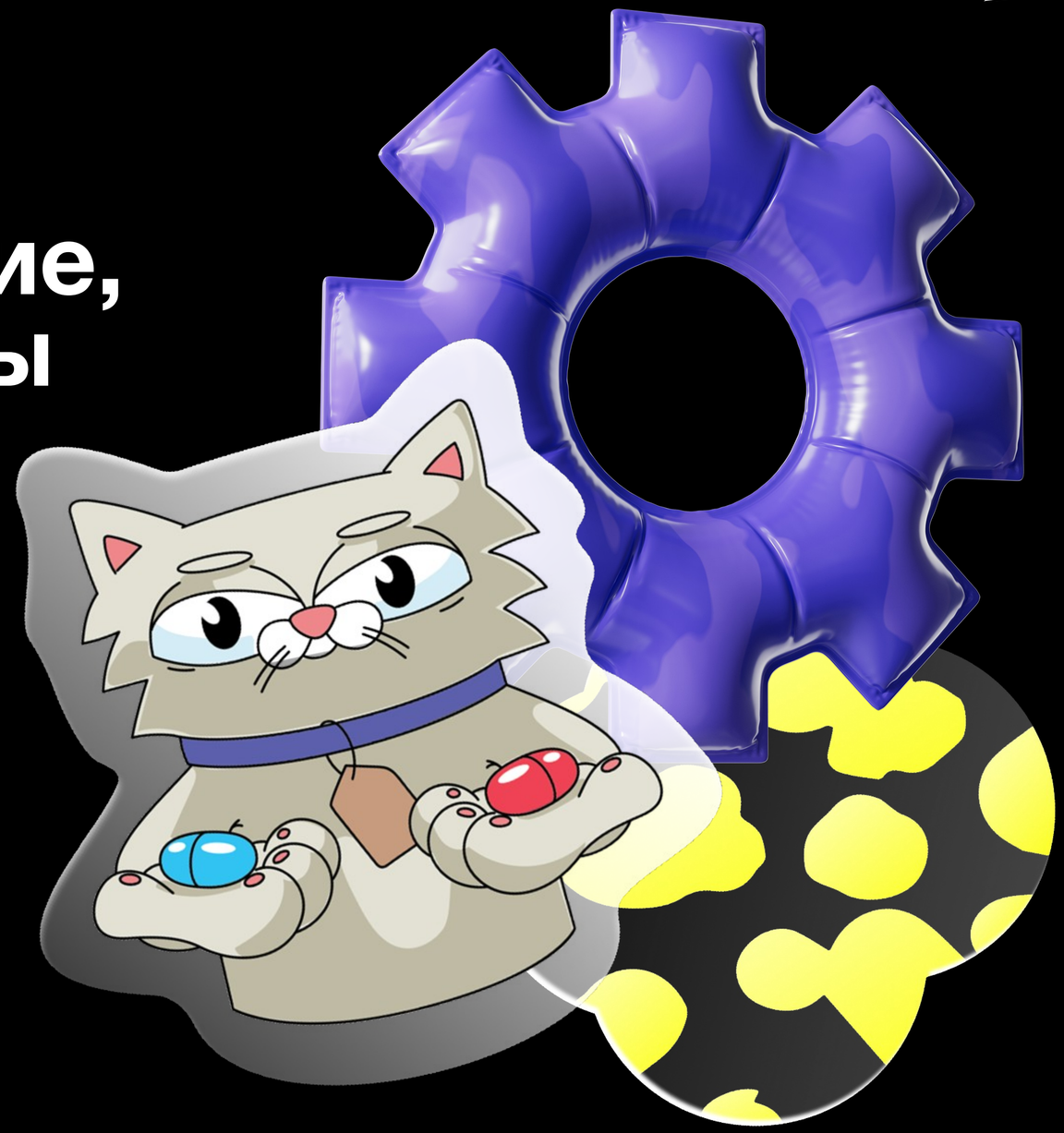
avito.tech Москва 2025

Отложенное шифрование, генетические алгоритмы и функциональное программирование в Web3: Хранение данных в вычислениях



