Environment. Technology. Resources. Rezekne, Latvia
Proceedings of the 12th International Scientific and Practical Conference. Volume II, 126-131

Development and Analysis of Authentication Method for Iot Devices Software on the Network Using Blockchain Technologies

Shtennikov Dmitry Gennadievich
Faculty of Software Engineering and Computer Systems
ITMO Universaty
Saint-Petersburg, Russia
dshtennikov@gmail.com

Kuzmin Kirill Sergeevich
Faculty of Software Engineering and Computer Systems
ITMO Universaty
Saint-Petersburg, Russia
wee3dood@gmail.com

Letov Nikolay Evgenievich
Faculty of Software Engineering and Computer Systems
ITMO Universaty
Saint-Petersburg, Russia
letov.nikolay@gmail.com

***Abstract* — Currently, there is an unsolved problem of IoT device authentication in networks to ensure their security. Due to low performance, work with traditional methods of protection is complicated and therefore a different approach is required. The article proposes a method for authenticating IoT devices (devices and software) by verifying their data and then entering it into the data storage. The implementation of a data storage using blockchain technology and the comparison of its efficiency with a classical database are considered. The aim of research is evelopment of the IoT device authentication method, analysis of the effectiveness and applicability of the developed method and comparison of various technological approaches to solving the problem. Obtained results is evaluation parameters of the resulting system and methods for constructing systems based on this method.**

Keywords— IoT, Software integrity, Blockchain.

I. Introduction

For different software, there is a problem of software integrity. Each manufacturer and developer solves the issue of protecting his software in his own way - encryption, hashing or simply refusing to operate on potentially dangerous devices - this task currently remains unsolved and the goal of the work done is to try to propose a solution that allows, in theory, to provide the necessary security level for mobile devices. [3]

Building a general architecture for the IoT a very complex task, mainly because of the extremely large variety of devices, link layer technologies, and services that may be involved in such a system. [9] An important factor in such systems is the timely identification of problems with devices. For this reason, there is a time response requirement - the shorter the response time, the sooner you can determine the cause of the fault.

II. MATERIALS AND METHODS

A prototype of a distributed system has been designed for software verification when updating and monitoring integrity for mobile devices to solve this problem.

The main purpose of such a system is to provide storage of authentication data and issue them upon request of the client, as well as to ensure that it is not possible to replace recorded records. The technology of private blockchain is quite suitable for this task - it allows you to store information about system states and software, and provides the ability to log changes to the system, so you can accurately determine where the failure occurred. [4]

With different goals of private blockchain as low cost of transactions and the high TPS (TPS- transactions per second) rate there is a problem – low security private blockchain. Comparing with an open blockchain, where more than half of all nodes have to be compromised, more than half of the validator nodes are enough to corrupt a private blockchain, and this is a smaller number. For now, the solution to this issue is the anchoring mechanism - the hash of the entire system is periodically recorded in a public blockchain. If the data of the block chains are changed, the system hash does not coincide with the one stored in the public network, which will reveal a substitution. [5]

The anchoring mechanism allows you to enhance the security of a private blockchain, but the implementation requires a public blockchain, which means that the solution will not be autonomous and energy consuming, since you still need to bring proof of work to the public blockchain.

As one of the options, it is proposed to consider a system of private blockchains with cross-anchorage. This architecture implies the presence of several private blockchains, which will periodically save their state in the chains of neighbors.

For this reason, the more blockchains are in the system, the more protected it becomes.

It is worth noting that in the case of anchoring the state of each private blockchain to all other blockchains of the

Print ISSN 1691-5402
Online ISSN 2256-070X
http://dx.doi.org/10.17770/etr2019vol2.4066

Published by Rezekne Academy of Technologies.

system, the number of connections between nodes would be equal to where K is the number of connections; N is the number of nodes. Thus, the increase in the number of links in the system will grow almost quadratically with respect to the number of nodes. Looking at this phenomenon from the point of view of practical implementation, this means that network traffic will also grow steadily until the moment comes when the network's physical bandwidth is not enough for quickly use anchoring, and therefore the system will be under threat.

