УДК 004:338
DOI: 10.25559/SITITO.15.201901.190-199

Blockchain Protocol Study

R. R. Giliazov
Lomonosov Moscow State University, Moscow, Russia
r_gilyazov@icloud.com

Abstract

Blockchain has great advantages over existing payment systems (bank cards, electronic money, etc.), such as decentralization and auditing capacity. There is a wide range of blockchain applications, ranging from cryptocurrency, financial services, risk management, the Internet of things and ending with public and social services. A number of studies had focused on the use of blockchain technology in various applications. We are doing research of blockchain technology soundness. In particular, this article focuses on analyzing the safety of using a blockchain, represents typical attacks on a protocol, examines blockchain applications and discusses technical problems, as well as recent innovations in solving difficult problems. It also provides an analysis of the two most popular cryptocurrencies and smart contracts, as well as the safety problems associated with them.

Keywords: Blockchain, cryptocurrency, Payment systems, Bitcoin, Payment safety, smart contracts, consensus algorithms, Ethereum, attacks.

For citation: Giliazov R.R. Blockchain Protocol Study. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2019; 15(1):190-199 DOI: 10.25559/SITITO.15.201901.190-199

Исследование протокола блокчейн

Р. Р. Гилязов
Московский государственный университет имени М.В. Ломоносова, г. Москва, Россия
r_gilyazov@icloud.com

Аннотация

Блокчейн имеет множество преимуществ по сравнению с существующими платежными системами (банковских карт, электронных денег и пр.), таких как децентрализация и возможность аудита. Существует широкий спектр приложений блокчейна, начиная от криптовалюты, финансовых услуг, управления рисками, Интернета вещей и заканчивая государственными и социальными услугами. В ряде исследований основное внимание уделяется использованию технологии блокчейна в различных прикладных аспектах. Мы проводим исследование безопасности технологии блокчейна. В частности, эта статья уделяет внимание анализу безопасности использования блокчейна, представляет типичные атаки на протокол, рассматривает приложения блокчейна и обсуждает технические проблемы, а также последние достижения в решении проблем. Также приводятся анализ двух наиболее популярных криптовалют и смарт-контрактов, а также проблем безопасности связанных с ними.

Ключевые слова: блокчейн, криптовалюта, платежные системы, биткойн, безопасность платежей, смарт-контракты, алгоритмы консенсуса, Ethereum, атаки.

Для цитирования: *Гилязов Р. Р.* Исследование протокола блокчейн // Современные информационные технологии и ИТ-образование. 2019. Т. 15, № 1. С. 190-199. DOI: 10.25559/SITITO.15.201901.190-199

Introduction

A blockchain is a distributed database of records or a public register of all transactions or digital events that have been executed and transferred to the parties involved. Each transaction in the database is verified by the consensus of most of the participants of the system. Once entered, information can never be deleted. The blockchain contains a defined and verifiable record of every transaction ever made. Bitcoin payment system, based on equality of participants, is the most popular example of the blockchain use.

One of the unique features of the technology is that the blockchain provides a system for creating distributed consensus in the digital online world. This allows participating organizations to know exactly that a digital event has taken place by creating irrefutable entries in the blockchain. This opens the door to the development of a democratic, open and scalable digital economy from a centralized one. This technology has a tremendous potential, and the revolution in this space has just begun. This document describes the blockchain technology and some important applications, both theoretical and practical.

Goals and objectives of the work:

- studying of the types and implementation of blockchain pools, attacks on them and the possibilities of additional protection;
- investigation of the possibility of using existing software tools to study the stability of Blockchain-type protocols;
- exploration of potential advantages of using these and other software tools by information exchange stakeholders.

Bitcoin

Bitcoin is a peer-to-peer payment system that uses a unit of the same name to accounting of transactions. To ensure the operation and protection of the system, cryptographic methods are used. At the same time, all information about transactions between system addresses is available in open form.

One of the main features of the system is total decentralization. The system has no central administrator or any equivalent. An essential component of the Bitcoin payment system is a basic client program[1], which has an open source code. Launched on many computers, client programs are interconnected into a peer-to-peer network, each node of which is equal and all-sufficient. The volume and time of production of new bitcoins (currency units) are known in advance, but they are distributed randomly among those who use their own equipment for calculations, whose outcomes are the mechanism for regulating and confirming the validity of operations in the Bitcoin system.

Bitcoin address

Bitcoin address, or just an address, is an identifier containing 26–35 alphanumeric characters, in the current version of the protocol it starts with the character 1 or 3, using this identifier you can perform operations with Bitcoin. Addresses can be obtained free of charge by any Bitcoin user. For example, using the software[2], clicking the "Get Address" button. A large proportion of existing Bitcoin addresses contain only 32 characters. Each Bitcoin address is equivalent to an account number. It happens that they start from zero, and when zeros are omitted, the coded address becomes shorter.

Technically, a Bitcoin address is a 160-bit hash-function value from an ECDSA public key of the key pair. Using math methods, it is possible to "sign" the data with your secret key, and anyone who knows your public key can make sure that the signature is valid. Since Bitcoin addresses are based on hash functions, it is possible, although very unlikely, that two people will generate the same addresses independently of each other. This is called a collision. If this happens, then both owners of this address can spend money sent to that address. If you deliberately decided to choose an address that would cause a collision, then at the moment you would have to spend 2^{107} times more time to generate such an address than to create a new block. At the time of writing, the creation time takes on average of 10 minutes[3] [11].

