A Blockchain-Based Access Control System for Cloud Storage

Ilya Sukhodolskiy, Sergey Zapechnikov
Department of Cryptology and Cybersecurity
National Research Nuclear University MEPhI (Moscow Engineering Physics Institute)
Moscow, Russia
suhodolik@gmail.com, SVZapechnikov@mephi.ru

***Abstract*—In this paper, we present a prototype of multi-user system for access control to datasets stored in an untrusted cloud environment. Cloud storage like any other untrusted environment needs the ability to secure share information. Our approach provides an access control over the data stored in the cloud without the provider participation. The main tool of access control mechanism is ciphertext-policy attribute-based encryption scheme with dynamic attributes. Using a blockchain-based decentralized ledger, our system provides immutable log of all meaningful security events, such as key generation, access policy assignment, change or revocation, access request. We propose a set of cryptographic protocols ensuring privacy of cryptographic operations requiring secret or private keys. Only ciphertexts of hash codes are transferred through the blockchain ledger. The prototype of our system is implemented using smart contracts and tested on Ethereum blockchain platform.**

Keywords—cloud storage; attribute-based access control; ciphertext-policy attribute-based encryption; blockchain

I. INTRODUCTION

In the last few years, services to remotely store and sync user data on cloud-based services have increased. A lot of users store their documents in clouds. Nevertheless, there are some security problems and copyright aspect. The essential problem is transferring data to the external environment, such that anyone else other than the owner can get access to information.

On the other hand, it is difficult to give in to the numerous facilities that provide services for data storage: backup files, the ability to access their documents from any device from anywhere in the world, easy transfer of files to other users. You can find several ways to solve the problem of secure remote file storage. But the most effective of them is to encrypt data before sending. Encryption is one of the main protective mechanisms recommended by the Cloud Security Alliance. However, encryption imposes certain difficulty to use the data and the collective access to them.

Currently, there are not so many tools and techniques to protect data stored on cloud servers and at the same time providing tools for a comfortable management. Some utilities propose to encrypt individual files before sending to the cloud, e.g. "BoxCrypt" [1]. There are also various tools for developing secure web applications with access to databases, such as «CryptDB» [2], «ARX» [3]. They use different encryption schemes, different approach to their use.

There are means to ensure the integrity and non-repudiation, their operation based on blockchain usage. In particular, "BigchainDB" [4] is designed for distributed cloud storage of information with a guaranteed assurance of its integrity and non-repudiation.

The rest of the paper is organized as follows. In section 2 we describe the concept of the project system and the main advantages of the chosen approach. Further, in section 3 the selected scheme of attribute-based encryption and modifying it. Section 4 describes the stage review of the policies and interaction protocols for the Ethereum virtual machine. Section 5 concludes the study and identifies a few directions of further research.

II. ACCESS CONTROL SYSTEM

The intended approach to solving the problem is to develop an access control model based on blockchain transactions, storing data in untrusted storage, and implementation of attribute-based encryption-based Ethereum smart contracts. We use attribute-based access control model [5]. The most widely-used standard for attribute-based access control is XACML [6]. This standard describes the necessary components of access control system, its purpose, interaction and using methods.

It is expected that the system can be applicable for different data type, for example, multimedia information, electronic documents, etc. To store this amount of data directly in the blockchain is not advisable, as increasing the number and increasing the size of the blocks, the complexity of Ethereum will increase multiple, which will primarily affect the cost of transactions. Therefore, data will be stored in cloud storage, wherein the information identifying the file, will only be available in the blockchain.

To determine the set of security mechanisms applicable to the user's information resources, it is necessary to classify them firstly as either publicly available or restricted. To do this, the user must be given the opportunity to convert files and directories with the appropriate attributes.

It is assumed that public information resources does not require additional security measures to prevent access of cloud

 978-1-5386-4340-2/18/$31.00©2018 IEEE

service provider. At the same time, the restricted information resources require protection from unauthorized access of any persons not authorized by the end user in an explicit form, including cloud services provider and other third parties. For this reason, the restricted information should be encrypted by the user before they made any attempts to transfer it to the external environment, and thus, it can be placed and stored in the cloud only in encrypted form.

Thus, in the case of restricted information is required to obtain all necessary encoding information, encryption to send data to the cloud and add an appropriate entry in the blockchain. In the case of public information, the implementation of the first and the third item is skipped.

The blockchain ensures the integrity and non-repudiation of information. A list of all changes can be tracked by means of the chain blocks, thus, to change the earlier recording is not possible. A copy of such chain is stored at each participant of the network that also allows you to always recover the information. The unit is also information about the author of the document, rights and other data.

