Безопасность ICO контрактов (2)

Александр Половьян
alex@ledgers.world

Yellow paper

- ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER BYZANTIUM VERSION

- https://ethereum.github.io/yellowpaper/

Парадигма блокчейн

- Ethereum: транзакционный конечный автомат
- Genesis — начальное состояние
- Транзакция — валидный переход между двумя валидными состояниями

- $\sigma_{t+1} \equiv Y(\sigma_t, T)$
  Y – функция перехода (вычисление)
  σ – состояние (информация)
  T – транзакция

Парадигма блокчейн

- Блок — комбинация транзакций

- $\sigma_{t+1} \equiv \Pi(\sigma_t, B)$
  B – блок
  Π – функция блокового перехода

- $B \equiv (..., (T_0, T_1, ...))$

- $\Pi(\sigma, B) \equiv \Omega(B, Y(Y(\sigma, T_0), T_1)...)$
  Ω – функция финализации блока

- Майнинг – proof of work

Парадигма блокчейн

- Wei — базовая ценность внутри сети Ethereum
- Целочисленная арифметика
- 1 ETH = 10^{18} wei

Парадигма блокчейн

- Форк – состояние сети, при котором цепочка узлов становится деревом

- Форк нежелателен и неизбежен

Структуры в Ethereum

- Состояние сети (world state) – отображение адресов на их состояния

- Адрес – 160 битный идентификатор

- KEC — Keccak256 алгоритм хеширования

- RLP — низкоуровневый формат байтового представления данных (https://github.com/ethereum/wiki/wiki/RLP)

- trie – Merkle tree
  иммутабельная структура данных, дерево из хешей пар

Состояние аккаунта

- Состояние адреса – структура:

    1. $\sigma[a]_n$, *nonce* – количество транзакций отправленных с адреса

    2. $\sigma[a]_b$, *balance* – количество wei на счете

    3. $\sigma[a]_s$, *storageRoot* – хеш хранимых данных для trie
    $TRIE(L^*_I(\sigma[a]_s)) \equiv \sigma[a]_s$, виртуальный, не сериализуется

    4. $\sigma[a]_c$, *codeHash* – хеш байткода
    $KEC(b) = \sigma[a]_c$, b - код

Специальные состояния аккаунтов

- Обычный аккаунт: *codeHash = KEC(())*
  Если есть код – аккаунт смарт-контракта

- *EMPTY(σ,a)* ≡ $\sigma[a]_c = KEC(\text{“”}) \wedge \sigma[a]_n = 0 \wedge \sigma[a]_b = 0$
  пустые аккаунты – нет кода, нет денег, нет истории

- *DEAD(σ, a)* ≡ *(σ[a] = ∅ ∨ EMPTY(σ, a))*
  мертвые аккаунты — состояние пустое или несуществующее

Компактное представление аккаунта

- $L_S(\sigma) \equiv \{p(a) : \sigma[a] \neq \varnothing\}$
- $p(a) \equiv (KEC(a), RLP(\sigma[a]_n, \sigma[a]_b, \sigma[a]_s, \sigma[a]_c))$
- $\forall a: \sigma[a] = \varnothing \vee (a \in B_{20} \wedge v(\sigma[a]))$
- $v(X) \equiv x_n \in N_{256} \wedge$
  $x_b \in N_{256} \wedge$
  $x_s \in B_{32} \wedge$
  $x_c \in B_{32}$; валидность аккаунта

Транзакция

- Транзакция – единичная криптографически подписания инструкция составленная вне Ethereum

- Передает сообщения либо создает аккаунт с хранимым кодом

Структура транзакции

1. T_n, *nonce* – количество транзакций отправленных с адреса отправителя транзакции

2. T_p, *gasPrice* – сколько *wei* отправитель заплатит за каждую единицу *gas*

3. T_g, *gasLimit* – максимальное количество *gas*

4. T_t, *to* – получатель сообщения, Ø для создания нового аккаунта

5. T_v, *value* – пересылаемые средства

6. *v, r, s* – подпись отправителя

7. T_i, *init* – EVM-код (только для создания аккаунтов)

8. T_d, *data* – байты (только для отправления сообщений)

Чек

- $R \equiv (R_u, R_b, R_l, R_z)$ – чек (receipt) транзакции
  - R_u – использованный газ (кумулятивно)
  - R_b – хеш логов
  - R_l – последовательность логов $(O_0, O_1, \ldots)$
  - R_z – status code

Лог

- $O \equiv (O_a, (O_{t0}, O_{t1}, \ldots), O_d)$
- O_a – адрес логгера
- $(O_{t0}, O_{t1}, \ldots)$ – топики (индексирование)
- O_d – логируемые данные

Блок

- $B \equiv (B_H, B_T, B_U)$
- B_H – заголовок (описан выше)
- B_T – список транзакций
- B_U – список заголовок оммеров

Заголовок блока

- H_p, *parentHash* – хеш заголовка предыдущего блока
- H_o, *ommersHash* – хеш оммеров этого блока
- H_c, *beneficiary* – адрес майнера
- H_r, *stateRoot* – хеш trie заголовка
- H_t, *transactionsRoot* – хеш trie узла с транзакциями
- H_e, *receiptsRoot* – хеш trie узла с чеками транзакций
- H_b, *logsBloom* – логи блока
- H_d, *difficulty* – сложность блока, зависит от сложности предыдущего блока и времени

Заголовок блока (2)

- H_i, *number* – порядковый номер блока
- H_l, *gasLimit* – газ лимит блока
- H_g, *gasUsed* – использованный газ
- H_s, *timestamp* – Unix time блока
- H_x, *extraData* – дополнительные данные
- H_m, *mixHash* – хеш подтверждения майнинга
- H_n, *nonce* – хеш подтверждения майнинга

Собственная валидность блока

- Соответствует включенным в блок оммерам
- Соответствуют хешам транзакций
- Исполнение упорядоченного набора транзакций B_T переводит состояние предыдущего блока в другое валидное состояние

Валидность заголовка блока

- Номер блока больше на единицу
- Сложность соответствует спецификации
  - ice age, difficulty bomb – PoS мотиватор
- Время больше времени предыдущего блока
- Nonce блока соответствует условию сложности (майнинг)

Gas

- *gasLimit* — отправитель транзакции покупает столько gas за *wei*
- по цене *gasPrice*
- использованный *gas* переводится в *wei* и зачисляется на счет *beneficiary* (майнер)
- *wei* за неиспользованный газ возвращается обратно
- покупатель выставляет любой gasPrice
- майнер может игнорировать любую транзакцию