Марат Зимнуров Team Lead

Марат Зимнуров

Руководитель группы разработки в Avito.tech

- ***fun of:*** bsd, wow hc, genetic algorithms, functional programming
- ***education:*** ассистент кафедры ИТиЦЭ в ИГХТУ
- ***role:*** lead full-stack engineer | team/tech/dev lead engineer
- ***project:*** digitable.ru
- ***ex-worker:*** Microsoft, Adjust, Roadmunk, Xsolla, Tproger, Kontur, EPAM Anywhere, Akvelon



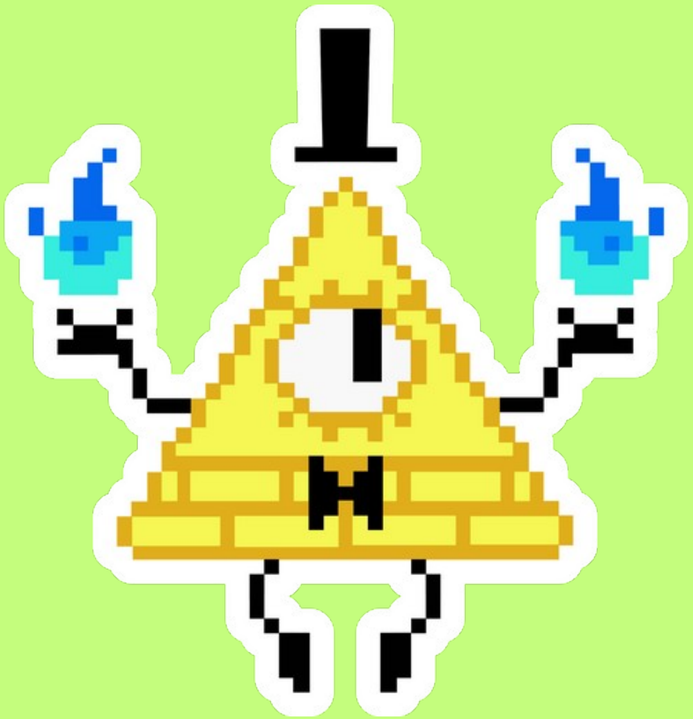
github: https://github.com/the-homeless-god

