Short-term smart-contracts, simple and powerful.

- Presentation plan:
  - smart-contract deployment and processing in Ethereum network
  - short-term and long-term contracts
  - some business cases for short-term contracts:
    - "invoice-paid" contract
    - "commit-reveal" contract
    - "one-time multisig" contract
  - problems and advantages

ethereum

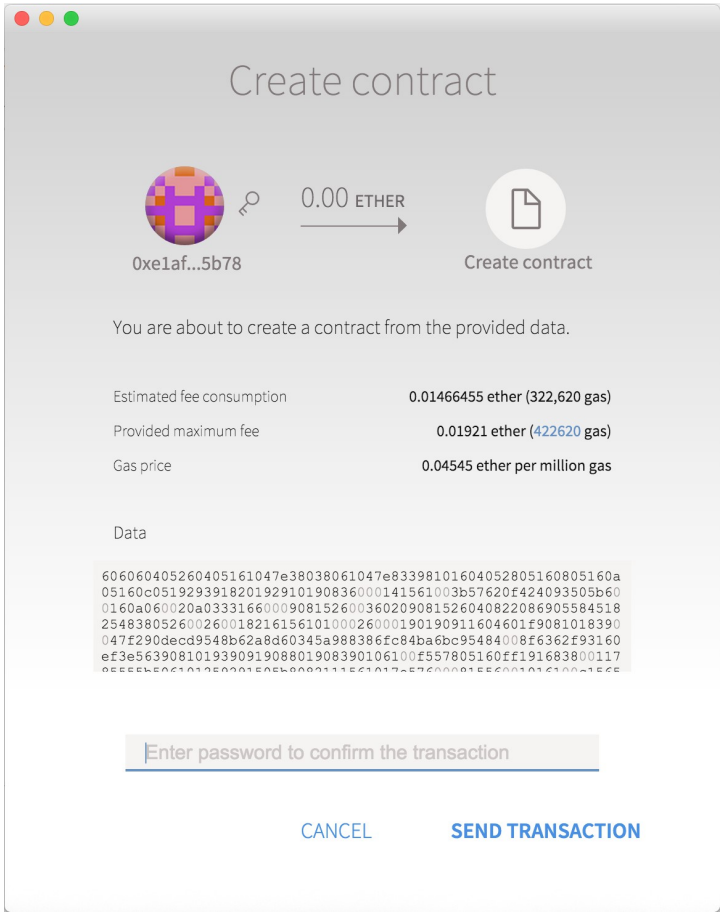
smartz

MixBytes()

Ethereum smart-contract deployment and processing, simplified view

• can imagine the process like deployment of new web-site in WWW, such as:

- code of this "site" cannot be changed
- every request is a undeniable transaction, saved forever in blockchain
- every request from user to this "site" can contain any amount of funds
- every request is fully visible to anyone
- every request, even unsuccessful (error in contract, not enough funds) is also fully visible to anyone



• after network "accepts" code of new contract, contract receives its own address, similar to creator's address, and now able to receive ether

• deployed contract can be also imagined, as automatized wallet, that can send funds outside if some internal code decides so

• if you want to call contract (execute one of its functions) you must create transaction with some ether inside, and send it to contract's address with the name of called function

Ethereum smart-contract deployment and processing, more technical view

- deploying contract
  - client prepares transaction to deploy contract
  - client put contract code and start parameters (price and ttl) into transaction
  - client signs the transaction with his secret key (user wallet, or code on web page can do this)
  - client sends the transaction to any Ethereum node
  - "any Ethereum node" broadcast the transaction using p2p network to many nodes, including block-producing
  - block-producing node takes transaction, validates contract code
  - block-producing node puts contract code into currently producing block with other valid transactions
  - block-producing node "finish" block with "proof-of-work" or "proof-of-stake"
  - block-producing node publish its new block to p2p network
  - "any other node" downloads new block
  - "any other node" checks if block is valid, and add it to chain
  - "any other node" "applies" every transaction to it's own local database (if block resides in main chain)
  - "any other node" now has our contract code, placed at the same address on all nodes

