) mode encryption

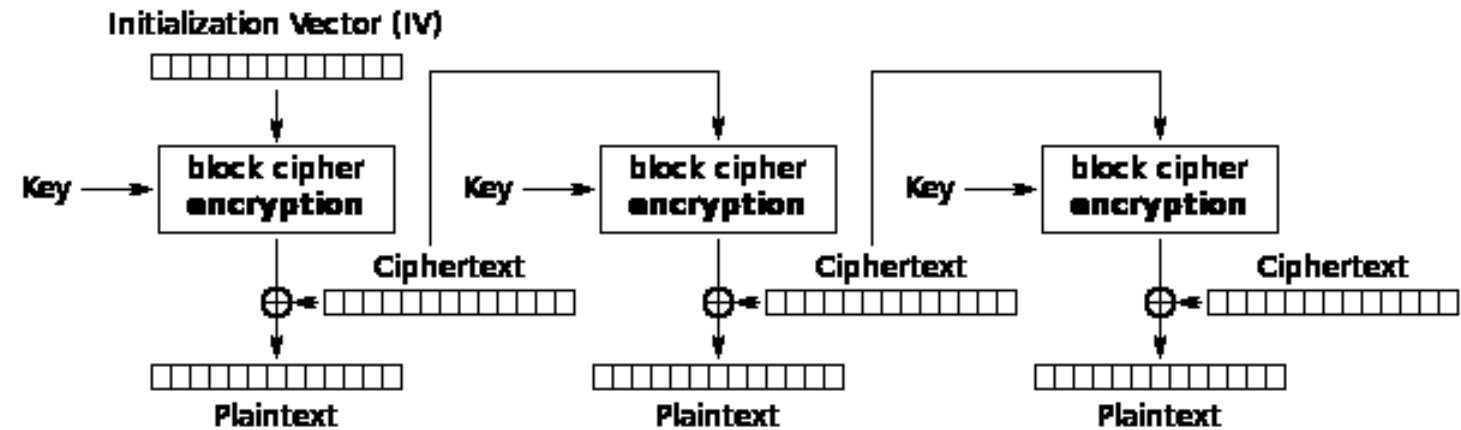


Cipher Block Chaining (CBC) mode decryption

