Технология блокчейн. Избыточные циклические коды в протоколах криптовалют

Жиляков Сергей ККСО-01-20

27 февраля 2025 г.

Введение

Определение из ISO 22739:2024.

Определение (blockchain)

distributed ledger (реестр) (3.23) with confirmed blocks (3.10) organized in an append-only, sequential chain using hash links (3.47).

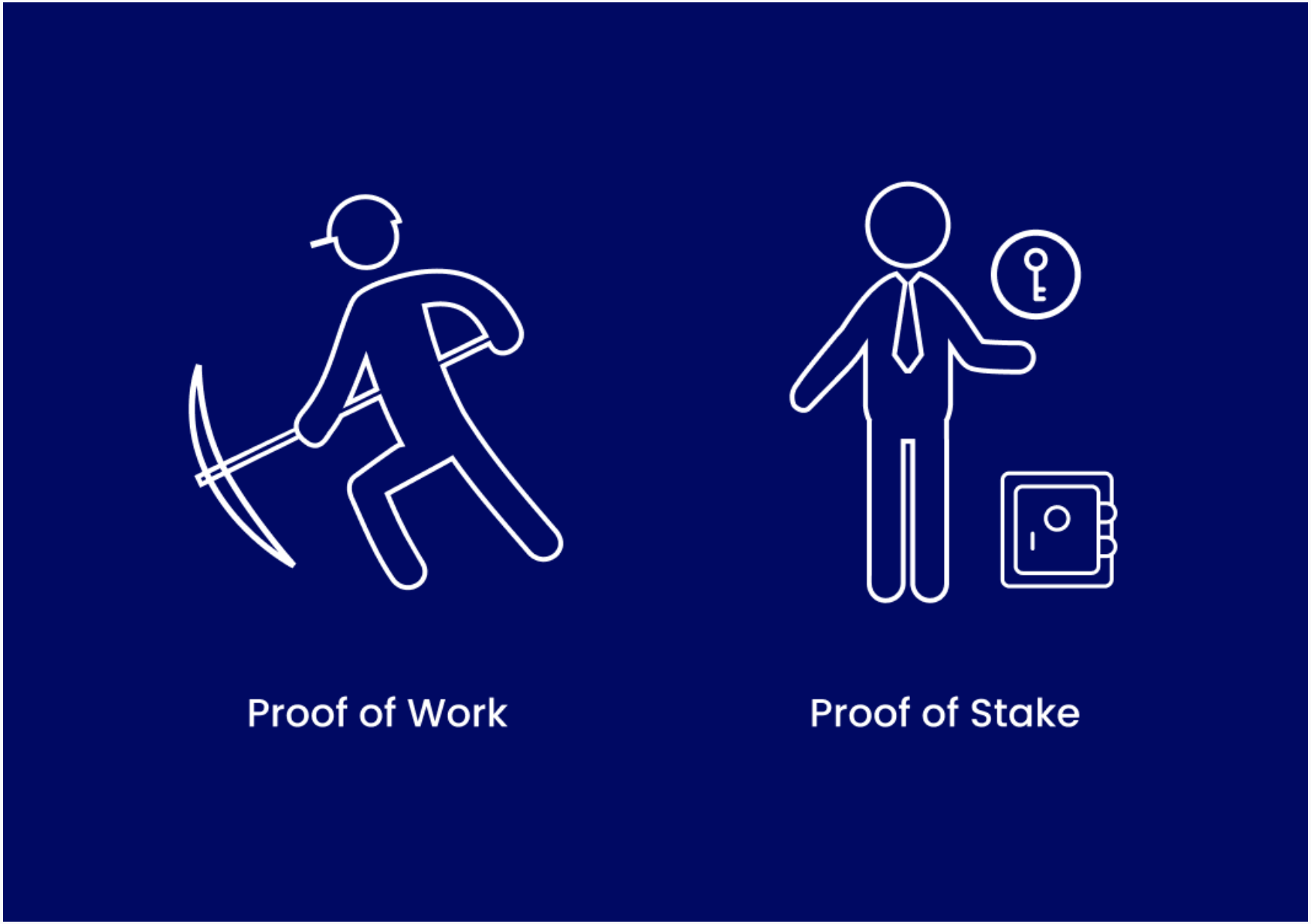
Введение

Nakamoto, S. (2008) *Bitcoin: A Peer-to-Peer Electronic Cash System*

https://bitcoin.org/bitcoin.pdf

…electronic payment system based on cryptographic proof instead of trust…

Введение



Введение

Неизменяемость. Транзакции в существующем блокчейне нельзя изменить; можно лишь создать разветвление (fork) блокчейна, начиная с некоторого блока.

