

# Современная криптография

ХОТЯ ЗДЕСЬ СКОРЕЕ ПРО КРИПТОАНАЛИЗ

# Типы шифрования

- Симметричное
- Асимметричное

# Симметричное шифрование

- Ключ шифрования равен ключу расшифровки
- Самые древние виды шифрования относятся к этому типу
- Самые известные представители – AES, DES, RC4

Для секретного ключа  $K$

$$\text{Encrypt}(M, K) = C$$

$$\text{Decrypt}(C, K) = M$$

# Применения симметричного шифрования

- Быстрое шифрование данных
- Возможность потокового шифрования

# Асимметричное шифрование

- Ключ шифрования НЕ равен ключу расшифровки
- Изобретен 40 лет назад
- Самые известные представители – RSA, ECC

Для секретной пары ключей  $K$  и  $K'$

$$\text{Encrypt}(M, K) = C$$

$$\text{Decrypt}(C, K') = M$$

$$K' \neq K$$

# Применения асимметричного шифрования

- Обмен ключами симметричного шифрования
- Проверка подлинности

# Блочные шифры

- Шифруют блоками постоянного размера
- Все асимметричные шифры относятся к блочным
- Популярные симметричные – тоже (AES, DES)
- Идеальный блочный шифр является собой шифр подстановки, заменяющий один блок текста на другой, причем взаимосвязь полностью задается ключом, однако восстановить по ней ключ невозможно

# Потоковые шифры

- Генерируют ключевой поток
- Позволяют шифровать буквально по одному биту
- Шифрование часто совпадает с расшифровкой

Операция шифрования обычно выглядит как

???????? – ключевой поток

⊕ – XOR (побитовое сложение по модулю 2)