Bitcoin address example: 1BQ9qza7fn9snSCyJQB3ZcN46biBtkt4ee

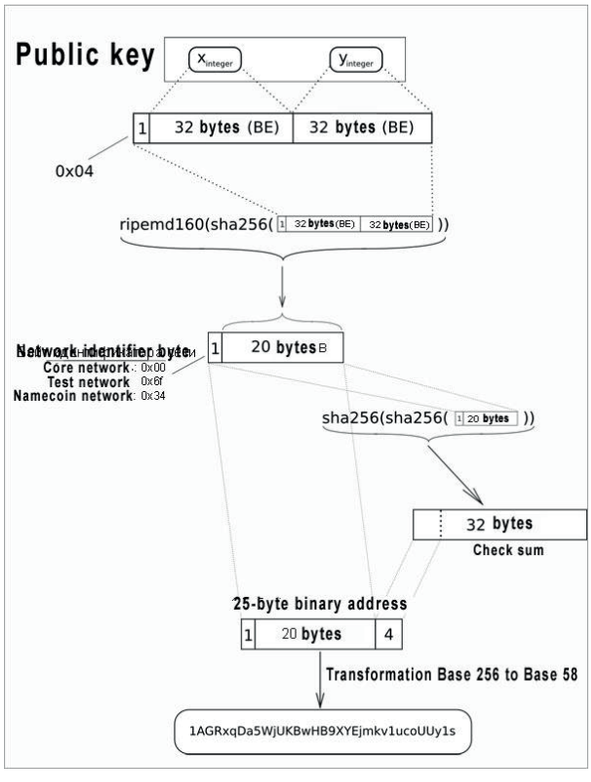

Fig. 1. The establishment of Bitcoin address

Fig. 1 shows the scheme for creating a Bitcoin address[4]:

1. You take a public key (65 bytes, 1 byte 0x04, 32 bytes correspond to the X coordinate, 32 bytes correspond to the Y coordinate).
2. SHA-256 public key hashing is performed.
3. The SHA-256 result is RIPEMD-160 hashed.
4. A byte network identifier is added in front of the RIPEMD-160 hash (0x00 for the main network).

[1] Bitcoin Core [Electronic recourse]. Available at: https://bitcoincore.org/ (accessed 19.01.2019). (In Eng.)

[2] Ibid.

[3] Bitcoin Block Time historical chart [Electronic recourse]. Available at: https://bitinfocharts.com/comparison/bitcoin-confirmationtime.html (accessed 19.01.2019). (In Eng.)

[4] Villa M. Bitcoin. 2014. p. 15-23. [Electronic recourse]. Available at: https://wiki.uio.no/mn/ifi/AFSecurity/images/1/1b/AFSec20140116-Villa-Bitcoin.pdf (accessed 19.01.2019). (In Eng.)

5. SHA-256 hashing for the extended result from RIPEMD-160 is performed:
6. A SHA-256 hash is performed on the previous SHA-256 hash, the result will be a checksum.
7. You take the first 4 bytes of the resulting hash.
8. These 4 bytes of the checksum from clause 7 are added to the end of the extended RIPEMD-160 hash from paragraph 4. This is a 25-byte binary Bitcoin address.
9. The result of paragraph 8 is converted to base58 string. This is the most commonly used format for bitcoin addresses.

Payments and fees

The minimum currency value of the Bitcoin payment system transferred from one address to another is 10^{-8} bitcoin.

Electronic payment between the two parties occurs without intermediaries and is irreversible. Irreversibility means the absence of a mechanism for canceling a confirmed transaction, including cases when the payment was sent to an erroneous or non-existent address, or when the transaction was signed with a private key that was compromised. None of the participants is able to block their own or other people's funds, even temporarily, with the exception of the participant who owns the private key or the person for whom he has become known.

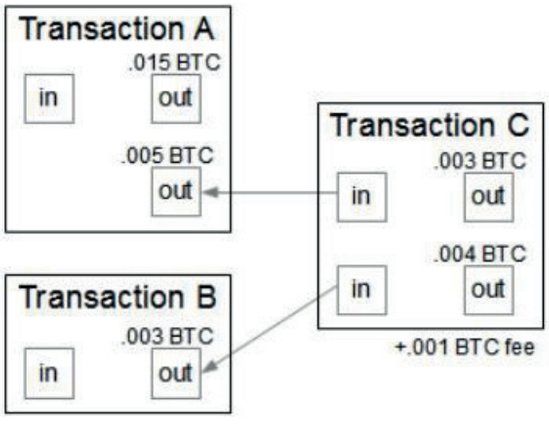

Fig. 2. Bitcoin transaction fees & commissions

The commission fee for operations is set by the sender voluntarily, the size of the it affects the priority when processing a transaction. The Bitcoin Core client program[5] suggests the recommended fee. Commission-free transactions are possible and are also processed, but the average processing time is long. In Fig. 2, you can see the connection between incoming *A, B* and an outgoing transfer *C*. The funds necessary for sending, including the commission, are taken from the funds transferred by the transactions *A* and *B*.

Bitcoin Elliptic Curve Digital Signature Algorithm

ECDSA [8] (Elliptic Curve Digital Signature Algorithm) is a public key algorithm for creating a digital signature, with the structure similar to DSA, but defined in a group of points of an elliptic curve.

Elliptic Curve Requirements

In order to avoid known attacks based on the discrete logarithm problem in a group of points of an elliptic curve, it is essential that the number of points of an elliptic curve E be divisible by a sufficiently large prime number n. The ANSI X9.62 standard requires $n > 2^{160}$. The equation of an elliptic curve is constructed in a specific way using random / pseudo-random coefficients.