- call contract code
  - client see contract code in blockchain (in one of blocks) and wants to send some ETH in this contract
  - client creates transaction with some ether, signs it
  - client sends transaction to p2p network
  - block-producing node takes transaction, and process it
  - block-producing node actions can be imagined as:
    - block-producer calls function in contract-receiver and pass the data form transaction into callee function
    - contract "knows" that somebody from address A sent X ETH into it, and can do something with this ether, or with internal contract variables
    - many block-producers execute the same program with same input data, and save same data into their databases
  - block-producing node "executes" contract code, using data from transaction, and save results in resulting block
  - same as in previous scheme (block propagation on network, validation by other nodes, choice of main chain, applying transaction-emitted changes in local databases on nodes)

Short-term and long-term contracts

- long-term contracts
  - contains data about many entities
  - have long lifetime
  - one contract for many users interaction

- long term contracts examples:
  - token
  - multisig
  - equity

- long-term contracts deployed once, their addresses are saved in many sources for a long time
  - examples: token contract address, saved on crypto exchanges

- short-term contracts
  - contains only few variables
  - have short lifetime
  - once-per-user-or-task

- short term contracts examples:
  - invoice-paid
  - commit-reveal
  - one-time multisig

- short-term contracts can be used to present single commodity, single invoice, one-time event fixation and similar

Short-term contracts: "invoice-paid"

- can be simply the analog of cash register in shop

- Example with mobile phones:
  - shop publishes such "invoice-paid" contract separately for each phone in shop, as short-term contracts with limited lifetime (1 day)
  - shop publishes new contracts every day, according to its price strategy
  - after deployment, contract waits until someboy send a payment to it
  - buyer simply send enough ether to given "invoice-paid" contract
  - contract takes buyer's ether, split it into three parts: "price", "tax", "charge" and pays "price" to shop, "tax" to tax collector, "charge" back to client, and becomes inactive (returns an error on every request, but still present on blockchain)
  - the history of succeded transactions to given contract is, technically, analog of cash register machine
  - the history of unsucceeded transactions is a history of unsusccessful trials to sell phones, with all needed data (maybe anonymized)
  - time-to-live (TTL) parameter is a common way to restrict amount of such one-time contracts, allowing to build fast and functional search engines over it
  - after TTL is expired, contract counts as inactive (returns an error on every request, but still present on blockchain)

Short-term contracts: "commit-reveal"

- can be used as one-time secret storage, that can be opened only "once" by target user with a "magic" word

- Example with delivering phone to user:
  - shop generates secret word "yoyo", and calculates it's hash
  - shop sends phone to customer, and creates contract "commit-reveal" with this hash(not a word "yoyo"!) and address of courier, that will receive payment for delivery
  - contract will pay ether to courier only if he will present "yoyo" to contract until delivery deadline and from his own address
  - shop sends a word "yoyo" to customer
  - cutomer receives bought phone and reveal secret word "yoyo" to courier
  - courier sends "yoyo" to contract and receives his ether
  - in case of deadline or error, shop can return his funds after deadline

Short-term contracts: "one-time multisig"

- can be used in any deal with 3 participants, such as escrow, letter of credit, and any scheme with third trusted party

