



Основы криптографии

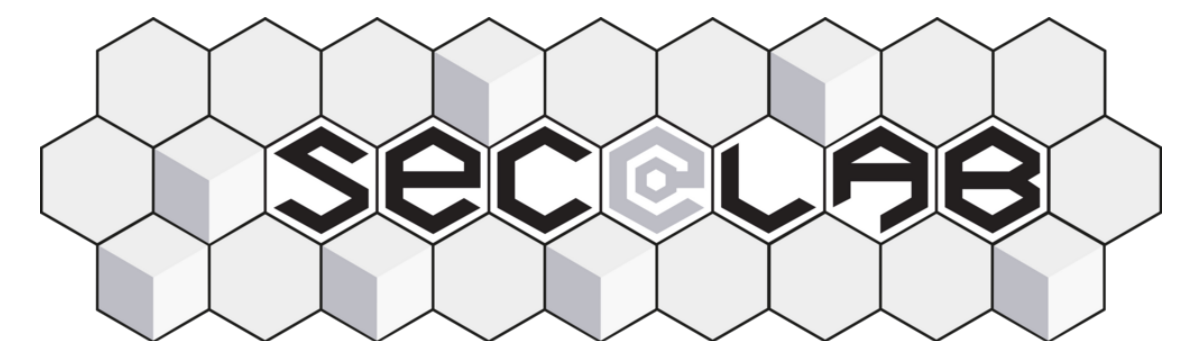


Артур Хашаев

arthur@khashaev.ru

МГУ им. М.В. Ломоносова

Лаборатория интеллектуальных систем кибербезопасности



Содержание

1. **Основные понятия** 👉
2. Переборные задачи
3. Односторонние функции
4. Хэширование
5. Проверка целостности
6. Шифрование

Свойства безопасности

- Конфиденциальность
- Целостность
- Подлинность
- Неотказуемость от авторства



Криптографические примитивы

Без ключа

- **Односторонняя функция**
 - Хэш-функция
- Генератор случайных последовательностей

На основе секретных значений

- Шифрование
- Код аутентификации
- Цифровая подпись
- Функция формирования ключа
- Генератор псевдослучайных последовательностей

Криптографические протоколы

- Протокол – алгоритм решения некоторой задачи $n \geq 2$ сторонами
- Криптографический протокол – протокол, в котором используются криптографические примитивы

Принцип Керкхоффа

- Правило разработки криптографических систем:
 - Надежность криптосистемы не должна быть основана на ее секретности
 - Система должна оставаться надежной, даже если все ее компоненты, кроме ключа, стали известны атакующему
- Иными словами, «враг знает систему»
- Антипринцип: «безопасность через неясность»

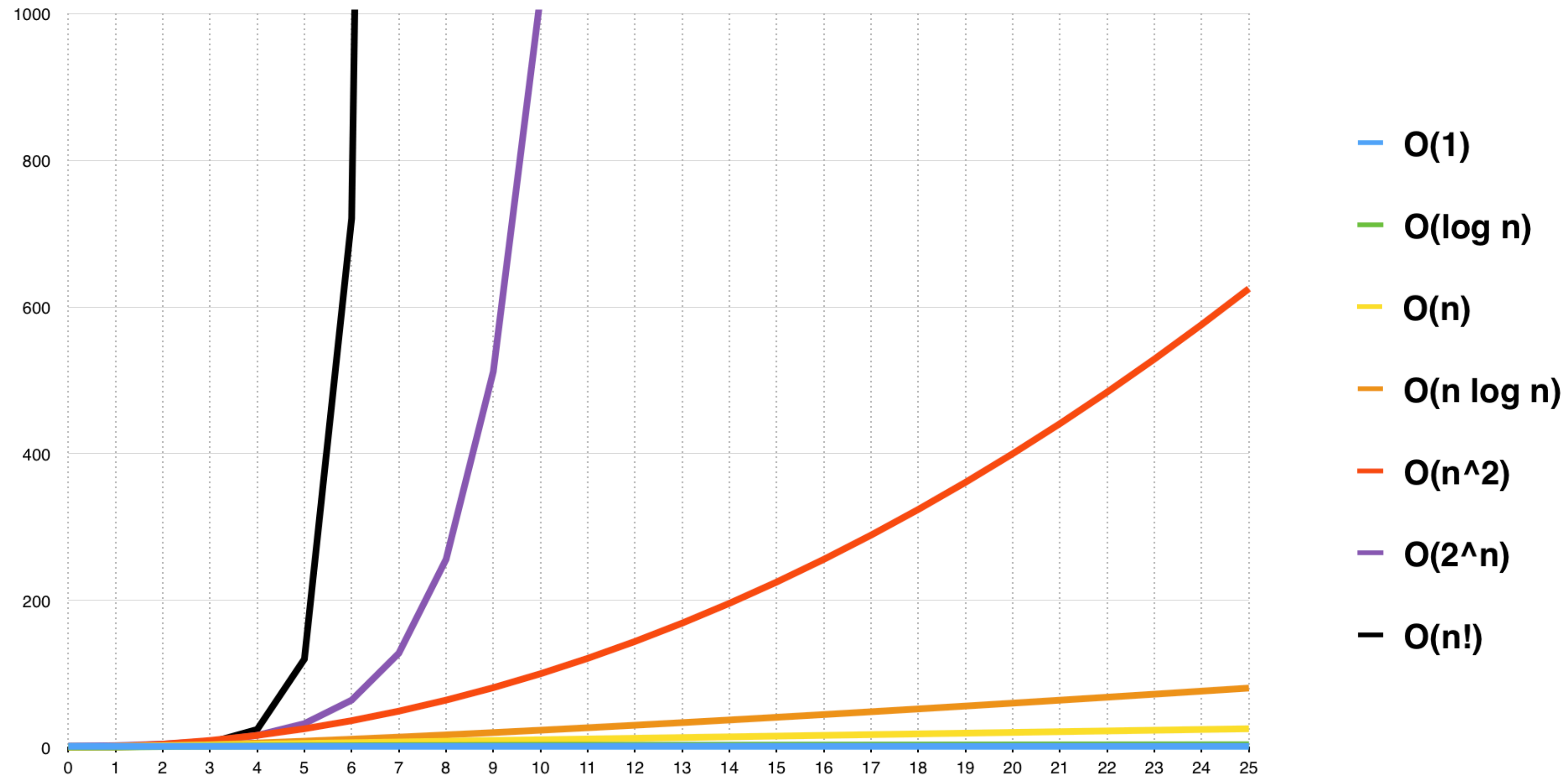
Содержание

1. Основные понятия
- 2. Переборные задачи** 🙌
3. Односторонние функции
4. Хэширование
5. Проверка целостности
6. Шифрование

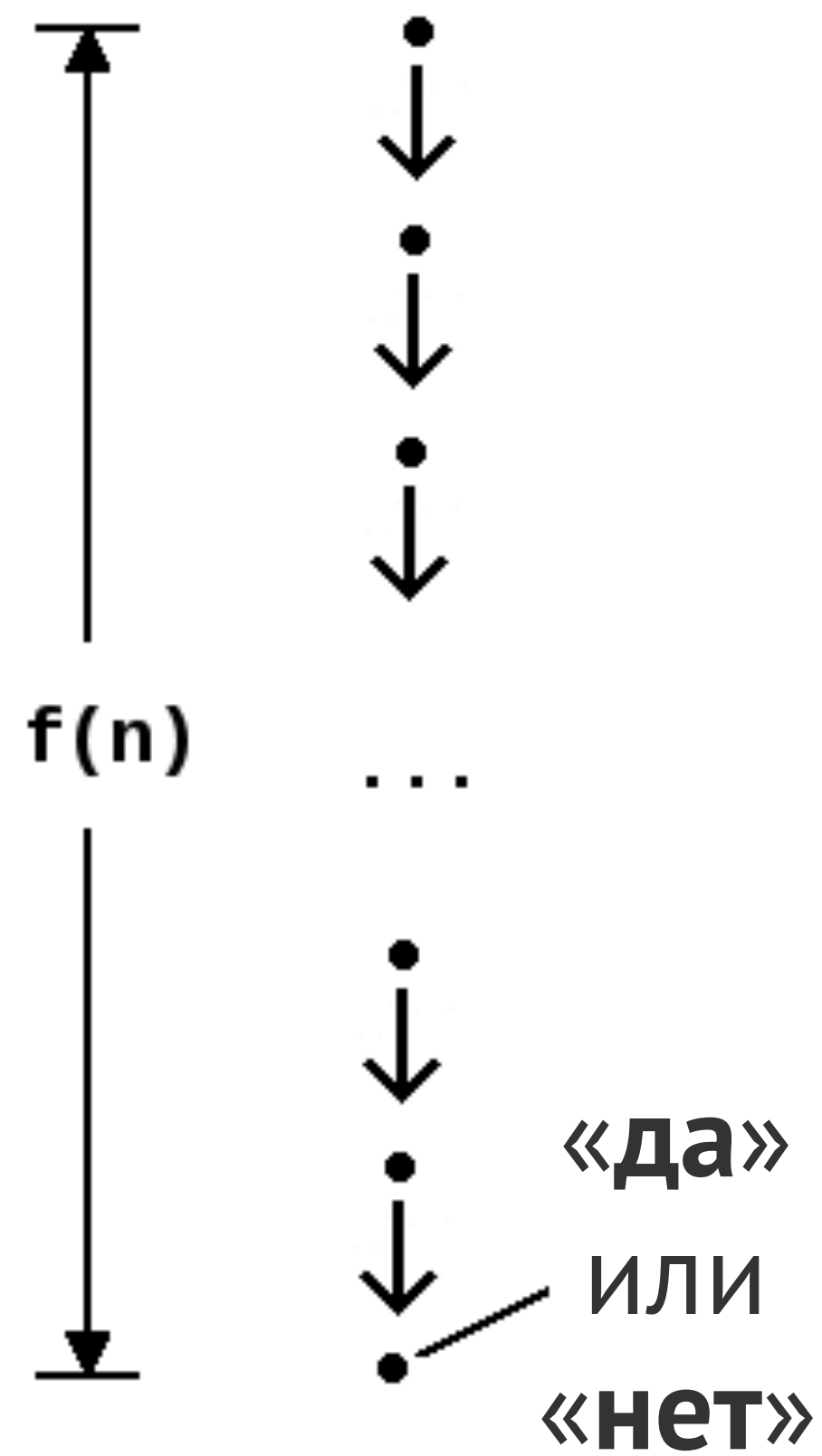
Проблема разрешимости

- Задача со входными данными, требующая ответа «да» или «нет»
- $f: X \rightarrow \{0, 1\}$
- Для любого входа $x \in X$ мы знаем размер входных данных $n = \|x\|$
- Машина Тьюринга – абстракция вычислителя, «компьютера»
- Нас интересуют алгоритмы вычисления f на машине Тьюринга

Временная сложность алгоритмов



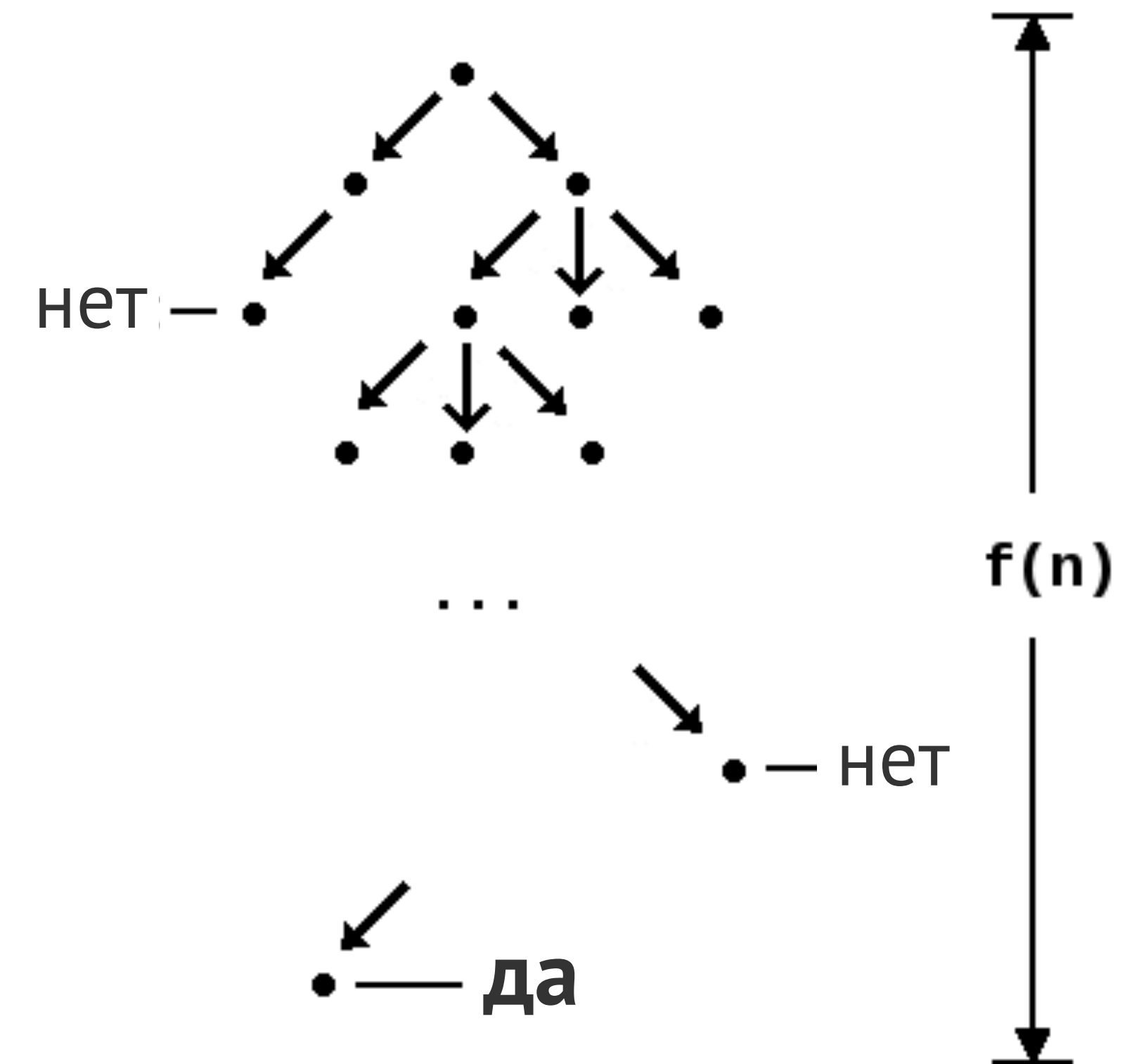
Класс P



P – класс проблем разрешимости, которые решаются за полиномиальное время на детерминированной машине Тьюринга.