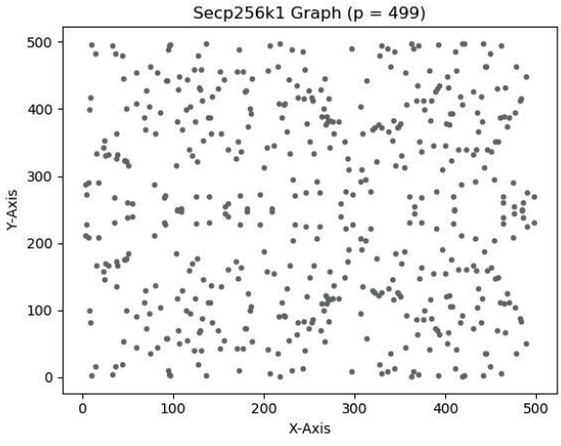

Fig. 3. The plot of the elliptic curve secp256k1 with $p = 499$

In Bitcoin, the secp256k1[6] [12] $y^2 = x^3 + ax + b \bmod p$ elliptic curve is used, where $a = 0, b = 7,$ the plot of which is shown in Fig. 3.

Transactions

A Bitcoin transaction is a signed data section (transaction signature) transmitted over the Bitcoin network and assembled into blocks. Usually it contains links to previous transactions and associates a certain number of Bitcoins with one or several public keys (Bitcoin addresses). It is not encrypted.

The blockchain browser is the place where all transactions combined into a blockchain can be found and checked, all Bitcoin transactions can be tracked. This is necessary not only to determine the technical parameters of the transaction, but also to check the payment quality.

Typically upon receipt of Bitcoin, the new owner cannot immediately dispose of them. Once a transaction is made, it is sent to the Bitcoin network for execution and must be included in the block in order to become legitimate. The process of including a transaction in the found block is called a transaction confirmation. The inclusion is carried out under the condition of *1 block = 1 confirmation*, and when such confirmations are accumulate as many as 6 and above, the transaction is considered confirmed. Such a function was introduced for protection against the repeated waste of the same bitcoins [10].

Bitcoin transaction cannot be undone.

Blocks

These transactions are recorded continuously in files called blocks. They can be viewed as separate pages of the city registrar records book (where changes in the ownership of real estate are recorded) or in the book of stock transactions. Over time, blocks are lined up

[5] Bitcoin Core [Electronic recourse]. Available at: https://bitcoincore.org/ (accessed 19.01.2019). (In Eng.)

[6] Brown D.R.L. Standards for Efficient Cryptography Group. SEC 2: Recommended Elliptic Curve Domain Parameters. Version 2.0. Certicom Corp., 2010. Available at: https://webencrypt.org/ecc/sec2-v2.pdf (accessed 19.01.2019). (In Eng.)

in a linear sequence (also known as a chain of blocks). New transactions are processed continuously by miners in new blocks, which are added to the end of the chain. As blocks get deeper and deeper into the blockchain, it becomes more and more difficult to change or delete them.
The block structure is shown in Table 1.

Table 1. The block structure

Field	Description	Size in Bytes
magic number	constant 0xD9B4BEF9	4
block size	number of bytes occupied by the block	4
block header	service parameters	80
transaction counter	natural value *N*	1–9
transactions	list of *N* transactions	varies

Transaction confirmation
Bitcoin client "Bitcoin Core"[7] will display the transaction as "unconfirmed" until 6 confirmations are counted (6 blocks found). Sites or services accepting Bitcoin to pay for their goods or services can put their own limits on the number of blocks needed to confirm a transaction. The number 6 was not chosen by chance: it is based on the theory that it's far less likely that an attacker can accumulate more than 10% of the computing power for faking transactions, and that a minor risk (less than 0.1%) is acceptable [9]. According to calculations [9], if wrongdoers do not have a capacity of less than 10% of the total, then 6 confirmations are a fairly reliable barrier to the attack. In turn, for people possessing more than 10% of capacity it will not be difficult to obtain 6 confirmations in a row. However, the possession of such computing power requires an investment of millions of dollars, which reduces the risk of attack.
Bitcoins issued by the network for finding a block can only be used after 100 confirmations, i.e. 100 found blocks. Bitcoin client "Bitcoin Core" will not display the coins received for block generation until 120 confirmations have been accumulated.

Mining

Mining[8] [1] is the activity aimed at creating new blocks in the blockchain to ensure the functioning of payment systems based on blockchains. The creation of the next structural unit (block) is rewarded through the transferred cryptocurrency units and / or commission fees. In the general case, mining comes down to a series of calculations with the enumeration of parameters to find the result of a hash function with specified properties. Different cryptocurrencies use different computational models that take a long enough time to find an acceptable option and provide for a quick check of the solution found. Such calculations are used by cryptocurrency algorithms to solve the double spending problem [10].
Proof-of-Work
The first consensus algorithm used in the Bitcoin blockchain is "proof of work done" (Proof-of-Work, PoW). PoW requires each user involved in the confirmation to prove performed computational actions to prevent an attack on the network in the form of spam or DoS attacks. Each node tries to solve complex cryptographic tasks using its own computational resources — the one who finds the solution gets the right to confirm transactions and to write the block to the blockchain. It means that miners are competing with each other for creating the next block of transactions in the blockchain. The winning miner, in turn, receives cryptocurrency tokens as a reward for the time and resources spent on finding a solution. For example, Bitcoin miners receive rewards in bitcoins. This reward system motivates miners to generate the right solution and ensures network safety. At the same time, the newly created coins are added to the circulation [5].
Proof of work means that it is possible to look at a part of the data (proof) and with just a little bit of effort to verify that someone had invested a lot of computing power in order to calculate this data, at least with a high probability. Bitcoins use the SHA256 hash function for this purpose. To use SHA256 as proof of work, the Bitcoin protocol requires miners to find the input value (which should include a random one-time nonce value changed by the miner), the result of applying SHA256 for which there is certain number of 0 characters at the beginning of the line. The amount depends on the complexity of the task. Since SHA256 is a hash function, the only way to find such a header is to try applying SHA256 to different one-time values (nonce) until a solution is obtained. On the other hand, if there is a valid input value, one SHA256 calculation will suffice. For technical reasons, SHA256 is actually used twice in bitcoins, so the SHA256 output (SHA256 (block header)) must be less than the target one. Until someone finds a valid proof of work, the number of expected tests can be estimated as 2^{256} / solution.
Bitcoin uses Merkle trees [13] (Fig. 4) or the so-called "Hashcash" [14] function to prove the performance of work (PoW).

