# Distributed Ledger Technology based PKI implementation using cloud-based IPFS Ethereum-gateway

Pavel Yu. Vladenkov, Tatiana A. Potlova, and Alexander V. Demidov

Orel State University, 95, Komsomolskaya str., Orel, 302026, Russian Federation
p.vladenkov@oreluniver.ru

**Abstract.** During last decade, the core of PKI – different CA's – were proved vulnerable. The article presents an alternative to the commonly used approach to implementation of PKI by using the blockhain. The reasonability to use IPFS to store the credentials is justified. The developed test bench allowed to obtain the experimental results for money costs on basic operations (adding, revoking and signing certificate) in Ethereum Gas (from 20,000 to 50,000 units) as well as time consumption of certificate's life-cycle operation (average 18.6 s) and of obtaining of its up-to-date information (870 ms).

**Keywords:** Public Key Infrastructure, Blockchain, InterPlanetary File System, Infura, Ethereum.

## 1 Introduction

The transmission of private data via the Internet requires protection from unauthorized access. It is essential to identify the participants, assure the confidentiality of transactions (information for the recipient should be available only to him and the sender) and data integrity (sent information should not be damaged or received in a modified form). This is typically provided using the Public Key Infrastructure (PKI) Its main functions are creating, distributing, signing, and revoking digital certificates and public keys. The primary drawback of traditional PKI implementations is the reliance on a third-party certification authority (CA), which operates at all stages of the certificate lifecycle. Certification authorities are expected to act according to some rules that are defined by the Certification Policy (CP) and Certification Practice Statement (CPS) [1].

However, this does not provide guarantees of protection against external interference or the malicious issuance of inaccurate certificates, which is confirmed by the known cases of attacks of various kinds [2]. The search for an improved solution is relevant. Usually, PKI is based on DBMS and functionates at the level of OS file

system. An alternative would be to implement a PKI based on Distributed Ledger Technology (DLT, blockchain) and InterPlanetary File System (IPFS).

## 2 Materials and methods

### 2.1 Ethereum Blockchain

To implement the solution, it is demanded to create a decentralized PKI. Including creating software based on smart contracts of the Ethereum blockchain, which will allow users to manage identifiers and attributes (and their types) associated with cryptographic keys. To get rid of centralization in the system, CA will be replaced by an alternative method of issuing certificates. The certificate authorities don't exist on the Web of trust, it's required a chain of signatures to be established from users instead. This approach provides greater security when authenticating. If one signing party is compromised, the other party's key will be revoked, and the impact on the network as a whole is limited.

For a DLT-based implementation to work, it must support smart contract creation. A smart contract is a software in the Ethereum network that automates the process of concluding transactions and monitoring their execution [3]. It stores and manages identification attributes, their types, and keys. It replaces trusted third parties as well, that is, they act as intermediaries between the parties to the contract. When executed, the smart contract consumes Gas.

Gas Ethereum is a resource (in ETH cryptocurrency) that is spent on sending a transaction, publishing, and executing contracts. Together with the transaction, the user specifies the amount of gas for the operation and its cost per unit in Gwei. The latter has a ratio of 1:1,000,000,000 to ETH. The volatility of the ETH rate is very high at the moment. On December 15, 2020 the ETH:USD rate was 1:588.79, on December 28, 2020 – 1:730.41 [4]. That is why Gas itself will be used to estimate future monetary costs, not fiat currency or Ether.

The language for writing smart contracts in Ethereum is Solidity. It is designed directly for the Ethereum virtual machine, statically typed, and supports inheritance, libraries, and complex custom types [5].

The memory inside the blockchain is very expensive, so there is a need to store large files separate from the blockchain. IPFS is a good alternative in this regard.

IPFS is located one-dimensionally on all devices of the network and is available to everyone, each user can make their changes and view all changes [6]. Each file or block receives a unique fingerprint – a cryptographic hash. Thus, the system avoids duplication. Each node stores only the files or blocks it needs on its device and also indexing information, which includes data on the location of files in the node. If the user needs a file that is not on his device, he searches the network for nodes that store content by the hash of this file.