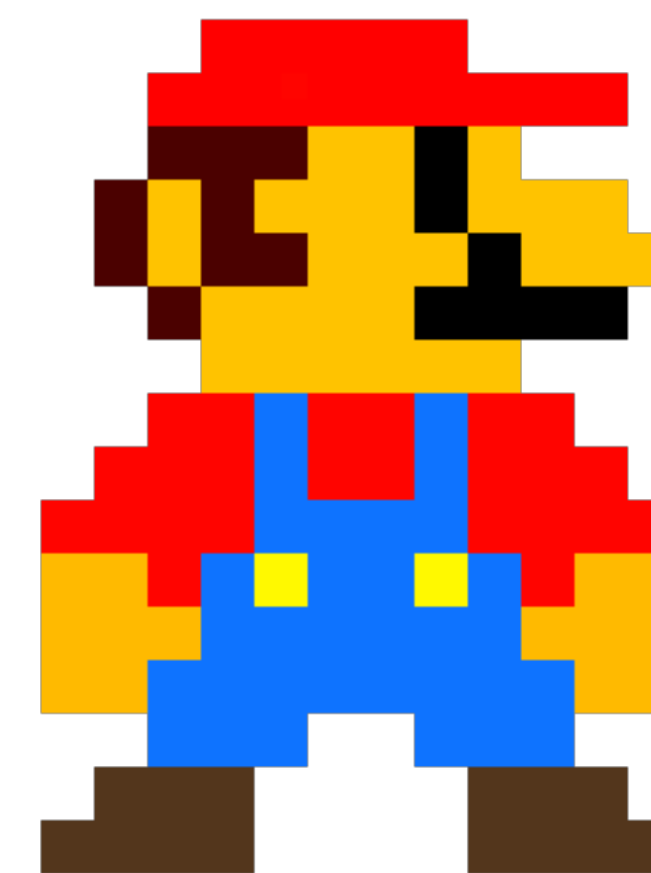
Класс NP

- **NP** – класс проблем разрешимости, которые решаются за полиномиальное время на недетерминированной машине Тьюринга
- Неформально говоря, НДМТ – много МТ, работающих параллельно:
 - Если хотя бы одна ответит «да», то ответ «**да**»
 - Ответ «**нет**», если все ответят «нет»



NP-полные задачи

- **NPC** – задачи из класса NP, к которым можно (за полиномиальное время) свести любую задачу из класса NP.
- Известные NP-полные задачи:
 - SAT – задача проверки выполнимости булевой формулы
 - Задача о рюкзаке
 - Сапер, тетрис, ...

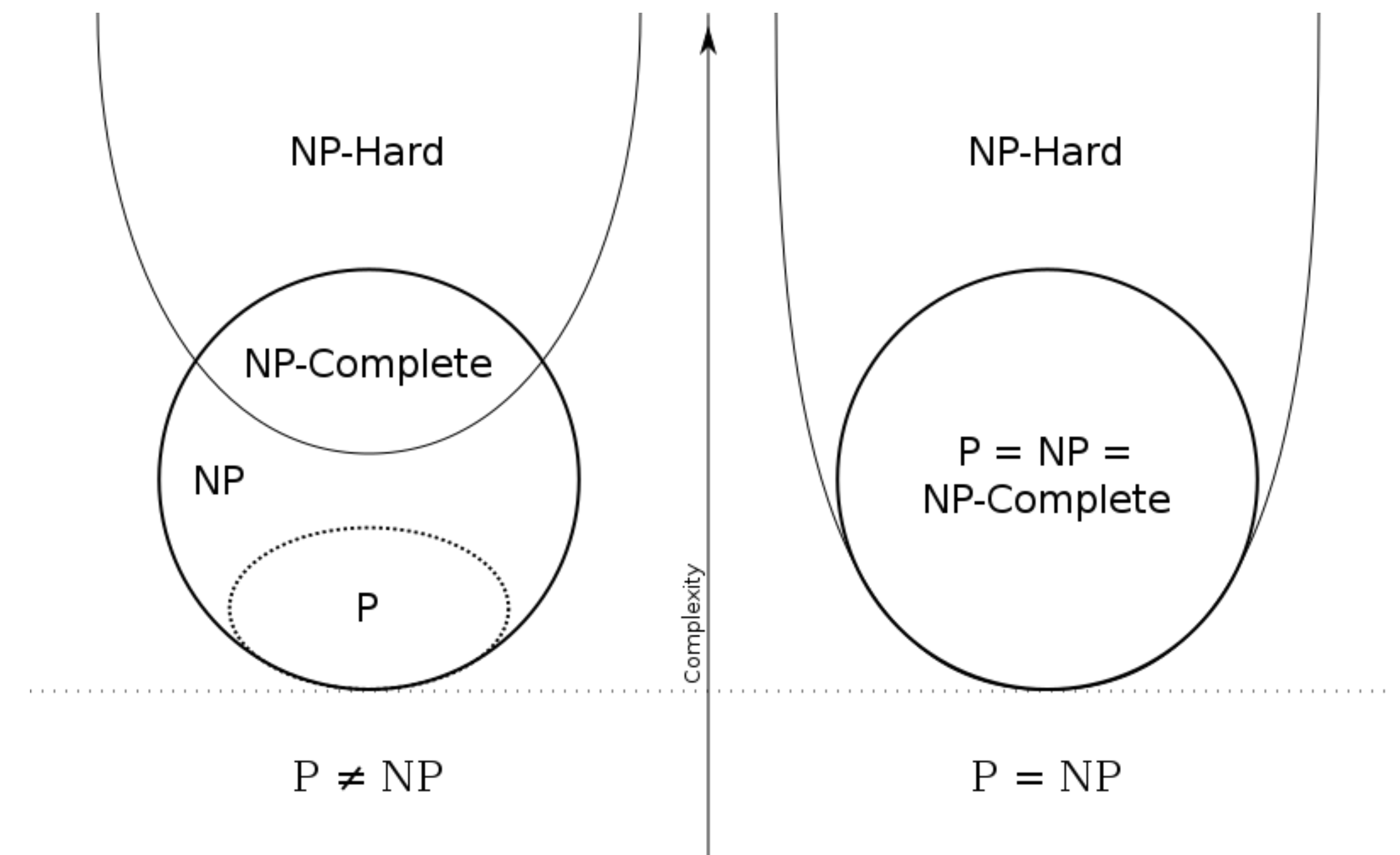


Задача SAT

- Дана булева формула $f: \{0, 1\}^n \rightarrow \{0, 1\}$
- Формула представлена в виде имен переменных, скобок, а также операций «и», «или», «не»
- Есть ли такой набор входных значений $x = (x_1, x_2, \dots, x_n)$, что $f(x) = 1$?
- Пример:
$$f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \& (\bar{x}_1 \vee x_2) \& (x_1 \vee \bar{x}_3) \& (\bar{x}_2 \vee x_3) \& (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$
- Есть множество эффективных решателей, например, [z3](#) (хотя он умеет намного больше)

Вопрос равенства классов сложности P и NP

- Очевидно, $P \subseteq NP$
- Открытая задача: $P \neq NP$ или $P = NP$?
- Одна из семи «задач тысячелетия»
- За ответ с доказательством математический институт Клэя предлагает \$ 1,000,000

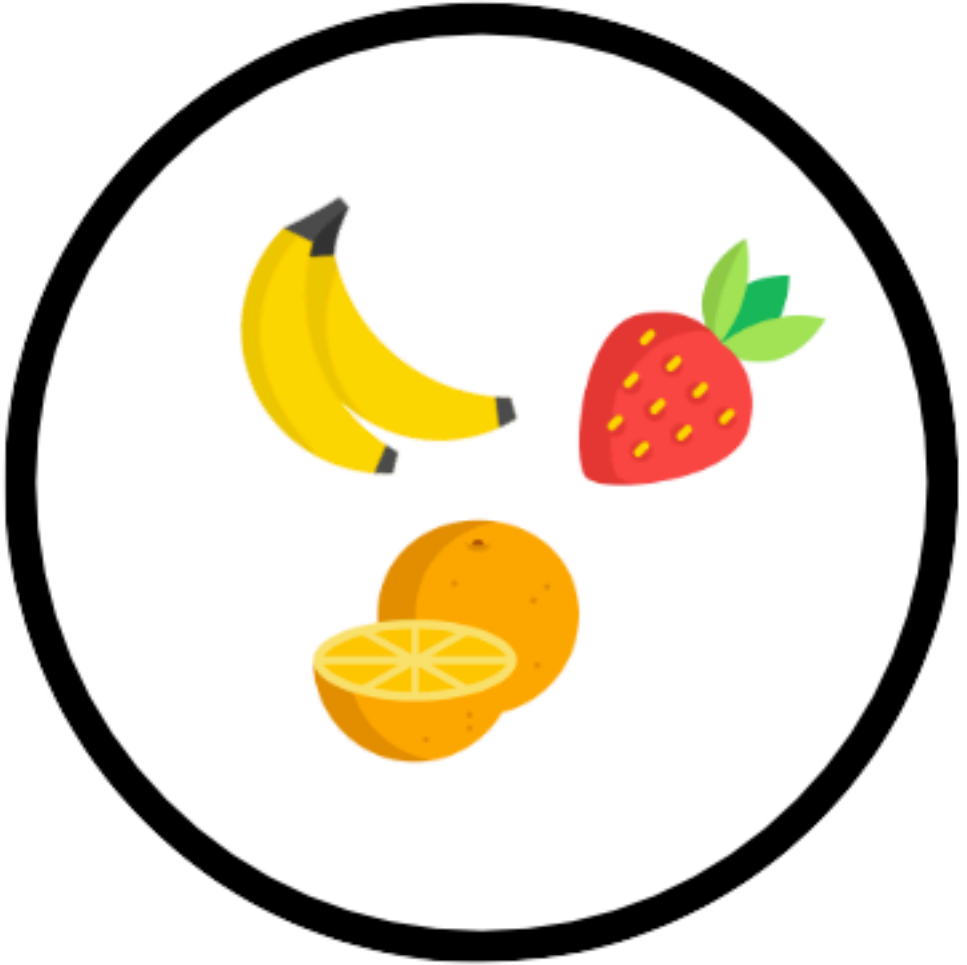


Содержание

1. Основные понятия
2. Переборные задачи
- 3. Односторонние функции** 🙌
4. Хэширование
5. Проверка целостности
6. Шифрование

Односторонняя функция

Просто



Сложно

Односторонняя функция

- Функция $f: X \rightarrow Y$, которая «легко» вычисляется для любого входного значения $x \in X$, при этом «трудно» найти $f^{-1}(y)$ для $y \in Y$
- «Легко» и «трудно» понимаются с точки зрения теории сложности вычислений
- Существование таких функций до сих пор не доказано
- Из их существования следует, что $P \neq NP$
- Тем не менее, есть хорошие кандидаты

Применение односторонних функций

Существование односторонних функций влечет существование многих других полезных криптографических объектов

Примеры:

- Генератор псевдослучайных чисел
- Имитовставка — код аутентификации
- Цифровая подпись
- Криптографическая хэш-функция
- Доказательство выполнения работы (например, для блокчейна или защиты от спама)

Пример протокола аутентификации

- Алиса генерирует последовательность: $m_0, m_1 = f(m_0), \dots, m_n = f(m_{n-1})$
- Алиса передает m_n Бобу – защищенный канал или личная встреча
- Когда Алисе необходимо подтвердить свою личность, она передает Бобу по открытому каналу m_{n-1}
- Боб проверяет: $f(m_{n-1}) = m_n?$
- Перехват сообщений в открытом канале на i -ом этапе ничего не дает злоумышленнику Еве, так как она не сможет узнать значения m_{i-1}

Односторонняя функция с потайным входом

- Односторонняя функция $f: K \times X \rightarrow Y$
- Однако, если знаем часть $k \in K$, то можем эффективно обратить функцию.
- Это свойство называется «потайным входом» или «лазейкой»
- Такие функции используются, например, для шифрования