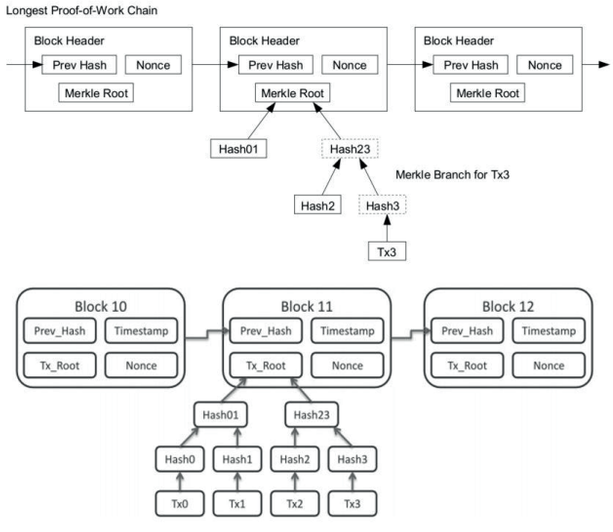

Fig. 4. PoW with the use of Merkle tree

[7] Bitcoin Core [Electronic recourse]. Available at: https://bitcoincore.org/ (accessed 19.01.2019). (In Eng.)

[8] Bonneau J. Bitcoin mining. All about mining. Lecture 4 [Electronic recourse]. Presented at IACR Summer School on Blockchain Technologies. Corfu, Greece, May 30 2016. p. 1-34. Available at: https://docs.google.com/presentation/d/1TPeEa-i8GVX1xGmjYgqf67GdSpKYSo7d7_iLuea-y-Y/view#slide=id.p (accessed 19.01.2019). (In Eng.)

Pools

In order to obtain bitcoins in a more equitable manner, miners use specialized servers – pools. Each participant is looking for his own version of the block and sends the results to the pool. A poole is rewarded as a single miner. The pool distributes the resulting cryptocurrency among the participants in accordance with the rules established by the owner of the pool. As of 2016, most of the large pools are located in the PRC: as of March 2016, more than half of the network's capacity is divided between three large Chinese pools, the fourth place is taken by the pool of BitFury, the one of the first producers of mining chips. It was founded by nationals of the post-Soviet region.

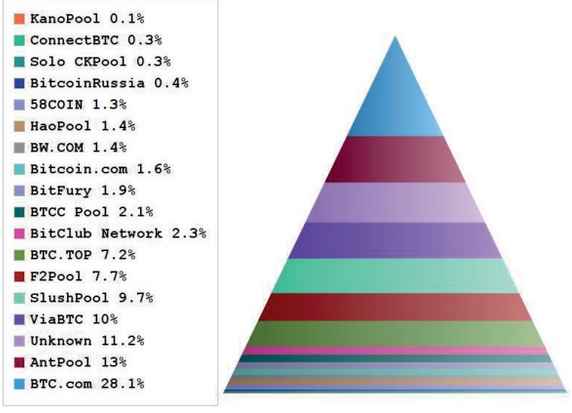

Fig. 5. The distribution of Bitcoin-pools at the end of 2018

According to our study, 68% of the computational power of all miners is concentrated in 10 pools, the diagram can be seen in Fig. 5.

Ethereum

Ethereum is a platform for creating decentralized blockchain-based online services operating on the basis of smart contracts. It is implemented as a single decentralized virtual machine. Smart contracts are cross-platform programs compiled from a high-level program code into an intermediate state (byte code), which is transferred to the execution of the Ethereum virtual machine (Fig. 6).

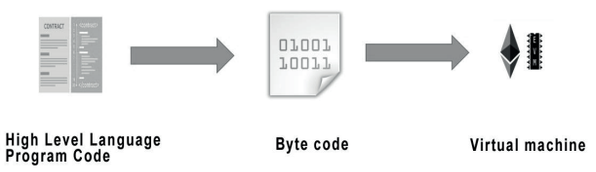

Fig. 6. Compiling and launching smart contracts

Ethereum exchange units are called ether. Fractional parts have their own names: 0.001 – finney, 0.0094 – szabo, 0.0001 – wei.

Unlike for other cryptocurrencies, the authors do not reduce the role of the ether to payments alone, but offer it as a means for sharing resources or registering transactions with assets using smart contracts, in particular, the authors called the ether "crypto fuel" to execute smart contracts with a peer-to-peer network. The total capitalization of cryptocurrency Ethereum in January 2018 exceeded $100 billion, but by August 2018 the capitalization decreased to $30 billion.