## 2.2 Ethereum-based Public Key Infrastructure

The developed solution is based on the SCPKI project Trustery [7], which is adapted to the problem of accurate identity management in the framework of a decentralized PKI and allows transparently operate with certificates while using Etherium. In the presented implementation it has been adapted to work with the Infura service and some procedures have been modified to be used with modern versions of the libraries.

To reduce the cost of a transaction, IPFS technology is used. In this case, the hash of the certificate will be stored inside the blockchain. Logging is automatic, based on the character of the blockchain: the complete history of transactions performed will always be located entirely in the blockchain.

Three methods have been developed to implement PKI: adding a certificate, certificate signature, and certificate revocation.

Using the proposed solution involves connecting to a private blockchain network and IPFS.

Figure 1 shows a UML-deployment diagram of the test bench.

To launch the solution it is required to publish a smart contract in advance in a private network and enter its address and the actual data on the server addresses.

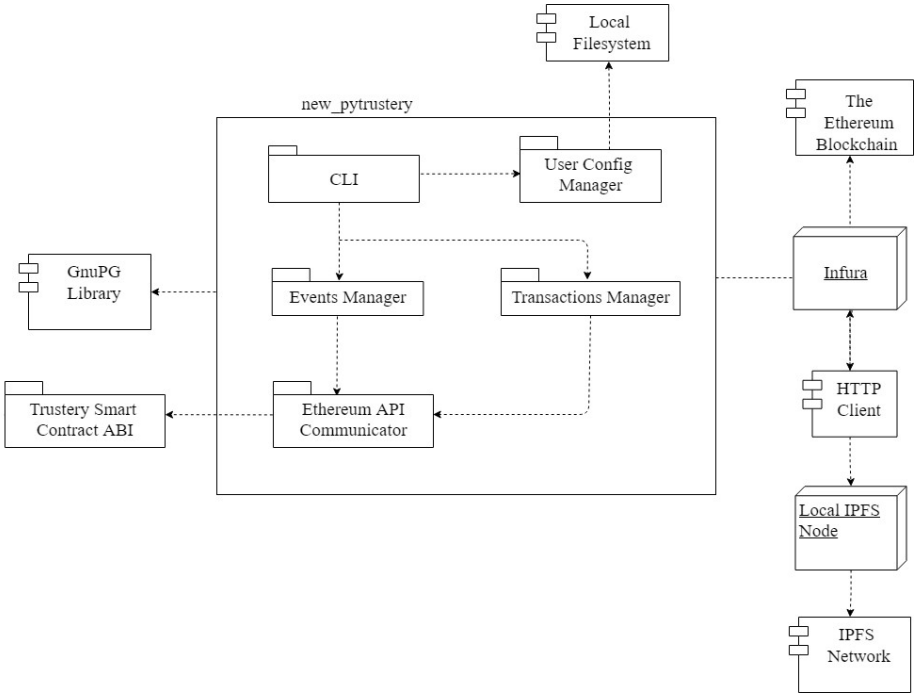


**Fig. 1.** UML-deployment diagram of the test bench.

# 3 Results

To evaluate the solution, money spendings, resource costs, and time costs were analyzed.

## 3.1 Money spendings

The efficiency of implementation in IPFS technology was evaluated by comparing gas costs for each type of transaction, the results are presented below.

Smart Contract Publishing Transaction. The cost of publishing a new smart contract is expressed in the following formula:

$$Gas_1 = FixedGas_1 + CodeGas + InitGas \quad (1)$$

$FixedGas_1$ is a fixed gas amount that must be deposited for any smart contract code in order to this code be processed and sent to the DLT (for instance, the fixed cost of creating a new transaction is 21,000 Gas, and the fixed cost of deploying a new contract is 32,000 Gas on December 2020 [9]).

CodeGas is the cost of storing a smart contract (i.e. code) on the blockchain.

InitGas is the total cost of calculations when executing contract constructors and contract initialization.

Storing the identification information of the owner of the contract requires a store operation (which currently costs 20,000 Gas), which is included in InitGas.