Кандидаты в односторонние функции

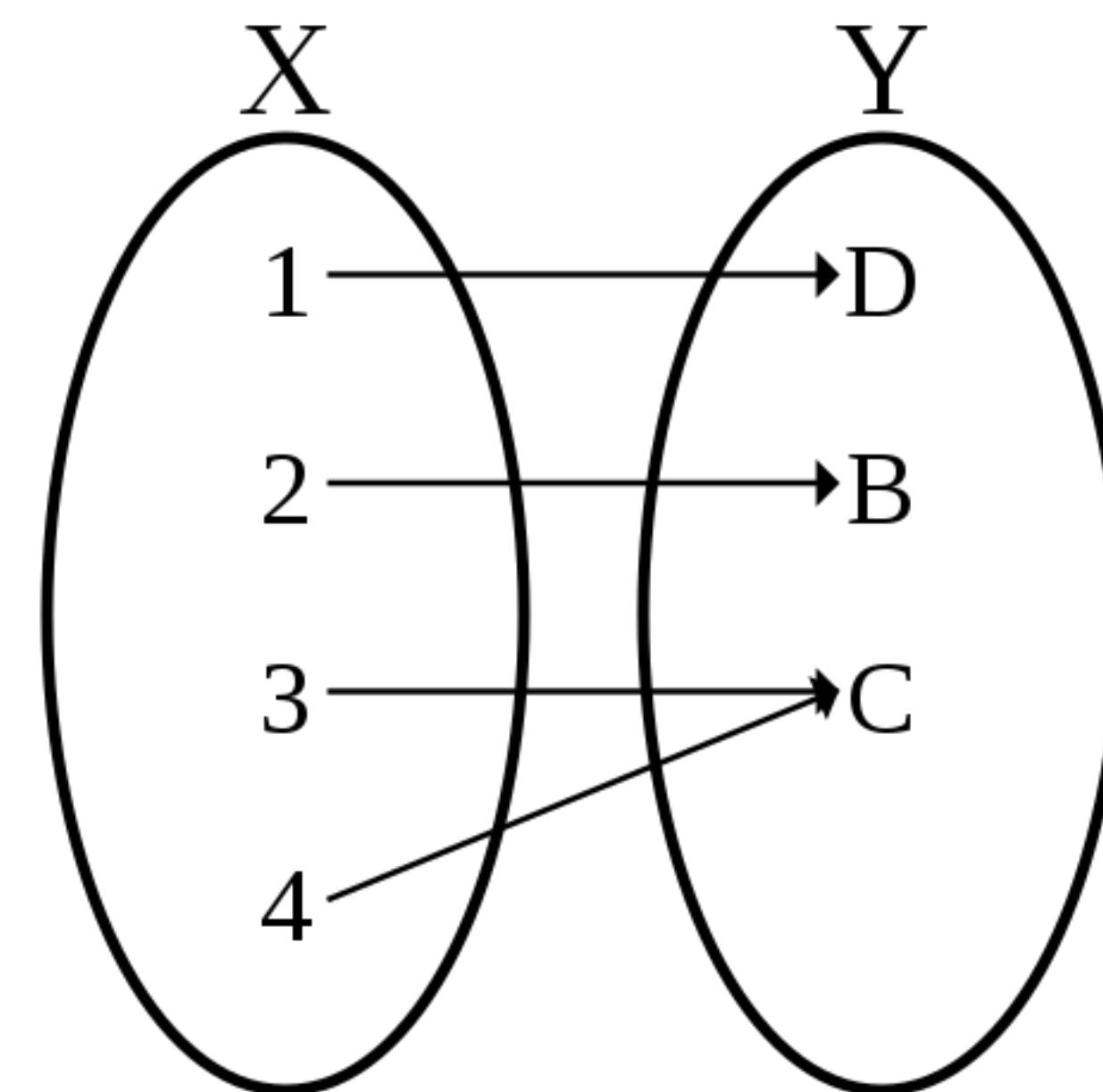
- Умножение и факторизация
Используется, например, в RSA
- Возведение в квадрат и извлечение квадратного корня по модулю
Например, криптосистема Рабина
- Дискретное экспоненцирование и логарифмирование
Например, протокол Диффи-Хеллмана
- Семейство криптографических хэш-функций
Например, SHA-256 и прочие

Содержание

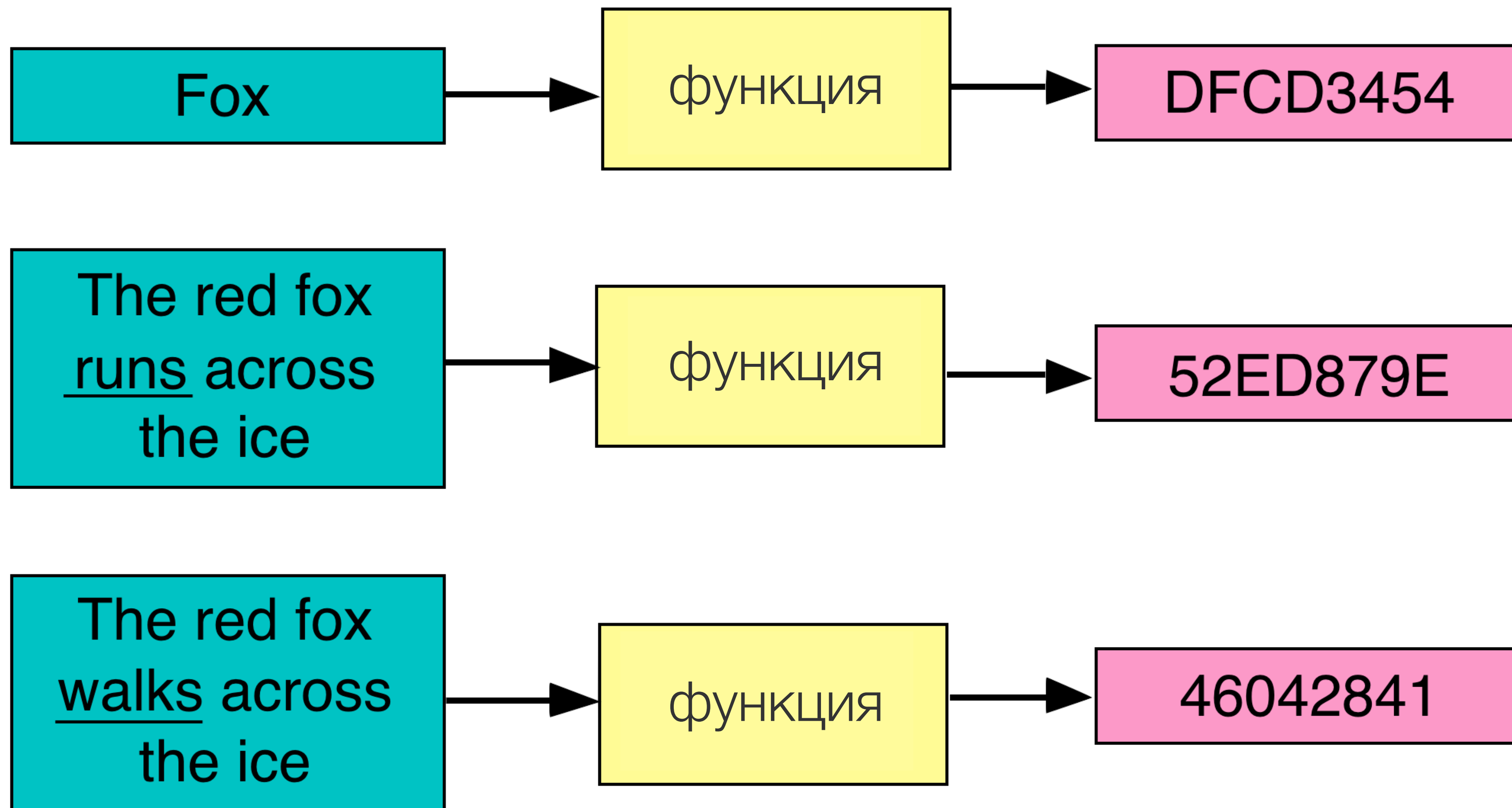
1. Основные понятия
2. Переборные задачи
3. Односторонние функции
- 4. Хэширование** 🙌
5. Проверка целостности
6. Шифрование

Хэш-функции и коллизии

- Хэш-функция $h: \{0, 1\}^* \rightarrow \{0, 1\}^m$ – функция, преобразующая сообщение произвольной длины в последовательность фиксированной длины
- Коллизия для хэш-функции:
пара сообщений $m_1 \neq m_2$, $h(m_1) = h(m_2)$



Лавинный эффект



Простой лавинный критерий

- Функция $h: \{0, 1\}^n \rightarrow \{0, 1\}^m$ удовлетворяет лавинному критерию, если при изменении одного бита входной последовательности изменяется в среднем половина выходных битов
- Есть более строгие и сложные определения критерия
- Понятие лавинного критерия обычно применяют к криптографическим хэш-функциям и блочным шифрам

Требования к криптографическим хэш-функциям

- Стойкость к поиску прообраза I-го рода:
нельзя эффективно обратить – функция является односторонней
- Стойкость к поиску прообраза II-го рода:
нельзя эффективно найти другое сообщение с такой же хэш-суммой
 - Стойкость к коллизиям:
нет эффективного алгоритма, позволяющего находить коллизии
- Лавинный эффект

Коллизии известных хэш-функций

SHattered
The first concrete collision attack against SHA-1
<https://shattered.io>

CWI
Marc Stevens
Pierre Karpman

Google
Elie Bursztein
Ange Albertini
Yarik Markov

SHattered
The first concrete collision attack against SHA-1
<https://shattered.io>

CWI
Marc Stevens
Pierre Karpman

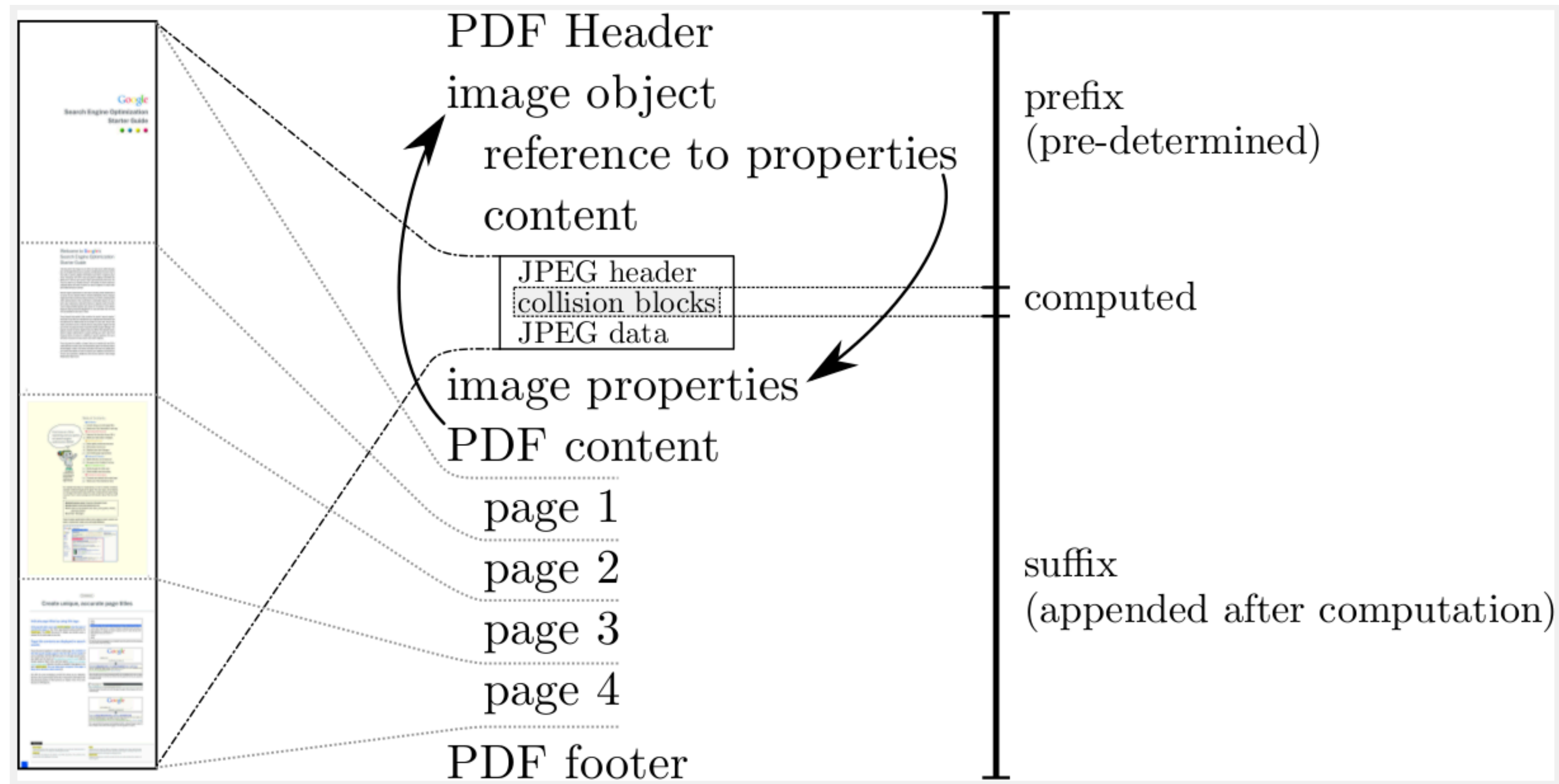
Google
Elie Bursztein
Ange Albertini
Yarik Markov

```
└─ sha1sum *.pdf
38762cf7f55934b34d179ae6a4c80cadccb7f0a 1.pdf
38762cf7f55934b34d179ae6a4c80cadccb7f0a 2.pdf
└─ /tmp/sha1
└─ sha256sum *.pdf
2bb787a73e37352f92383abe7e2902936d1059ad9f1ba6daaa9c1e58ee6970d0 1.pdf
d4488775d29bdef7993367d541064dbdda50d383f89f0aa13a6ff2e0894ba5ff 2.pdf
```