Децентрализованность. Состояние блокчейна одновременно хранится у всех участников сети.

Безопасность. В основе механизмов лежит стойкая криптография.

Прозрачность. История транзакций доступна всем пользователям.

Согласованность. Механизмы proof-of-work (bitcoin) и proof-of-stake (ethereum) позволяют добиваться согласованности между участниками сети в вопросе добавляемых в блокчейн транзакций.

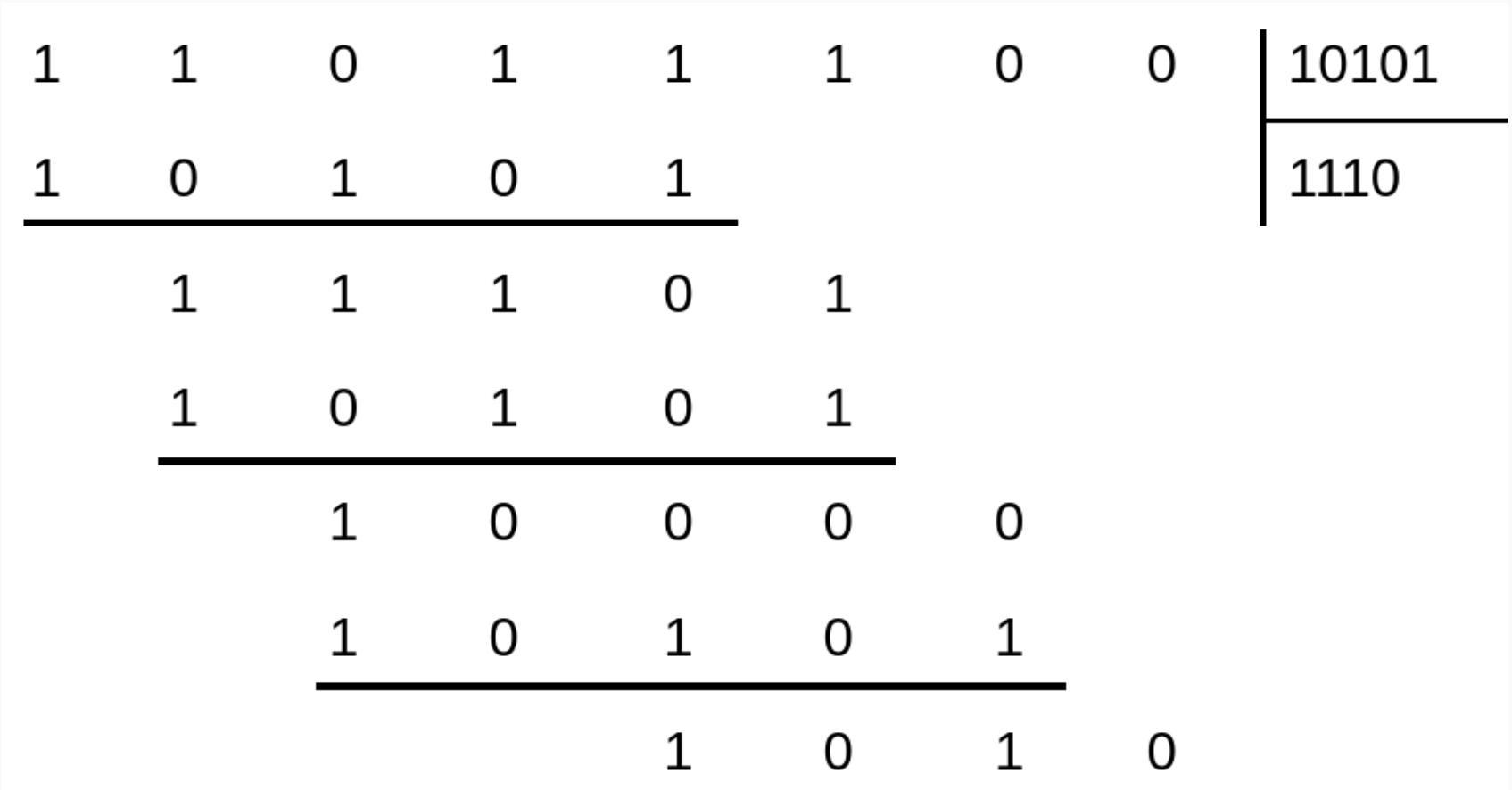
Избыточный циклический код

В статье 1961 года “Cyclic Codes for Error Detection” Вильям Питерсон предложил идею выявления ошибок при передачи двоичных данных с помощью циклических кодов.

$$11011100 \rightarrow X^7 + X^6 + 0 \cdot X^5 + X^4 + X^3 + X^2 + 0 \cdot X^1 + 0 \cdot X^0$$

Избыточный циклический код

Контрольное значение можно получить как остаток от деления информационного многочлена на многочлен-генератор.



Bitcoin

Для хранения состояния блокчейна в Bitcoin используется LevelDB[1].

```
Status ReadBlock(RandomAccessFile* file, const ReadOptions& options,const BlockHandle& handle,
      BlockContents* result) {
    ...
    Slice contents;
    Status s = file->Read(handle.offset(), n + kBlockTrailerSize, &contents, buf);
    ...
    // Check the crc of the type and the block contents
    const char* data = contents.data();  // Pointer to where Read put the data
    if (options.verify_checksums) {
        const uint32_t crc = crc32c::Unmask(DecodeFixed32(data + n + 1));
        const uint32_t actual = crc32c::Value(data, n + 1);
        if (actual != crc) {
            delete[] buf;
            s = Status::Corruption("block␣checksum␣mismatch", file->GetName());
            return s;
        }
    }
    ...
}
```

[1] https://github.com/bitcoin/bitcoin/src/leveldb/table/format.cc

Ethereum. EIP-2124: Fork identifier for chain compatibility checks