tg: https://github.com/the_homeless_god

linkedin: https://www.linkedin.com/in/marat-z/

habr: https://habr.com/ru/users/the_homeless_god/

Шифрование



1

2

3

4

От Цезаря до AES:
Как развивалось шифрование

01
Шифр Цезаря

02
DES

03
AES

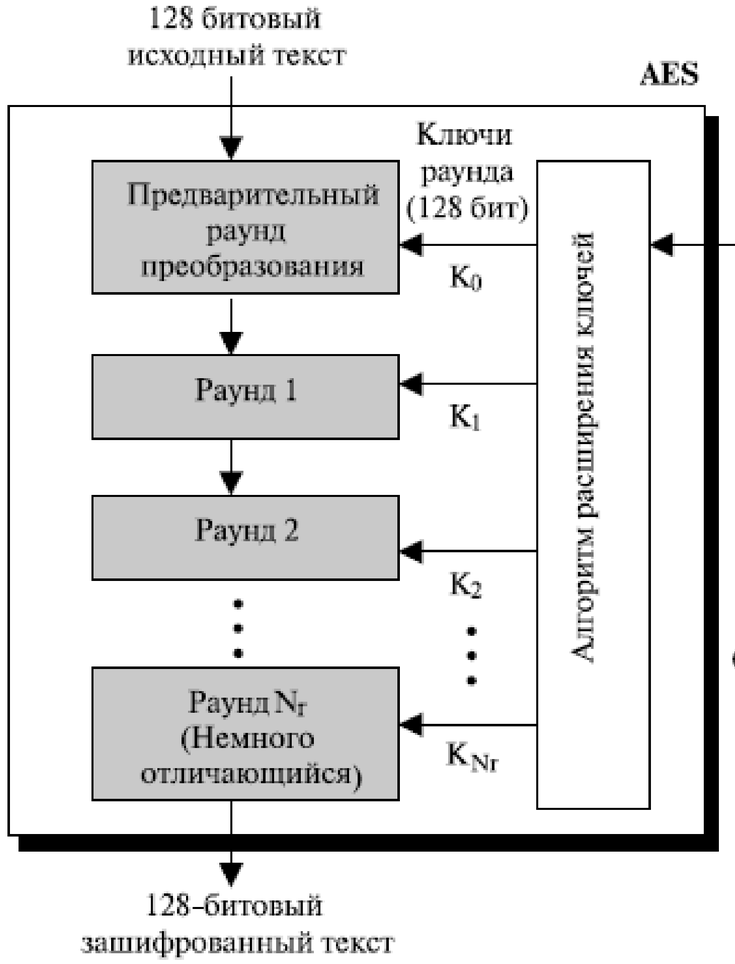
04
Base64



AES: Почему его взлом стоит $3.5 млрд

256 бит
Ключ

10^{77}
вариантов
больше, чем атомов во Вселенной



Nr	Размер ключа
10	128
12	192
14	256

Соотношение между номерами раундов и размерами шифра

Почему взлом **невозможен**:

- Даже квантовые компьютеры не взломают AES-256 до 2030+.
- Стоимость атаки: $3.5 млрд (оценка NIST).

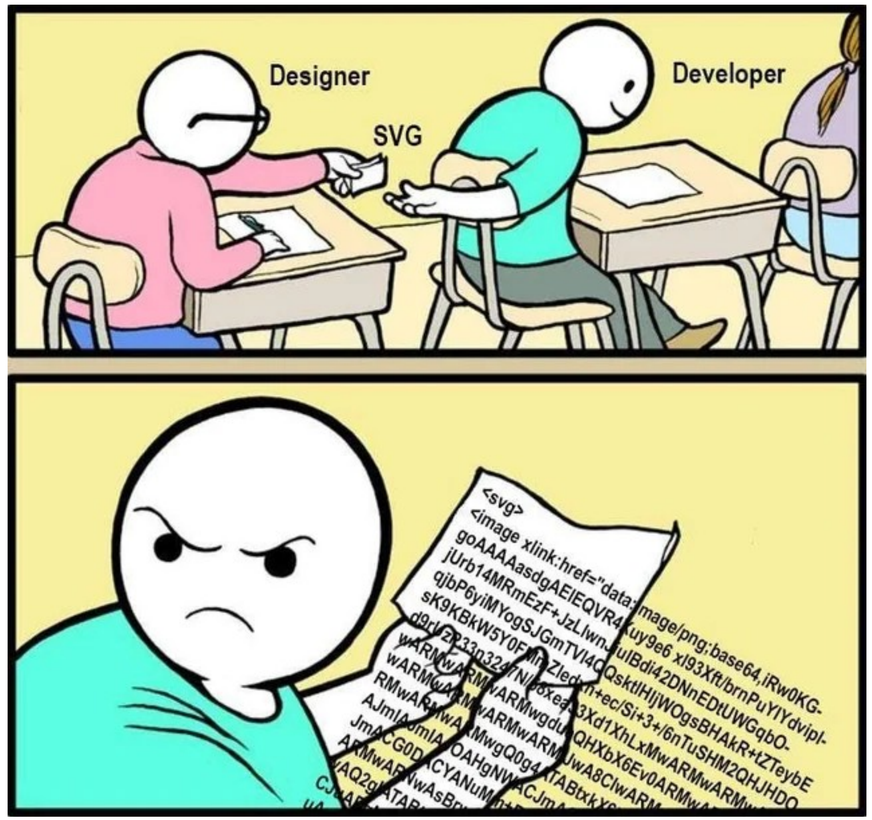
Base64 — «шифрование» для бедных

Это **кодирование**, а не шифрование (нет ключа).