MESSAGE – исходное сообщение

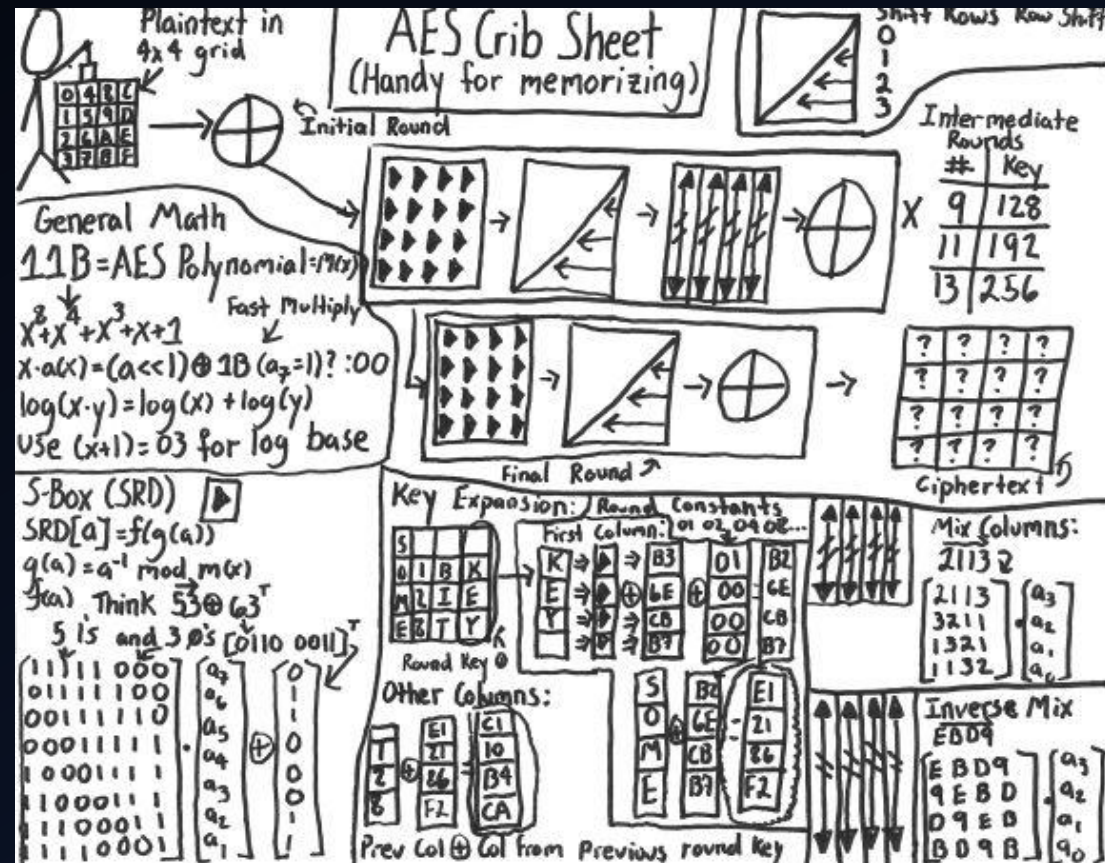
---

CIPHERTEXT – зашифрованное сообщение



# AES, DES и компания

Какая-то ацкая муть, является предметом изучения математиков, можно наблюдать на NSUCRYPTO, например.



# Режим связывания блоков

- Атаки, как правило, вообще можно осуществлять на любой блочный шифр, даже идеальный
- И главное, совершенно не нужно разбираться как оно работает
- Создан чтобы избежать поблочного анализа

Очевидно, что для каждого блока  $X$  для таблицы подстановки  $S$

$S(X_1) == S(X_2)$  тогда и только тогда когда  $X_1 == X_2$

Это беда

# Режим связывания блоков



PLAIN



ECB



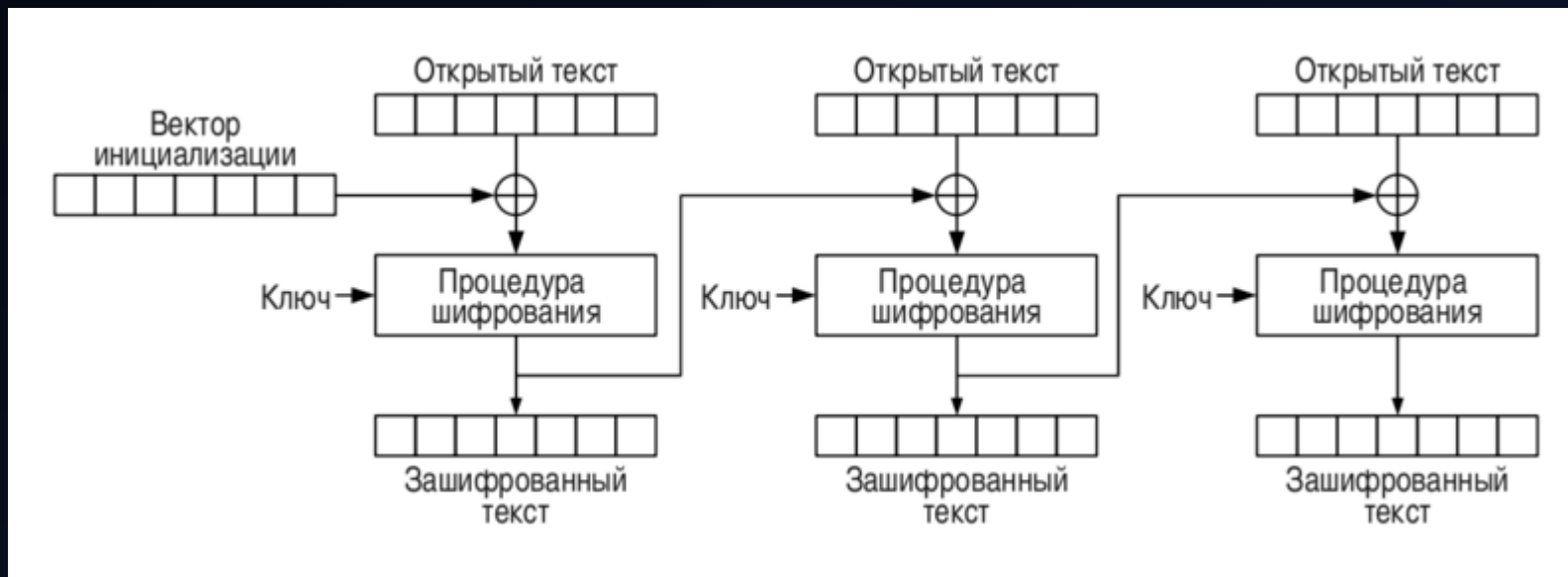
CBC

# Режим связывания блоков

В режиме CBC для шифрования:

