Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 169 (2020) 479–488

Procedia Computer Science

www.elsevier.com/locate/procedia

Postproceedings of the 10th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2019 (Tenth Annual Meeting of the BICA Society)

Integration of distributed ledger technology into software electronic signature exchange service

Grigoriy Krylov, Alena Gaybatova*, Valentina Davydenko and Asmik Grigoryan

National Research Nuclear University «MEPhI» (Moscow Engineering Physics Institute), Russian Federation, Moscow

Abstract

Object of the research: information storage, transmission and processing systems. Subject of the research: approaches to implementing the distributed ledger technology, in particular, the blockchain technology, as well as implementation issues and feasibility. Purpose of the work: enhancing the safety of data storage, transmission and processing by embedding distributed ledgers in the existing information system. Research methods: the theoretical and analytical research is supposed to underlie the study of the experience building upon the current application and possible options for boosting the potential of the distributed ledger technology, in particular, the blockchain technology. Scientific novelty. The results of the research are supposed to be used to issue proposals and recommendations on how to improve the security of data storage and processing through the use of distributed ledger technology. Practical importance. Practical perspectives consist in the fact that the discussed process of developing and implementing a distributed system simplifies further integration of the technology under consideration with the resulting blockchain service enabling data storage inside a non-modifiable registry.

Keywords: distributed ledger technology, blockchain, information security, electronic signature, distributed system, network architecture

* Corresponding author.
E-mail address: alena.gaibatova@gmail.com, davydenko-valyusha@mail.ru

10.1016/j.procs.2020.02.221

1 Relevance

The word “blockchain” has recently become popular and is increasingly used for marketing purposes to draw attention to the products and services of one or another company. The ability to exchange information in areas exposed to well-established low-level trust between participants is claimed to be the main advantage of decentralized data storage as such participants deal through intermediaries who lay down convenient rules for themselves, store data in private and closed systems that often have doubtful reliability and, most importantly, significantly slow down the existing processes. It is centralization that triggers many attack scenarios when an attacker penetrates into a shared protected resource and inflicts great damage on the entire system. The blockchain is claimed to be able to put an end to such offences because if information is duplicated on thousands of nodes, hacking one of them will not be enough. This feature of a distributed ledger is often positioned as a preemptory approach to protecting against destructive impacts and designated using the term "Attack 51%" meaning that the attacker must possess more capacity than the rest of the network, a kind of "controlling block" of generating capacities [7]. In practice, however, the immutability of a ledger does not always turn out to be a definite advantage as the efficiency of a distributed system depends on the implementation approach with the accessibility nature of the distributed network being of no little interest and given the features embedded in the very principles of the technology the latter also gives rise to numerous questions relating to monitoring, implementation, risk analysis and measures required to prevent potential threats which necessitates thorough analysis and study.

There is considerable interest in technology on the part of Russian government authorities. During the Sochi Economic Forum that took place in February 2017, Dmitry Medvedev voiced the need to embrace blockchain technology and ordered the Ministry of Communications and Mass Media to identify sectors that would benefit from it most of all. Moreover, the blockchain has been included in the digital economy program of the Russian Federation [9].

Therefore, the relevance of the topic lies in the fact that the distributed ledger technology that has emerged recently undoubtedly features high potential for the information security domain and requires in-depth exploration. Even now, business entities and government organizations around the world are trying to actively implement distributed ledgers in their information transfer systems and the number of new players seeking to popularize the use of blockchain technology is increasingly growing. New projects using this technology benefit from its key advantages such as transparency, security, safety. Therefore, it is a good reinforcement for any services exhibiting the well-established low level of trust between parties involved in the information exchange process where users are especially concerned about the risk of possible fraud or general data integrity [6].

Currently, the technology is being comprehensively explored in terms of integration possibilities, approbation, legal monitoring and review of possible pros and cons as compared to well-established approaches to the design of information systems [5].

2 Task definition

Modern information processing systems generate a huge amount of heterogeneous data at tremendous speeds. Accordingly, appropriate methods are required to structure not only safe, but also fairly fast, fail-safe and easily scalable information systems.