The Ethereum platform is designed to create decentralized services based on the blockchain. It is a single distributed virtual machine. Smart contracts Ethereum, unlike Bitcoin, support cycles that, on the one hand, led to the introduction of fees for their implementation, called gas, and has greatly expanded their possible applications on the other. Change the virtual machine state can be written in Turing complete scripting language. For each file the user creates a smart contract, which will store information about the owner, access policy, a hash sum of the stored information, information to identify the cloud, and all changes that will occur with the file. Due to the fact that the information stored in the blockchain is public it is necessary to encrypt information before sending it to storage, and control access to it.

III. Attribute-Based Encryption

The project uses a decentralized scheme [7] to control access to encrypted data. This scheme is most suitable for controlling access to encrypted data in cloud environments, but there is no opportunity to change attributes or to specify dynamic access policy.

Mate Horvath proposed in [8] a multi-authority CP-ABE scheme for effective revocation of user's attributes based on their identities. A formal proof of his scheme's security is carried out in a generalized model of bilinear groups and random oracle model. The Horvath's attribute encryption scheme consists of the following algorithms. The illustration of the proposed scheme is shown in Fig. 1.

Circuit initialization algorithm takes a security setting and provides global system's parameters $GP: (\lambda) \to GP$.

Certification Authority configuring algorithm takes as input global parameters GP and outputs a pair of private and public keys (SK^*, PK^*) of Certification Authority (CA): $(GP) \to (SK^*, PK^*)$.

Private key generation algorithm is performed by Certification Authority after a user's request for an authenticated secret key. It checks if request is correct and generates private key using global parameters and revoked users list: $(GP, RL, GID, SK^*) \to K^*_{GID}$.

Public key attributes generation algorithm outputs a key pair (SK, PK) for each user taking global parameters GP as the input: $(GP) \to (SK, PK)$.

Attribute key generation algorithm takes as input global parameters, global user's identifier, an attribute belonging to a user, and the secret key. The output is a key for this attribute/identifier pairs: $(GP, SK, GID, i) \to K_{i,GID}$.

Encryption algorithm takes a message M, access matrix (A, p), a set of receiver's public keys $\{PK\}$, public key of CA PK^*, revoked users list RL and global settings GP and outputs a ciphertext CT: $(GP, M, (A, p), \{PK\}, PK^*, RL) \to CT$.

Decrypion algorithmt takes global parameters GP, revoked users list RL, ciphertext CT, identify key CT and a set the appropriate keys for attribute/identifier pairs $\{PK\}$ for the fixed value of GID. At the output we obtain a plaintext message M if the set of attributes corresponds to the ciphertext access matrix $(GP, CT, (A, p), \{K_{i,GID}\}, K^*_{GID}, RL) \to M$. Otherwise, the decryption will fail.

To add the ability to specify a dynamic access policy was used the method of Yuan, which does not require the participation of other participants in the system. This method allows you to ensure that all system participants will not be able to ignore the new rules, when the owner of the file to install them. Previously, it was required that the user interacted with attribute authority (AA) to update your key, but the attacker had the ability to opt out and keep the old permissions.

Yuan proposed in [9] a way to define dynamic access policies without changing user keys. The method is based on the observation, which says that all ciphertexts consist of several components: the primary ciphertext; components of the ciphertext, which are related with attributes.

Thus, to change the access policy you want to change just some components. In the first step the user generates a new matrix in accordance with the new policy. Then you need to change only the components of the ciphertext are associated with variable attributes. This operation requires fewer resources than encrypting the whole file.

Yuan proposed a way to define dynamic access policies without changing user keys. The method is based on the observation, which says that all ciphertexts consist of several components:

- the primary ciphertext;
- components of the ciphertext, which are related with attributes.

