Безопасность ICO контрактов

Александр Половьян
alex@ledgers.world

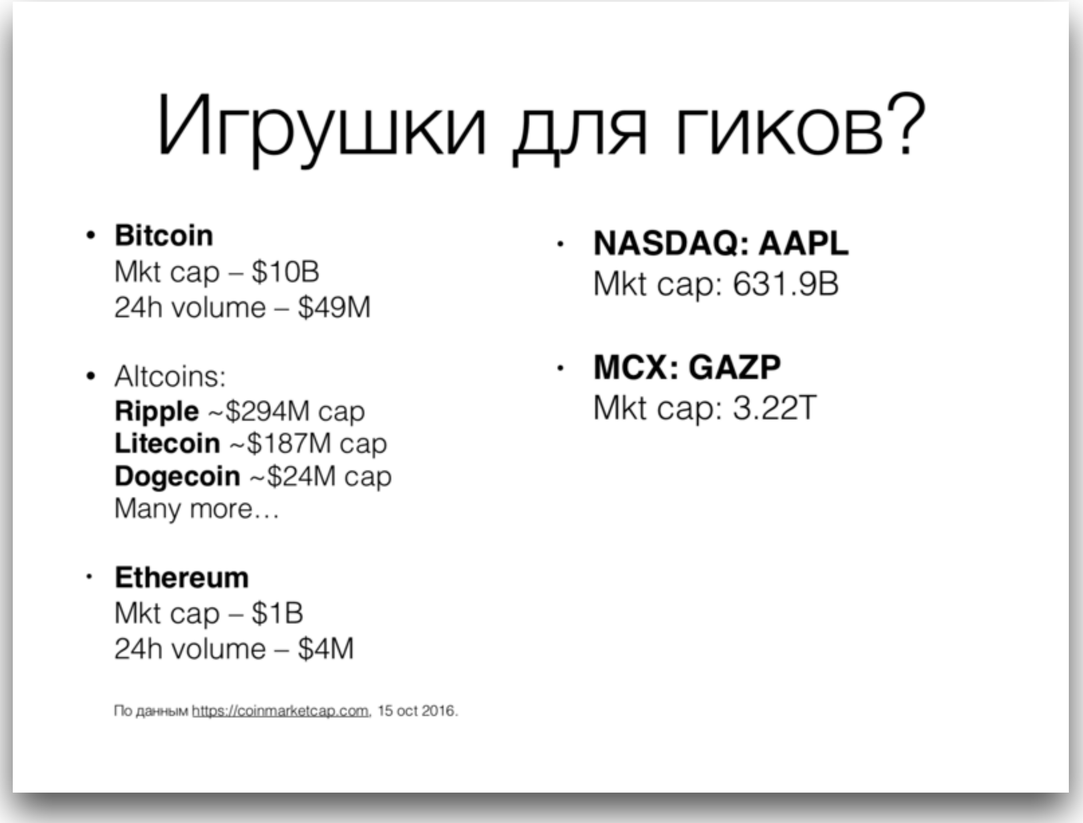
Что не входит в этот курс

- Майнинг
- Спекулятивные сделки
- Финансовые прогнозы
- Соответствие законодательству
- Альтчейны и альткойны

Знания и навыки по результатам курса

- Ethereum: зачем нужен и как работает
- Использование блокчейна в приложениях
- Разработка на языке solidity
- Анализ безопасности смарт-контрактов

Зачем этим заниматься?



- DAO attack
  3 600 000 ETH
  Потери: $60M (сейчас $3B) + репутация

- https://www.coindesk.com/hacks-scams-attacks-blockchains-biggest-2017-disasters/

Блокчейн

Блокчейн

- Криптографически защищенный
  транзакционный
  конечный автомат

- Распределенная система

- Может достичь консенсуса за конечное время

Заблуждения о распределенных системах

- Сеть надежна
- Нет задержки передачи информации
- Ширина канала связи не ограничена
- Сеть безопасна
- Топология сети не меняется
- У сети только один администратор
- Передача данных бесплатна
- Сеть однородна

http://www.rgoarchitects.com/Files/fallacies.pdf

Конечный автомат

- Простейший блокчейн это записи о средствах на счете *“идентификатор счета (bytes32)” — “баланс (uint256)”*
- Идентификатор счета – кошелек
- Состояние конечного автомата:
  все балансы всех счетов

Транзакции

- Переход между состояниями КА – перевод средств
  “Уменьшить баланс счета 0xaaaa на 5 единиц и повысить баланс счета 0xbbbb на 5 единиц”