Due to the existence of the listed flaws of the standard blockchain network protocol used for cryptocurrencies, it requires significant reworking, which, while retaining the benefits of "immutability" as a fundamental solution to many information security issues relating to the integrity of the data bank, would provide a solution to solve the problem of scalability and unacceptably low transaction speed, for adapting and integrating DLT (Distributed Ledger Technology) technology into information protection systems.

Despite the mentioned flaws, DLT (Distributed Ledger Technology) is evolving and is not limited to public blockchain networks like cryptocurrencies and in fact departs from the mandatory participant anonymity requirement and lack of administration and it is these aspects which were animadverted upon by supervisory government authorities and representatives of the business community.

The properties of DTL based systems may vary depending on the specific implementation and may be either open or interoperable for a limited circle of people and may be also controlled by a special group of trusted users who must

not necessarily be randomly selected from the entire mass of participants involved in the exchange of information 10, 2]. A controlled ledger may be exemplified by BankChain, a banking software platform designed to facilitate the exchange of information between banking institutions connected to the system. A controlled ledger with an unlimited circle of users has been implemented by Ripple [3].

3 Analysis of the remote signature request process

Due to the development of information technology and increasingly growing presence of banks in cyberspace, it is necessary to put in place protective measures to counter attempts to exert a destructive influence on the information systems of the bank and commits fraud on the part of technically savvy cybercriminals.

The process of servicing a large mass of technical devices, including updating the firmware and executable programs, is of not less importance. Devices are located remotely and communication channels may be exposed to attacks aimed to violate the integrity of messages. DLT (Distributed Ledger Technology) [4] can help put in place the process of exchanging trusted information.

One of the possible application areas for the distributed ledger technology is the development by the bank of its own POS (Point of Sale Terminal) terminal firmware whereby it is normally sent to a vendor for affixing its electronic signature to the distribution kit, failing which it is impossible to download the distribution kit to its POS terminal [1].

Such process usually runs insecurely. Due to the low weight of compiled firmware, the distribution kit is often attached to an email and emailed to a vendor as an attachment. Mail services are not highly secure and such message may be intercepted and, worse, changed while in transit. As a result, the vendor may sign files after they have been injected with malicious code and then return them to the bank. The cybercriminal would then intercept such response email in the same way used to replace the distribution kit and will be able to upload malware onto any devices of the vendor in order to steel tangible assets and collect data on payment transactions. Moreover, if an employee of the bank recklessly uses not a received (not original) version of a payment transaction, the malicious logic will penetrate into the work environment. Thus, the exchange takes place in the manual mode without the mandatory monitoring of the integrity of the transmitted data which expose them to the risk of the distribution kit being replaced by an intruder while being transmitted to a vendor.

The commissioning of a blockchain service for a bank and its equipment vendors enabling the bank to send the distribution kit for signature and a vendor to respond by enclosing its electronic signature file or stating reason for refusal to sign must enable the trusted automated exchange of distribution kits for payment transactions, history storage and integrity control.

4 Definition of the service's business logic

The target service must be flexible enough to adapt to the special conditions that may arise within the bank's specific software development project. Its architecture must be as versatile as possible. Therefore, in determining the requirements to a payment transaction signature exchange system and its capabilities, it’s crucial to consider the peculiarities of the process being modeled and Fabric as a platform.

Depending on the target hardware platform and software to be used, distribution kits may range from a couple of tens of megabytes to several hundred megabytes and more. It is not reasonable to store heavy data in a distributed ledger. The data duplication process will take much more physical memory to save one distribution kit in proportion to the number of peers existing in the network and storing the blockchain; in addition, the load on data transmission channels would increase due to the excessively large block size. Moreover, permanent storage of payment transactions in a database, even if compiled into a machine code, is undesirable from a security point of view: developers maintain a history of code changes in a separate version control system, and the vendor does not necessarily need to have permanent access to all of the bank's written programs; only a potential criminal who manages to somehow gain access to the ledger and decides to engage in reverse engineering would stand to gain from accessing distribution kits in the blockchain.

Certificates with an electronic signature relating to a hardware vendor' distribution kit are very small in size. They are substantially a set of symbols, including metadata and electronic signature in the form of a symbol sequence. For example, X.509 type certificates used by network members in Fabric rarely exceed a quarter of kilobyte. There are

detached electronic signatures. They can be freely alienated from the signed information. No signature may be generated without a secret key and no signature will be suitable for any other digital objects. However, a signature that has already been calculated for specific files can be properly attached when installing the distribution kit.