Зачем нужно:

- Передача бинарных данных через текстовые протоколы (HTTP, JSON).

Пример: *echo "hello" | base64 → aGVsbG8K*

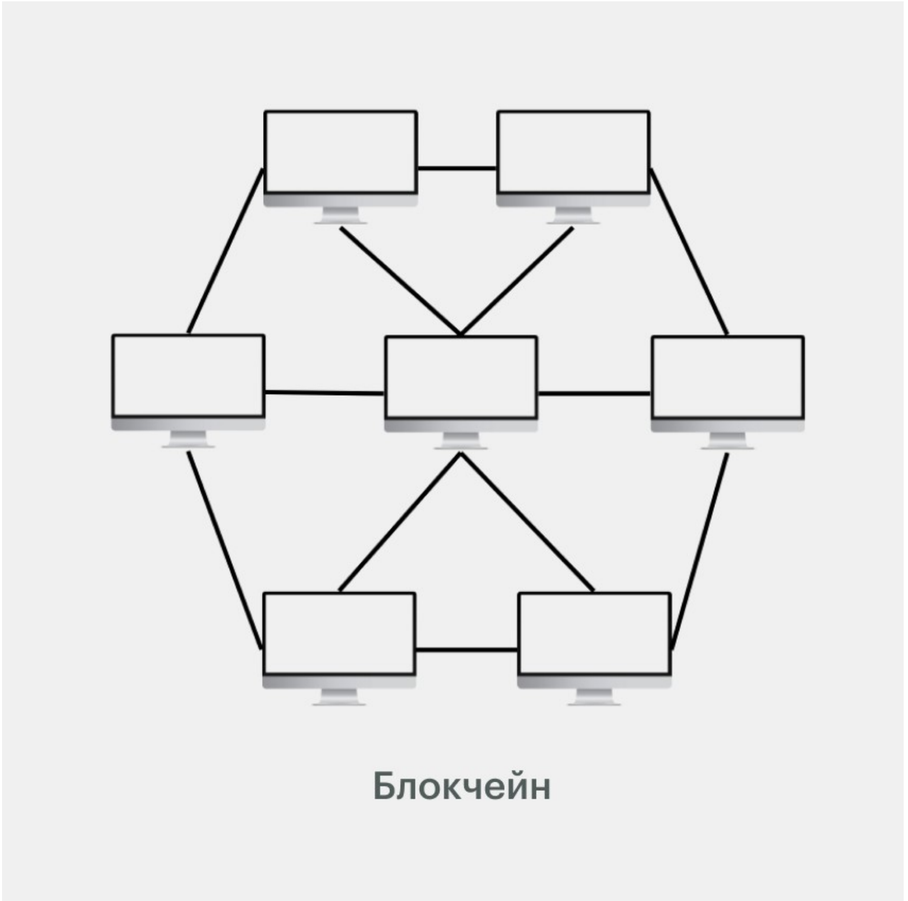


Шифрование в блокчейне

1. Данные в блокчейне открыты (прозрачность > приватность).

1. **Шифруют** только:
   - **Приватные ключи** (AES-256 в кошельках).
   - **Транзакции** в приватных сетях (Zcash, Monero).