To resolve this issue, it is reasonable to divide the total space of blockchains into several anchorage zones. Blockchains that are in the same zone will be trustees for each other - each of them will send a hash of its state to blockchain trustees.

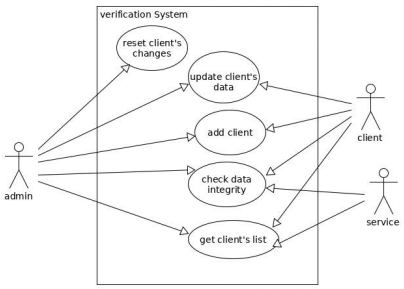

Fig 1 Use case diagram.

The overall system architecture shown on the picture:

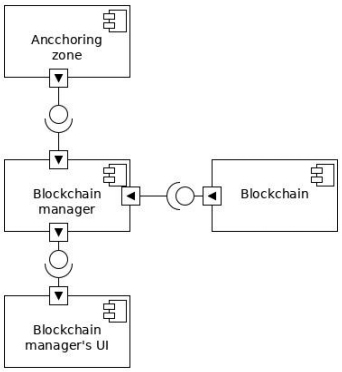

Fig. 2. System architecture.

The blockchain module implements all the functionality related to the blockchain structure itself, including: creating a blockchain, adding blocks, checking integrity, etc.

Module of zones of anchoring - this module provides opportunities for creating zones of anchoring and interaction of blockchains inside them.

The blockchain manager module is a control module for creating and managing blockchains, anchorage zones, and system configuration settings.

Scenarios for adding data about a client device and getting a list of clients shown in a Fig. 3, 4 and are not of particular interest - there are ordinary operations with a database.

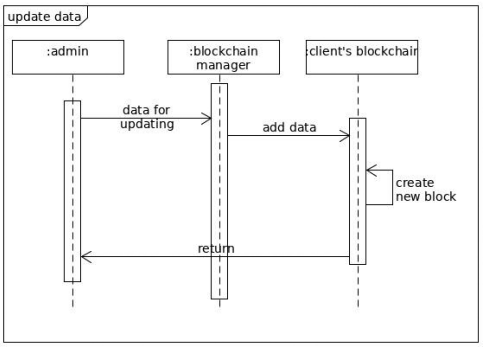

Fig. 3. Client add data scenario

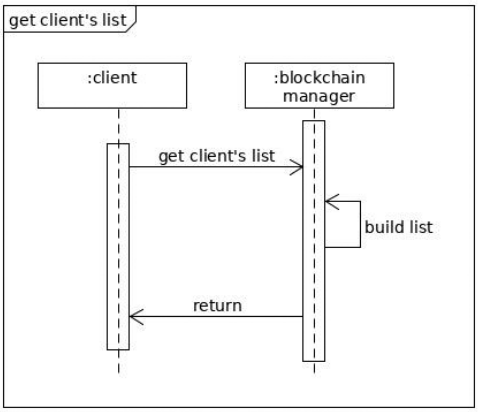

Fig. 4. Client list receiving script

The scenario of correcting client data looks the same as adding data, since the blockchain structure does not allow a simple change of data, since a change of one block will entail a violation of the connectivity of the entire system.

Of particular interest are the operations of adding a new client, creating bindings blocks and checking the integrity of customer data. In the first case, assumed change the system configuration — add or change anchorage zones, in the second and third, the anchorage mechanism is implemented and the integrity of the client's blockchain is checked, and anchor entries in the blockchain validators of the specified chain.

The sequence diagram of adding a client shown in Fig. 5.

