Pavel STETSENKO[1], Andrii VLASOV[2]

Opiekun naukowy: Gennady KHALIMOV[3]

ARCHITEKTURA SYSTEMÓW ZDECENTRALIZOWANYCH NA BLOCKCHAIN-IE

Streszczenie: Technologia Blockchain (łańcucha bloków) poprawia wiele obszarów technologicznych, ponadto istnieją funkcje zapewniające bezpieczeństwo informacji. Tak więc formalizacja przedstawiona w tym artykule jest kluczowym elementem w budowaniu modelu zagrożenia. Przedstawiono podejście formalne ze strukturą warstw oraz definicją funkcjonalności każdego poziomu opisu architektury zdecentralizowanych systemów opartych na technologii Blockchain.

Słowa kluczowe: architektura, atak, blockchain, zdecentralizowany system, warstwa, bezpieczeństwo

BLOCKACHAIN-BASED DECENTRALIZED SYSTEMS ARCHITECTURE

Summary: Blockchain technology improves many technological areas, but there are features of ensuring information security. Thus, the formalization presented in this paper is a key block in building a threat model. A formal approach is presented with the structure of layers and the definition of the functionality of each level of the description of the architecture of decentralized systems based on Blockchain technology.

Keywords: architecture, attack, blockchain, decentralized system, layer, security.

1. Introduction

In the process of development of digital technologies, there is a significant amount, speed and variety of data in the Internet. The most common data sources are mobile devices, sensors, individual archives, social networks, software logs, the Internet of

[1] Kharkiv National University of Radio Electronics, Ukraine, faculty of Computer engineering and management, specialty: cybersecurity, steps.ps93@gmail.com
[2] Kharkiv National University of Air Force by named I. Kozhedub, Ukraine, Air Force Science center, candidate of technical sciences, specialty: information security, vavand2011@gmail.com
[3] Doctor of Technical Sciences, Professor, Kharkiv National University of Radio Electronics, Ukraine, Head of the Department of Security Information Technology, faculty of Computer engineering and management, hennadii.khalimov@nure.ua

things (IoT), enterprises corporate networks, cameras, etc. This growth and variety of data makes the problem of effective and optimal data management, development of an analysis methodology and secure data storage more and more actual.
Huge amount of digital transactions is created and processed on the global financial market every day. It is, in turn, represents a large size of heterogeneous confidential data that must be safely stored, transferred and processed. The annual number of digital payments is expected to exceed 726 billions by 2020, according to a research by Capgemini & BNP Paribas in 2017.
There is a growing usage of Blockchain technology in a huge variety of industrial sectors for sharing health information, performing fast financial transactions, developing secure smart contracts. Blockchain technology itself is a computer algorithm to provide distributed communication in a peer-to-peer network of subscribers where the transactions are transparent among the untrusted parties involved.
The classic approach to build an infrastructure for processing large amounts of financial data is the deployment of local computing data centers with trusted isolated environment (secure perimeter, corporate inner networks, etc).

2. Blockchain-based decentralized system architecture

Blockchain architecture for most of the existing decentralized platforms is in general the same. Typical Blockchain-based decentralized system consists of 5 main abstract layers in its internal architecture: network, consensus achievement, data model, process, application.
Decomposing and formalizing general Blockchain architecture will help to determine main attack vectors and methods for mitigating negative impact or minimizing possibility of successful attack. Such methods can be unified into complex security approach.

2.1. Network layer

A peer-to-peer (P2P) network is the basis for the interaction and dissemination of data between nodes in decentralized systems based on Blockchain technology. This approach was not chosen by chance, because in such a network model each participant in the network has equal rights and opportunities and each of the participants can initiate a communication session. Equality of participants, in turn, ensures decentralization, since, unlike the client-server interaction model, each participant in a peer-to-peer network can function both as a client and as a server.
In a P2P network model, each node has 100% (or as close to it) amount of system data, and updates are propagated to all nodes. Data replication and distribution of updates occur multiple times and quite often depending on system workload, which entails an increase in network traffic and total storage space for storing data by each node. This aspect makes the P2P network less efficient than the classical client-server architecture [1]. However, each node is more independent and may continue to operate even after losing communication with most other nodes. In other words, P2P provides a solution to the problem of a single point of failure, which increases the reliability of P2P network, since there is no central server that monitors and processes

all data flows, so shutting down a P2P network is much more difficult [2].
Another problem in P2P networks is the different state of the data on the nodes due to the absence of a central server and the different speed of data distribution in the network. This problem is solved at the next level of the Blockchain architecture by using various mechanisms of consensus achievement.
Main protocols which are used in network layer are presented at Figure 1.