The cost of deploying a contract is about 401,000 Gas and it remains constant since it does not accept dynamic variables during the creation process. It should be noted that only one smart contract needs to be published for the application to work.

Function call transactions. In this implementation, to work with a certificate on the blockchain, events are used that are stored in a separate data structure called Transaction Receipt. The transaction cost is calculated using the following formula:

$$Gas_2 = FixedGas_2 + FunctionGas \quad (2)$$

$FixedGas_2$ – the cost of the transaction that calls the contract while passes parameters to it. FunctionGas – the cost of the function call.

Adding a certificate. The function accepts input values that are dynamic in size, it requires 35,000 Gas or more. The function call itself is cheap because it is a transfer of attributes to event and an increment of one variable. When adding an attribute from IPFS, an input is a fixed-size string containing the hash of the file in IPFS and the function has a fixed cost of about 36,000 Gas. Figure 2 shows a graph of changes in the cost of adding an attribute with and without IPFS.

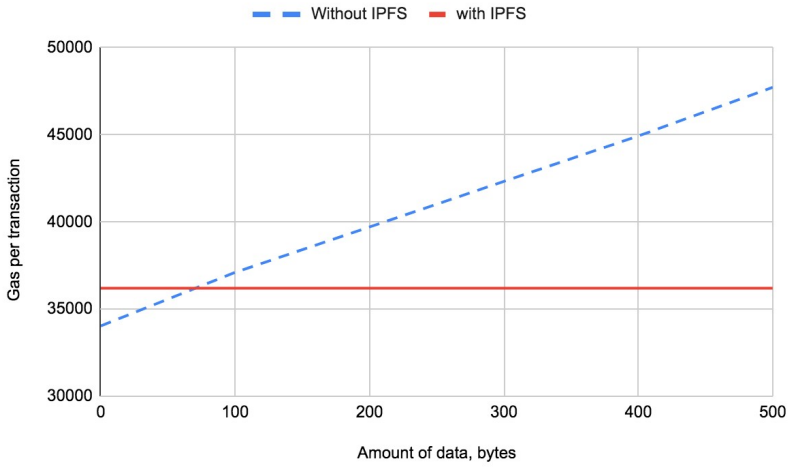


**Fig. 2.** Comparison of amount of Gas per translation for solutions with and without IPFS.

It is obvious that the use of IPFS can significantly reduce costs as the volume of data increases.

Adding a public key. The cost of adding a public key to the blockchain via IPFS is around 36,000 Gas. The public part of the key, the text containing the Ethereum address from which the key was sent and the fingerprint from this text are sent to IPFS so that it can be verified. The hash of the file with the key and text is sent to the contract function for it to be written to the event.

Certificate signature. The cost of calling a function is about 51,000 Gas, this is because the address from which the signature was made is recorded when calling. This is necessary so that the signature can only be revoked by the same account that created it. Information about the time for which it is valid is sent along with the signature.

Certificate revocation. It takes about 30,000 Gas to call the revocation function. A signature to revoked is submitted as an input. As already mentioned, it can only be revoked by the owner.

### 3.2 Resource costs

The main needs are computing resources that are aimed at launching the blockchain client. The current implementation of the system is based on the Ethereum GETH client, it starts up and performs all the necessary functions properly on simple hardware (two CPU cores, 4 GB of RAM and free space of no more than 200 GB will be enough). A good Internet connection is important. All processes are organized on the Ubuntu operating system.

### 3.3 Time costs

In the Ethereum blockchain, the time of the creating a new block changes over the time. On December 2020, it takes about 13 seconds to create a new block (it used to be up to 17 seconds) [10]. This time is valid if there is enough free space in the new blocks [11].

To estimate the actual time cost, it is important to consider how many times the transaction has propagated. In case of transaction overload, waiting for the addition to the block can take longer. But this can be overcome by offering a higher price for a unit of gas. (in other words, by willingness to pay more for faster confirmations). As an illustration of the random mining time, the transaction confirmation time was measured as the difference in seconds between the time the transaction receipt was received and the time the transaction was sent. The results are shown in Figure 3.