1. Base64 тут: Для кодирования данных в смарт-контрактах.



Блокчейн

Emoji-шифрование



1

2

3

4

🔐 Emoji как ключи



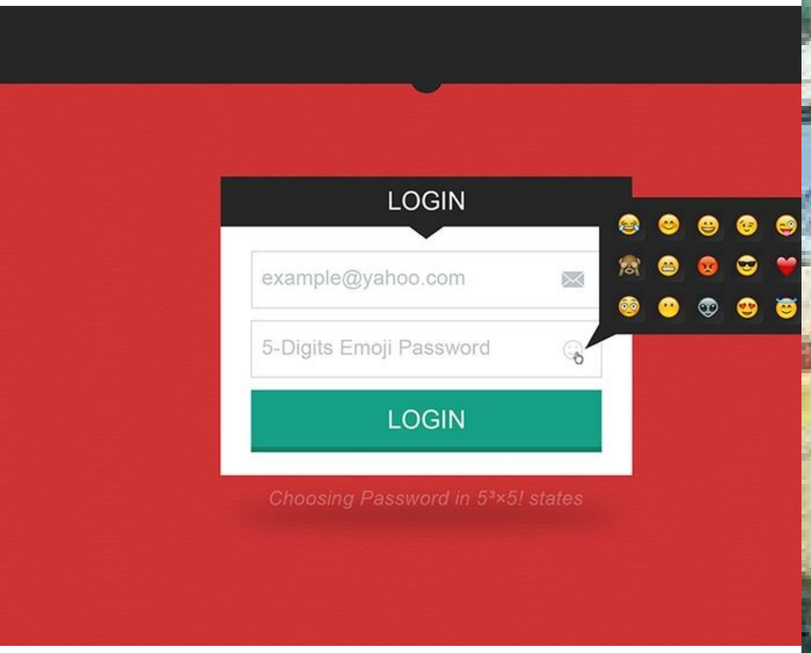
Проблема: Shared Secret в env — это уязвимость.

Решение: Emoji-ключи + генетический алгоритм для поиска сессии

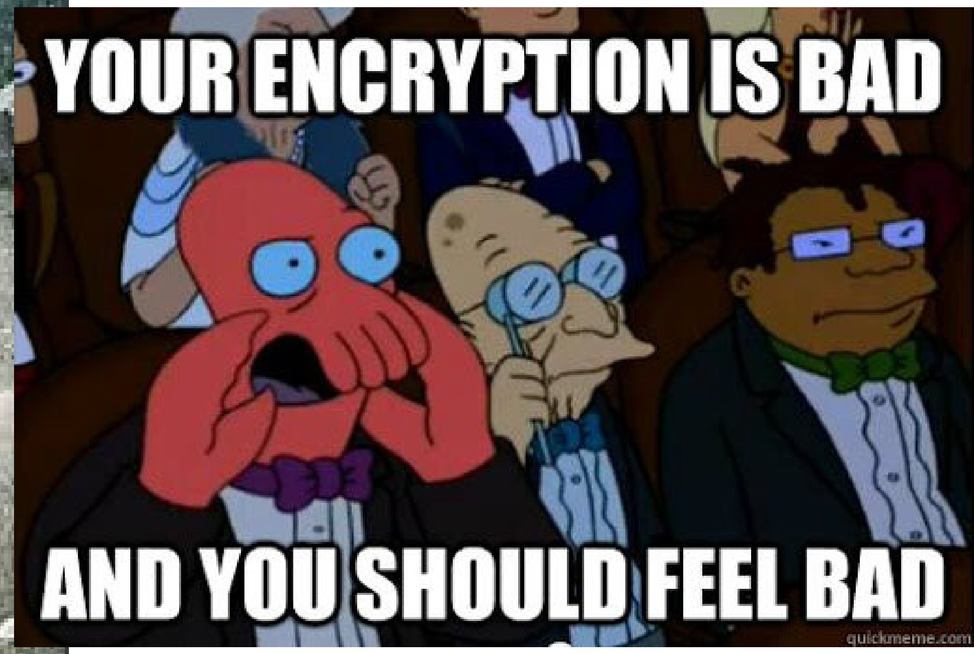
Telegram, Jitsi и Matrix: Почему 🦄🍣⚡ — это новый AES