- Блок – набор операций по переводу средств

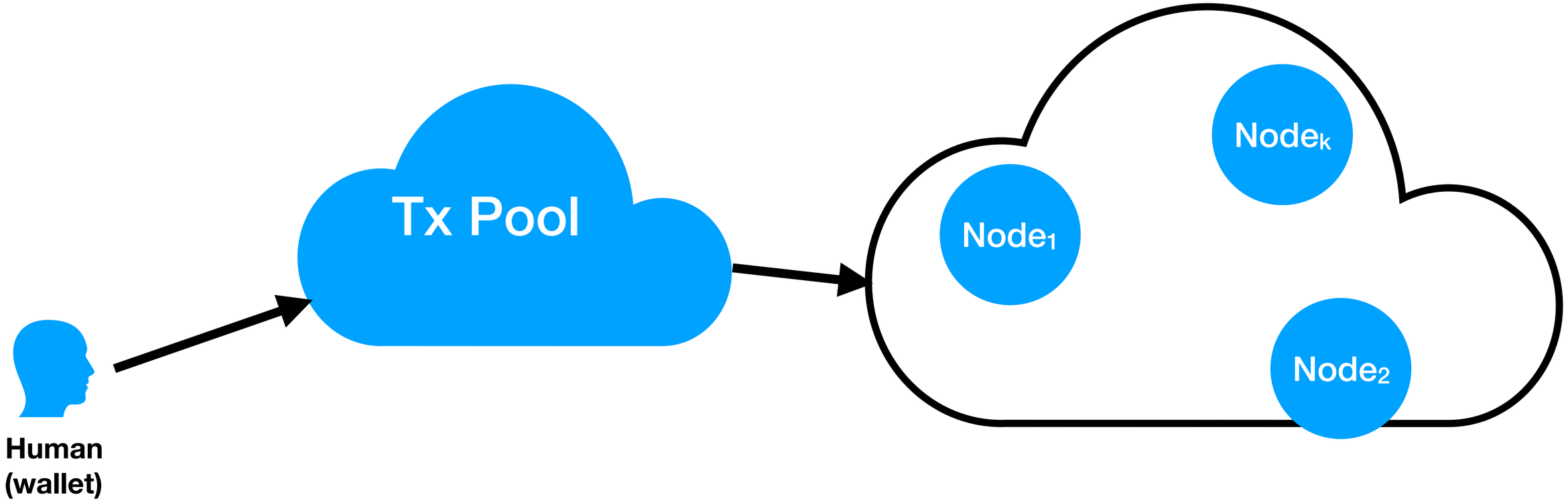
- Валидность транзакции зависит от состояния КА
  нет овердрафта

- Порядок операций в блоке имеет значение
  — double spend
  — frontrunning

Блоки

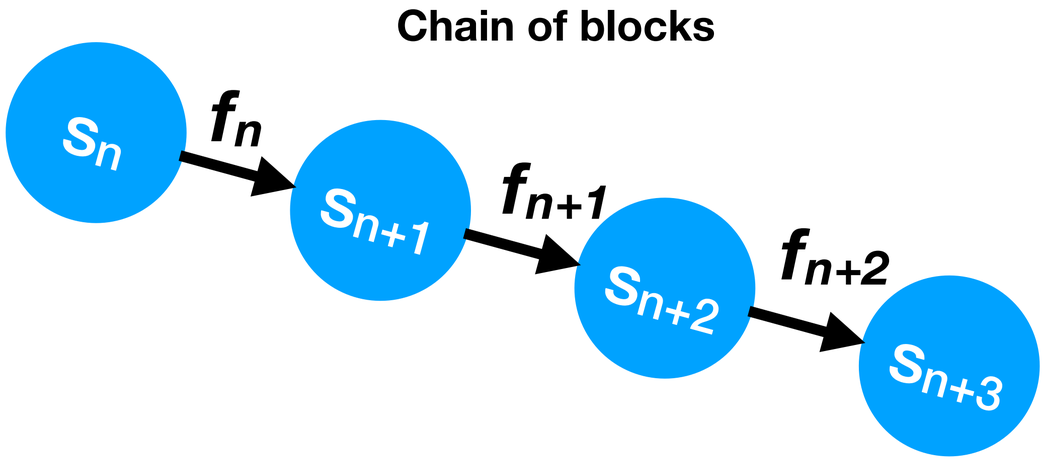
- Состояния КА s_n, s_{n+1}, s_{n+2}
- Блоки это функции перехода f: $f_n(s_n) = s_{n+1}$
- КА, двудольный граф…