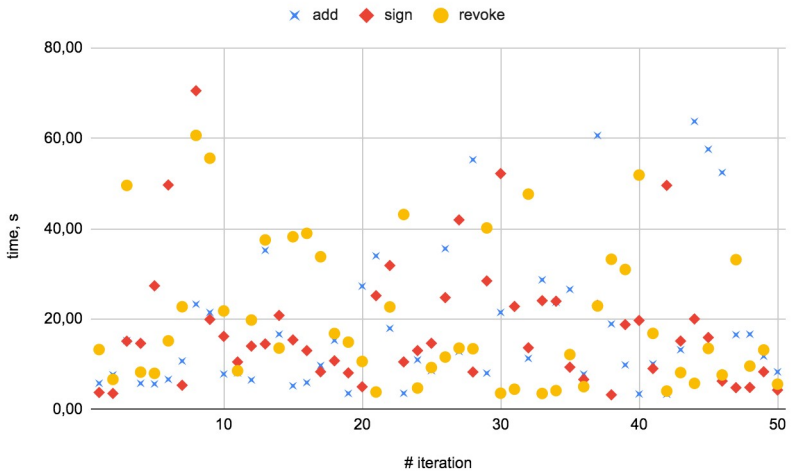


**Fig. 3.** Results of the check of the transactions confirmation time.

To reduce the number of variables for determining the transaction confirmation time, it is possible to consider the blockchain itself as a measurement of time along the block height (that is, the distance of the block from the genesis block) [11]. The blockchain is regarded as a timestamping service. Based on the solution, the transaction confirmation timestamp is the height of the block in which it was mined. This means that we can measure the time it takes to confirm a transaction as the difference in block height between the last known block that the network mined during transaction deployment (relative to the originator of the transaction) and the block in which the transaction was inserted. To check the confirmation time of transactions, it was accordingly measured and expressed in the difference of the block number in which the transaction with the last known block fell when the transaction was sent (Figure 4).

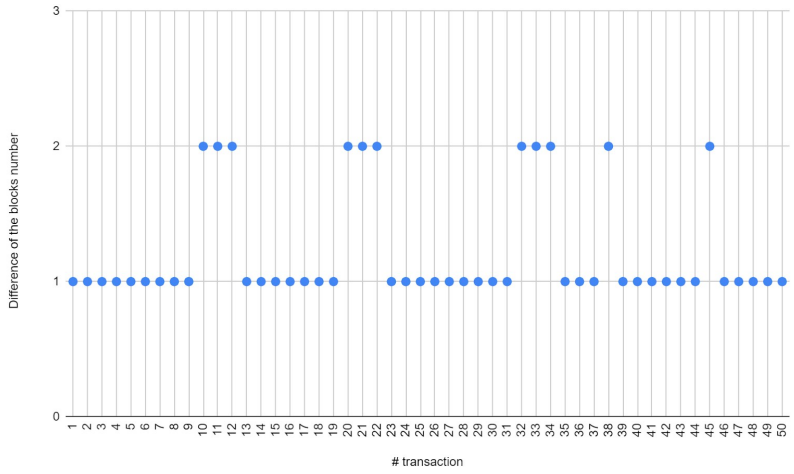


**Fig. 4.** Traction of confirmation time in blocks.

To evaluate the usability of the solution for the user, the time taken to process a request to obtain up-to-date information about the certificate was measured, the results are shown in Figure 5.

The average processing time for a request to obtain up-to-date information about a certificate is 890 ms. The average response time of existing OCSPs is up to 375 ms [12]. The increase in speed in the developed solution is due to the use of the Infura cloud service, which can be overcome by storing the entire blockchain locally.

## 4 Discussion

The disadvantage of the solution is that the storage of a distributed ledger in network nodes takes more and more disk space with an increase in the number of transactions over time. Blockchain means storing a complete copy of all consensus data transactions on each storage device. Geth provides various types of synchronization (full, fast, and light), but this still does not completely solve this problem. Even with private blockchain development, registry size will grow rapidly.