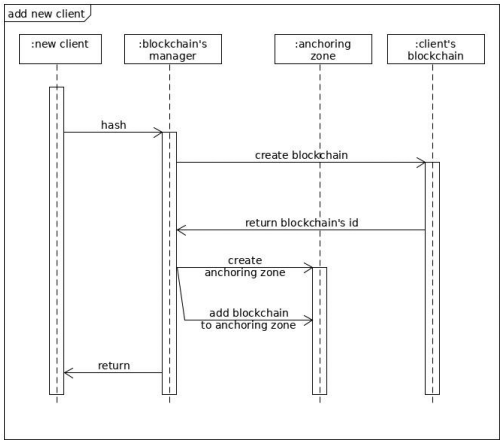

Fig. 5. Sequence diagram of adding a client

When we add a new blockchain, the blockchain manager creates a new anchoring zone and adds the created blockchain as the chain check, and all the blockchains known up to this point as the validators. This algorithm implements the principle "every with each", which is not applicable for a large number of chains, but, the simplicity of implementation and the ability to simply change the algorithm without changing the zone mechanism is quite suitable for implementation in the prototype.

The mechanism for creating anchor blocks is simple - when performing a binding in each zone, a block with a hash of the bounded blockchain added to each blockchain validator.

Scheme of process shown on a fig. 6.

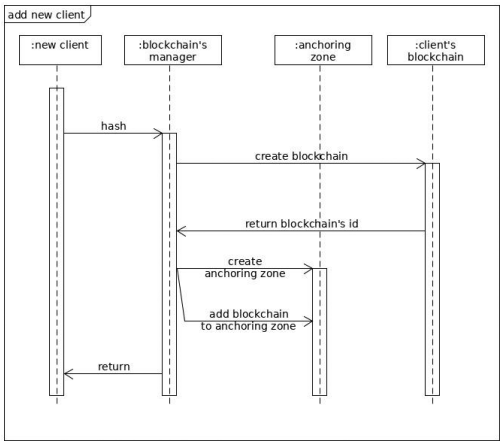

Fig. 6. Creation anchoring blocks

Integrity checking is a key mechanism for the entire system. Take a look in more detail:

1. There is a request to check the integrity of the blockchain N
2. The blockchain manager finds a zone in which blockchain N is a verifiable.
3. The last anchor block of the blockchain N is requested from each validator in this zone.
4. The current hash is requested from the blockchain being checked.
5. The resulting hash of the blockchain N comparing with the hashes of validators.
6. If more than 2/3 of the validator hashes coincided with the current hash, then the blockchain is considered verified. Otherwise, compromised.

This algorithm is simple, but effective - in order to quietly replace the data, you need to replace the data in blockchains, where N is the number of validators in the anchorage zone.

A schematic description presented in Fig. 7.

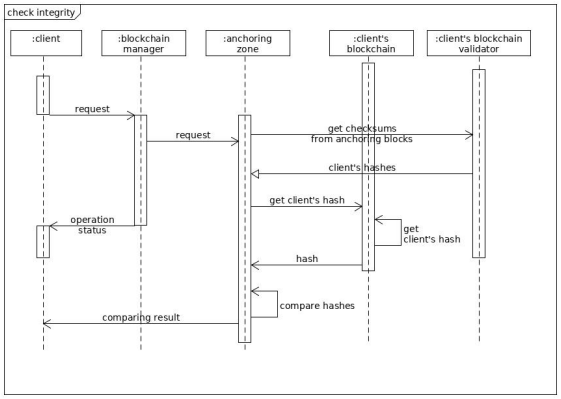

Fig. 7. Sequence diagram of checking the integrity of the client's blockchain.

