

On some methods of storing data in the cloud for a given time

Banu Yergaliyeva, Yerzhan Seitkulov, Dina Satybaldina, Ruslan Ospanov

Department of Information Security, Information Technology Faculty, L. N. Gumilyov Eurasian National University,
Nur-Sultan, Republic of Kazakhstan

Article Info

Article history:

Received Oct 05, 2021

Revised Feb 25, 2022

Accepted Mar 05, 2022

Keywords:

Client-server communications

Cloud computing

IoT

Secure cloud storage

Secure data storage

Time-lapse cryptography

ABSTRACT

In this work we study some methods of storing data in the cloud for a given time using secret sharing technology. Such problems are especially relevant in the context of the rapid development of the internet of things (IoT). Chips, smart cards and other physically small devices, as a rule, have significant memory limitations, so there is a need to use cloud storage as an auxiliary tool for secure data storage. We present method for secure storing data in distributed servers, provided that servers remote from each other do not collude with each other. In the paper we also consider two diferent methods for storing data using various cryptographic solutions, such as Shamir's secret sharing method, El Gamal scheme, diffie-hellman key distribution protocol.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Banu Yergaliyeva

Department of Information Security, Information Technology Faculty

L. N. Gumilyov Eurasian National University, Nur-Sultan, Republic of Kazakhstan

Email: banu.yergaliyeva@gmail.com

1. INTRODUCTION

In this article we research some methods of storing data in the cloud for a given time using secret sharing technology. This task is to encrypt data that can only be decrypted after a specified time in the future. There are interesting practical applications for solving this problem. For example, we can ensure that diaries, records, or other data are “sealed” for a certain period of time, and in such a way that even the author of this data could not “unseal” them before the specified period. A useful practical application can be to protect important data that has been obtained as a result of some scientific research or experiments, until they are completed and published. This may be necessary to prevent information leaks or pressure from any interested parties. For example, when bidding, you can hide bidders' price offers until the end of the trading session. Another case is when intermediate voting data can be protected until it is completed in order to avoid affecting the voting process. The scope of the solution to the problem of sending a secret client's data to the future can be very extensive and includes not only auctions and voting, but also e-commerce, financial markets and their regulation, and law.

Over the past time, a number of interesting approaches to solving the problem of encryption in the future have been described. Many reseachers used the so-called “time-lock puzzles” to solve this problem and described an encryption scheme with partial key escrow (partial key escrow protocol). In [1], a cryptographic protocol is built that encrypts client data in such a way that decryption of this data is guaranteed no earlier than the specified exact time, even if this decryption is undesirable for the sender. This protocol is based on the Pedersen distributed key generation protocol, the Feldman verifiable threshold secret sharing protocol, and the El Gamal encryption algorithm. Rabin and Thorpe noted a difference between existing protocols such as theirs, in which the time from the moment of encryption to the moment of decryption is fixed, and other protocols that only give an estimate of this time or set a lower limit to the estimate. Their solution to the

problem of encryption in the future was called time-lapse cryptography (TLC). The authors of TLC received a patent [2] for their invention. In [3]-[6] presented a cryptographic protocol for encrypting data for a given time, based on TLC. This protocol is based on the distributed key generation protocol based on discrete logarithm on elliptic curves, the Pedersen verifiable threshold secret sharing protocol and the El Gamal encryption algorithm based on elliptic curves. The protocol was called elliptic curve time-lapse cryptography (ECTLC).

This research is intended to solve the problem of deploying a secure and reliable distributed network of participants of a service that provide key generation. The scientific novelty of this research is that it is assumed that the protocol of encryption for a given time will be based on new more efficient algorithms that expand its functionality. In particular, to ensure data encryption for a sufficiently long time, it is assumed to use the proactive secret sharing protocol. To date, various variants of proactive secret separation have been developed, such as, for example, in the works [7], [8]. The protocol presented in section 2 will be a combination of 4 known protocols: i) distributed key generation protocol, ii) asymmetric encryption algorithm protocol, and iii) proactive secret exchange protocol, and iv) electronic digital signature algorithm.