In view of the above, the service is structured in such way that only the necessary distribution information is stored in the distributed ledger as a set of string values. Grouped together, they constitute an object in the database whose state change procedure is used to build a smart contract. The distribution kit as such is transmitted to a vendor via a secure connection, for example, via SFTP (SSH File Transfer Protocol) protocol. For these purposes, the bank may use an in-house written service. For starting download, authentication may be required involving the use of a password or a public key based on a cryptographic algorithm, for example, RSA (Rivest – Shamir – Adleman). However, the separation of the process of sending requests and electronic signatures from the transmission of distribution kits is not an exceptional measure. Given the fact that signatures are not requested very frequently and the small size of distribution kits designed for POS terminals, they can be stored in the blockchain as this will require minimal changes in applications and contract [8].

Only the required and sufficient minimum of information is entered into the blockchain, which must allow starting the download of the distribution kit and, upon completion of the transmission, identify it as a complete set of data that were not damaged during transmission. No attacker will be able to replace files during the downloading process since any violation will be detected. The identification information will be reliably and independently saved in the blockchain. Upon the consideration of the distribution kit, notes on the consideration results and an electronic signature file must be added to the corresponding object of the ledger to enable the bank to download it for installing the software on the vendor's hardware.

Thus, from a user's perspective, the service offers the same capabilities as usual e-mail box – the bank sends a request for the consideration of the distribution kit and vendor responds to the request with a signature file or reasons for rejection. Instead of e-mail clients, employees would use special applications featuring simple functionality with the complex mechanism of the service being encapsulated and hidden from users behind a simple user interface.

For the sake of simplicity, a reference to a distribution kit will hereinafter include the object in the ledger describing it.

5 Creation of a smart contract

Any Fabric service is underlain by a smart contract. It describes an object stored in the ledger and functions called by users that determine the transition between object states in due course.

For the target system, it is implemented as a set of software modules written using JavaScript. Despite the fact that initially the above mentioned interpretable scripting language was intended to run simple scripts on web pages, numerous tools have been released in recent years that extend its capabilities, such as the Node.js runtime environment. It allows running a code in server hardware, outside a browser page, and offers a set of ready-made components for accessing a file system, sending network messages, launching the server listening on a specific network port and featuring other functionality needed by a full-fledged application. Fabric contains a stable version of SDK (Software Development Kit) for writing a code for both a smart contract and custom JavaScript applications. For this reason and also due to the simplicity of the syntax and convenient testing opportunities, this particular language was chosen from among other supported ones to write all necessary program code.

An object is substantially a structure consisting of text fields. It is described as the Distributive class that contains a constructor, getters and setters, an object listing all admissible states, methods for converting an instance into a stream of binary data and back into the object. The names and descriptions of class fields are shown in Table 1 below.

Table 1. List of fields in a stored object

Field name	Description
hash	Hash of the distribution kit calculated using the specified cryptographic algorithm. The popular SHA-256 algorithm can be used. This value is transmitted separately from the software distribution kit and enables the vendor to verify the authenticity of files downloaded via SFTP.
url	Link to the file of the distribution kit. May have an arbitrary form. Must contain information using which a vendor can download the distribution kit for verification and signing.
certificate	Electronic signature of the vendor. It is the contents of the certificate file in string representation. Any file can be uploaded in such form regardless of the specific format of electronic signature.
status	Status; current state of the object. Shows the presence or absence of electronic signature. It can take one of unvarying values: UNSIGNED, SIGNED, REJECTED, REVOKED.
revokeOrRejectReason	Text describing the reasons for refusing to sign the distribution kit or revoking the signature that has already been downloaded.

To prevent mistakes in the course of finalizing a contract in the future and creation of a loophole for attacks, class requires the presence of getters and setters that will perform special functions and does not imply any direct reference to the value of a variable. The status field is not available for direct recording; the state of an object is set using a separate method for each of the possible options listed in the class module's nested object.

The functions of a smart contract are described in a separate module. They can be called by users from outside and accept a tuple of arguments from a remote client. This set should be limited to the necessary minimum to enable the updating of an object or a query, so they also do not accept an input state variable directly. Chaincode functions create an interface enabling interaction between the user and ledger; their list and internal structures must be consistent with the business logic that defines the rules for changing an object state.