0.64G 8-11h

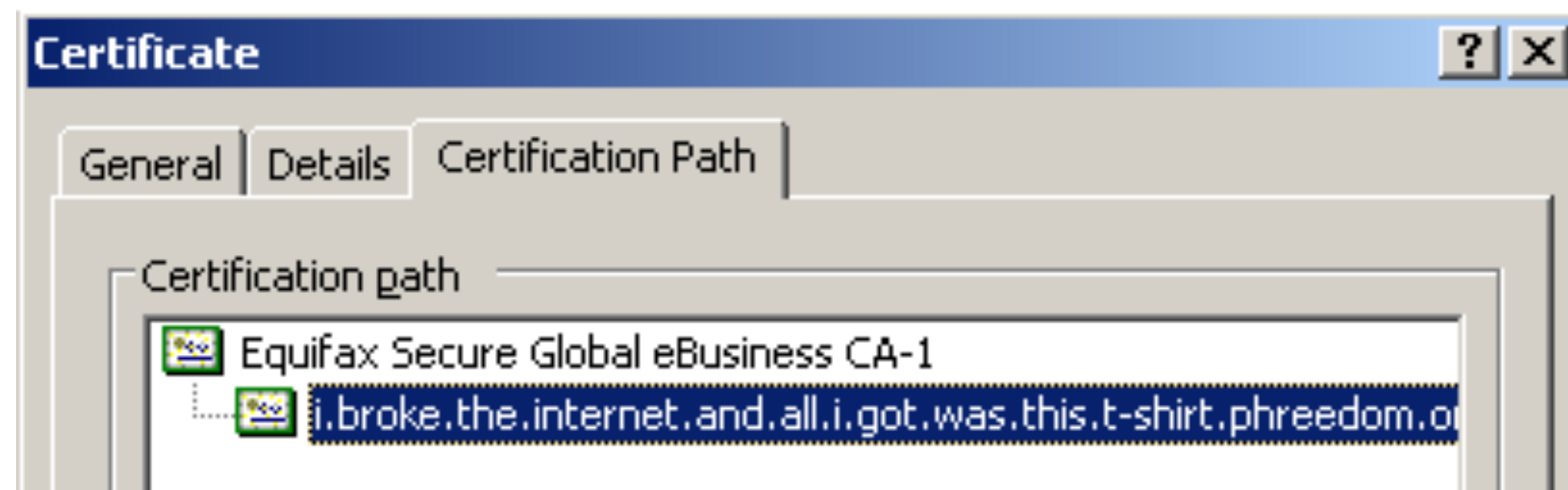
- MD5: ≈2007
- SHA-1: 2017, CWI & Google

Коллизия SHA-1

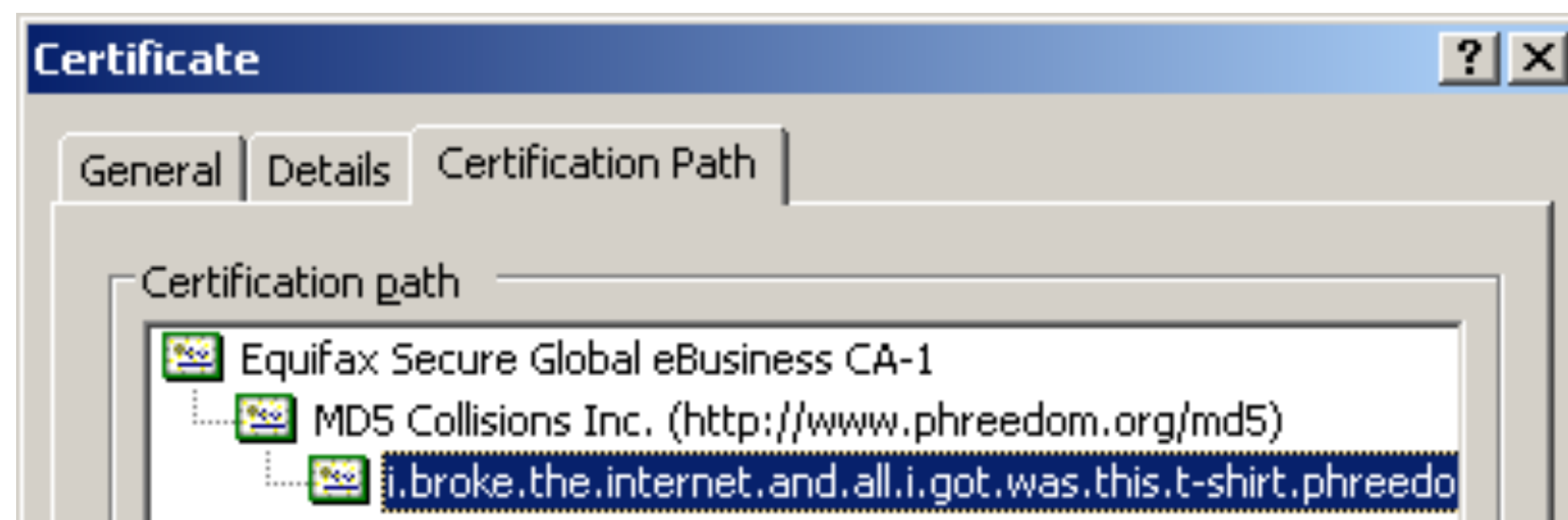


Атака на электронную подпись коллизией

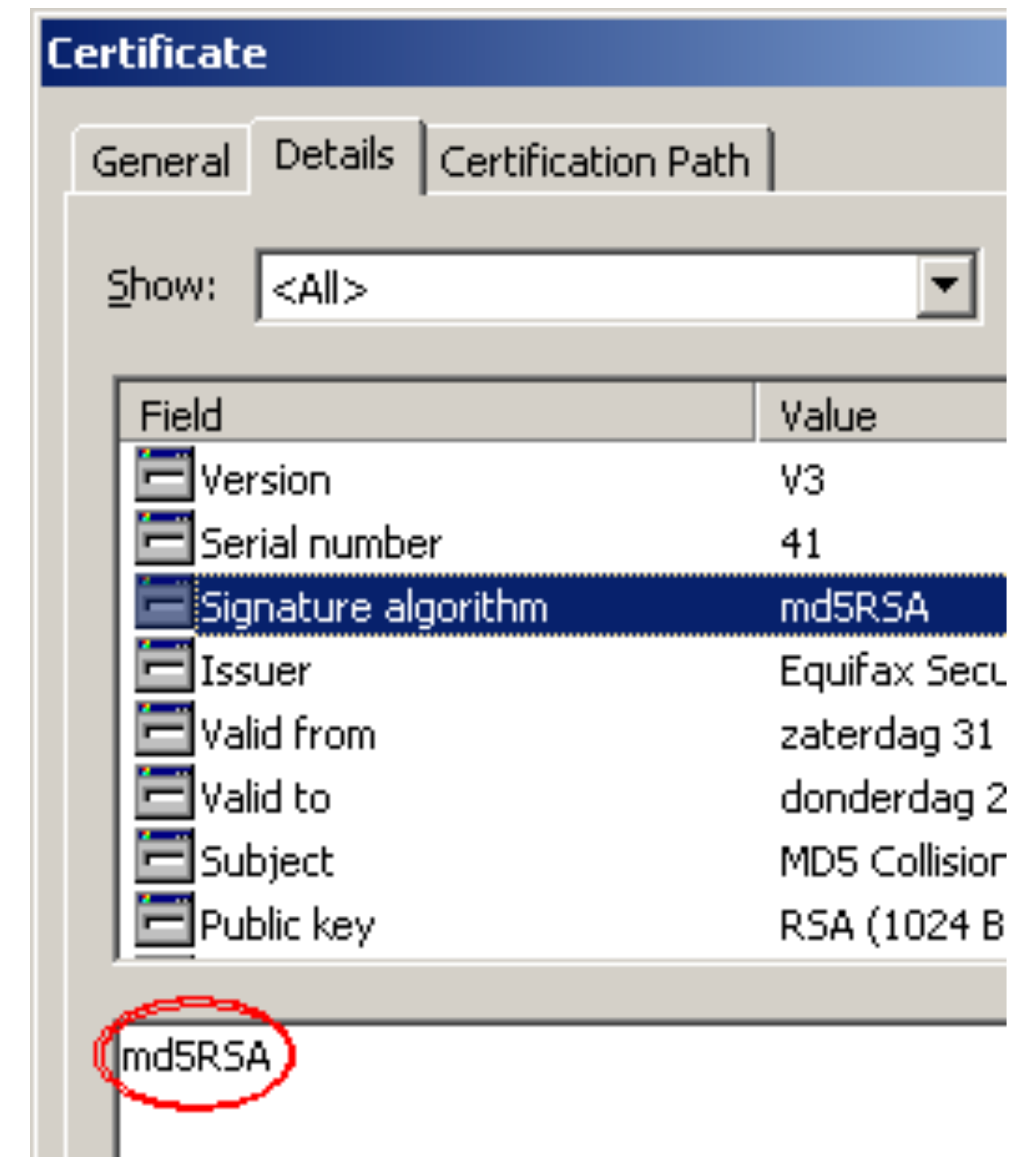
При использовании цифровой подписи обычно подписывают не само сообщение, а его хэш-сумму



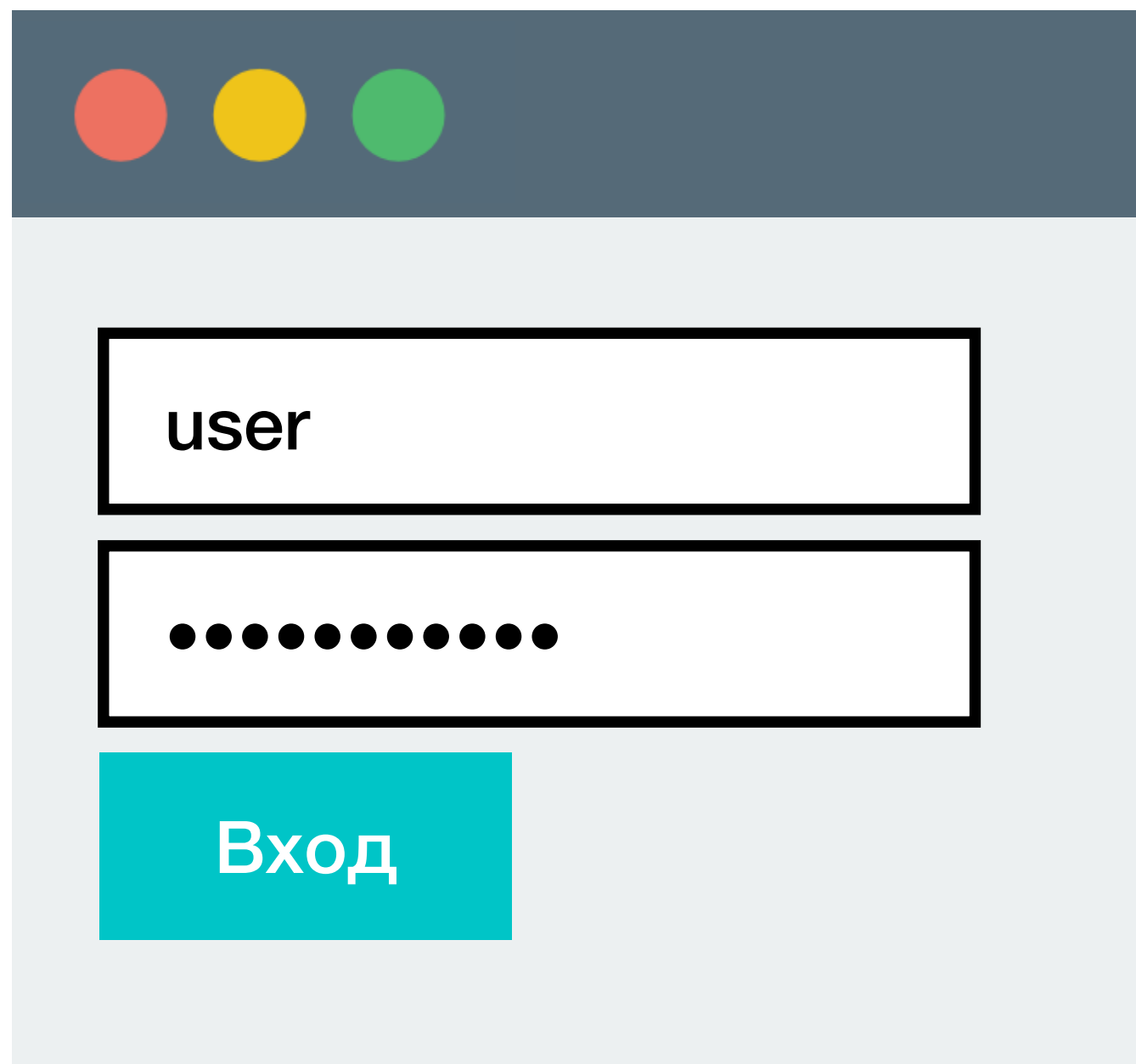
подписано у центра сертификации



использовали подпись на коллизии



Хранение паролей в открытом виде

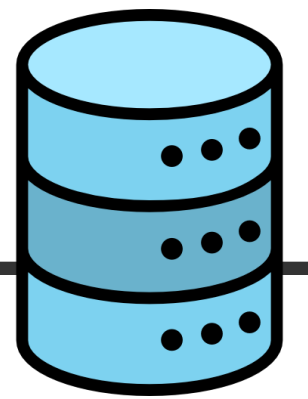


user

.....