The protocol provides for the use of the agreed parameters of an asymmetric encryption algorithm such as, for example, a prime number p , a generating element g of prime order q in the case of the El Gamal encryption algorithm, or elliptic curve modulo a prime number p , the elliptic curve equation, the equation coefficients a and b of the field F_p , the elliptic curve point G of prime order q in the case of the El Gamal encryption algorithm on elliptic curves. Therefore, it will be necessary to research and select the most effective algorithms, protocols, and parameters for the protocol being developed.

The main stages of the protocol are supposed to be as follows:

- 1) The key generation using distributed key generation and proactive secret sharing protocols. The service can generate keys on a repeating basis; for instance, it can create keys with a service life of one month every week, or it can create keys with a service life of 2 hours every 30 minutes. This schedule is posted by managers on an open information stand. In addition, the service can receive some requests from clients to generate new keys with a specified service life; workers of the server receive these requests, and then post these requests on an open information stand. Service workers create keys according to the protocol. After, managers of the service sign set of keys, and publish the signed a set of keys on an open information stand.
- 2) Encrypting data using public keys generated by the Service, with a specified period.
- 3) Decryption of data using private keys that are generated by the Service when the specified period is reached.

Section 3 is dedicated to the research and development of secure outsourcing methods for storing data using secret sharing technology. Such problems are especially relevant in the context of the rapid development of the internet of things (IoT) [9]-[25]. Chips, smart cards and other physically small devices, as a rule, have significant memory limitations, so there is a need to use cloud storage as an auxiliary tool for secure data storage. The idea of the new approach is to develop a method for storing data using various cryptographic solutions, such as shamir's secret sharing method, diffie-hellman key distribution protocol.

2. METHOD OF STORING DATA IN DISTRIBUTED SERVERS

So, in this section, we explore a simple protocol model for storing client data for a given time. Participants in the model under consideration: i) portal for accepting applications from clients for storing confidential information (data), ii) client is a user of the portal, and iii) service is three distributed servers, remote from each other. it is assumed that the servers do not collude with each other, that is, they do not transmit confidential data to each other.

General description of the model: i) the client makes a standard entrance to the portal, ii) the client sends a request to the portal to encrypt the data, and iii) the portal sends a client's request to the service specifying the client's identifier, the time the request was sent and the time before which the client's data cannot be decrypted.

- Step 1: each server receives a request from the client to encrypt data for a specified time, generates its own private key, calculates its own public key and sends it to other servers in the service. That is, let g and p are some public parameters, while the number g is the primitive root modulo p , and numbers x_1, x_2, x_3 are secret keys of three servers, respectively.

Server 1 generates its public key $h_1 = g^{x_1} \bmod p$, similarly servers 2 and 3 generate their public keys $h_2 = g^{x_2} \bmod p$ and $h_3 = g^{x_3} \bmod p$ respectively. And all three servers exchange their public keys with each other Figure 1.

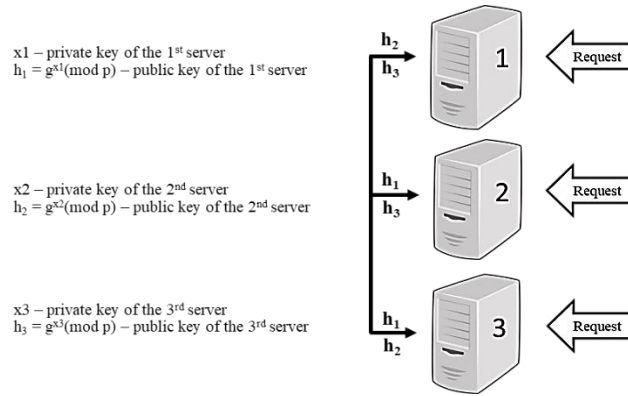


Figure 1. Description of step 1

- Step 2: each server receives the public keys of the other servers, calculates the shared public key $h = h_1 h_2 h_3 \pmod p$, and save it in their database with binding to the client identifier (ID) and time parameters specified in the request and sends the calculated public key h to the portal.
- Step 3: the portal sends the received public key h to the client. Next, the client generates a symmetric key s for data encryption, encrypts his data with this key, then encrypts the symmetric key with the received public key h according to the El Gamal scheme:

$$a = g^k \pmod p, \quad b = h^k s \pmod p, \quad (k, p - 1) = 1, \quad 1 < k < p$$

and sends encrypted data and encrypted symmetric key (a, b) to the portal.