Let's consider the life cycle of a distribution kit in the system (Figure 1):

The system receives the bank's request to sign the distribution kit and input hash and url values are accepted which results in an object with the UNSIGNED status being created in the ledger.

The vendor requests the list of all unsigned distribution kits that are currently located in the ledger. By downloading and verifying one of them, it can generate an electronic signature or record an identified problem. If an entry rejection request is sent with the "revokeOrRejectReason" field value, then the object will switch to the REJECTED state. If a signature request with the contents of an electronic certificate is sent, then such contents will be recorded in the "certificate" field of the object and the distribution kit will switch to the SIGNED state.

Finally, if a problem in the distribution kit is detected too late, after the certificate has been sent, then the vendor is required to record such fact in the ledger by sending a signature revoke request with a text for the "revokeOrRejectReason" field. This will result in the object switching to the REVOKED state.

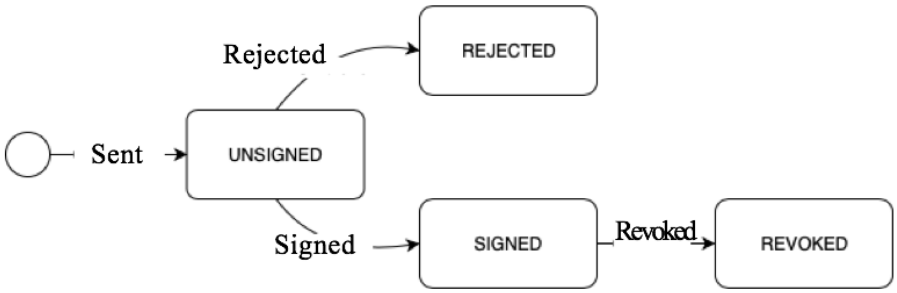

Fig. 1. Object state transition diagram

The smart contract methods allow initiating a request for switching over between the above mentioned stages of the distribution kit's life cycle and running search requests for world state objects. For the sake of compactness, all used input arguments are described as a single list:

1. ctx is a service object where a transaction's context is not sent directly from client applications but is determined by a peer when calling a smart contract. It provides access to the context of the transaction being processed, ClientIdentity and ChaincodeStubts. The former enables access to such information as organization ID, user ID, participant certificate, its attributes, etc. The latter allows creating and updating an object in the ledger, retrieving the history of its changes, deleting the object, generating a key-value pair based on object attributes, etc.
2. hash is a hash string of a distribution kit.
3. url is a link or data for downloading a distribution kit file in arbitrary form. url may not contain any authorization information.
4. certificate is the content of an electronic signature file in a symbolic representation.
5. rejectReason is a text describing the reason for refusing to sign a distribution kit.
6. revokeReason is a text describing the reason for revoking the signature affixed to a distribution kit.
7. queryString is a string representation of a NoSQL request for world state in a JSON (JavaScript Object Notation) format. It is used only within test environment; in the production environment the code of any necessary request must be specified inside the contract method. In a basic case, one method involving requesting all entries awaiting the vendor's signature without input parameters and one method involving directly requesting the result of considering a particular distribution kit by key would suffice.

Each method features protection against exceptional situations. The error returned to the client contains the description text explaining the reason for the failure to execute or refusal to approve the transaction.

When installed on a peer, the contract will be launched in a separate Docker container while modules of third-party dependencies will be ignored (node_modules folder). The required Fabric modules are included in the peer's container and will be transferred from a secure environment.

In addition to the modules of executable code, the contract catalog also contains files for generating indexes in a database that stores the world state in peers. This allows speeding up the search operation in the ledger. For recorded requests, indices were created based on object type ("distributive", being a service field specified in the contract class) and the "status" field that stores the current state.

The function first searches for an entry based on transmitted key fields, then checks that the status of the entry is “UNSIGNED”, i.e. this distribution kit has not yet been signed or rejected, following which an attempt is made to enter certificate information in the appropriate field. If any error occurs during the process, it will be returned to the client and the transaction will not be able to update the ledger.