От теории к практике: Emoji → ДНК → Ключ



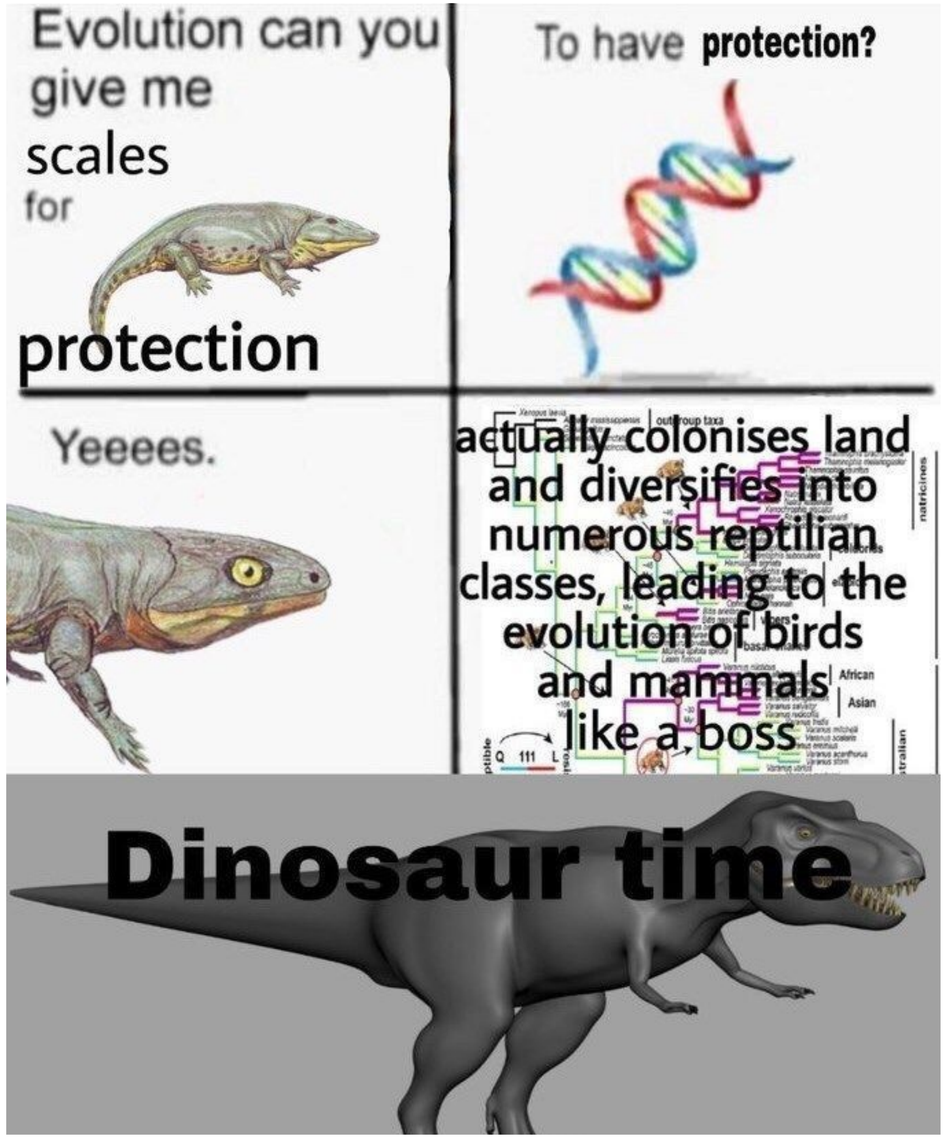




Генетические алгоритмы как конфиг для брутфорса

Генетические алгоритмы (ГА) — это “**умный**” брутфорс, где перебор оптимизируется через эволюцию.

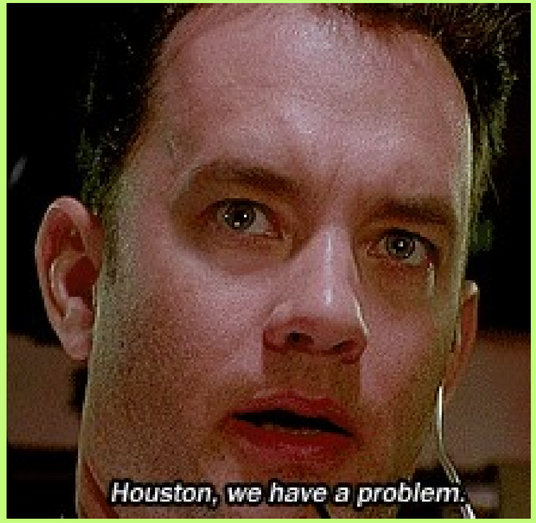
Пример: Взлом AES не полным перебором, а через “скрещивание” перспективных ключей.