- Step 4: the portal saves the received encrypted data and the encrypted key in its database with reference to the client ID and time parameters specified in the request.
- Step 5: the portal, upon reaching a time before which it is impossible to decrypt the client’s data, or later than this time, sends a request to the service to obtain a private key indicating the client’s identifier, public key and time parameters.
- Step 6: each service server receives the request, sends its own private key to the rest of the servers. Each server of the service receives the private keys of the other servers, calculates the shared private key, stores it in its database with binding to the corresponding shared public key $x = x_1 + x_2 + x_3 \pmod{\varphi(p)}$ and sends the private key to the portal Figure 2.

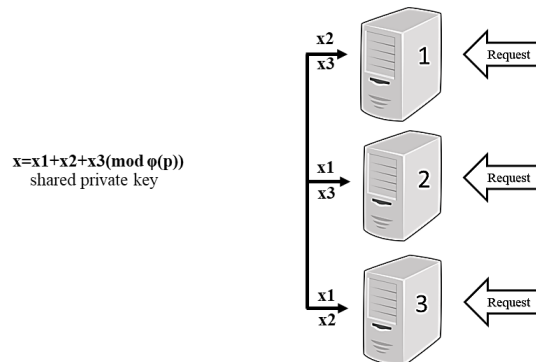


Figure 2. Description of step 6

- Step 7: the portal receives the private key x , and decrypts the symmetric key with this key:

$$s = b(a^x)^{-1} \pmod p$$

And then it decrypts the client’s data. Indeed, there is a chain of equalities:

$$h = h_1 h_2 h_3 \pmod p = \{(g^{x_1} \pmod p)(g^{x_2} \pmod p)(g^{x_3} \pmod p)\} \pmod p = g^x \pmod p$$

3. METHOD OF STORING DATA IN AN UNSECURE SERVER

Suppose client wants to store his data (files, drawings, and photographs) on outsourcing, that is, in the cloud storage. For the exchange of information between clients and the server, standard secure information exchange protocols can be used. The problem is that the cloud storage (server) is not trusted. That is, during the storage period of the received data, intruders may intervene, although the server itself, which is considered an automated system, does not deviate from the interaction protocol. That is, the server, for its part, protocolly wants to protect itself from such unauthorized interventions as substitution of information, and distortion of data content, precisely during the period of data storage. Thus, the data transmitted to the server is not secret, but the server wants to avoid distortion and substitution of information by intruders, and therefore performs all the actions described in the information exchange and data storage protocols.

In addition, let us assume that the server itself does not violate the interaction protocol at the time of transmission and exchange of information with clients, that is, there is no information leakage. But long-term storage of the data itself in the cloud can be unsafe, therefore, the original data, after being transferred to the server, must be kept encrypted using standard symmetric encryption algorithms, such as block cipher (GOST), and advanced encryption standard (AES). Thus, for each client and server, the task is to generate a shared secret key that will be used by the server as the encryption key for the data of a particular client. In such a model, the server is protocolly interested in “forgetting” this public key, but being able to recover this key to decrypt the data only with the participation of the client who owns the data.

3.1. Data storage method using classical asymmetric cryptography

So, let's say client i needs to send data to the cloud for storage in the encrypted type.

- Step 1: client i and server choose a sufficiently large prime p and number d . The client i and the server, independently of each other, choose random natural numbers a and b , respectively.

Next, client i calculates the number A_i :

$$A_i = d^a \bmod p;$$

and the server finds the number B :

$$B = d^b \bmod p.$$

Here the number a is the secret key of the client i ; b is the server's private key. Numbers A_i and B are the public keys of the client and server, respectively.