Ethereum Virtual Machine

Ethereum Virtual Machine is a smart contract execution environment in Ethereum. Its main feature is isolation from the outside world, that is, the code running inside the EVM does not have access to the network, file system, or other similar elements. Intelligent contracts may also have limited access to other smart contracts.

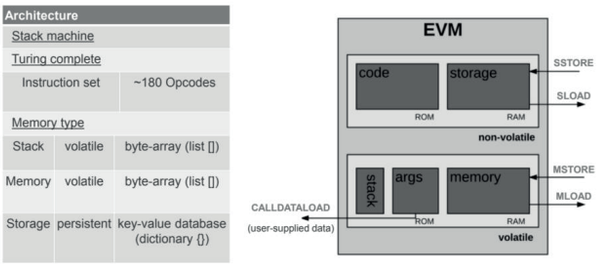

Fig. 7. Ethereum Virtual Machine

There are 2 types of accounts in the Ethereum network that have the same address space: external accounts controlled by public access key pairs, and contract accounts controlled by the code stored with the account. The address of the external account is determined from the public key, while the address of the contract is determined at the time the contract is created (it is obtained from the address of the creator and the number of transactions sent from this address). Regardless of whether the code stores an account or not, the two types are handled equally by EVM. Each account has a permanent storage of key values comparing 256-bit words to 256-bit words, called repository.

A transaction is a message that is sent from one account to another account. It may include binary data (useful information) and ether (Ether). If the target account contains code, this code is executed and useful information is provided as input. If the target account is a zero account (with address 0), the transaction creates a new contract. The payload of such a contract creation transaction is accepted as an EVM bytecode and is executed. The result of this performance is permanently stored as a contract code.

As mentioned earlier, each account has a permanent area of memory called a key storage (storage). It is impossible to "sort out" the storage within the framework of the contract, and it is relatively expensive to read, much less to change this storage. The contract can neither read nor write to any other storage, except its own. The second area is called memory, from which a contract receives only a "cleaned" copy of an object for each message call. The memory is linear and can be addressed at the byte level, but the reading is limited to 256 bits wide, while the writing can be either 8-bit or 256-bit format.

EVM is not a register machine, it uses a stack model, so all calculations are performed in an area called the stack. It has a maximum size of 1024 elements and contains 256-bit words. Access to the stack is limited to the top bar as follows: it is possible to copy one of the top 16 items to the top of the stack, or replace the top item with one of the 16 items below it. All other operations take the top two (or one or more, depending on the operation) elements from the stack and push the result onto the stack. Of course, you can move stack items to storage or to memory, but you cannot just gain access to arbitrary items deep in the stack without first removing the top part of the stack. This is schematically illustrated in Fig. 7.

Comparing Ethereum and Bitcoin

We carried out a comprehensive comparison of the characteristics of the Ethereum and Bitcoin payment systems, the results are shown in Table 2.

Table 2. Comparing Ethereum and Bitcoin

	Bitcoin	Ethereum
Launch	2008	2005
Consensus	PoW	PoW
Hash function	SHA-256	KECCAK-256 (SHA-3)
Virtual Machine	Turing-Incomplete	Turing-Complete
Transaction cost	1.63$	0.3$
Reward per block	81 190 $	1 734$
Blockchain size	201 GB	667 GB
Mining	ASIC	GPU
Max. performance	11 TH/s (1 ASIC)	32 MH/s
Mining profitability 1 Mh	0.0000003348 $ / day	0.0374 $ / day
Total network hash rate	39.106 EH / s (18 zeros)	273 TH / s (12 zeros)

Ethereum and Bitcoin attacks and vulnerabilities

Transaction Ordering Dependency

In the Ethereum Blockchain network, miners control the order of transactions, which means that your transaction can be ignored if you pay more ether for another (the higher the amount of ether, the higher your transaction priority for the miner).

Only the miner who generates the block determines the order of transactions, and this is a vulnerability called transaction order dependency or transaction ordering dependence (TOD).

In TOD, you may encounter unexpected malicious miner behavior. Imagine, for example, a smart contract offering remuneration for the correct solution of a task. The contract owner can change the prize until a decision is made, and users can submit their assignment solutions to get a reward.

However, the miner can keep track of all incoming solutions and manage the order of transactions. For example, when there is a raw user transaction with a solution, the miner is able to check it and then send the transaction to change the prize, thus reducing the reward to zero. If this transaction is processed in the first place, the user will not receive a prize.

Integer overflow

Integer overflow is a situation in computer arithmetic when the value calculated as a result of an operation cannot be placed in the n -bit integer data type (Fig. 8). There is an overflow through the upper boundary of the view and through the lower one.

In 2018, the 360 Red Team provided a report[9] [3] for CVE-2018-11561. An integer overflow in the unprotected distributeToken function of implementing an intelligent contract for EETHER (EETHER), the token Ethereum ERC20, will lead to an unauthorized increase in the attacker's digital assets, which is a critical vulnerability of the Solidity language, which defines the uint type as a 256-bit integer unsigned number.