Блокчейн



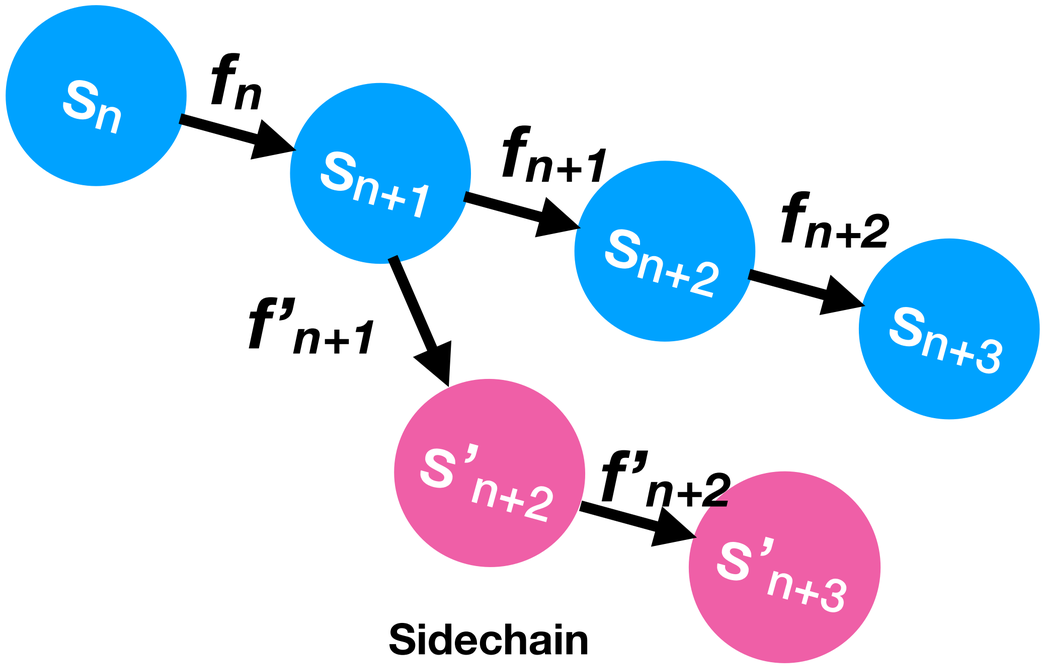
Блоки

- Состояния s_n, s_{n+1}, s_{n+2}
- Блоки это функции перехода f: $f_n(s_n) = s_{n+1}$
- KA, двудольный граф…



Блоки

- Состояния s_n, s_{n+1}, s_{n+2}
- Блоки это функции перехода f: $f_n(s_n) = s_{n+1}$
- KA, двудольный граф…



Криптография

- Пользователи → заявки на перевод → узлы сети
- Узлы выбирают заявки (формируют блок),
  обновляют состояние и
  сообщают соседям
- Узлы экономически заинтересованы формировать блоки
- Для предотвращения злоупотреблений, для каждого блока надо решить вычислительно дорогую задачу

Криптография (2)

- Сеть одноранговых узлов
- Только владелец счета может управлять его средствами
- Заявление на перевод можно оставить на любом узле
- Для авторизации ЭЦП
  приватный ключ → публичный ключ
  публичный ключ = кошелек

Ethereum

Новый тип кошельков

- Блокчейн бухгалтера
  “идентификатор счета (bytes32)” — “баланс”

- Блокчейн программиста
  “адрес (bytes32)” — “баланс” — “байткод”

Ничего не изменилось (почти ничего)

- Состояния s_n, s_{n+1}, s_{n+2}
- Блоки это функции перехода f: $f_n(s_n) = s_{n+1}$

- Пользователи могут не только переводить средства, но и загружать код
- И исполнять его внутри Ethereum Virtual Machine
- Результатом исполнения может быть изменение состояния

EVM

- Виртуальный стековый процессор
- Включен в каждую ноду
- Исполняет специальный байткод ассемблерного вида
- Исполнение может изменять состояние блокчейна
- Тьюринг-полный

Проблема останова

- Программа на полном по тьюрингу языке может реализовать бесконечный цикл
- Для того что бы определить, содержит ли программа бесконечный цикл надо её выполнить
- …и дождаться завершения
- В Ethereum решается тем что за каждую выполненную инструкцию нужно заплатить

Установка

Окружение

- Virtual Box
  http://virtualbox.org

- go-ethereum
  https://github.com/ethereum/go-ethereum
  https://github.com/ethereum/go-ethereum/wiki/Building-Ethereum

- Альтернативный вариант: установка через Docker
  https://hub.docker.com/r/ethereum/client-go/

Запуск

- Первый блок — genesis.json
  $> geth --datadir myfolder init genesis.json
- Запуск узла
  $> geth --datadir myfolder
- Подключение к узлу
  $> geth attach ipc:myfolder/geth.ipc

```
{
  "config": {
    "chainId": 73456378946589376 45,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "nonce": "0x0000000000000033",
  "timestamp": "0x0",
  "gasLimit": "0x8000000",
  "difficulty": "0x10000",
  "mixhash":
"0x0000000000000000000000000000000000000000000000000000000000000000"
}
```

Проверка работоспособности

- > personal.listAccounts
  [ ]

- > personal.newAccount()
  Passphrase:
  Repeat passphrase:
  “0x8948b03bdf5ce035f731a09bd691e8c051286f56"

- > personal.listAccounts
  [“0x8948b03bdf5ce035f731a09bd691e8c051286f56”]

Майнинг

- > miner.setEtherbase(personal.listAccounts[0])
  true
- > eth.coinbase
  "0x8948b03bdf5ce035f731a09bd691e8c051286f56"
- > miner.start()
  true
- > eth.getBalance(personal.listAccounts[0])
  1.5265625e+21

Что еще можно сделать?

- https://github.com/ethereum/wiki/wiki/JavaScript-API

Ссылки

- https://github.com/ethereum/wiki/wiki/White-Paper
- http://gavwood.com/Paper.pdf

Домашнее задание

- Как будем общаться вне курса?
- Приватная сеть
  - Развернуть приватную ноду
  - Добавить вторую ноду к приватной сети
- Подключиться к mainnet’у
- Запустить сеть на спецкурс