CFB

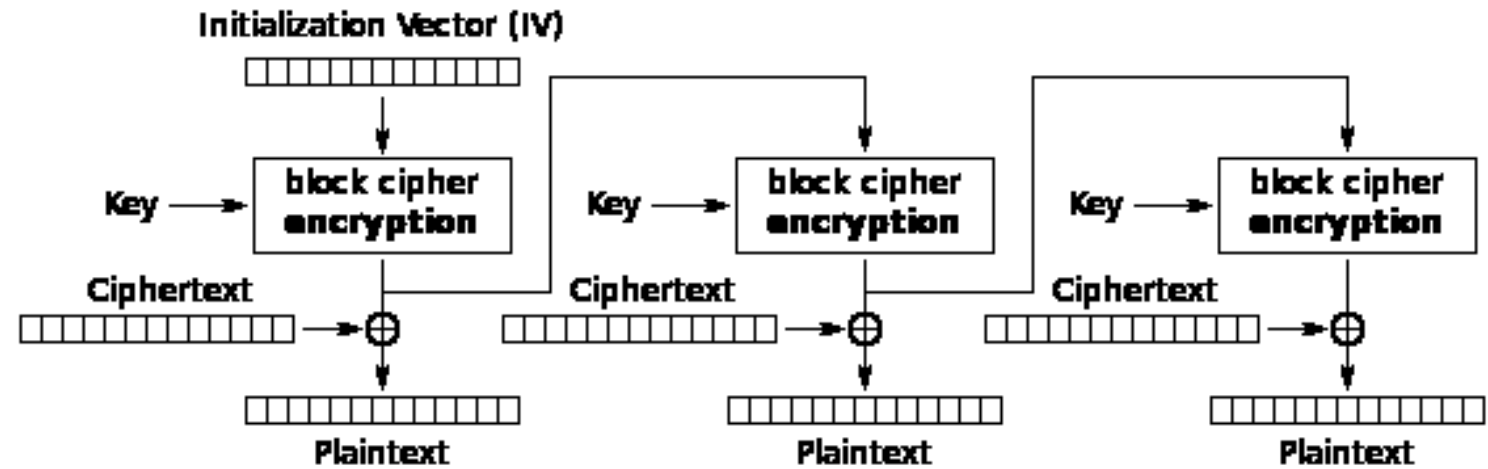
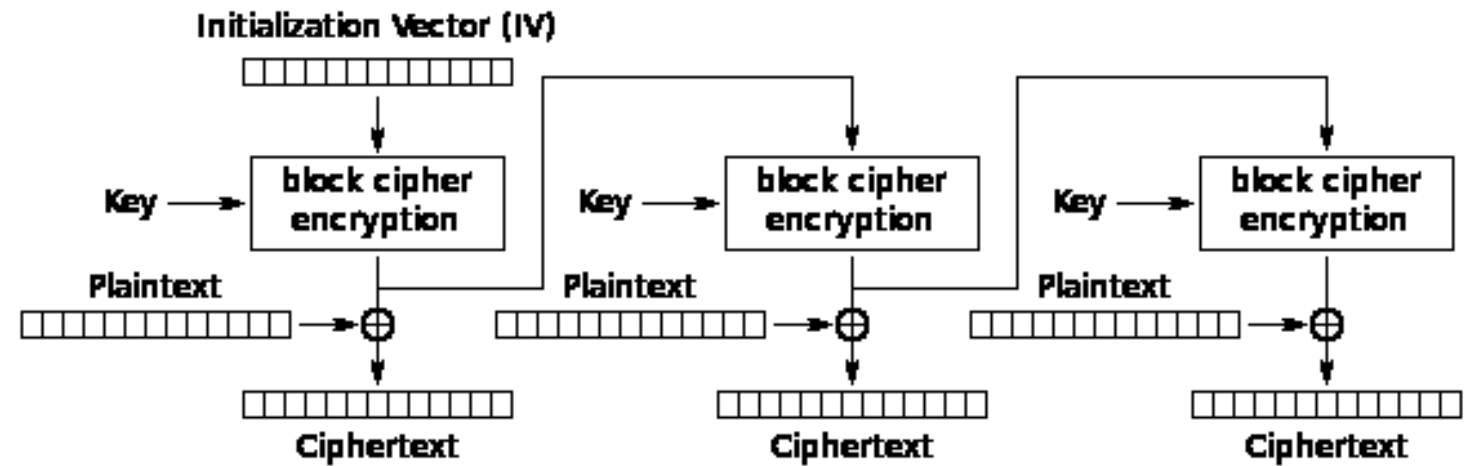