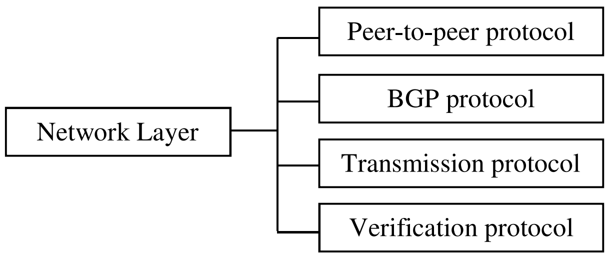

Figure 1. Main protocols of the network layer of Blockchain architecture

Thus, the network layer is the first layer in the architecture of decentralized systems based on Blockchain technology and is responsible for inter-node communication, which includes the discovery of new nodes and the transfer of data (usually transactions and distribution of blocks). Network layer includes peer-to-peer, BGP, transmission and verification protocols.

2.2. Consensus achievement layer

The role of the consensus achievement layer is to make all nodes in the Blockchain system have a unified view of the state of the ledger i.e., of completed transactions. In other words, if a node adds a block to a Blockchain ledger, other nodes in the network must approve and apply the same copy of the new state of the ledger to its local Blockchain copy.
Main mechanisms for consensus achievement layer are presented at Figure 2.

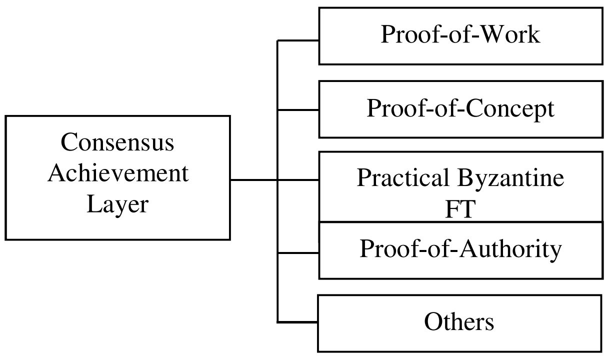

Figure 2. Main mechanisms of the consensus achievement layer of Blockchain architecture

The consensus achievement layer includes mechanisms (algorithms) by which a block

is considered verified and added to the block chain. It should be emphasized that the mechanism for achieving consensus plays an important role, since it solves the problem of trust in the system without a trusted central authority: the participants in the system trust not each other, but the protocol. This, in turn, makes it extremely important to ensure the security of this level of architecture of decentralized systems. Description of existing consensus algorithms and their pros and cons is out of the scope of this work but can be found in the numerous publications [3]-[6].
The level of consensus is closely interacting with the next level of architecture - the level of the data model, responsible for creating, checking and adding a block to the blockchain register. The level of the data model, in turn, refers more to the structure of the block.

2.3. Data model layer

The data model layer contains the structure and transaction data of the Blockchain ledger. Transactions in the context of Blockchain systems are data structures that encode the transfer of value (in the case of cryptocurrencies) between network participants [7]. The block contains a list of operations and a list of completed smart contracts, as well as their latest statuses. At the data model level, each block is identified by a cryptographic hash of its contents and is associated with the hash of the previous block. This linking of blocks with each other forms a chain - Blockchain or digital register. The data model layer structure is presented at Figure 3.

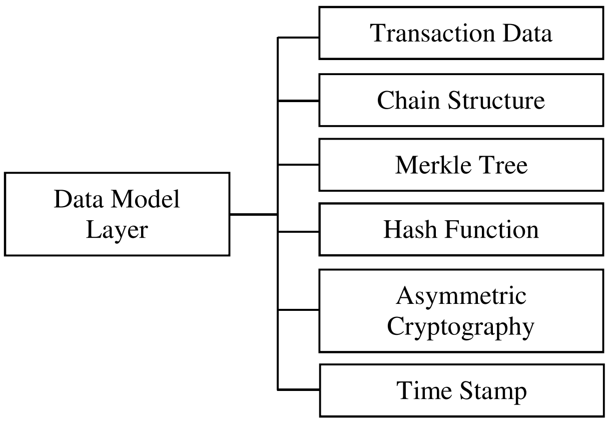

Figure 3. Main components of the data model layer of Blockchain architecture

The transaction hashes in the block are grouped in the Merkle tree. Each transaction is a public entry in the transaction register. Transactions are an integral part of the Blockchain network. All components and services in the Blockchain network are designed to ensure that transactions can be created, distributed on the network, verified, and finally added to the Blockchain. Typically, a data model includes a block, block header, and transactions.
For example, in Bitcoin, transactions are system states representing digital coins on a network. Bitcoin introduces a transaction data model that is also used by other protocols such as MultiChain and Chain core. Ethereum, on the other hand, uses a state replication model where the state of each new block is the result of transactions

that were included in the block. For example, in Bitcoins, the transfer of money between the sender and the recipient includes searching for the transaction belonging to the sender, and then marks some of them as spent, while in Ethereum it is easy to do this by updating two accounts in one transaction. An Ethereum account has a balance as its state and is updated when a transaction is received.