Есть проблема пространства

4

2



3

1

Идемпотентные генетические алгоритмы:

Настройка → Шифрование → Гарантии

Проблема: Как гарантировать,
что генетический алгоритм:

- Даст одинаковый результат при одинаковых входных данных.
- Не мутирует состояние на бекенде.

Решение: ФП + строгие параметры ГА.

GA (params) → AlwaysSameOutput



GA + ФП = Отложенное шифрование без боли

Вы знаете AES, но **GA** —
это **“криптография с эволюцией”**:

- **Ключ** = параметры ГА (популяция, мутация).
- **Шифрование** = итеративный GA,
  где результат — это N-ое поколение.



Параметры ГА

```
seed: Int                    -- Фиксирует случайность
populationSize: Int          -- Число кандидатов
mutationRate: Float          -- Вероятность мутации
fitnessFn: Gene → Float  -- Критерий "качества" гена
```

↓

Инициализация

Создать N случайных генов (на основе seed)

↓

Эволюция (поколения)

```
1. Селекция:
-- Отбор топ-10% генов по fitnessFn
   filter (population, fitnessFn)
2. Кроссовер:          -- Скрещивание родителей
   map (parents → crossover(parent1, parent2))
3. Мутация:            -- Случайные изменения
map (genes → mutate(gene, mutationRate, seed))
```

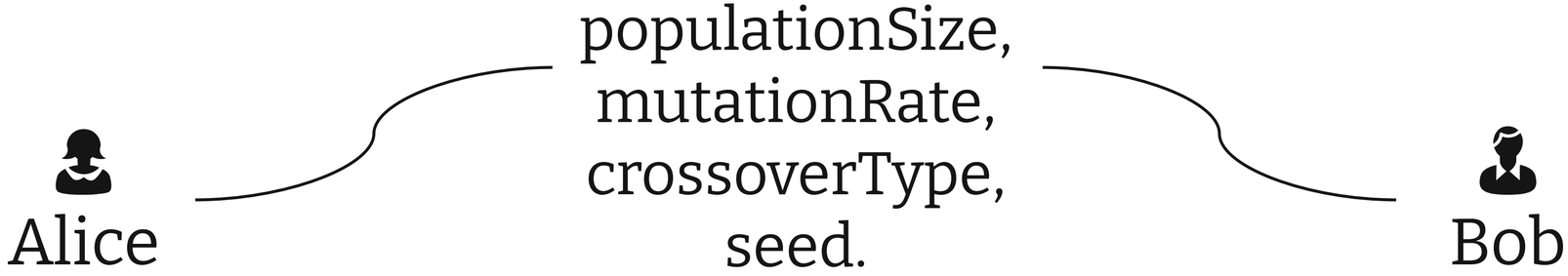
↓

Условие остановки

generations ≥ maxGenerations? → ✓ **Результат**

GA как функция

(для тех, кто не в теме)👆



encrypt(data) = GA(params, data, generations=n)

AES vs. GA: Кто кого?

	AES-256	GA + ФП
Стойкость	10^{177} вариантов	Зависит от params
Гибкость	Фиксированный	Настраиваемый (популяция,мутации)
Идемпотентность	Да	Да (если seed фиксирован)
Применение	Банки, VPN	Web3, P2P, отложенное шифрование

Энтропия ключа ≥ 256 бит:
Размер популяции = 1 000 000 ключей.
Длина каждого ключа = 256 бит.
Итоговая сложность: $\log_2(1\,000\,000 \times 2^{256}) \approx 256$ бит.