Cipher Feedback (CFB) mode encryption

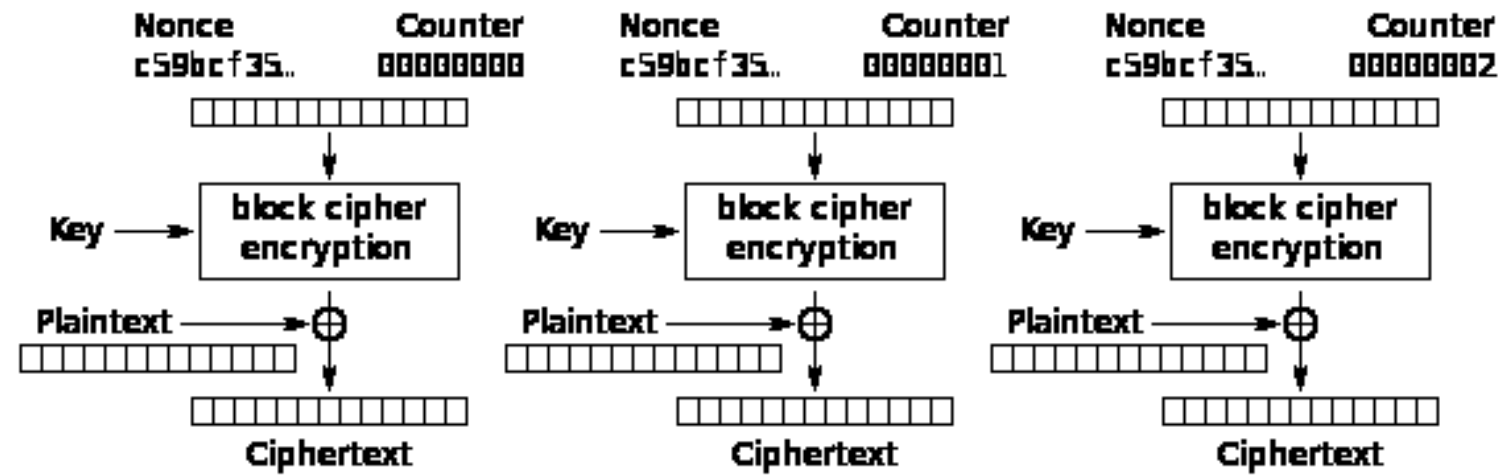


Cipher Feedback (CFB) mode decryption

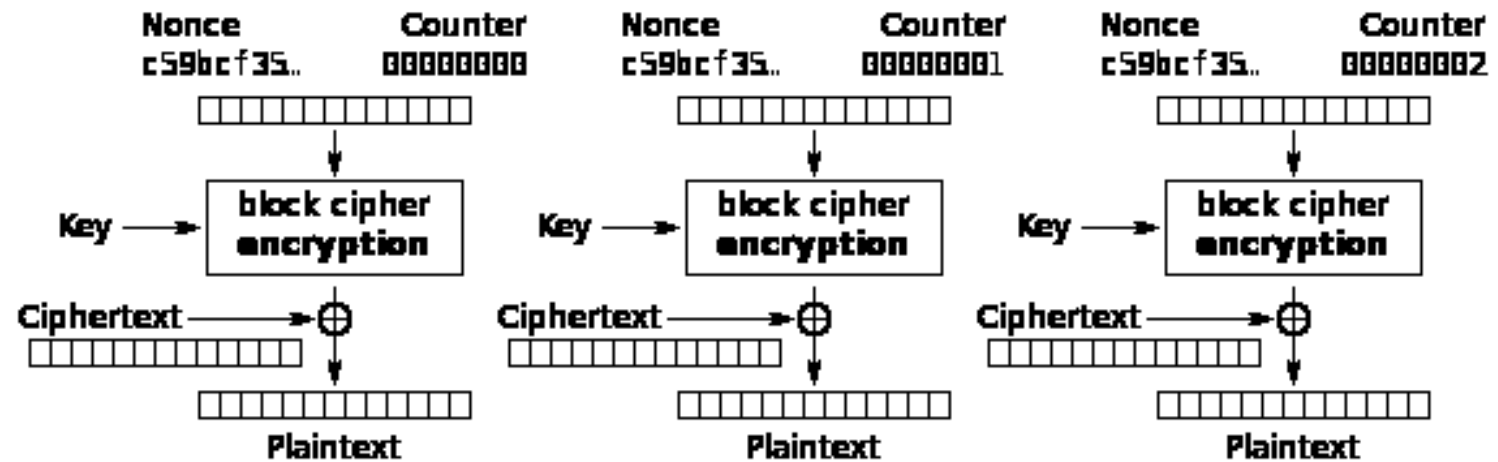
OFB



CTR



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Безопасное программирование

- Проверка входных значений (длины входов, типы, корректность)
 - Должна производиться по возможности в начале функции
 - Параноидальные проверки
- Криптографические методы должны быть вынесены в отдельные функции и модули
- Все внешние криптографические интерфейсы работают только с массивом байт
- Не использование «магических чисел» – все константы должны быть определены
- Не использовать «алгоритмы по умолчанию», т.е. явно задавать алгоритмы шифрования через параметры

Проверка входных данных

```
def decrypt(key, data, cipher_suite):
    if len(key) != AES_KEY_SIZE:
        raise Exception($'invalid key size, expecting {AES_KEY_SIZE}')
    if len(data) < NONCE_SIZE:
        raise Exception('invalid ciphertext length')
    if cipher_suite = AES_CBC_WITH_CBC_MAC:
        return aes.cbc.decrypt(key, data)
    else if cipher_suite = AES_CTR_WITH_CBC_MAC :
        return aes.ctr.decrypt(key, data)
    else:
        raise Exception($'cipher_suite {cipher_suite} is not supported')
```

Отдельные функции, работа только с массивом байт

```
def generate_aes_key():  
    return Crypto.random.getBytes(AES_KEY_SIZE)  
  
-----  
  
def encrypt_user_input(user_string):  
    user_bytes = Encode.utf8.getBytes(user_string)  
    key = generateAesKey()  
    encrypted = encrypt(key, user_bytes , AES_CBC_WITH_CBC_MAC)  
    encrypted_hex = Converter.toHex(encrypted)  
    key_string = Converter.toHex(key)  
    return key_string, encrypted_hex
```