A special type of account, called a "smart contract", contains executable code and private conditions, and when a transaction is received, in addition to updating its balance, the contract code is called and executes the specified arguments in the transaction. The code can read the status of other non-contract accounts and send new transactions at runtime. Ethereum uses another transaction model which allows multi-stage contracts or scenarios in which the cloud maintains an internal state [8].

2.4. Process layer

The process layer contains details of the runtime environment that support Blockchain operations [8]. Each Blockchain system uses its own programming and scripting languages which entail runtimes environment including compiler, decoder, virtual machine and others. Most of these runtime environment operations are maintained at the process layer of the blockchain systems. Main components of the process layer are presented at Figure 4.

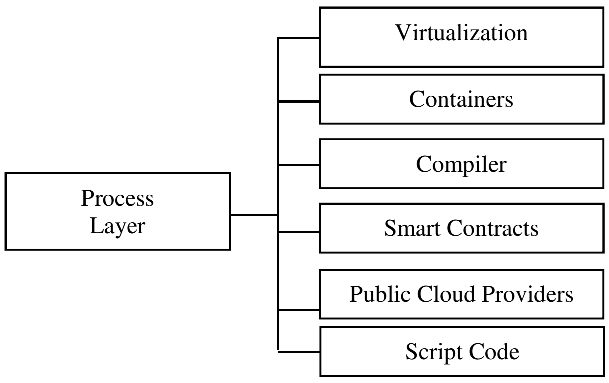

Figure 4. Main components of the process layer of Blockchain architecture

A contract (or chaincode) is executed in a runtime environment, and there are two main requirements of execution. The first one: execution must be fast because there are multiple contracts and transactions in one block and they must all be verified by the node. The second one: execution must be deterministic; ideally the same at all the nodes.

Deterministic execution prevents unnecessary inconsistency in transaction input and output states which leads to blocks being aborted and aborting a block usually wastes computer resources.

Smart contracts are programs that are used to automate coordination and ensure autonomous execution of agreements. The terms and conditions of the contract are programmed in a programming language. Such smart contracts can complement or replace legal contracts.

Blockchain-based platform Ethereum develops its own machine language (bytecode) and a virtual machine called Ethereum Virtual Machine (EVM) for executing the code. EVM is a runtime environment, which provides all the needed support and services for executing the smart contracts in Ethereum blockchain system. EVM is optimized for Ethereum specific operations. For instance, executing a code (running a transaction or contract) in Ethereum costs a certain amount of gas, and the total cost must be properly tracked and charged to the transaction's sender. Gas is the internal pricing for running a transaction or contract in Ethereum blockchain system. The originator of the transaction sets the price of gas, to which the miner may accept or ignore. That is, miners are free to ignore transactions whose gas prices limit is too low, on the other hand, with Bitcoin miners' priorities transaction with the highest mining fees. The idea behind using gas is to limit infinite loops. For instance, 10 Szabo (which is about 1/100,000 of an Ether), or 0.00001 Ether (Ethereum currency) or 1 gas can execute a line of code or some commands in the contract. So, if there is not enough Ether in the account to perform the transaction or execute the smart contract, then it is considered invalid. The idea is to stop denial of service attacks from infinite loops and to make an attacker pay for the resources they use, from bandwidth to CPU calculations through to storage [9].

Blockchain functionality can be executed on containers (for example inside Docker images) or deployed at the public clouds such as Amazon Web Services, Microsoft Azure and Google Cloud Platform.

Process level can be divided on two branches: API, used by on-chain applications in runtime, and programming languages, used in development time and compiled for runtime into a binary code that can be put into blockchain and understood by the virtual machines.

For example, API branch of the process layer is presented by an interface of Blockchain nodes that can be accessed from off-chain applications. The most famous example is Ethereum JSON RPC protocol and Web3.js based on it.