Since clients are low-performance devices, they do not participate in the blockchain architecture themselves. Clients prepare data for processing and interact with the system as follows:

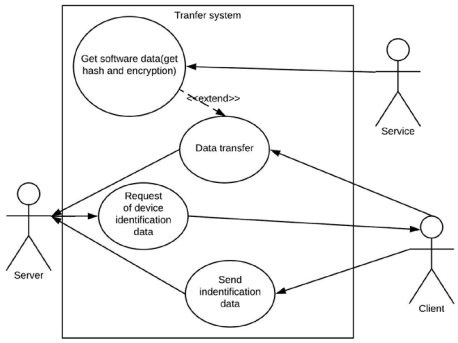

Fig. 8. Use case of client side

The client part of the system consists of 3 blocks, each of which contains its component implementation:

- Hashing block;
- Encryprion block;
- Data transfer block;

The authentication and encryption algorithms are used to protect the confidentiality and integrity of IoT data. [10]

The system works according to the following principle:

1. Made request to the client device.
2. The device returns device identification
3. The device collects a hash of predefined files / information and additional device parameters.
4. Data encryption occurs
5. Transfer data to the server.
6. Server decrypts data

The system operation sequence is presented in more detail in the Fig 9

.:

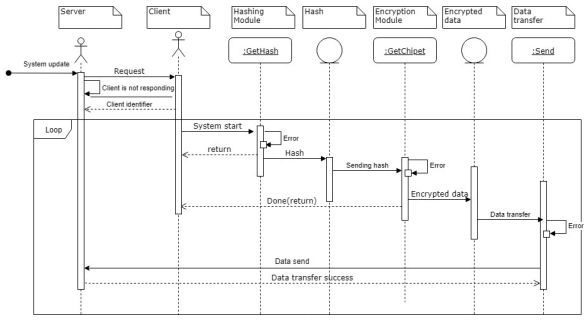

Fig 9 – sequence diagram

The server, after receiving and processing data, transfers them to the verification system.

The transactional information of IoT applications will remain secure, because all transactions are protected using cryptographic encryption. [11].

All process is shown in Fig: 10

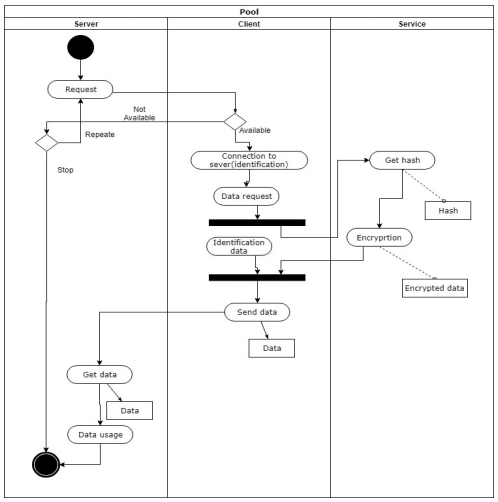

Fig –10 Sequence diagram

III. RESULTS

As encryption algorithms, we will use the following algorithms:

- SHA256;
- SHA512;
- SHA3-256;
- SHA3-512;
- Tiger;
- Whirlpool;

MD5 is not usable due to the possibility of collisions. The following are the results of testing various algorithms for different sizes of input data (Figure 11-14):