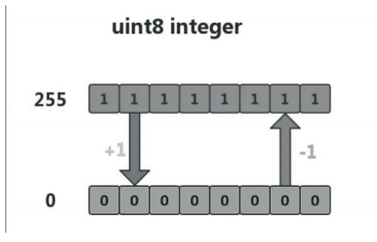

Fig. 8. Integer overflow vulnerability

BGP attack

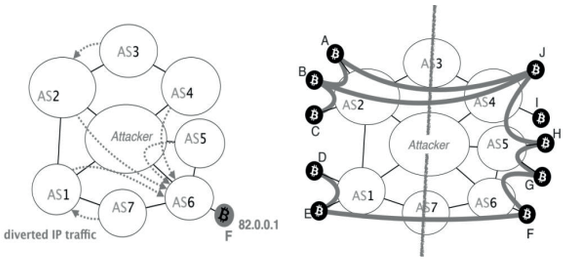

Fig. 9. Splitting the network into two parts and intercepting traffic to node F

BGP (Border Gateway Protocol), the main dynamic Internet routing protocol, is vulnerable to routing-changing attacks [4] when a certain node starts pretending to be something else. Each node uses BGP to distribute network prefixes which can be given traffic.

Suppose an attacker wishes to divide the network into right and left parts (Fig. 9), and in the right there is a pool. To do this, the attacker will manipulate BGP routes in order to intercept traffic directed to the nodes on the right side.

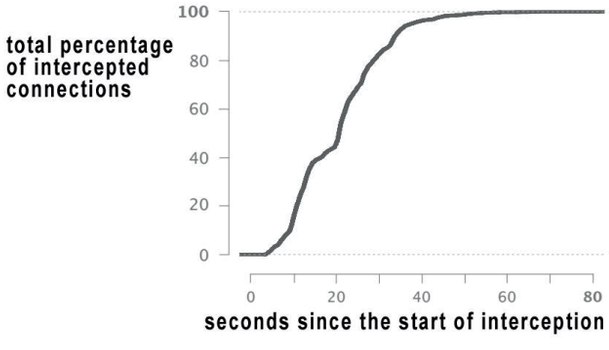

Fig. 10. Time spent for intercepting all connections

[9] Jiafeng L., Changcheng Y. How to Exploit Blockchain Public Chain and Smart Contract Vulnerability [Electronic recourse]. D2T2 - Back on the Chain Gang - Blockchain, Public Chain and Contract Security. Proceedings of the HITBSecConf2018, Dubai, 2018. p. 42-45. Available at: https://conference.hitb.org/hitbsecconf2018dxb/materials/D2T2%20-%20Back%20On%20The%20Chain%20Gang%20-%20Blockchain,%20Public%20Chain%20and%20Contract%20Security%20-%20Li%20Jiafeng%20and%20Yang%20Changcheng.pdf (accessed 19.01.2019). (In Eng.)

Let us assume that there is a node F, whose provider has IP 82.0.0.1, and let F create a BGP ad. Ad F will start to spread across the network until all similar nodes learn about it. The vulnerability is that BGP does not check ad validity, so any node can announce any prefix. The attacker spreads the advertisement, specifying a more specific prefix, overlapping the IP of the node F. As the routers choose more specific prefixes, the attacker's route will be preferred. Traffic to node F is intercepted.

Then, using a similar interception of traffic to each of the nodes on the right side, the attacker can turn off the traffic flow from the right side to the left side. Thus, the separation will be achieved and the attacker will gain the computing power of the nodes to his left.

In practice, not all networks can be separated, since not all connections can be intercepted.

We did an experiment on intercepting active compounds in the experimental network and obtained the results shown in figure 10 as a graph. The graph shows that it took only 80 seconds to intercept 100% of the connections.

DAO attack

DAO, a unique investment project based on the Ethereum blockchain, was subjected to a large-scale attack in July 2016. As a result of operating errors in the DAO code, tens of millions of dollars were stolen from the accounts of the project.

The DAO project was opened at the end of May 2016 and is a fully automatic investment fund. Those who are interested can submit their proposals to the public. Those who managed to find support from the community receive funding, and all the "investors" share part of their profits. Voting, and financing, and the distribution of profits occur automatically. DAO employees or managers are not able to influence these processes.

DAO is based on Ethereum. The DAO rules are described on it.

An unknown attacker used a vulnerability in the function SimpleDAO, designed to create child versions of the project. This function is part of the DAO code. Vulnerability allows SimpleDAO to be called recursively in the process of each branch and thus receive multiple "ethers" (Ethereum currency) during a single transaction.

As a result of the study, two attack algorithms were constructed. Figure 11 shows the source codes of the functions SimpleDAO and Mallory at the time of the attack.

First algorithm:

1. Call SimpleDAO.
2. Call Mallory, passing the DAO address to the input.
3. Spend a certain amount of ether for Mallory in DAO. The attack speed is proportional to the amount of ether spent.
4. Wait for others to increase the balance of the DAO.
5. Use Fallback Mallory to clean up the DAO.
6.

Second algorithm:

1. Call SimpleDAO.
2. Call Mallory2, passing the DAO address to the input.
3. Wait for others to increase the balance of the DAO.
4. Call the Mallory2 attack function by spending 1 wei to lower the balance of Mallory2 DAO below zero and thus get a uint-underflow.
5. Call getJackpot to clear the DAO.

Within a few hours, funds equivalent to tens of millions of dollars were transferred to the attacker's address.

```
contract SimpleDAO {
  mapping (address => uint) public credit;
  function donate(address to){credit[to] += msg.value;}
  function queryCredit(address to) returns (uint){
    return credit[to];
  }
  function withdraw(uint amount) {
    if (credit[msg.sender]>= amount) {
      msg.sender.call.value(amount)();
      credit[msg.sender]-= amount;
    }
  }
}
```

```
1 contract Mallory {
2   SimpleDAO public dao = SimpleDAO(0x354...);
3   address owner;
4
5   function Mallory(){ owner = msg.sender; }
6   function() { dao.withdraw(dao.queryCredit(this
        )); }
7   function getJackpot(){ owner.send(this.balance
        ); }
8 }
```

Fig. 11. The implementation of the functions SimpleDAO and Mallory

Security Software

Smart Contract Auditing Tools

This software provides the functionality of automatic search in the smart contract code of common vulnerabilities and undesirable code writing practices.