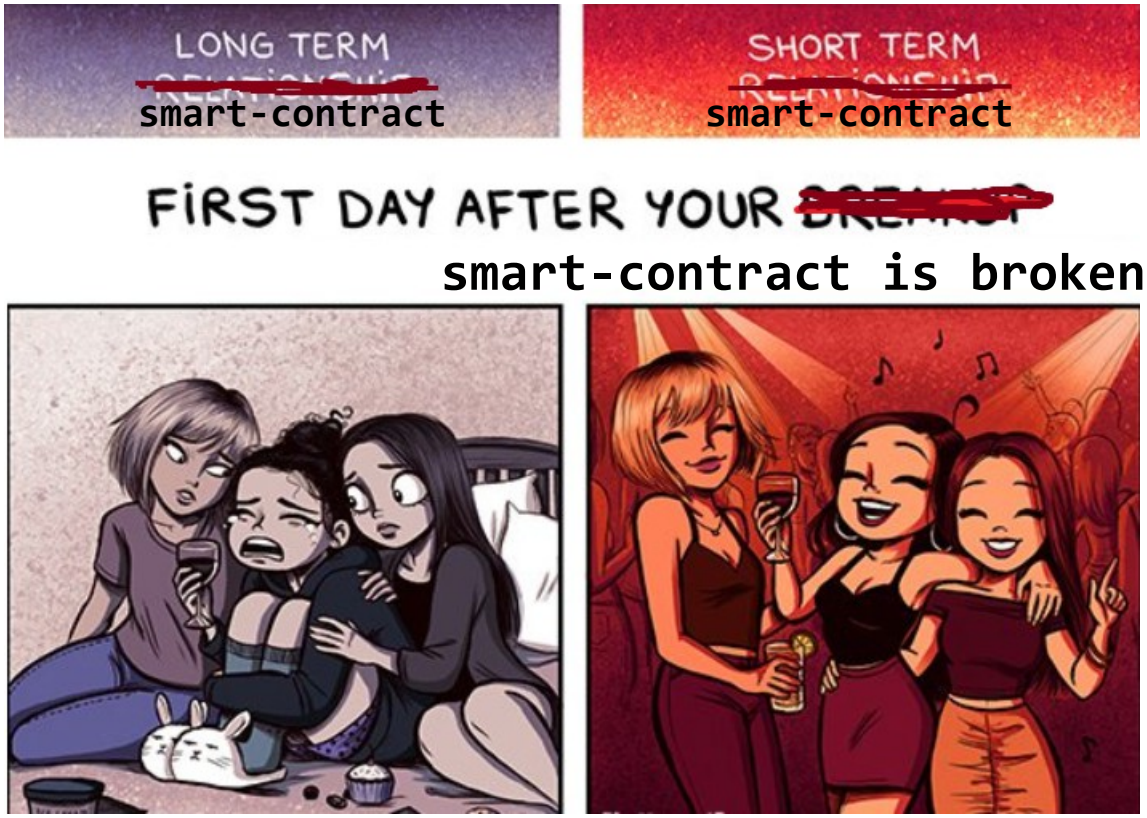
- Example with bank letter of credit:
  - "multisig" term in our case means: "contract will send funds to given address only, if it will see 2 of 3 signatures from group of 3 addresses, passed to contract on creation)
  - three participants: seller, buyer, bank
  - seller should receive funds only if 2 of 3 participants agreed
  - seller agreed to "release"(and take) the funds, "freezed" on multisig contract always
  - buyer agreed to "release" funds only if he is absolutely sure, he want to unlock funds, but he doesn't have to
  - bank agreed to "release" funds only if papers are ok
  - buyer creates "one-time multisig" contract, places ether into it, and adds bank and seller as co-signers
  - seller wants to take ether and sends transaction to contract with his signature and request to withdraw ETH. 1 of 3 signatures is ready
  - buyer can agree with seller (no bank needed), and send approving transaction, "releasing" funds
  - bank can check papers and send approving transaction, "releasing" funds
  - if nothing happens for enough long time, buyer can simply take his funds back

Advantages and problems of short-term contracts

- Advantages of short-term contracts:
  - simplicity:
    - more secure
    - cheaper (comparing to long-term contracts)
    - unified interfaces
    - simple portability to other blockchains
  - used once
    - simplify fixes in contracts' code
    - allows to purge items from local database
  - time-to-live
    - allow fast search in blockchain (we can simply skip all items older, than TTL)

- Problems of short-term contracts:
  - gaz, gaz, gaz
    - profitable only for expensive deals
  - a lot of dead code in blockchain
    - not optimal for blockchain performance
  - complex deploy infrastructure
    - need to use deployment and monitoring services around blockchain, such as smartz.io

:)


LONG TERM
smart-contract
SHORT TERM
smart-contract
FIRST DAY AFTER YOUR
smart-contract is broken

Thanks :)

Sergey Prilutskiy

Chief of Research @ MixBytes() & Smartz

MixBytes()

smartz

https://smartz.io

https://t.me/smartz_ru