EIP-2124[2] предлагает вычислять одно из полей идентификатора с помощью CRC32.

Specification

Each node maintains the following values:

- `FORK_HASH`: IEEE CRC32 checksum (`[4]byte`) of the genesis hash and fork blocks numbers that already passed.
  - The fork block numbers are fed into the CRC32 checksum in ascending order.
  - If multiple forks are applied at the same block, the block number is checksummed only once.
  - Block numbers are regarded as `uint64` integers, encoded in big endian format when checksumming.
  - If a chain is configured to start with a non-Frontier ruleset already in its genesis, that is NOT considered a fork.
- `FORK_NEXT`: Block number (`uint64`) of the next upcoming fork, or `0` if no next fork is known.

E.g. `FORK_HASH` for mainnet would be:

- forkhash_0 = `0xfc64ec04` (Genesis) = `CRC32(<genesis-hash>)`
- forkhash_1 = `0x97c2c34c` (Homestead) = `CRC32(<genesis-hash> || uint64(1150000))`
- forkhash_2 = `0x91d1f948` (DAO fork) = `CRC32(<genesis-hash> || uint64(1150000) || uint64(1920000))`

[2] https://eips.ethereum.org/EIPS/eip-2124

The Open Network. Адрес

TEP-2[3] предлагает формат адреса, который включает в себя CRC16.

Under the conditions stated above, the smart-contract address can be represented in the following forms:

A) "Raw": :<64 hexadecimal digits with address>

B) "User-friendly", which is obtained by first generating:

- one tag byte (0x11 for "bounceable" addresses, 0x51 for "non-bounceable"; add +0x80 if the address should not be accepted by software running in the mainnet network)
- one byte containing a signed 8-bit integer with the workchain_id (0x00 for the basic workchain, 0xff for the masterchain)
- 32 bytes containing 256 bits of the smart-contract address inside the workchain (big-endian)
- 2 bytes containing CRC16-CCITT of the previous 34 bytes