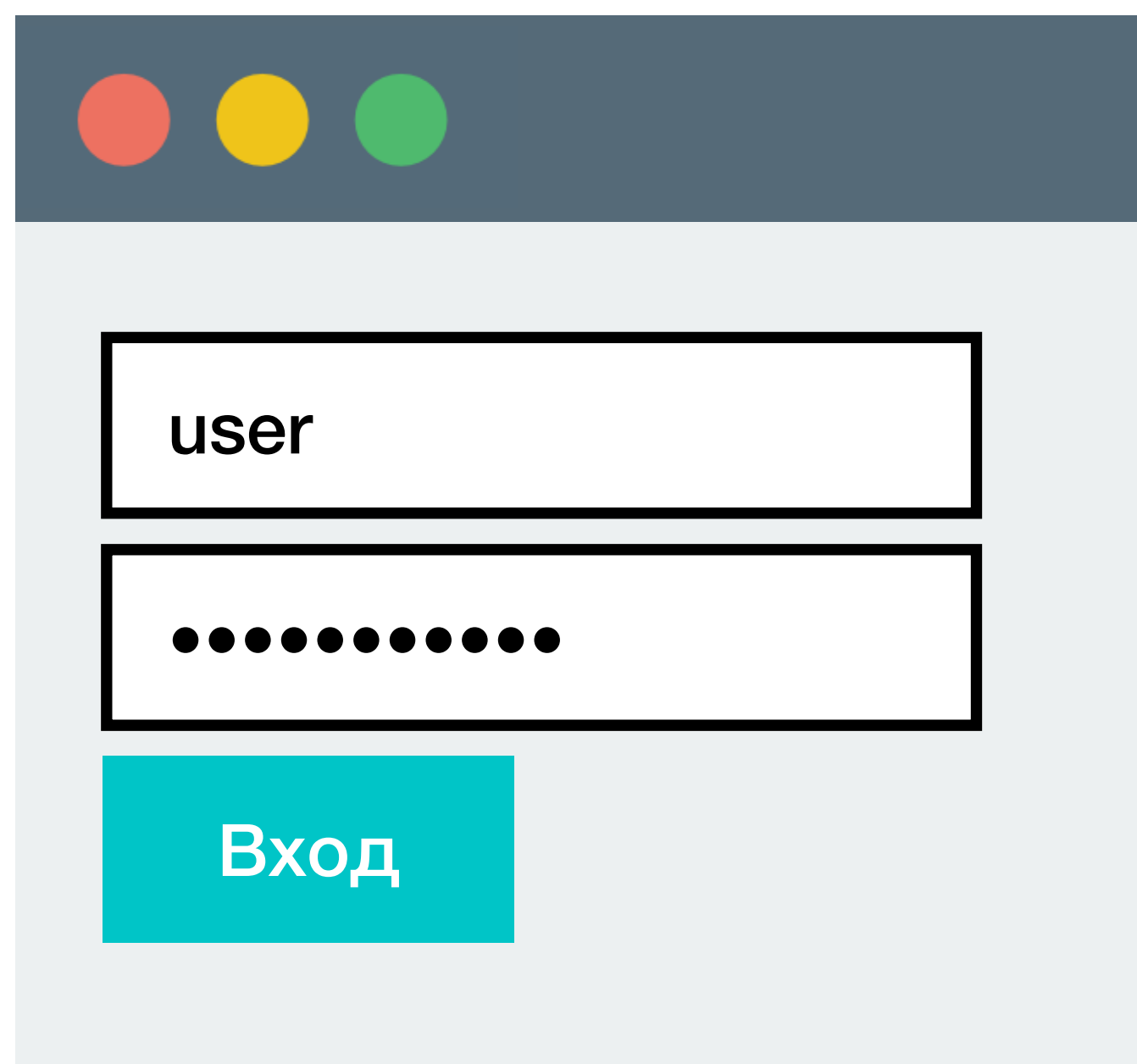
Вход

- Злоумышленник получает доступ к БД
- Сразу узнает пароли всех пользователей

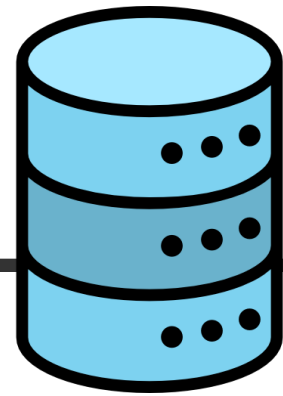


username	password
admin	qwerty1337
user	password123

Хэширование паролей

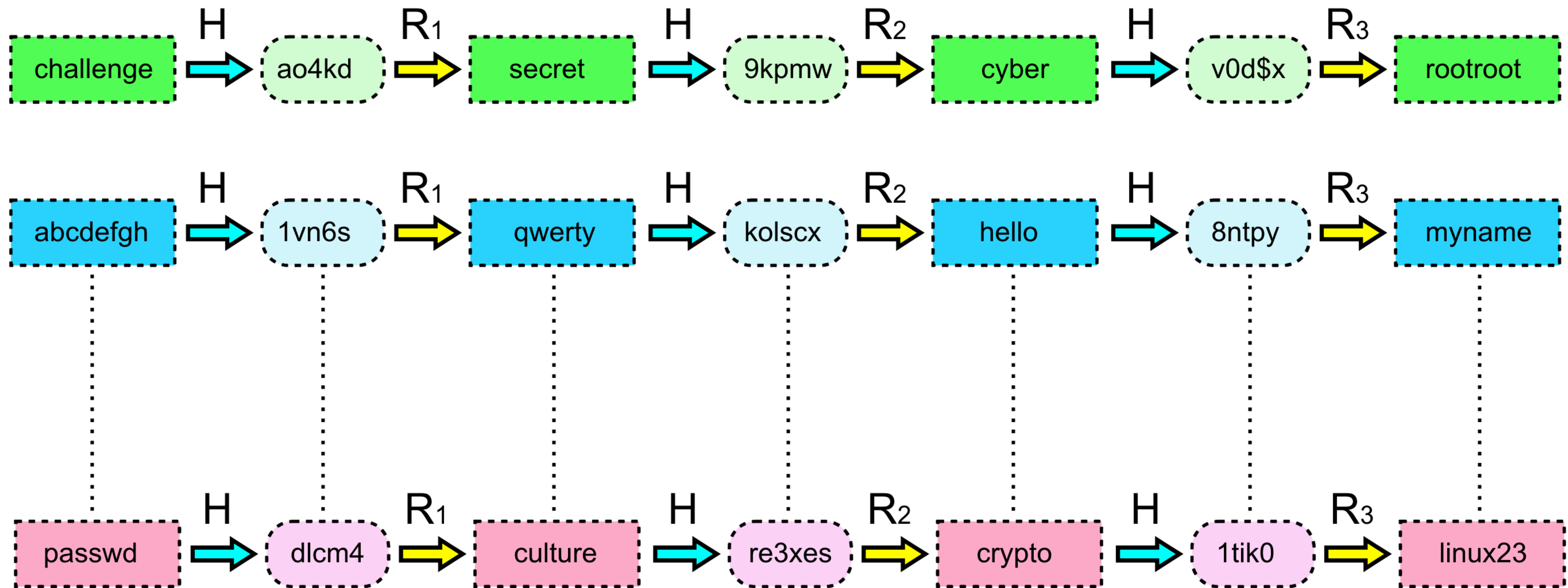


A login form interface with a dark blue header containing three colored circles (red, yellow, green). Below the header, there is a text input field containing the word "user". Underneath that is a password input field represented by ten black dots. At the bottom of the form is a teal button with the Russian word "Вход" (Login) written in white.



username	password
admin	\$2a\$10\$0mp0ludyiPhhxyrU0Sy/b.muu4WLQ0EmpnghwfktHC/5yorciYAB0
user	\$2a\$10\$WL1hPhzfrBqHdqjXYQTffeEPnRbNC/KkBHsav/iAdWhPvE.vaiVnK

Радужные таблицы



Радужные таблицы

- Эффективная структура данных, которая позволяет получать соответствие хэш → прообраз хэша
- Идея: берем словарь паролей, хэшируем, сохраняем в радужную таблицу
- Такие таблицы могут занимать много терабайт памяти
- Можно не только составлять их самому, но и:
 - скачать готовые таблицы в интернете
 - воспользоваться веб-сервисами, которые ищут по таблице
- Поиск по ним позволяет атакующему восстановить пароли

Хэширование с солью

- Обычное хэширование: `hash(password)`
- Хэширование с солью: `hash(combine(password, salt))`, где
 - `salt` – некоторое случайное значение для каждого пользователя, которое сохраняется в базу вместе с хэшем пароля
 - `combine` – функция, некоторым образом комбинирующая пароль с солью
- Соление хэшей позволяет усложнить применение радужных таблиц
- Однако, соль не может защитить от перебора каждого отдельного пароля

Есть ли разница?

- `hash(password + ':' + salt)`
- `hash(salt + ':' + password)`

Замедление хэширования

- Известные криптографические хэш-функции: MD5, SHA-256 и подобные
- Проблема: они быстро вычисляются, особенно на специализированном «железе»
- Поэтому используют KDF – функции формирования ключа

Криптографические функции формирования ключа

$KDF(cost, key, salt, input)$ – адаптивная функция формирования ключа:

- $cost$ – сложность вычисления, обычно порядок раундов алгоритма
- key – начальный ключ, из которого надо сформировать другой
- $salt$ – соль
- $input$ – входные данные

Криптографические функции формирования ключа

- Известные KDF: PBKDF2, bcrypt, scrypt, Argon2
- У всех разная степень защиты от атак на CPU, GPU, FPGA, ASIC, ...
- Пример **bcrypt** с 2^5 раундами и **солью**:

\$2a\$05\$3WivJJ2ECsLVE9u.F37DveukbyJP4kx0mDm2QH/KF0ubhE7xv.Hk0

СЛОЖНОСТЬ

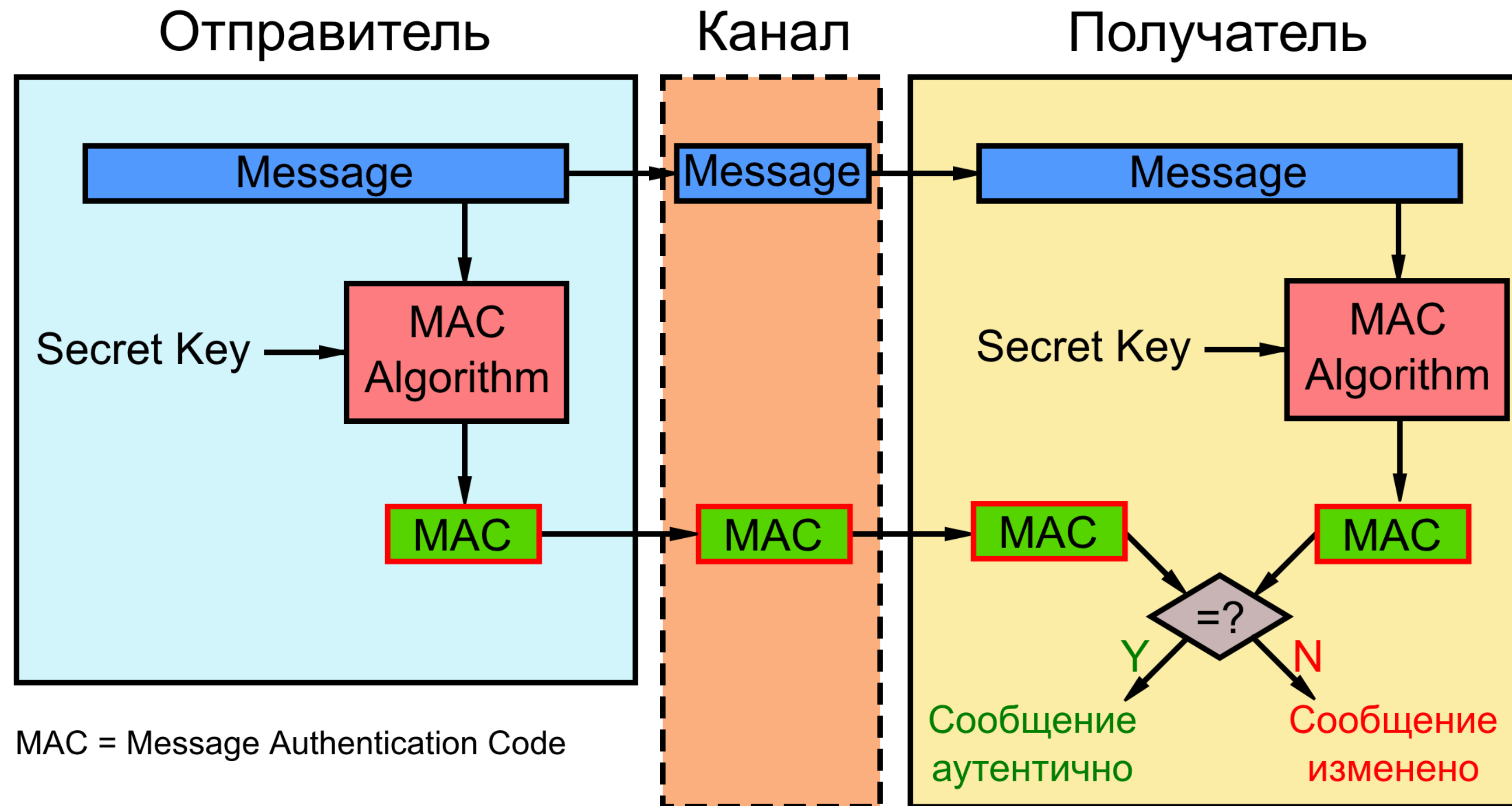
Стоимость взлома пароля за 1 год (2010)

KDF	6 letters	8 letters	8 chars	10 chars	40-char text	80-char text
DES CRYPT	< \$1	< \$1	< \$1	< \$1	< \$1	< \$1
MD5	< \$1	< \$1	< \$1	\$1.1k	\$1	\$1.5T
MD5 CRYPT	< \$1	< \$1	\$130	\$1.1M	\$1.4k	1.5×10^{15}
PBKDF2 (100 ms)	< \$1	< \$1	\$18k	\$160M	\$200k	2.2×10^{17}
bcrypt (95 ms)	< \$1	\$4	\$130k	\$1.2B	\$1.5M	\$48B
scrypt (64 ms)	< \$1	\$150	\$4.8M	\$43B	\$52M	6×10^{19}
PBKDF2 (5.0 s)	< \$1	\$29	\$920k	\$8.3B	\$10M	11×10^{18}
bcrypt (3.0 s)	< \$1	\$130	\$4.3M	\$39B	\$47M	\$1.5T
scrypt (3.8 s)	\$900	\$610k	\$19B	\$175T	\$210B	2.3×10^{23}

Содержание

1. Основные понятия
2. Переборные задачи
3. Односторонние функции
4. Хэширование
- 5. Проверка целостности** 🙌
6. Шифрование

Имитовставка – код аутентификации сообщения



Свойство целостности и подлинности

- Имитовставка не заменяет собой электронную подпись!
- В чем отличие?