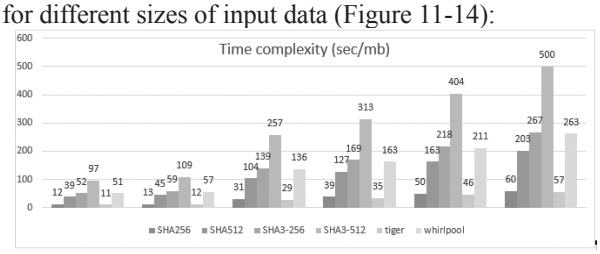

Fig 11 – Hash algorithms test

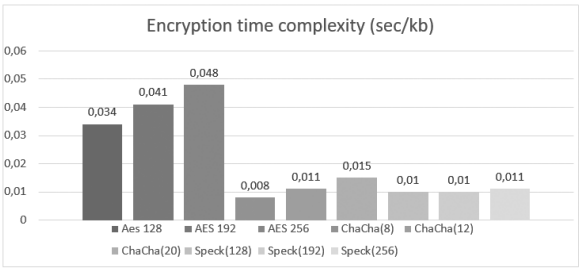

Fig 12 – Encryption test

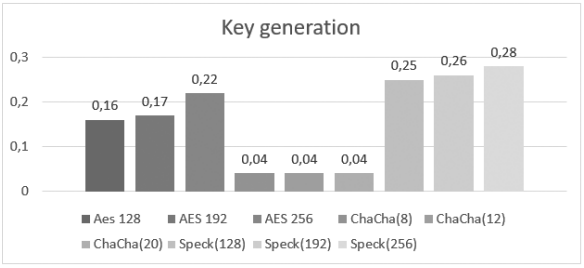

Fig 13 – Time generation test

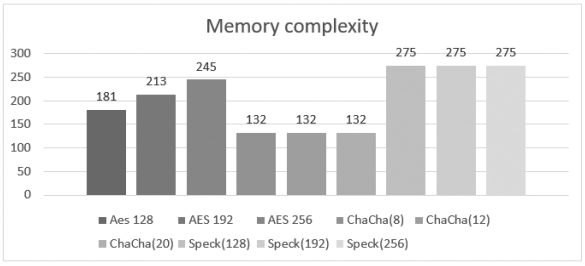

Fig 14 – Memory complexity test

Encryption algorithms are used for testing:

- AES 128/192/256;
- ChaCha 8/12/20;
- Speck 128/192/256;

The main indicators of encryption in this experiment are:

- Encryption / decryption speed
- Key generation time
- Memory usage

The client-side architecture of the resulting client interaction system is as follows:

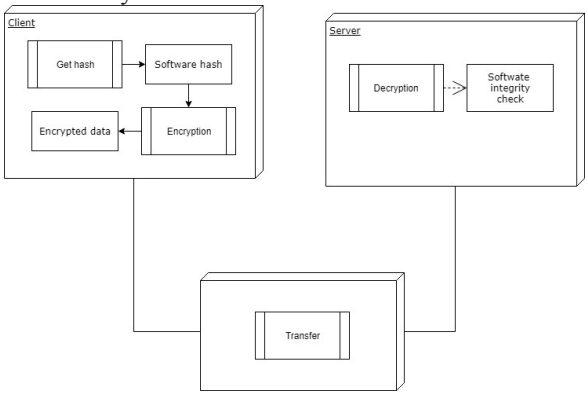

Fig 15– Client-side archetecture

Since the hardware platform itself for the implementation of the client part of the system and testing is different in capabilities, the implementation for different devices is different.

The hardware can be divided into two parts, Arduino-compatible devices [7] (such as Trekduino and ESP 8266) and with the UNIX system on board (Raspberry pi 2 and 3[8])

Accordingly, for each of these groups, the implementation requires different combinations of technologies.

In the case of Arduino compatible devices used:

- C / C ++ programming language
- Arduino cryptography Library

In the case of Raspberry 2 and 3:

- Python 3.6 programming language
- Unix-based encryption and hashing algorithms
- Socket / Django

Server implementations are used:

- Python 3.6 programming language
- Socket server

The server part of the system consists of three classes:

- Blockchain
- Blockchain Manager
- Anchorage space

Consider these classes in more detail:

The blockchain class describes, in fact, the very structure of the blockchain with support for all its functions, such as

- Adding a block;
- Integrity check;
- Getting a block;
- Getting the hash of the entire chain.

This class contains the path to the repository, and the current number of blocks, and has a unique identifier.

The class of anchorage zone (space) is an entity that describes the connections between different blockchains using the following functionality:

- Adding a blockchain to anchorage space for verification (who binds);
- Adding a blockchain to the anchorage space for validation (where bindings are made);
- Removal of the verified blockchain from the space;
- Remove a validating blockchain from the space;
- Creating bindings between blockchains;
- Check bindings between blockchains.