[3] https://github.com/ton-blockchain/TEPs/blob/master/text/0002-address.md

The Open Network. Идентификатор метода

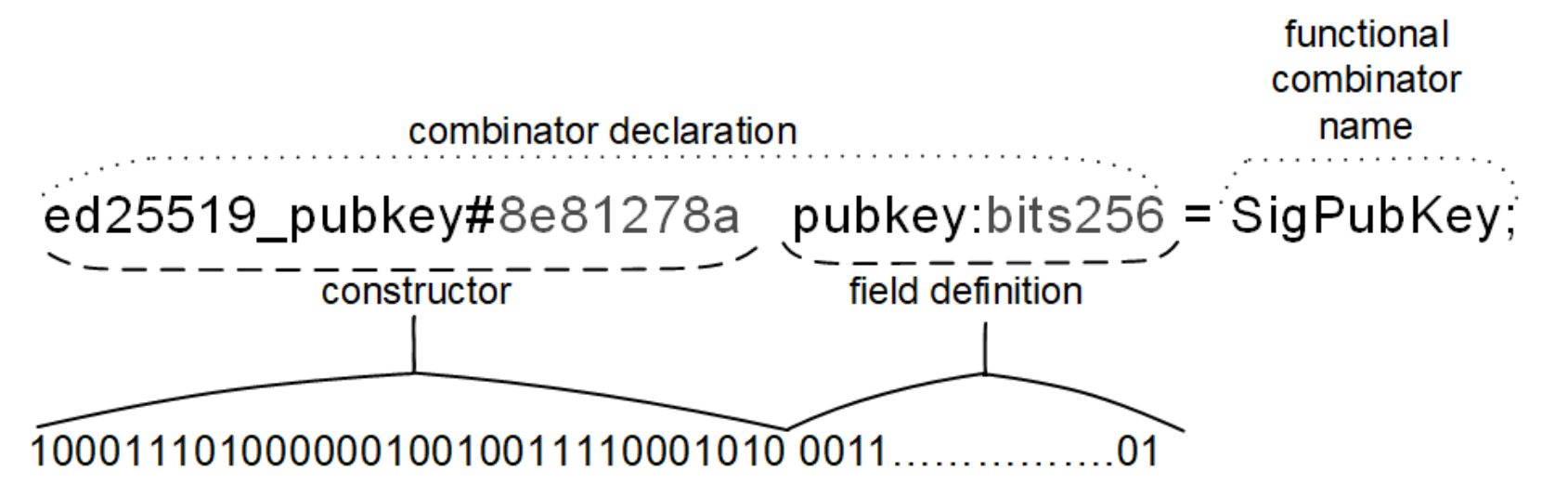
Для функции можно указать её идентификатор с помощью спецификатора method_id, который вычисляется с помощью CRC16[4].

```
td::int64 TestNode::compute_method_id(std::string method) {
    td::int64 method_id;
    if (!convert_int64(method, method_id)) {
        method_id = (td::crc16(td::Slice{method}) & 0xffff) | 0x10000;
    }
    return method_id;
}
```

[4] https://github.com/ton-blockchain/ton/blob/master/lite-client/lite-client.cpp

The Open Network. TL-B

Для сериализации данных в TON используется язык TL-B[5].



Идентификатор конструктора можно указать в явном виде, или он будет вычислен с помощью CRC32.

[5]https://docs.ton.org/v3/documentation/data-formats/tlb/tl-b-language

The Open Network. Bag of Cells

При сериализации и десериализации структуры “Bag of Cells” осуществляется проверка отсутствия ошибок с помощью CRC32[6].

```
td::Result<long long> BagOfCells::deserialize(const td::Slice& data, int max_roots) {
  ...
  if (info.has_crc32c) {
    unsigned crc_computed = td::crc32c(td::Slice{data.ubegin(), data.uend() - 4});
    unsigned crc_stored = td::as<unsigned>(data.uend() - 4);
    if (crc_computed != crc_stored) {
      return td::Status::Error(PSLICE() << "bag-of-cells␣CRC32C␣mismatch:␣expected␣" << td::format
             ::as_hex(crc_computed) << ",␣found␣" << td::format::as_hex(crc_stored));
    }
  }
  ...
}
```

[6] https://github.com/ton-blockchain/ton/blob/master/crypto/vm/boc.cpp

Спасибо

спасибо [7]

[7] спасибо