6 Distributed system network architecture

To deploy the service, it is necessary to determine what companies, channels and nodes will be part of the network, what transaction approval policies will apply to channels and what implementation option will be used for each modular component.

The following solution was applied for modular components:

1. Kafka was used as a consensus protocol as there was no fail save and proven alternative.
2. In view of the multi-attribute structure of the object, need to regularly run search requests and theoretical need to add more complex conditions in the future, it was decided to use CouchDB, a NoSQL database, whose Docker containers are included in the standard Fabric solution as a DMS (Data Management System) for world state.
3. As mentioned earlier, JavaScript is used for writing smart contracts and applications. JavaScript uses modules from the toolkit developed by Fabric Node.js SDK.
4. At least three companies must operate within the blockchain network: BankOrg, VendorOrg, OrdererOrg for the normal functioning of the service. There is no one-to-one correspondence between each of the companies and real legal entities in terms of the Fabric platform. Each of the companies is described below to give a clue as to why such lineup was selected:
5. BankOrg represents the bank and users who are allowed to input a signature request in respect of a distribution kit in the ledger will act on its behalf. Given the fact that the service is being developed primarily for the benefit of a bank, it will be a party to each of the information exchange processes running in the blockchain network and such organization will exist as a unique instance. It has its own certificate authority and peers connected to a separate distribution channel assigned to each vendor for signing distribution kits.
6. VendorOrg is an equipment vendor's organization. Its personnel engaged in the acceptance and signing of software will act on its behalf. There may exist several vendors having access to the service. Each of them has its own separate organization respectively (and a separate channel so that information about distribution kits relating to different equipment does not overlap). It has its own certification authority and peers connected to the channel designed for communication with the bank.
7. OrdererOrg is a virtual organization actually managed by the bank. It is responsible for the general operation of the network. The nodes of the prioritization service and cluster of Kafka and ZooKeeper servers for the selected consensus protocol are deployed on its behalf. It is not authorized to write information to the ledger in any of the channels and does not store its own copies.

The network architecture is structured in a way minimizing any potential attack on the service. Therefore, separate OrdererOrg was created for security reasons – compromised certificates of one organization will not allow access to all logical links in the course of transaction processing. The channel transaction approval policy requires that a transaction must be mandatorily signed by at least two peers pertaining to each organization (BankOrg and VendorOrg channel). This will not allow entering incorrect data into the general ledger once full access to the peers of one organization has been acquired.

Docker containers are used to deploy server processes in network nodes; their images are included in the standard list of the Fabric platform's materials. They can be migrated to server equipment either manually via SSH (Secure Shell) or using tools designed for deploying and administering service applications based on Docker containers, such as Kubernetes and Docker Swarm.

For its operation, Fabric requires that logical entities' containers communicate with each other. It is not important whether a virtual overlay network is used between machines as in Docker Swarm or they will get access to the ports of the tail computer to send messages directly. The main thing is to provide access for sending network messages between entities interacting within the network.

It is quite sufficient to create a configuration file for Docker Compose in respect of each physical machine, which configuration file must specify the addresses and ports of related containers in remote network nodes (extra_hosts section in the YAML file) and environment parameters for containers to be launched. E.g., for a peer, it must specify the ports on which it will listen to incoming transactions of a smart contract, node and requests addressed directly to the blockchain, its organization ID, credentials, address and port of its CouchDB database container, etc. Launching

triggers, the mapping of host ports to the ports of Docker containers and remote containers will be available through the host network while containers on the same machine will communicate via a virtual local Docker network.

It is not recommended to use several network entities work on the same host. E.g., a node and a peer or several peer containers should not be maintained on the same orderer server. Storing copies of the ledger in the same environment does not make sense. To ensure that the service is truly decentralized, each logical unit of the fabric network must have its own operating machine. This would prevent a failure of several elements in the system if any equipment goes down.

For greater security, the physical machines being part of the service may communicate via VPN with preset firewall rules containing a white list of addresses that are allowed to connect.