Programming languages branch can be quite well distinguished into a number of categories — according to the underlying VM that they are designed for:

- EVM languages: include original LLL, Serpent (outdated due to some compiler bugs), Solidity (the most popular smart contract language as for today), pre-released Viper (Serpent offspring) and some highly-experimental attempts like Bamboo (state-machine functional-type language) and Babbage (visual programming language);
- WebAssembly-compiled languages: include the whole rich family of LLVM languages, including C/C++, Swift, Python, Ruby, Rust and many, many others. Unfortunately, modern Blockchain VMs do not support the whole set of LLVM instructions, not mentioning language standard libraries, without which language usage could become quite painful. Thus, these are quite early experiments not ready for real-world projects yet.
- CLR languages: Common Language Runtime includes all .NET-based languages from Microsoft – C#, F#, VisualBasic.NET etc;
- Functional smart contracts languages: these languages put a big emphasis on formal verification methods, proving that smart contracts will function as expected at their design. This approach can be called – it will work if it was compiled. Examples of such languages can be Plutos language for Cardano Blockchain platfotm and Rholang used by RChain platfom.

2.5. Application layer

The application layer includes Blockchain-based application usage scenarios. Most of these Blockchain use cases are mainly due to two main features of this technology. Firstly, the data on the Blockchain ledger is immutable and transparent to participants. This means that after adding a record, it cannot be changed, and this function ensures data integrity. Secondly, it is resistant to intruders. Main use cases of the application layer are presented at Figure 5.

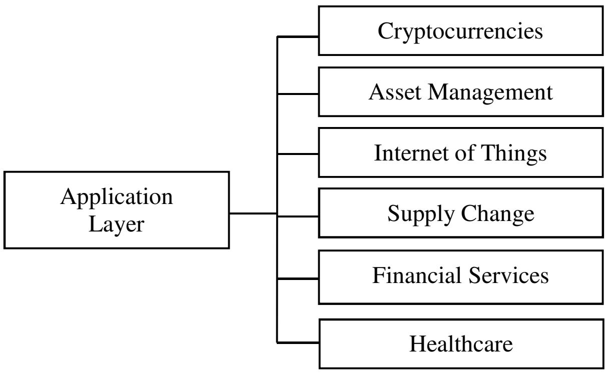

Figure 5. Main components of the process layer of Blockchain architecture

Cryptocurrencies remain the most common Blockchain application. According to Coinmarketcap, there are currently more than 1,500 different cryptocurrency projects on the market. In addition to cryptocurrencies, there are other options for using the Blockchain on the market, such as asset management, supply change, identity management, security calculations, IoT, cloud security, financial services, healthcare and many others.

3. Conclusion

Summarizing information about layers of Blockchain-based decentralized system architecture, the whole structure can be built as at Figure 6.

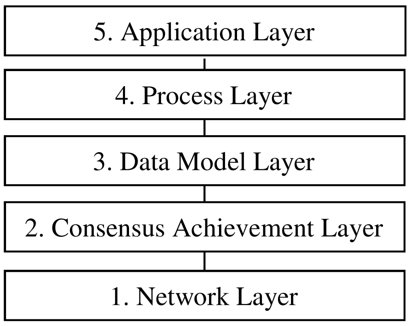

Figure 6. Blockchain-based decentralized system architecture

This 5-layered architecture is used in most current Blockchain-based decentralized systems with various combinations of items on each layer.

REFERENCES

1. LEWIS A..: A gentle introduction to Bitcoin. BraveNewCoin. 2018, 14 p.
2. DECKER C., WATTENHOFER R..: Information propagation in the Bitcoin network. IEEE P2P 2013 Proceedings, pp. 231-243, 2013.
3. GERVAIS A., KARAME G.O., WÜST K., GLYKANTZIS V., RITZDORF H..: On the security and performance of proof of work Blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 3-16, 2016.
4. BENTOV I., LEE C., MIZRAHI A., ROSENFELD M..: Proof of activity: Extending Bitcoin's proof of work via proof of stake. No. 452. 2014, 19p.
5. KING S..: Primecoin: Cryptocurrency with prime number proof-of-work. 2013, 12 p.
6. KING S., NADAL S..: Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. Self-published Paper. 2012, 16p.
7. ANTONOPOULOS A..: Mastering bitcoin. O'Reilly Media. 2014, 282p.
8. DINH T., WANG J., CHEN G., LIU R., OOI B., TAN K..: Blockbench: A framework for analyzing private blockchains. In Proceedings of the 2017 ACM International Conference on Management of Data, ACM, 2017, pp. 1085-1100.