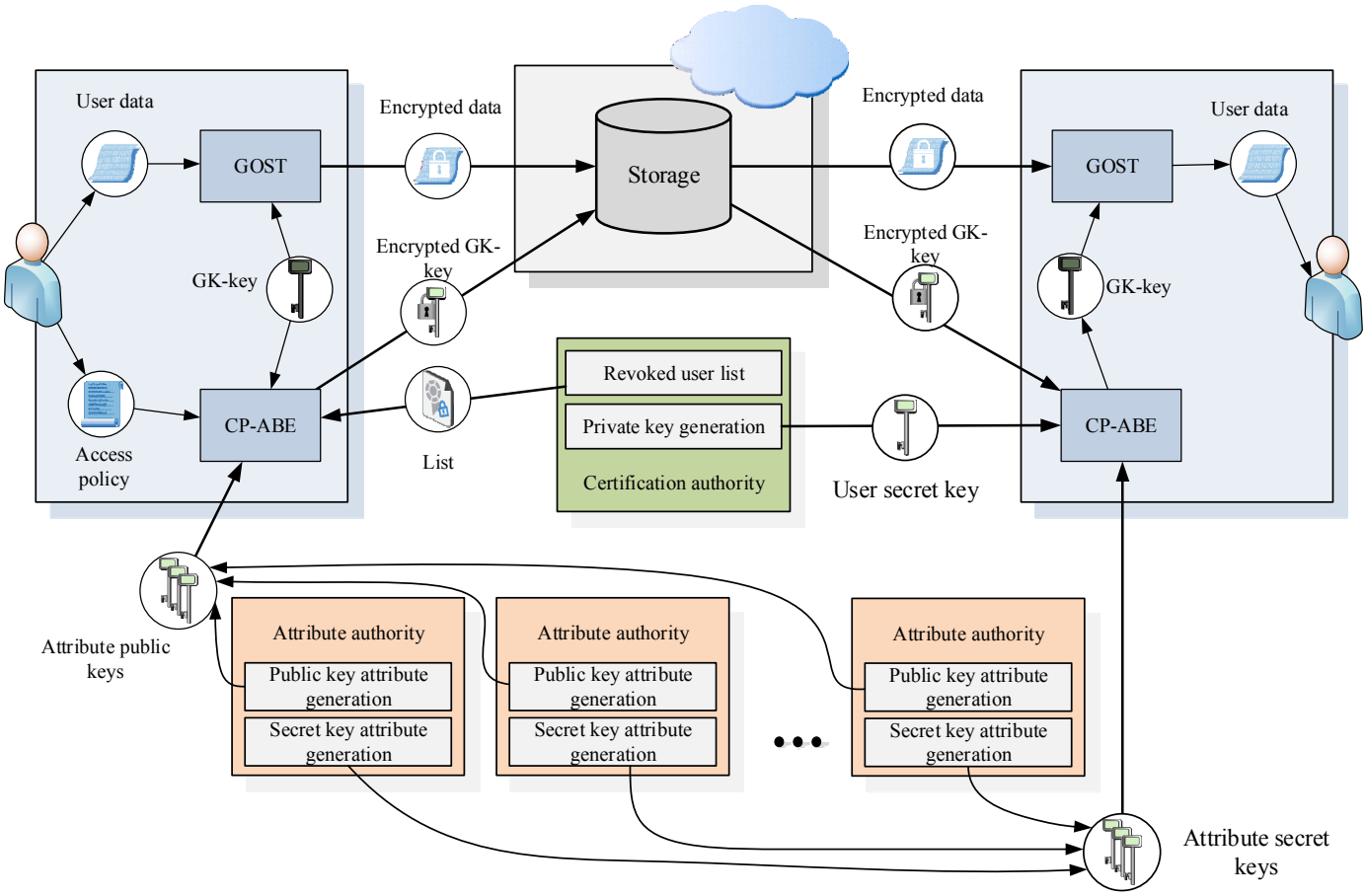

Fig. 1. ABE scheme to access control in cloud storage.

Thus, to change the access policy you want to change just some components. In the first step the user generates a new matrix in accordance with the new policy. Then you need to change only the components of the ciphertext are associated with variable attributes. This operation requires fewer resources than encrypting the whole file.

The matrix update receives on input the global parameters GP, the public keys of the attributes $\{PK\}$, public key of the user PK^*, a list of revoked user RL, secret key of the user K^*_{GID}, the old access structure (A,ρ), new access patterns (A',ρ) and returns the update components UC: $(GP, K^*_{GID}, (A,\rho), (A',\rho), \{PK\}, PK^*, RL) \rightarrow UC$.

Access policy in the Horvath schema, as in many other ciphertext-policy attribute-based encryption schemes based on the linear secret sharing scheme. This means that it is possible to apply the method of the recalculation of the access matrix to maintain a dynamic access policy. There are only four ways to change the matrix: delete attribute associated with "OR" or "AND", or add an attribute that is associated with "OR" or "AND".

IV. Interaction Protocols

For the implementation of the prototype a set of protocols for interaction among the participants of the system was constructed.

Most functions are performed on the client device (CD), due to the fact that most exchanging data is private. The remaining functions are performed using smart contracts loaded in the Ethereum virtual machine (EVM). All operations in the EVM is presented in the form of contracts that were related to user, AA or CA.

The system generated CA-contract to initialize in a EVM. As well, the certification authority can be trusted dedicated server. Keys are generated at client device. When registering a new AA CA-contract sends a request to CD. Then CD releases a new certificate and sends it to the storage in the CA-contract. Any participant in the system can verify its validity by contacting CA-contract after you publish the certificate.

The certification authority creates a new user contract and generates the keys for interaction with the system when registering new user. Further actions in the system can only be done through this contract. The user contract should refer to the CA-contract. Then CA generates a key on the server and transmits it in encrypted form [10] to the user contract.

To register new attribute authority AA-contract is created in EVM. The AA server generates keys for all attributes, and then sends a registration request to the CA. Central authority records the certificate data in the AA-contract after confirmation of the registration.