The current solution for accessing the resource requires the client to be a member of the distributed ledger. To overcome this limitation, it is possible, for example, to supplement the solution with BlockQuick technology [13] or FlyClient [14], which provide less resource-intensive protocols for general network users.

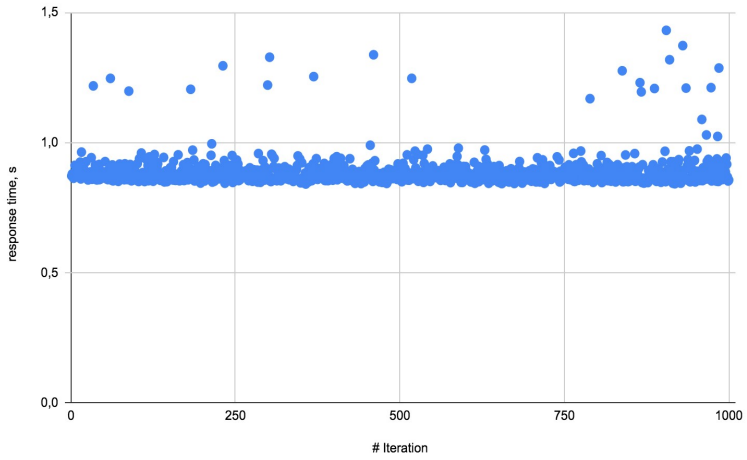


**Fig. 5.** Response time for access to certificate.

## 5 Conclusion

In the paper, an Ethereum-based Public Key Infrastructure implementation was presented. The main advantage of the new solution is the absence of a centralized authority and better protection against unauthorized interference into the system. Although there are earlier solutions with this approach (Trustery, IKP, etc.), in this work new results were obtained: a cloud-based IPFS-gateway solution Infura was applied, which makes it easier to design a test bench and allows to perform more experiments on fewer hardware resources. The performance was assessed and the preferability to use decentralized storage for storing files outside the blockchain, such as IPFS, was substantiated.

## 6 Acknowledgments

The work was supported by The President of Russia Scholarship for young scientists SP-2130.2019.5.

## References

1. Kubilay, M.Y., Kiraz M.S., Mantar H.A.: CertLedger: A new PKI model with Certificate Transparency based on blockchain. In Computer Security, 85. 333–352, Elsevier Ltd (2019).
2. What will Happen to WoSign, https://sslmate.com/blog/post/history_of_ca_sanctions, last accessed 2020/10/30.

3. Introduction to smart contracts, https://ethereum.org/en/developers/docs/smart-contracts, last accessed 2020/11/07.
4. ETH/USD – Ethereum US Dollar, https://investing.com/crypto/ethereum/eth-usd-historical-data, last accessed 2020/12/28.
5. Solidity, https://solidity.readthedocs.io/en/v0.7.3/, last accessed 2020/10/30.
6. IPFS Decentralization, https://docs.ipfs.io/concepts/what-is-ipfs/#decentralization, last accessed 2020/10/30.
7. Al-bassam, M: SCPKI: a smart contract-based PKI and identity system. In Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, 35–40, Association for Computing Machinery, NY, USA (2017).
8. Infura, https://infura.io/, last accessed 2020/10/30.
9. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 1–32 (2020).
10. Ethereum Average Block Time Chart, https://etherscan.io/chart/blocktime, last accessed 2020/12/28.
11. Di Francesco Maesa, D., Mori, P., Ricci, L.: A blockchain based approach for the definition of auditable Access Control systems. In Computer Security, 84, 93–119, Elsevier Ltd (2019).
12. Understanding OCSP Times and What They Mean for You, https://www.digicert.com/dc/blog/ocsp-times-and-what-they-mean-for-you/, last accessed 2020/11/22.
13. Letz, D.: BlockQuick: Super-Light Client Protocol for Blockchain Validation on Constrained Devices. IACR Cryptol. ePrint Arch. (2019).
14. Bunz, B., Kiffer, L., Luu, L., Zamani, M.: FlyClient: Super-Light Clients for Cryptocurrencies. IACR Cryptol. ePrint Arch. (2020).