Свойство целостности и конфиденциальности

- Могут возникнуть следующие вопросы:
 - Почему бы просто не зашифровать сообщение, а потом расшифровать?
 - Чем отличается шифрование от проверки целостности?
 - Что будет, если разработчик криптосистемы не понимает разницы?
- Кстати, иногда необходимо шифровать вместе с обеспечением целостности (см., например, тег в AES-GCM)

Пример уязвимости при отсутствии контроля целостности

```
def user_token(id):  
    return encrypt(key, 'u:{}'.format(id))
```

```
def admin_token(id):  
    return encrypt(key, 'a:{}'.format(id))
```

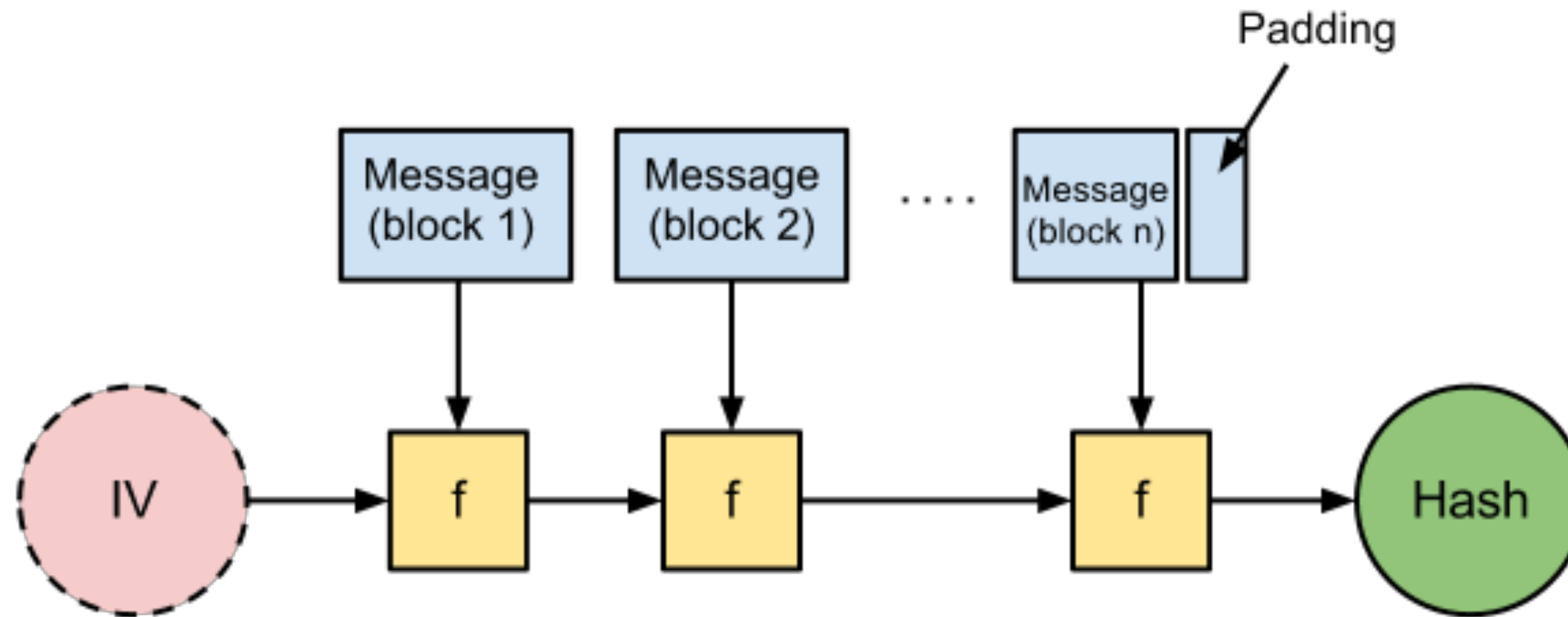
```
def is_admin(token):  
    return 'a:' in decrypt(key, token)
```

атакуется за $\approx (256^2 / 2)$ запросов

Простой MAC на основе хэш-функции

- $\text{SimpleMAC}_K(M) = H(K \parallel M)$
- H – хэш-функция, K – секретный ключ, M – сообщение

Блочное устройство хэш-функций



Атака удлинением сообщения на подпись $H(K || M)$

- Запрос:

```
user_id=1&count=5&item=nuggets
```

- Есть легитимная подпись:

```
6d5f807e23db210bc254a28be2d6759a0f5f5d99
```

- Угадав длину, атакующий формирует новый запрос:

```
user_id=1&count=5&item=nuggets \x80\x00\x00...  
\x00\x02\x28&item=wings
```

- Новая подпись получается удлинением сообщения

НМАС – код аутентификации на основе хэш-функций

- $$\text{НМАС}_K(M) = \text{H}\left((K \oplus \text{opad}) \parallel \text{H}\left((K \oplus \text{ipad}) \parallel M \right) \right)$$
- H – хэш-функция, K – секретный ключ, M – сообщение,
 $\text{opad} = (0x5c, 0x5c, \dots, 0x5c)$, $\text{ipad} = (0x36, 0x36, \dots, 0x36)$
- Доказано, что вероятность успешной атаки на НМАС эквивалентна вероятности атаки на используемую хэш-функцию

Содержание

1. Основные понятия
2. Переборные задачи
3. Односторонние функции
4. Хэширование
5. Проверка целостности
- 6. Шифрование** 🙌

Перестановочный шифр «Скитала» III век до н.э.



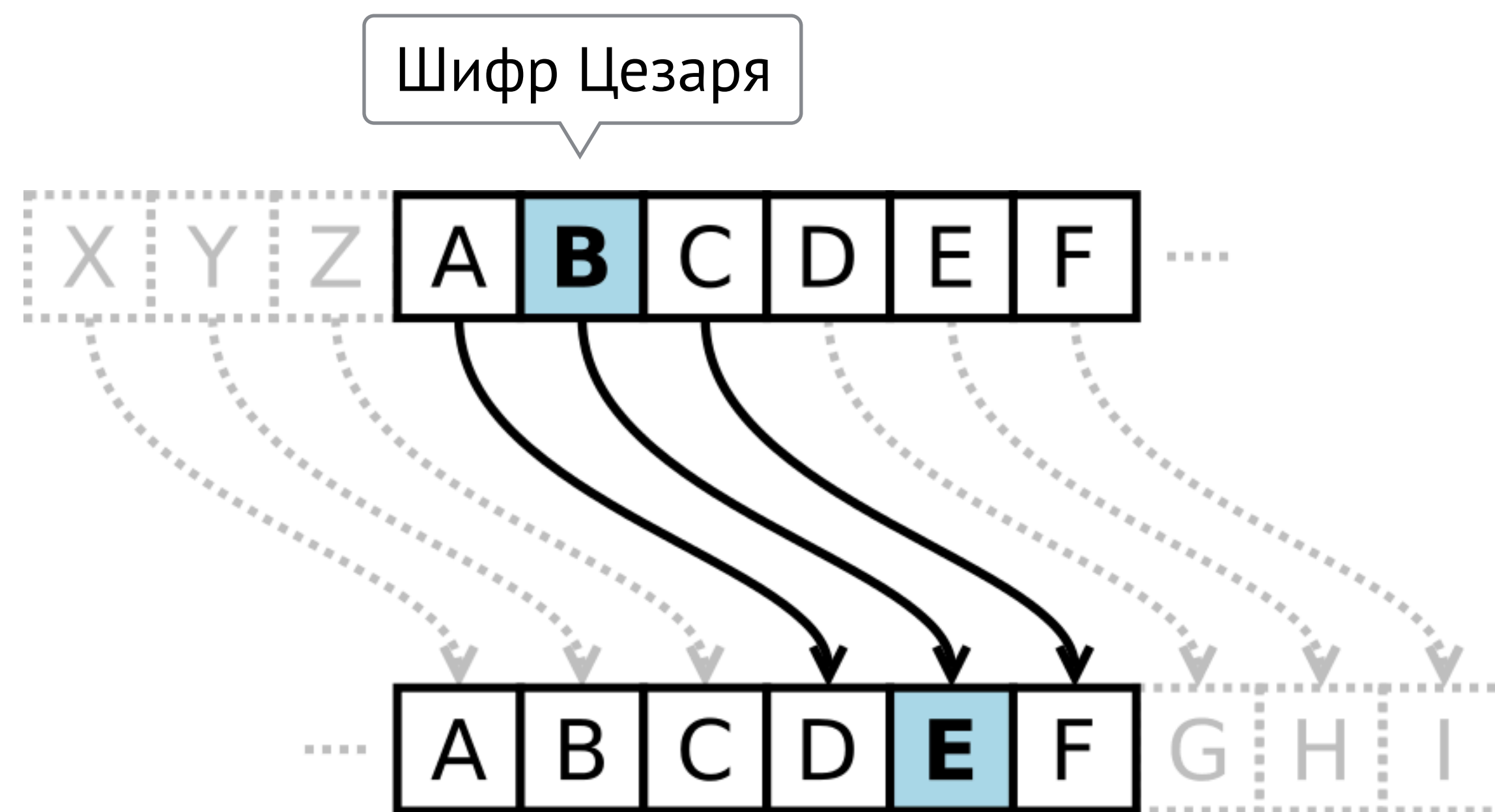
«Энигма»

WW II

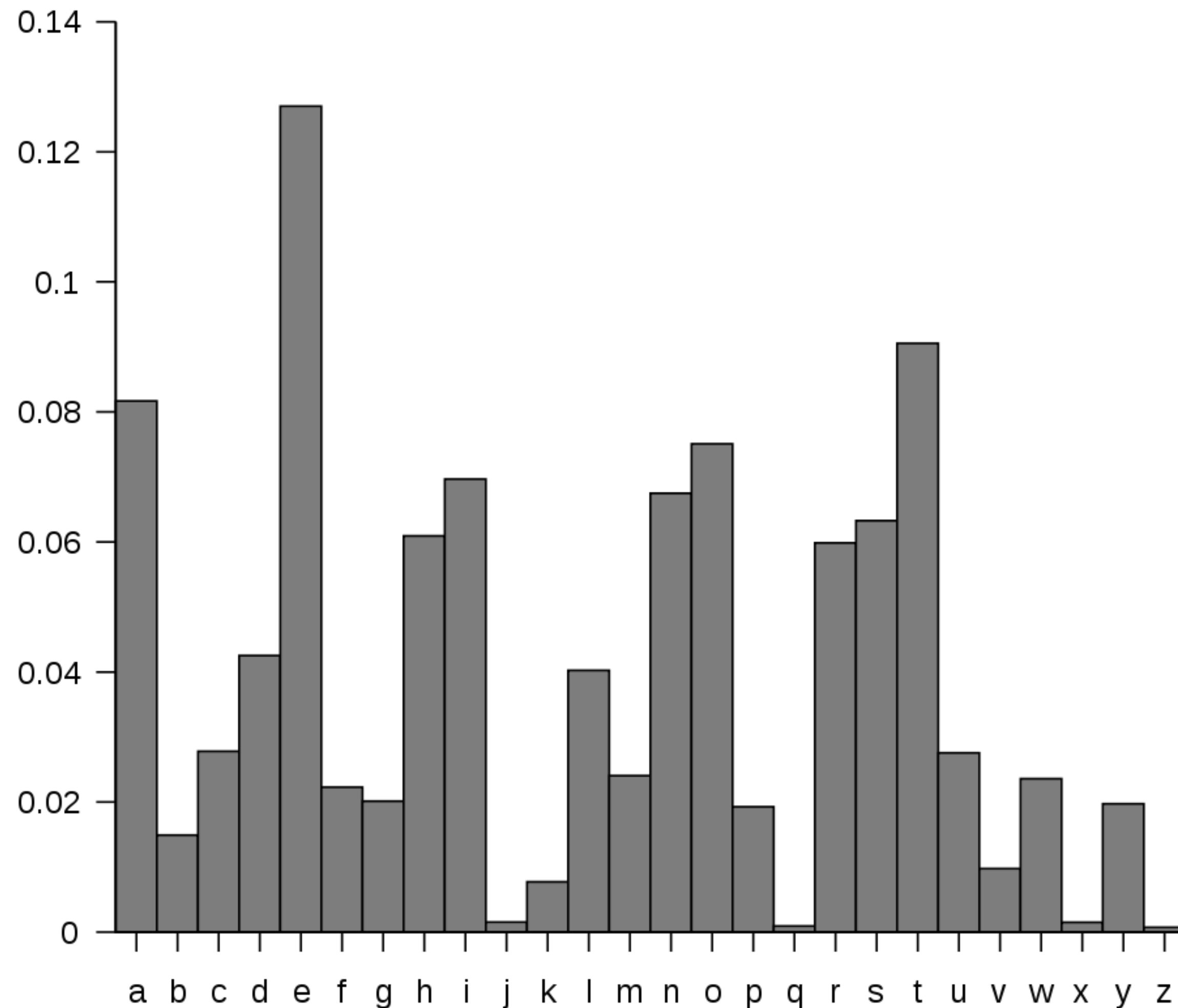


Простые методы шифрования

- Шифры подстановки
 - Шифр простой замены
 - Цезаря, табличный
 - Прочие
- Шифры перестановки