The anchorage zone class fields are two lists of blockchains: verifiable and validating.

The class manager provides the interaction of the external environment with the storage and manages the storage. Its features:

- Create blockchain
- Creating an anchor zone
- Storage reconfiguration
- Remove blockchain
- Remove anchorage zone

The fields are the lists of blockchains and anchorage zones.

The server class is a conductor of the system's functionality for the external environment - it implements functions for working with the main storage and interaction with the network.

Class diagram is shown (fig. 16)

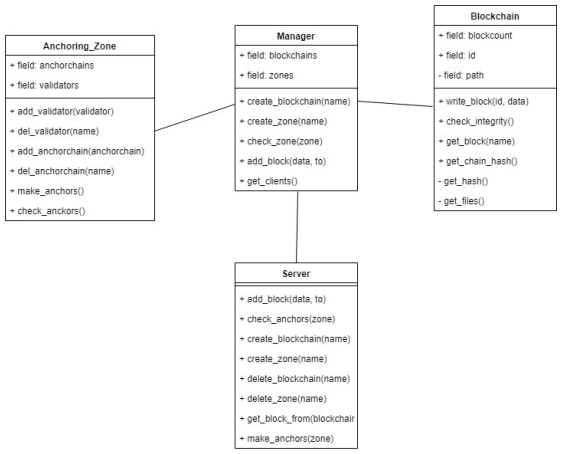

Fig 16 – class diagram.

Since it implies the ability to run the system on different devices, for the development language chosen Python3, combining power, speed of development, brevity and cross-platform.

CONCLUSIONS

The developed system allows you to ensure the security of data about the device and its software when updating and monitoring software. Each individual blocks were tested independently and the results of their testing are presented in this paper. In the future, it there is a plan to carry out testing and evaluation of the entire system as a whole with various parameters of the software and hardware.

References

[1] Donald E. Knuth. The Art of Computer Programming, vol.3. Sorting and Searching, Second Edition (Reading, Massachusetts: Addison-Wesley, 1998), xiv+780pp.+foldout. ISBN 0-201-89685-0

[2] Melanie Swan. Blockchain: Blueprint for a New Economy O'Reilly, 2015

[3] Blockchain and IoT: interaction perspectives and problems on the way of development [Online] https://forklog.com/blokchejn-i-iot-perspektivy-vzaimodejstviya-i-problemy-na-puti-razvitiya/ [Acessed: 27 Feb, 2019]

[4] Bitcoin: A Peer-to-Peer Electronic Cash System [Online] https://bitcoin.org/bitcoin.pdf [Acessed: March 2, 2019].

[5] The method of guaranteeing trust in blockchains [Online]. https://habr.com/post/338696/ [Acessed: Feb 25, 2019]

[6] Arduino vs. Raspberry Pi: Mortal enemies, or best friends? [Online] https://www.digitaltrends.com/computing/arduino-vs-raspberry-pi/ [Acessed: Feb 24, 2019]

[7] Arduino official site [Online] URL: https://www.arduino.cc/

[8] Raspberry PI official site[Online] https://www.raspberrypi.org/

[9] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of Things for Smart Cities," in IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22-32, Feb. 2014 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6740844&isnumber=6810798

[10] H.C. Pöhls, JSON sensor signatures (JSS): End-to-end integrity protection from constrained device to IoT application, in: Proc. 9th Int. Conf. Innovative Mobile Internet Serv. in Ubiquitous Comput., IMIS'15, 2015, pp. 306–312

[11] G. Prisco, Slock. it to introduce smart locks linked to smart ethereum contracts, decentralize the sharing economy, Bitcoin Mag. (Nov) (2015) (online), Available: https://bitcoinmagazine.com/articles/sloc- it-to-introduce-smart-locs-lined-to-smart-ethereum-contractsdecentralize-the-sharing-economy-1446746719, (Accessed:02 February 2019)