Шифроблок  $C_i = E(P_i \oplus C_{i-1})$ ,  $C_0 = IV$  (случайный вектор инициализации)

Расшифровка  $P_i = C_{i-1} \oplus D(C_i)$



## Режим связывания блоков

Напрямую из процедуры расшифровки получаем

Если  $P_i = C_{i-1} \oplus D(C_i)$  то заменяя  $C_{i-1}$  мы заменяем  $P_i$

То есть если нам известен некоторый текст скрытый за шифрованием, мы можем создать ЛЮБОЙ текст такой же длины как  $P_{evil} = C_{evil} \oplus D(C_i)$ ,  $C_{evil} = P_i \oplus C_{i-1} \oplus EVILTEXT$

Вывод:

Симметричная криптография хреново подходит для проверки подлинности (еще бы)

# RSA

- Няшка, все любят RSA
- Старый, но не бесполезный
- Опирается на задачу разложения числа на множители

Выглядит как  $M^e = C \pmod{N}$ ,  $C^d = M \pmod{N}$   
 $e * d = 1 \pmod{\phi(N)}$

$\phi$  – функция Эйлера, для  $N = p * q$ ,  $p$  и  $q$  – простые имеет вид  
 $\phi(N) = (p - 1) * (q - 1)$

# Как наскосячить с модулем RSA

- Использовать небольшое  $N$  (привет NSUCRYPTO)
- Использовать  $N$  такое где  $|p-q|$  мало, тогда применяется алгоритм факторизации Ферма
- Использовать  $N$  где  $p-1$  имеет малые делители (*гладкое*) – применяется  $p-1$  метод Полларда
- Использовать  $N$  где  $p+1$  имеет малые делители – метод  $p+1$  Уильямса
- Слить информацию о части бит  $p$  или  $q$  – Coppersmith method или “factoring with a hint” в CrypTool

## Как накосячить RSA (ещё)

- Использовать публичную экспоненту  $e = 3$  (привет МНСК).  
Очевидно если  $M^e = C$  то если  $M < N^{1/3}$  то  $M = \sqrt[3]{C}$
- Шифровать похожие сообщения одним ключом (т.е.  $|M1 - M2|$  мало) - Franklin-Reiter related-message attack
- Шифровать одно и то же сообщение разными ключами.  
Если  $e$  меньше числа вариантов сообщения, то по китайской теореме об остатках по набору  $C_i$  и  $N_i$  можно восстановить  $M^e$   
Как вы догадались, потом  $M = \sqrt[e]{M^e}$



## Как накосячить RSA (и ещё)

- Не проверять подлинность  $N$  при обмене ключами. Тогда можно осуществить атаку “человек посередине” подменив  $N$  своим и расшифровывая-зашифровывая данные прозрачно для двух собеседников. Но это в CTF уже экзотика.

# Секундочку, но ведь RSA шифрует числа?

Ага, есть такое дело.

Чтобы перевести строку в число достаточно просто взять составляющие ее байты и посчитать их байтами числа.

В языке Python это делается как  
`int('строка'.encode('hex'),16)`

В обратную сторону  
`hex(num)[2:].replace('L','').decode('hex')`

# Полезные инструменты

- Python, без него здесь почти никак
- Libnum, позволяет выполнять разные математические операции, близкие к STF. Или gmpy/gmpy2 (как более общее решение).
- Sagemath, для отчаянных. Обладает огромным списком возможностей, но странноват, большой и тормозит.
- Google + site:github.com для поиска готовых скриптов
- GIMP – позволяет открывать байты (расширение .data) как картинку, очень полезный инструмент для поверхностного анализа данных (энтропия “на глазок”)
- Xortool, позволяет выполнять операцию xor над файлами

Сервер с задачами

<http://dmz.n0n3m4.ru/tasks>

Вопросы? :|