Частотный криптоанализ

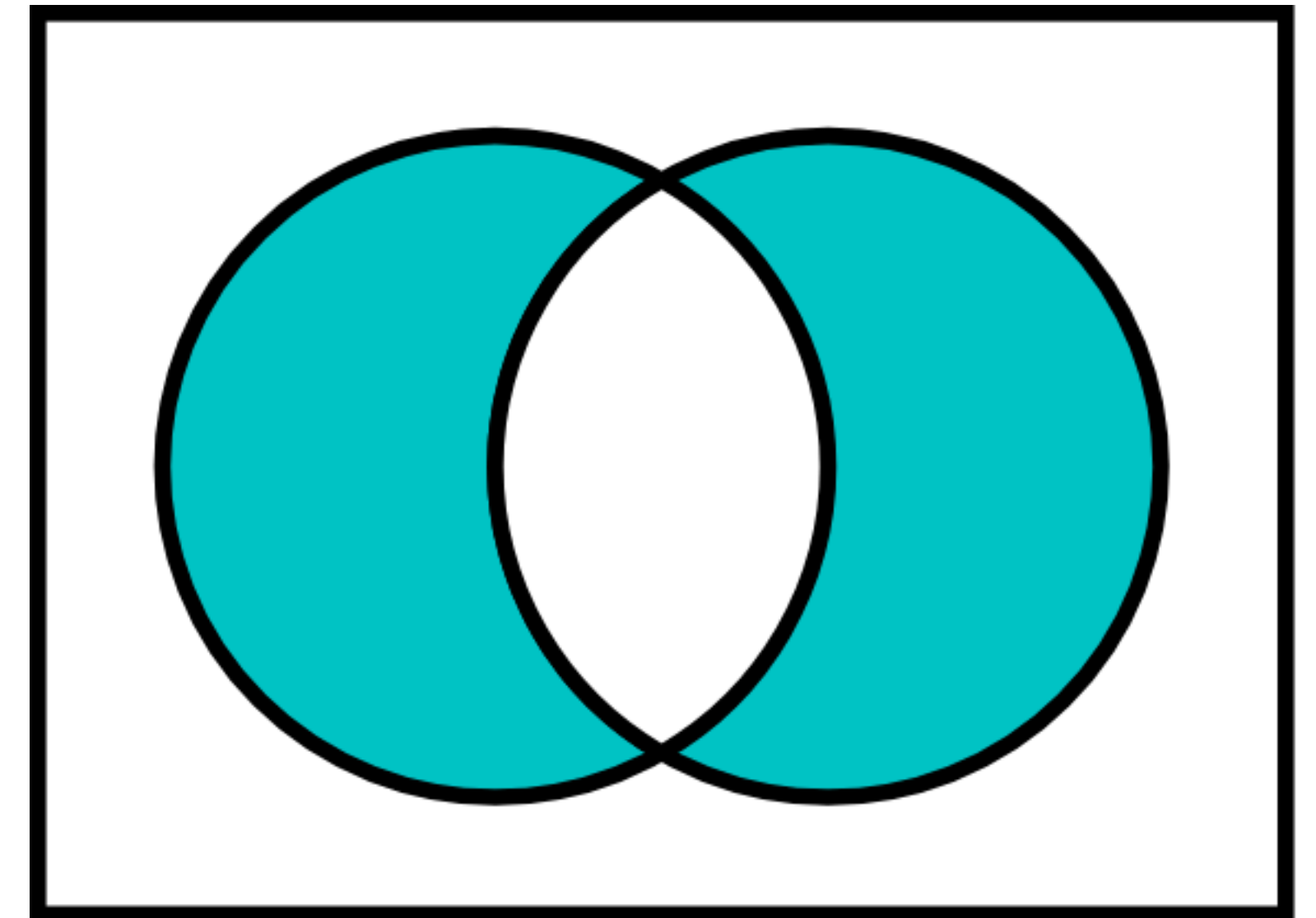


- Атака на простые шифры
- Можно рассмотреть распределение вероятностей:
 - символов алфавита
 - n-грамм, слов

Сложение по модулю 2

- Оно же «исключающее или», «XOR»
- $\oplus : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$

- | x | y | $x \oplus y$ |
|-----|-----|--------------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



Свойства сложения по модулю 2

- Ассоциативность:

$$(a \oplus b) \oplus c = a \oplus (b \oplus c)$$

- Коммутативность:

$$a \oplus b = b \oplus a$$

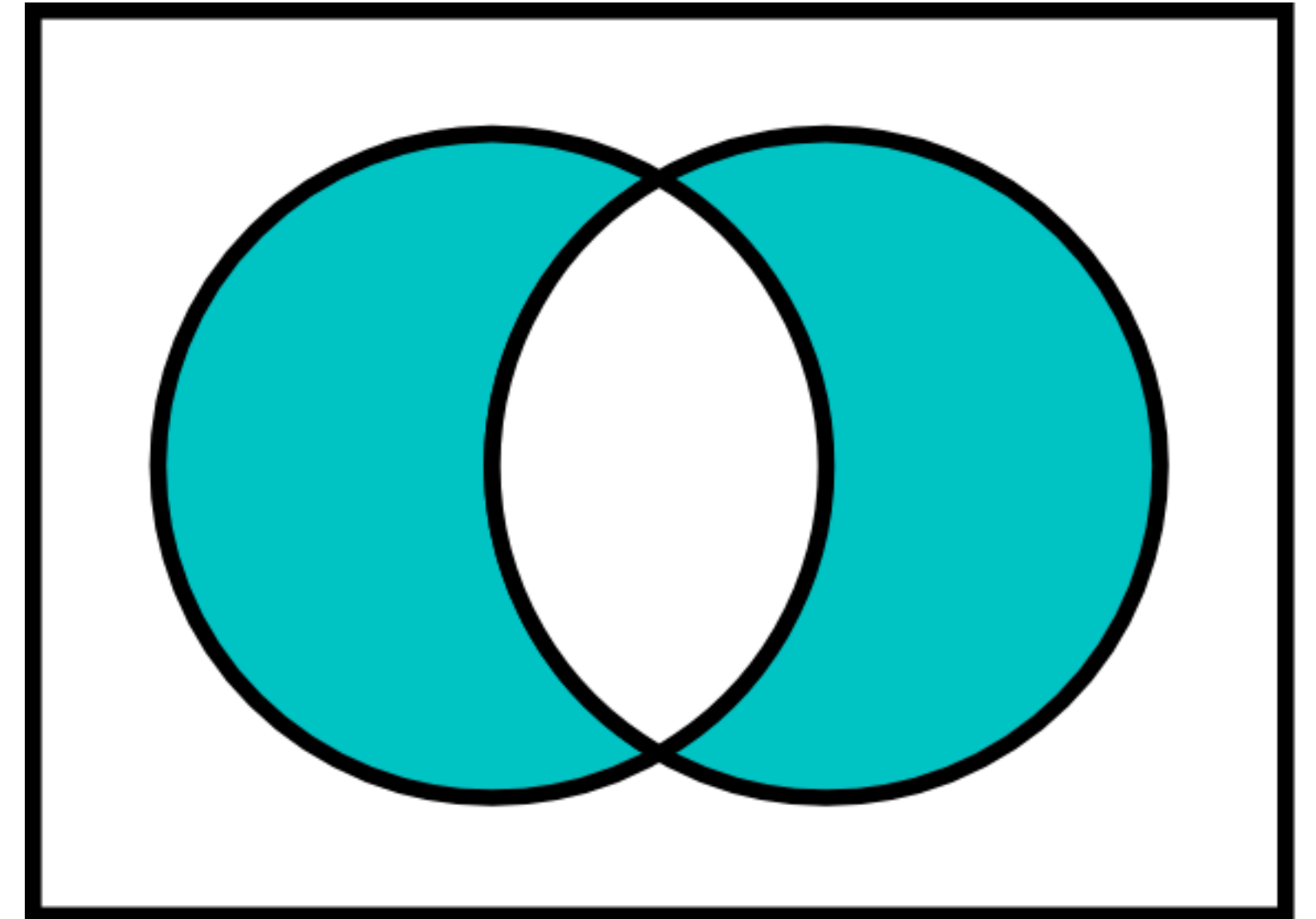
- Нильпотентность:

$$a \oplus a = 0$$

- Получение равенства и отрицания:

$$a \oplus 0 = a$$

$$a \oplus 1 = \bar{a}$$



Побитовое сложение по модулю 2

$$73 \oplus 87 = 0b1001001 \oplus 0b1010111$$

$$\begin{array}{ccccccc} & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ = & \oplus & \oplus & \oplus & \oplus & \oplus & \oplus & \oplus \\ & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ = & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ & = & 0b0011110 \\ & = & 30 \end{array}$$

Шифр Вернама

- $C = P \oplus K \Leftrightarrow P = C \oplus K$
- P – сообщение, C – шифртекст, K – ключ
- Ключ должен быть совершенно случайным и равномерно распределенным:

$$\mathbb{P}(K = k_1 k_2 \dots k_n) = \frac{1}{2^n}$$

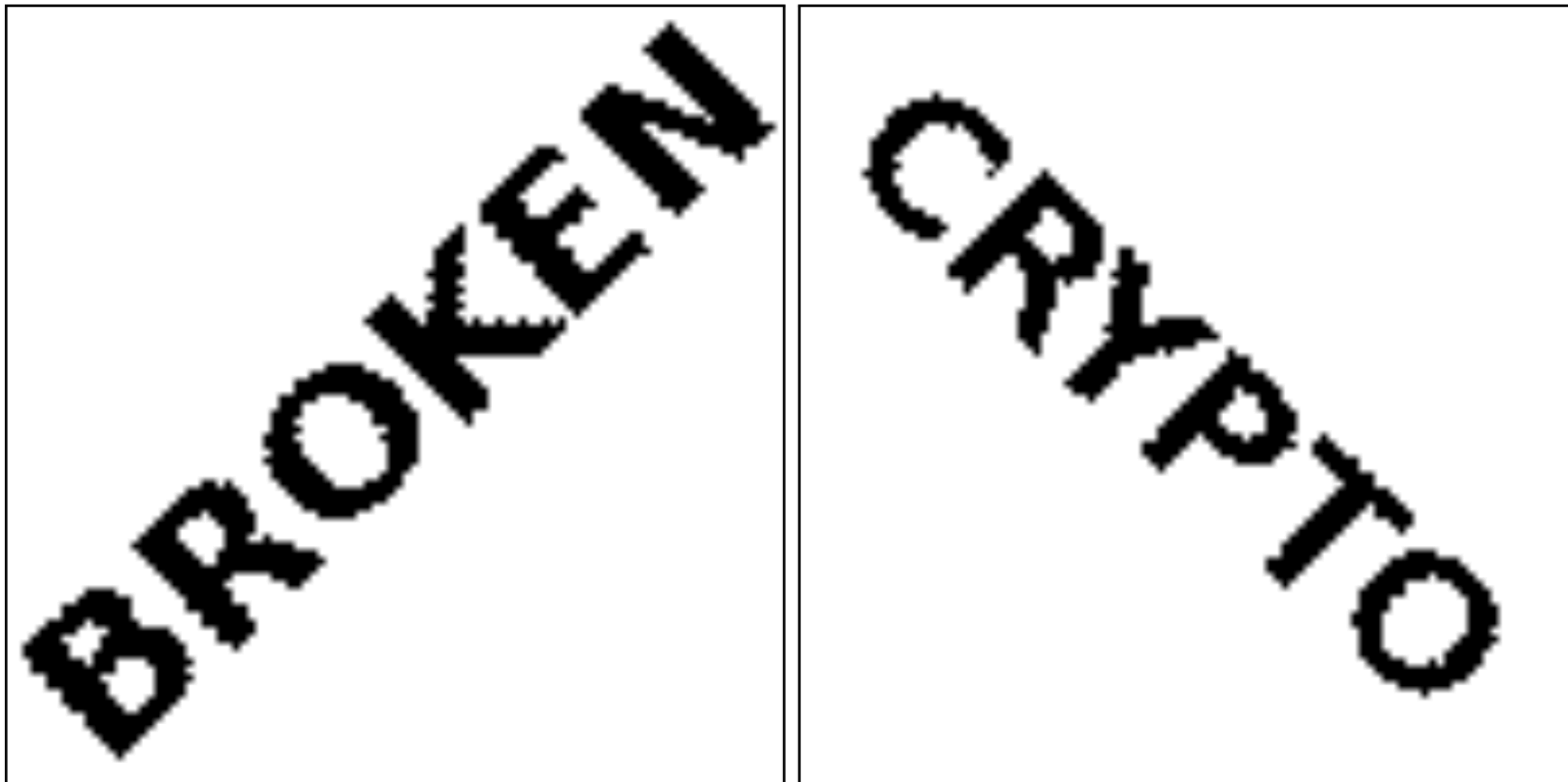
Шифр Вернама

- Симметричный потоковый шифр
- Абсолютно криптостойкий – нельзя придумать шифр лучше
- Однако, довольно часто неприменимый на практике:
 - Для генерации ключа нужен генератор истинно случайных чисел
 - Ключ должен иметь такую же длину, как и сообщение
 - Ключ должен использоваться только один раз, после чего уничтожаться