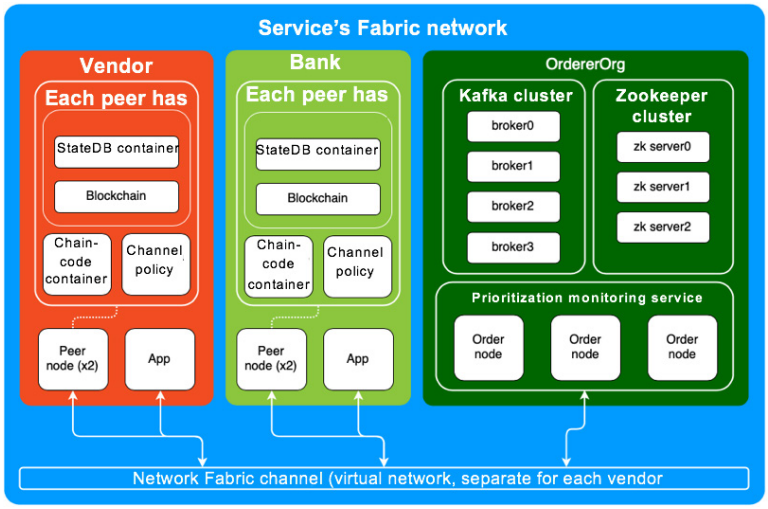

Fig. 2. Service network diagram

7 Development of user applications

Interaction between employees of organizations and a ledger requires special applications that allow sending network requests for updating and retrieving data in the distributed ledger by calling the functions specified in the smart counter. Similar to the service chaincode, they consist of Node.js executable modules, one of which serves as an entry point for any user command and a set of directories for files containing the signatures of distribution kits (a vendor's application allows sending a file with the preset name from a directory and the bank's application will save its signature file into it).

Applications for both the bank's and vendor's organizations are built in a similar way with interaction within the network taking place through Fabrik SDK modules. Therefore, let's begin by reviewing their generalized workflow that consists of the following set of sequential operations:

Selecting credentials from Wallet, a special component that locally stores users' identities. Wallet may be one of the following four types depending on its internal structure: FileSystem, In-memory, Hardware Security Module (HSM) or CouchDB. This means that user data are stored in a file system (public and private key files in an application's directory), RAM (certificates exist only while the application is running), secure hardware storage or database. Fabric SDK offers a single interface for identity management operations, which enables writing a polymorphic code that is independent of the specific implementation.

The process of connecting to the network via a gateway, a component responsible for networking, involves the use of a configuration file in YAML (Yet Another Markup Language) format that specifies IPs (Internet Protocol) and ports of peers, nodes and CA nodes as well as the absolute path to TLS (Transport Layer Security) certificates for establishing a secure connection. The application can interact with several channels in the network. To generate a request, it is required to specify the desired channel and name of the contract.

Generation of a transaction for calling a smart contract. The application must specify the channel, full name of the smart contract (name of the corresponding JavaScript class), set of input parameters and affix the signature from the user's identity.

Sending a transaction to the network. This is made via a gateway. The transaction will be sent to the node addresses listed in the channel description section of the network configuration file.

Processing of responses to a request that has been sent. The responses received from peers are compared with each other. If they contain contradictions or an error has been returned, then the text describing the situation will be displayed to the user for further action. If normal responses are returned, then, depending on the requested action, either their contents are displayed to the user (if a search request has been sent) or a transaction is automatically generated for the prioritization monitoring service's nodes to record the approved ledger update (if new data has been requested). The application subscribes to channel event notifications relating to transaction processing progress. Upon receiving a notification stating that the transaction proposal has been included in the ledger from peers, the client application will inform the user about the successful completion of the operation. If the transaction is rejected at the final processing stage, the error will also be displayed to the user.

It may seem that networking should take a long time. However, this is a delusion. Judging by the results of tests ran for the basic network configuration, it takes not more than three seconds of actual time between a user completes entering required request parameters and receives a message about the successful ledger update.

When saving the signature file of a distribution kit, the application will write the first twenty symbols from the object's hash into the file name, unless otherwise specified. This is done to avoid having too long file names that are not supported by most file systems. The code enables the safe handling of exceptions. If a user inputs incorrect information, a message explaining the error and a request for further actions will be displayed.

8 Conclusion

The developed blockchain service enables the creation of a process designed for sending distribution kits for signature to a manufacturer of electronic devices used by a bank and capable of storing exchange history and monitoring data integrity. However, its implementation involves costs, albeit low, for maintaining servers to handle rather ordinary operations and labor costs for administering the same.