- securify [http://securify.ch];
- smartcheck [https://tool.smartdec.net];
- remix [https://remix.ethereum.org];
- oyente [https://github.com/melonproject/oyente].

Oyente

"Oyente" started to develop rapidly after hacking of The DAO. This is a code analyzer that is also based on symbolic execution. There is its online version, which works in conjunction with *Remix* - the most common IDE for writing smart contracts.

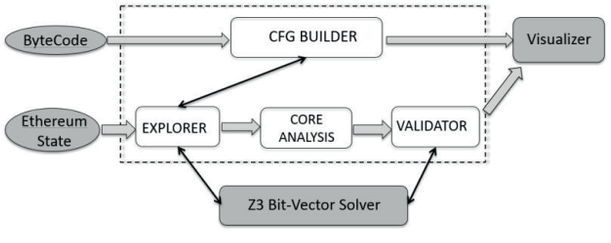

Fig. 12. Oyente structure

"Oyente" consists of several modules and contains a builder to build a program execution flow and a program execution path analyzer. The analyzer also provides visual images and validates the found vulnerabilities, proving that they are really "positive" and not "falsely positive". Oyente, like Mythril, uses the Z3 solver.

The analyzer automatically checks if assertions are violated (assert), and also supports working with cycles. In addition, this tool allows you to customize the state of the blockchain environment to analyze external contracts. Oyente generates tests to test path conditions that can be used in other frameworks, such as Truffle. The analyzer for streamlining and speeding up work can simplify character variables. It is worth mentioning that there is the ability to

work with most contracts, but it requires vast computing resources. All listed can be seen in the diagram (Fig.12).

SAT Solver

In order to automatically and efficiently perform software verification, you can apply SAT Solver, which can transform smart contract verification problems into a boolean executable problem.

SAT-solver technology is a tool for improving the performance of smart contract verification. The effectiveness of smart contract verification requires verification technology for quickly and accurately search of errors and vulnerabilities. A wide range of defect types and a low level of errors across all types of vulnerabilities and errors are also needed to provide optimized smart contract verification. In addition, the location and causes of errors must be identified quickly. The key to verifying software is solving a feasibility problem that requires a large number of irregular calculations. When the size of the limit condition exceeds 10 million, it must rely on a high-performance computing platform, and the problem can be solved within a reasonable period of time.

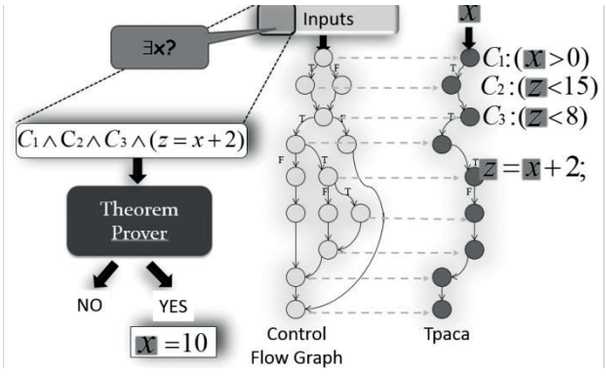

Fig. 13. SAT Solver

Satisfiability problems (SATs) consist of a set of Boolean variables and a set of short sentences consisting of these variables. The task is to obtain a set of solutions satisfying a set of short sentences. The SAT problem is the first NP-complete problem detected, and is also the core of a large class of NP-complete problems. Therefore, solving the SAT problem plays an important role in verifying smart contracts.

Fig. 13 shows the working pattern of SAT-Solver.

Smart Contract Disassembly Tools

Disassembling smart contracts[10] [2] is not a difficult task due to the fact that compiled smart contracts are stored as intermediate bytecode, which is easy to transform into source code. Listed below are the studied software tools for reverse engineering of smart contracts[11] [6].

- etherscan.io;
- quolab;
- capstone;
- IDA-EVM;
- ethersplay;
- evmdis, ethdasm.

Search for vulnerable smart contracts

We took a sample of 7251 contracts and analyzed using the Oyente software tool described above. The vulnerabilities of Callstack, TOD, Reentrancy and Timestamp were selected for identification. 100% of the smart contracts with the TOD vulnerability were identified by Oyente.

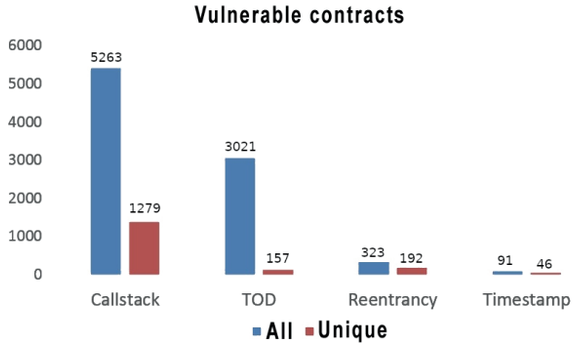

Fig.14. Scan results

Fig. 14 shows a diagram of the results we obtained during the study of a sample of smart contracts.

Conclusion

Blockchain has demonstrated its potential to transform the traditional industry with its key characteristics: decentralization, consistency, anonymity, and ease of use. This article gives you an overview of blockchain security. First, the most popular application of the technology was considered and the details of the implementation of building block chains and mining cryptocurrency were examined. Attention was drawn to typical attacks. Moreover, some vulnerabilities and problems hampering the development of the blockchain were listed, and some existing tools to solve these problems were described. The article also focused on smart contracts. Currently, the smart contract is developing rapidly, there are many applications based on this technology. Since there are still many gaps and limitations in the languages of smart contracts, many innovative applications are difficult to implement at present. It is planned to delve into the investigation of smart contracts in the future.