- Step 2: key distribution is carried out using the well-known Diffie-Hellman protocol:
 - 1) Client i calculates and sends its public key to the server $A_i = d^a \bmod p$, and the server sends its public key to the client $B = d^b \bmod p$.
 - 2) The server calculates the number $Q_i = A_i^b \bmod p$, and the client similarly calculates the same number $Q_i = B^a \bmod p$, because $B^a \bmod p = d^{ab} \bmod p = A_i^b \bmod p$.
 - 3) As a distributed key k_i take the number Q_i . That is, we have a shared secret key between a specific client i and the server: k_i .
 - 4) The server encrypts (for example, AES or GOST) the original data of client i using the shared secret k_i , and stores encrypted data in its own storage. We denote this encrypted data $F(i)$.
 - 5) Now the client i and the server delete from their storages the numbers a and b , respectively, that is, they “forget” them. This protocol agreement to “forget” your private keys is mandatory and is performed in an automated manner - both on the client side and on the server side.
- Step 3: using Shamir's secret sharing technology:
 - 1) Let l denote the value of the product of two numbers: $l = Q_i A_i \bmod p$;
 - 2) Server and client independently form the same polynomial

$$f(x) = k_i + lx \bmod p$$

- 3) Server and client randomly share a shared secret k_i into two keys using Shamir's secret sharing technology. Let us denote them $S(i) = (x_1, f(x_1))$ и $C(i) = (x_2, f(x_2))$. Both the server and the client keep these shared secrets and keep them secret.
- 4) Now the server and client are deleting numbers from their stores k_i and l from polynomial $f(x) = k_i + lx$, that is, they “forget” these parameters according to the protocol. Before removing the key k_i the server calculates the hash value $h(k_i)$ and this value is sent to the server store.

- Step 4: the server forms the client base, that is, for each client i , only the following information will be stored: i) $h(k_i)$ (hash value from key k_i), ii) $S(i)$ (shared server secret), iii) A_i (client's public key), and iv) $F(i)$ (encrypted customer data i).

Here $h(k_i)$, the value of a cryptographic hash function (such as SHA-3) from a distributed public key k_i between client i and server. Hash storage is required for client authentication. Note that the information in the server's client base is sufficient for the server to recover the shared secret k_i , which will then be used to decrypt the data and then transfer the decrypted data back to the client. Thus, this algorithm allows to safely store client data in the cloud in encrypted form. At the same time, it is easy to see that the resistance of the protocol to active and passive attacks is ensured due to the reliability of the cryptographic system Rivest-Shamir-Adleman (RSA), but on the condition that all parties of the client-server interaction do not deviate from the protocol.

3.2. Data storage method using elliptic curve cryptography

In this section, we will show an analogue of the data storage method described above, but using elliptic curve cryptography [21]. So let's say client i needs to send big data to the cloud for encrypted storage.

- Step 1: let the general elliptic curve be chosen

$$E_p(a, b): y^2 = x^3 + ax + b \pmod{p}, \quad (4a^3 + 27b^2) \pmod{p} \neq 0,$$

and point G on it is a generator, that is $G, [2]G, [3]G, \dots, [q]G$ are the different points, and $[q]G = O$ for some prime number q .

Client i chooses a random number r_i , $0 < r_i < q$, which it stores as its secret key, and calculates a point on the curve $R_i = [r_i]G$, which will be his public key. Likewise, the server randomly generates the number d_s , $0 < d_s < q$, which stores as its secret key and calculates a point on the curve $D_s = [d_s]G$.

The following parameters are also public and public data: p, a, b, G, q .

- Step 2: key distribution is carried out using the well-known Diffie-Hellman protocol on an elliptic curve:
 - 1) Client i calculates and sends its public key to the server $R_i = [r_i]G$, and the server sends its public key to the client $D_s = [d_s]G$;
 - 2) The server calculates the point $Q_i = [d_s]R_i$, and the client computes the same point in the same way $Q_i = [r_i]D_s$, since $d_s R_i = d_s r_i G = r_i D_s$;
 - 3) As a distributed key k_i take the first coordinate x_1 of the point $Q_i(x_1, y_1)$. That is, we have a shared secret between client i and server: $k_i = x_1$.
 - 4) The server encrypts the original data of client i using the shared secret k_i , and stores the data in encrypted form in its storage. We denote this encrypted data $F(i)$.
 - 5) Now server and client i are deleting private keys d_s и r_i respectively, that is, "forget" them.
- Step 3: using Shamir's secret sharing technology:
 - 1) Let us denote by x_2 the value of the first coordinate of the sum of two points on an elliptic curve $Q_i + R_i = L$;
 - 2) Server and client now independently form the same polynomial