The expediency of launching the service depends on a specific situation. If the target bank has to run a lot of projects requiring that payment transactions be signed by vendors, then the service may become the only solution to the mass problem: in real life, one can hardly encounter a project with more than ten working builds of distribution kits per day that are sent for signature to conduct equipment tests or for commissioning in the production environment; considering the fact that new versions of distribution kits are sent for signature within a single development project infrequently, a single service is capable of handling all developments of the bank that require signatures from the entire group of equipment vendors. For this purpose, a separate channel with its own ledger will be created for each vendor. The bank's peers and nodes will successfully handle the operation of several channels due to the low transaction load and small sizes of files stored in the blockchain.

Client applications are easy to use and will not require serious adaptation or interaction effort from the staff. Their functions can be called using command prompt shellcode which enables the integration of the service with continuous software engineering (or continuous integration (CI)) systems, such as Jenkins or TeamCity. A CI system determines the rules for considering the required tests of a new build as successfully passed and the assembled distribution kit will be sent to a vendor for signature. A user application provides an interface for interacting with the service via a short set of commands; from a CI system perspective, triggering an operation for emailing a distribution kit will not significantly differ from calling a Node.js script. Thus, the logic of the banking software development process is separated from components of the blockchain service and can be freely configured for any conditions.

Equipment vendors are often poorly motivated in transforming a process for the sake of enhanced security. Their equipment has already been purchased and certified with the bank bearing the risk of any flaws in its software. Fortunately, vendors are not required to take an active role in the implementation of the service. Their companies may elect not to participate in it if they wish so. A vendor's organization within the Fabric network and all nodes and users relating to it can be moderated by the bank's administrators. The main thing here is to structure access to resources in

such a way that control over all machines and certificates is not in the hands of a single person; a separate representative must be responsible for each organization and server group. Applications are self-sufficient and do not require a vendor to automate the distribution kit signing process. It is enough for a vendor to receive a container with client application and user credentials provided that the officer who previously emailed a response to its order has mastered a simple set of commands pertaining to the application.

REFERENCES

[1] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, Steven Goldfeder Draft. Bitcoin and cryptocurrency technology. January, 2015, Translated by ForkLog.com Version 1 (July 2015) –132 pages.
[2] Blockchain in the future: five main perspectives – [Electronic resource]. URL: https://baksman.com/news/Blokchejn-v-buducshem-pjat-osnovnyh-perspektiv.html / (accessed on: November 10, 2017).
[3] Blockchains: How They Work and Why They'll Change the World – [Electronic resource]. URL: https://spectrum.ieee.org/computing/networks/ blockchains-how-they-work-and-why-theyll-change-the-world / (accessed on: November 10, 2017).
[4] E.M. Shiryaev. Blockchain – a universal way to store and protect financial information // Student science for the development of the information society: collection of materials from the 8th All-Russian Scientific and Technical Conference: 2 volumes. V.1. Stavropol: NCFU Publishing House, 2017. 540 pages.
[5] G.O. Krylov, I.N. Loskutov Cryptocurrency – future without inflation or new AML/CFT problems? Collection of materials from international scientific-practical conference of network AML/CFT institute "Threats and risks for the global economy", November 1–3, 2016, Moscow, INI AML/CFT, ISBN978-59906571-8-2, p. 140–148/
[6] M.P. Voronov, V.P. Chasovskikh. Blockchain 0 basic concepts and role in the digital economy // Fundamental researches. - 2017. No. 9-1. - p. 30–35.
[7] N. Popper, Digital Gold. Unbelievable History of Bitcoin. How Idealists and Businessmen Invent Money all over Again / N. Popper - Williams, 2016 – 350 pages.
[8] Pedro Franco. The Blockchain//Understanding Bitcoin: Cryptography, Engineering and Economics. -John Wiley & Sons, 2014. -288 p. -ISBN 978-1-119-01916-9/
[9] The influence the blockchain on government projects - [Electronic resource]. URL: https://baksman.com/news/Vlijanie-blokchejna-na-pravitel-stvennye-proekty.html / (accessed on: November 10, 2017).
[10] Types of blockchain – [Electronic resource]. URL: http://cryptmaster.ru/howto/vidy-blokcheyna / (accessed on: November 10, 2017).