If the user who has the right to some attribute refers to the AA-contract attribute authority provides keys to users. Client device generates $K_{i,GID}$, then it encrypts the key $pk_{GID,Enc}$ and sends an encrypted copy of the key in the user contract. It is worth noting that the user may soon be convinced of the validity of its certificate due to the properties of EVM when accessing attribute authority.

The scheme of interaction between the Client, CA and AA is depicted on Fig. 2. To store data, a contract file is created. It contains information about the location of the file in the cloud storage, its access policy and additional owner's information. Interaction with the file may be carried out using the contract. Four types of interaction is supported in the system: create, modify, read and delete.

To create file the user encrypts file by attribute-encryption scheme on his own device, and then sends the ciphertext to the cloud, and records the public link, the hash code of the file and the access policy in the contract.

For changing the file's access policy, CD performs the update of the access matrix and components of the ciphertext. Then updates the information in the contract file, and replaces components of the ciphertext in the cloud.

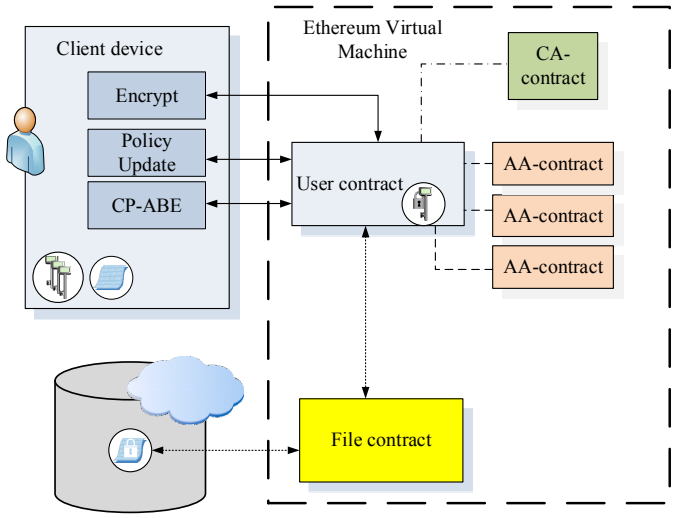

Fig. 2. Interaction of Client, CA and AA.

When deleting a file, the contract file self-destructs and CD should remove it from the cloud. After deleting the file, the link to it cannot be used again in the system to eliminate the possibility of disputes. A user wishing to read a file must match the access policy and have the necessary keys to decrypt. After checking for policy compliance, the user receives a link to the file and can download it, and then to decipher. If the user does not meet access policy, then the file it is to decipher even if he will be able to link to it.

V. Conclusion

The main result of this work is the implementation of a software system prototype that implements the access control model of the system to data stored in untrusted environments. To implement the system algorithms have been selected acceptable complexity, functionality, and complexity of implementation.

Key benefits of access control system are: the ability to customize the access policy for the encrypted data without duplicating them to a large number of participants; the ability to define dynamic access policies; access policy change does not require any additional action from other members of the system, which avoids the need for regular changes to user keys; the integrity of information about all transactions, including the granting and changing access, facts gain access to file, rejection of the fact and the inability to edit these data is guaranteed through the use of the blockchain and smart contracts.

Acknowledgment

The authors acknowledge support from the MEPhI Academic Excellence Project (Contract No. 02.a03.21.0005).

References

[1] The Boxcryptor website. [Online]. (2017) Available: https://www.boxcryptor.com/en/

[2] Popa R. A., Redfield M., Zeldovich N. CryptDB Protecting Confidentiality with Encrypted Query Processing. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, pages 85–100, 2011

[3] Poddar R., Boelter T., Popa R. Arx: A Strongly Encrypted Database System. (2016) IACR Cryptology ePrint Archive. [Online]. Available: https://eprint.iacr.org/2016/591.pdf

[4] McConaghy T., Marques R., Muller A. BigchainDB: A Scalable Blockchain Database. (2016) BigchainDB whitepaper. [Online]. Available: https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf

[5] Sukhodolskiy I. A., Zapechnikov S. V. An access control model for cloud storage using attribute-based encryption. In Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian (pp. 578-581). IEEE.

[6] OASIS Standard. eXtensible Access Control Markup Language (XACML) Version 3.0. 2013. 154 p.

[7] Lewko A. and Waters B. Decentralizing attribute-based encryption. Springer, 2011, pp. 568-588.

[8] Horvath M. Attribute-Based Encryption Optimized for Cloud Computing. In SOFSEM 2015, LNCS 8939, pp. 566-577.

[9] Yuan W. Dynamic Policy Update for Ciphertext-Policy Attribute-Based Encryption. IACR Cryptology ePrint Archive, 2016, 457.

[10] Russian State Standard 34.12 2015. Cryptographic protection of information. Moscow, Standartinform Publ., 2015. 25 p. (In Russian)