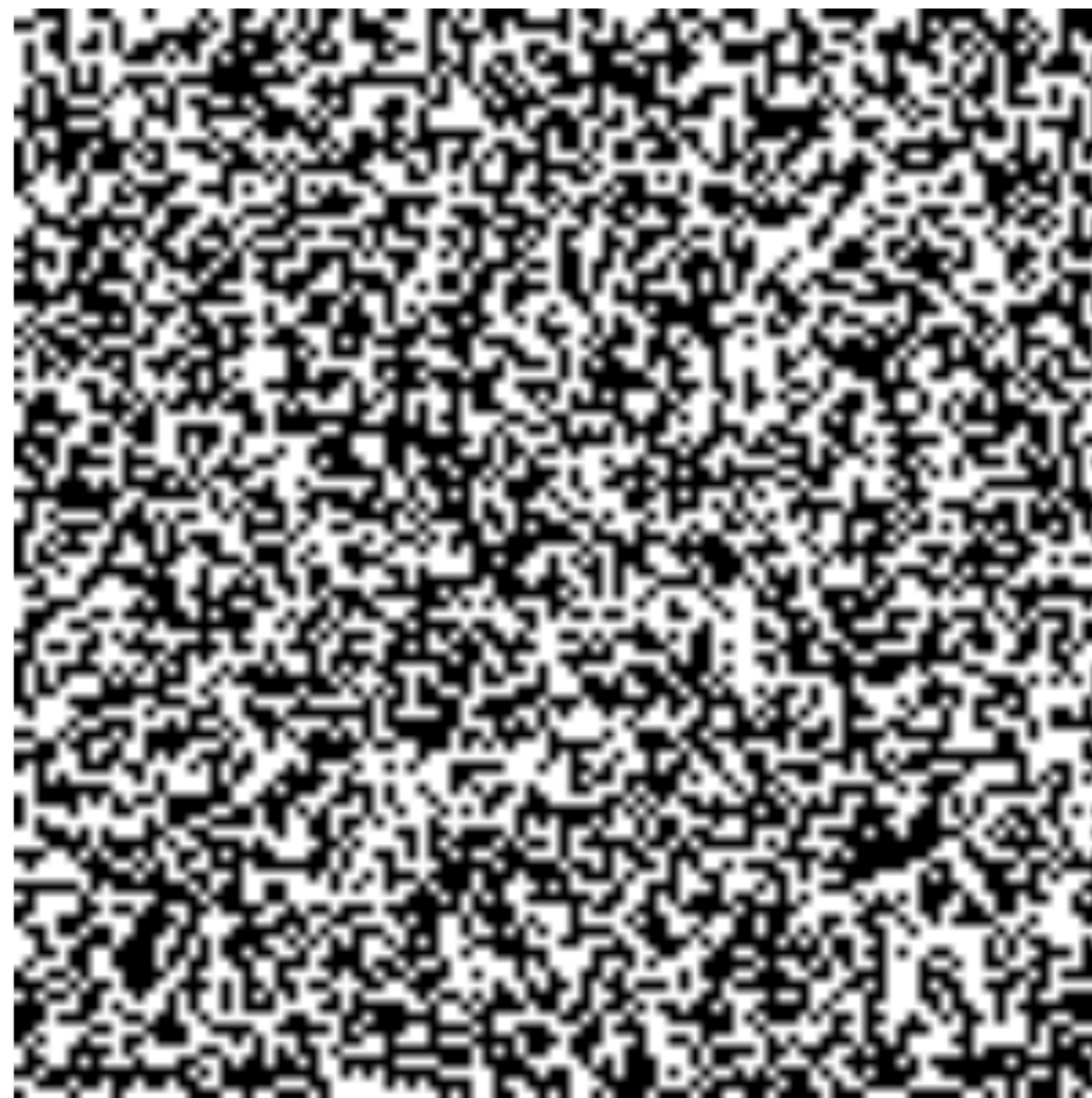
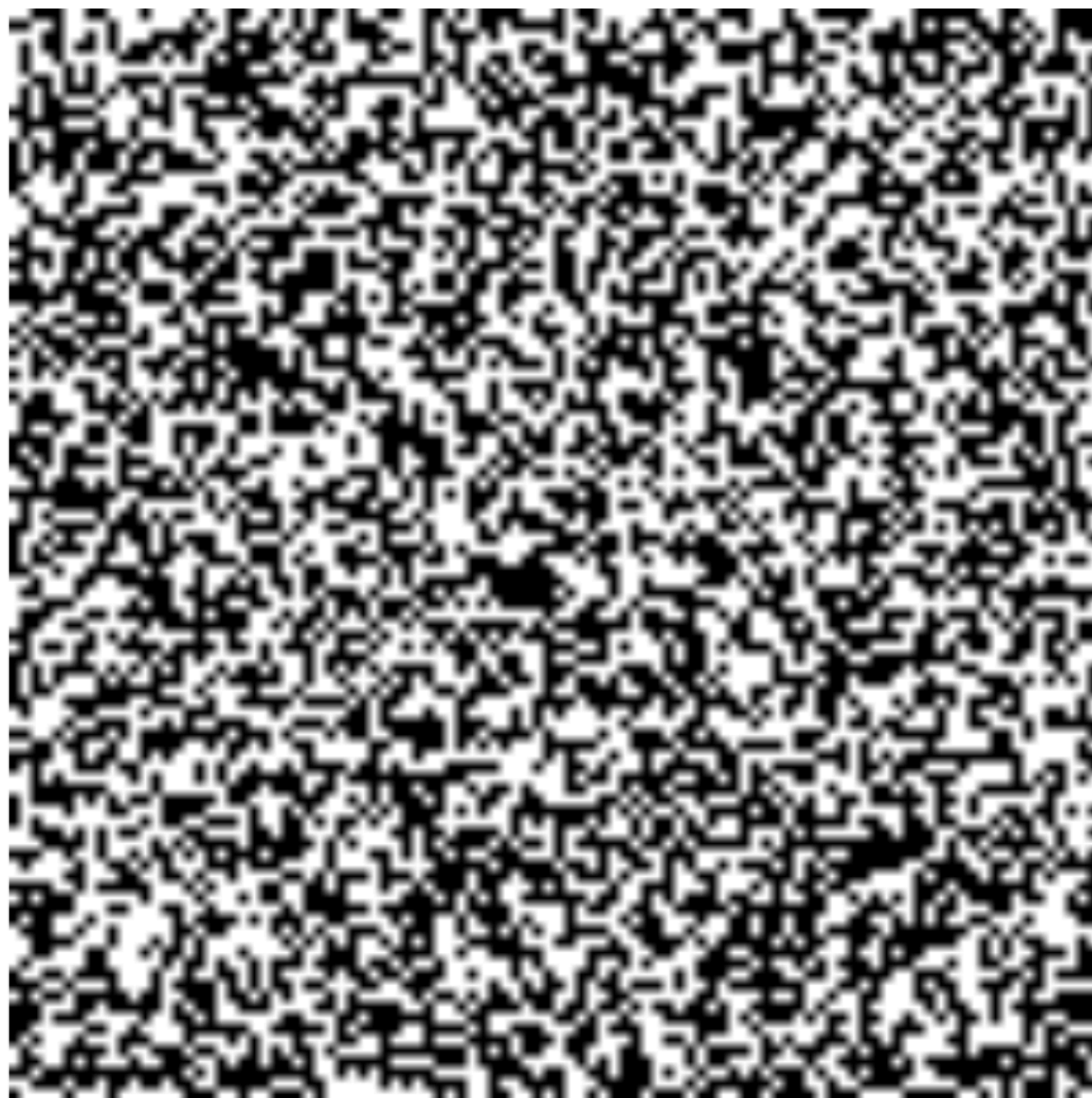
Атаки на шифр Вернама

- Использование не истинно случайно равномерно распределенных ключей
 - Использование повторяющегося укороченного ключа:
 $K = K'K'...K'$ – «XOR-шифр»
Атаки: частотный анализ, на основе открытых текстов.
- Переиспользование ключа:
 $K_1 = K_2$

Переиспользование ключа



Переиспользование ключа



Переиспользование ключа



- $C_1 = P_1 \oplus K_1$

- $C_2 = P_2 \oplus K_2$

- $K_1 = K_2 \implies$

$$\begin{aligned} C_1 \oplus C_2 &= (P_1 \oplus P_2) \oplus (K_1 \oplus K_2) \\ &= P_1 \oplus P_2 \end{aligned}$$

Виды шифрования

- Симметричное – один общий ключ для зашифрования и расшифрования
 - Поточное шифрование
 - Блочное шифрование
 - На основе сетей Фейстеля
 - На основе SP-сетей
 - Прочие
- Асимметричное – различные ключи: как правило, открытый и закрытый

Про поточные шифры

Это случайная последовательность?

01111011001101110001

Про поточные шифры

А это?

000000000000000000000000

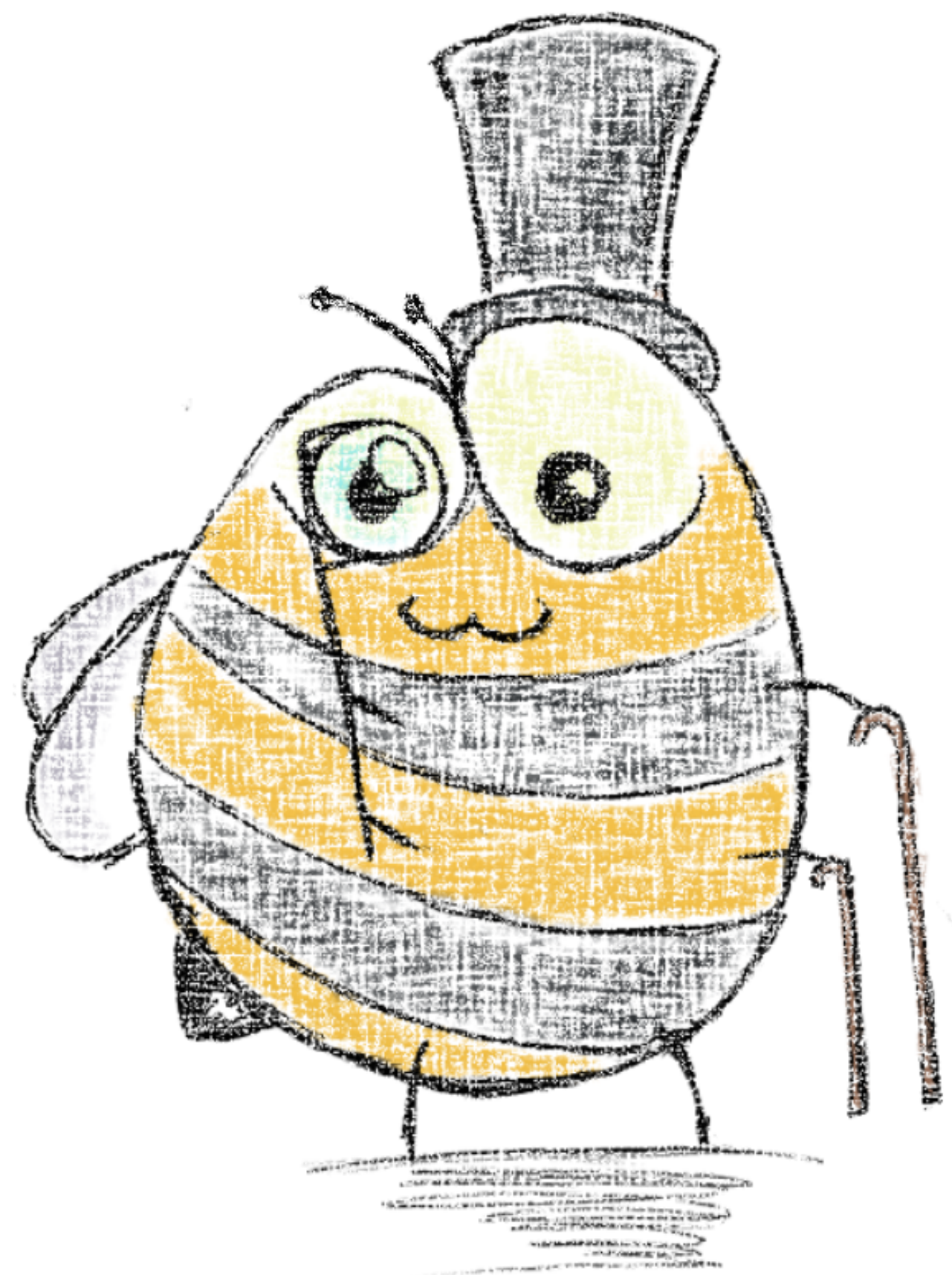
Про поточные шифры

А это?

01010101010101010101

Шифрование

Подробнее про симметричное и асимметричное шифрование —
в последующих лекциях по криптографии



Контакты

Хашаев Артур Акрамович

arthur@khashaev.ru · khashaev.ru · telegram: [inviz](https://t.me/inviz)

МГУ им. М.В. Ломоносова

Лаборатория интеллектуальных систем кибербезопасности