References

[1] Bonneau J., Miller A., Clark J., Narayanan A., Kroll J., Felten E.W. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. Proceedings of 2015 IEEE Symposium on Security and Privacy. San Jose, CA, 2015. p. 104-121. (In Eng.) DOI: 10.1109/SP.2015.14

[2] Anjana P.S., Kumari S., Peri S., Rathor S., Somani A. Entitling concurrency to smart contracts using optimistic transactional memory. Proceedings of the 20th International Con-

[10] Ventuzelo P. Reversing & Vulnerability Research of Ethereum Smart Contracts [Electronic recourse]. Workshop on Black Alps Cyber Security Conference. Switzerland, 2018. p. 10-15. Available at: http://archive.hack.lu/2018/hack_lu_2018_Reversing_and_Vulnerability_research_of_Ethereum_Smart_Contracts.pdf (accessed 19.01.2019). (In Eng.)

[11] Ventuzelo P. Reverse Engineering of Blockchain Smart Contracts [Electronic recourse]. REcon Montreal, 2018. p. 26-28. Available at: https://recon.cx/2018/montreal/schedule/system/event_attachments/attachments/000/000/053/original/RECON-MTL-2018-Reversing_blockchains_smart_contracts.pdf (accessed 19.01.2019). (In Eng.)

ference on Distributed Computing and Networking (ICDCN '19). ACM, New York, NY, USA, 2019. p. 508-508. (In Eng.) DOI: 10.1145/3288599.3299723

[3] Mense A., Flatscher M. Security Vulnerabilities in Ethereum Smart Contracts. In: Indrawan-Santiago M., Pardede E., Salvadori I.L., Steinbauer M., Khalil I., Anderst-Kotsis G. (eds) Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services (iiWAS2018), ACM, New York, NY, USA, 2018. p. 375-380. (In Eng.) DOI: 10.1145/3282373.3282419

[4] Apostolaki M., Zohar A., Vanbever L. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. Proceedings of 2017 IEEE Symposium on Security and Privacy (SP). San Jose, CA, 2017. p. 375-392. (In Eng.) DOI: 10.1109/SP.2017.29

[5] Zhang F., Eyal I., Escriva R., Juels A., van Renesse R. REM: Resource-Efficient Mining for Blockchains. Proceedings of the 26th USENIX Security Symposium. Vancouver, BC, 2017. p. 1427-1444. Available at: https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-zhang.pdf (accessed 19.01.2019). (In Eng.)

[6] Zhou Y., Kumar D., Bakshi S., Mason J., Miller A., Bailey M. Erays: Reverse Engineering Ethereum's Opaque Smart Contracts. Proceedings of the 27th USENIX Security Symposium. August 15-17, 2018. Baltimore, MD, USA, 2018. p. 1371-1385. Available at: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-zhou.pdf (accessed 19.01.2019). (In Eng.)

[7] Frantz C.K., Nowostawski M. From Institutions to Code: Towards Automated Generation of Smart Contracts. Proceedings of 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W). Augsburg, 2016. p. 210-215. (In Eng.) DOI: 10.1109/FAS-W.2016.53

[8] Johnson D., Menezes A., Vanstone S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*. 2001; 1(1):36-63. (In Eng.) DOI: 10.1007/s102070100002

[9] Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptography Mailing list*. 2009; 3:1-9. Available at: https://bitcoin.org/bitcoin.pdf (accessed 19.01.2019). (In Eng.)

[10] Pérez-Solà C., Delgado-Segura S., Navarro-Arribas G., Herrera-Joancomartí J. Double-spending prevention for Bitcoin zero-confirmation transactions. *International Journal of Information Security*. 2018. p. 1-13. (In Eng.) DOI: 10.1007/s10207-018-0422-4

[11] Narayanan A., Bonneau J., Felten E., Miller A., Goldfeder S. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press, Princeton, NJ, USA, 2016. Available at: https://www.lopp.net/pdf/princeton_bitcoin_book.pdf (accessed 19.01.2019). (In Eng.)

[12] Whitman M.E., Mattord H.J. Readings and Cases in Information Security: Law and Ethics. Cengage Learning, 2010. (In Eng.)

[13] Merkle Signatures for Real-World Use. Available at: http://www.mit.edu/~rio/merkle.pdf (accessed 19.01.2019). (In Eng.)

[14] Back A. Hashcash A Denial of Service Counter-Measure. 2002. Available at: http://www.hashcash.org/hashcash (accessed 19.01.2019). (In Eng.)

Submitted 19.01.2019; revised 10.03.2019;
published online 19.04.2019.

Поступила 19.01.2019; принята к публикации 10.03.2019;
опубликована онлайн 19.04.2019.

About the author:

Ruslan R. Giliazov, Researcher, Department of Information Security, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia), ORCID: http://orcid.org/0000-0001-6107-7907, r_gilyazov@icloud.com

The author has read and approved the final manuscript.

Об авторе:

Гилязов Руслан Раджабович, младший научный сотрудник, кафедра информационной безопасности, факультет вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова (119991, Россия, г. Москва, Ленинские горы, д. 1), ORCID: http://orcid.org/0000-0001-6107-7907, r_gilyazov@icloud.com

Автор прочитал и одобрил окончательный вариант рукописи.