Fitness-функция:
Пример: fitness(key) = (SHA3(key) == target_hash).

Параметры алгоритма:
seed, mutation_rate, crossover_type — часть ключа.
Пример: params = {seed: 42, mutation: "flip_0.1%", crossover: "SHA3-based"}.

Когда GA лучше?
- Нужен контроль над процессом (например, остановка на N-ом поколении).
- Данные зависят от контекста (сетевые задержки, TTL).

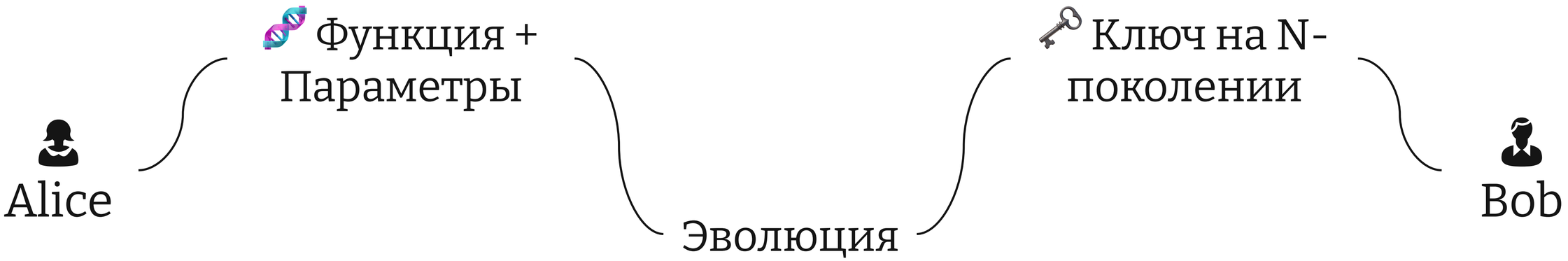
Когда AES лучше?
- Требуется стандартизированная защита (например, платежи).



А что если я хочу чтобы **GA** соответствовал **AES-256** ?

Как GA все находит?

GA сокращает пространство за счет «эволюции», а не грубой силы



Почему **нельзя взломать**:

🔢 **Ключ на выходе:** Миллиард символов (но это не строка, а результат вычислений).

🧩 **Функция:** Уникальна для сессии (можно встроить подпись).

⏳ **Ленивость:** Ключ "материализуется" только при необходимости.

Итоги



1

2

3

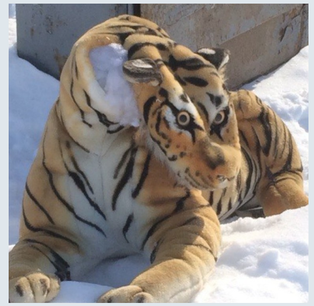
4

Мега крутое с генетическим алгоритмом



- 🧬 Эволюция ключей через поколения
- 🔄 Кастомизация под задачу
- ⏳ 1-5 сек (100k итераций)
- 🧩 Параметры = ключ

Любое другое шифрование



- 🔢 Жесткие мат. алгоритмы
- ⚖️ Стандартизировано
- 🚀 0.1 мс (AES) / 🐢 10 мс (RSA)
- 🔐 Секретный ключ / 🔄 Пары ключей

Что запомнить?

1. GA — это конфиг + генератор,
   а не «волшебная коробка».

1. Сложность взлома: Зависит от параметров
   (например, seed=42 → предсказуемо).

1. Для Web3: GA + ФП = Децентрализованное доверие.



Марат Зимнуров

Руководитель группы разработки







/the-homeless-god



@the_homeless_god

Вы можете хранить ваши данные в отложенных вычислениях

Генетические алгоритмы - это функции,
не знаешь параметров, то и не знаешь результат

Ваши функции сохраняют ссылочную прозрачность –
ваши результаты однозначны

Нельзя просто так взломать то, что генерируют
тысячи поколений по заданному маршруту,
потому что вместо ключей – вы передаете алгоритм