$$f(z) = k_i + x_2 z \pmod{q}$$

- 3) Server and client randomly share a shared secret k_i into two keys using Shamir's secret sharing technology. Let us denote them $S(i) = (z_1, f(z_1))$ и $C(i) = (z_2, f(z_2))$. Both the server and the client keep these shared secrets and keep them secret.
 - 4) Now the server and client are deleting points Q_i and L from their repositories, and also remove numbers k_i и x_2 from a polynomial $f(z) = k_i + x_2 z$, that is, these points and numbers "forget". Before removing the key k_i the server calculates the hash value $h(k_i)$ and this value is sent to the server store.
- Step 4: the server forms the client base, that is, for each client i , only the following information will be stored: i) $h(k_i)$ (hash value from key k_i), ii) $S(i)$ (shared server secret), iii) R_i (client's public key), and iv) $F(i)$ (encrypted customer data i).

Here $h(k_i)$ – the value of a cryptographic hash function (such as SHA-3) from a distributed public key k_i between client i and server. Hash storage is required for client authentication. As in the previous case, the information in the server's client base is sufficient for the server to recover the shared secret k_i , which will then be used to decrypt the data and then transfer the decrypted data back to the client. As we can see, the elliptic curve cryptographic system was used by analogy with classical cryptography in the task of storing

data in the cloud. But elliptic curve cryptography has clear advantages in practical implementation, has greater resistance to attacks, and so on.

4. CONCLUSION

In this paper, new protocols for secure storage outsourcing are presented. Based on the cryptographic protocol described in section 2, it is possible to design new cryptographic systems encryption data of client for a given time. The need for such a cryptographic application exists in the Republic of Kazakhstan. In particular, this problem is also relevant for the web portal of electronic public procurement of the Republic of Kazakhstan, which provides encryption of the data of users of the portal (suppliers), with the ability to decrypt clients' data through a certain time. The developed protocol will allow developing an alternative approach to solving the problem of the electronic public procurement portal, and will ensure the impossibility of interference in the public procurement process both on the part of the customer and supplier, and on the part of the portal administration.

ACKNOWLEDGEMENTS

This research work was carried out with the financial support of the Ministry of Digital Development, Innovations and Aerospace Industry of the Republic of Kazakhstan, grant No. AP06850817.

REFERENCES




- [1] M. O. Rabin and C. Thorpe, "Time-lapse cryptography," *Technical report TR-22-06, Harvard University School of Engineering and Computer Science*, 2006. [Online]. Available: <https://dash.harvard.edu/bitstream/handle/1/26506434/tr-22-06.pdf?sequence=1&isAllowed=y>.
- [2] M.O. Rabin and C.A. Thorpe, "Method and apparatus for time-lapse cryptography," *U.S. Patent*, 2007. [Online]. Available: <https://patentimages.storage.googleapis.com/e5/05/65/0919b8287335d9/US8526621.pdf>
- [3] R. Doku, D. B. Rawat, and C. Liu, "On the Blockchain-Based Decentralized Data Sharing for Event Based Encryption to Combat Adversarial Attacks," in *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1033-1043, 1 April-June 2021, doi: 10.1109/TNSE.2020.2987919.
- [4] R. Alvarez and M. Nojournian, "Comprehensive Survey on Privacy-Preserving Protocols for Sealed-Bid Auctions," *Computers & Security*, vol. 88, Jan. 2020., doi: 10.1016/j.cose.2019.03.023.
- [5] J. Sun and N. Liu, "Incentivizing Verifiable Privacy-Protection Mechanisms for Offline Crowdsensing Applications," *Sensors (Switzerland)*, vol. 17, no. 19, Sep. 2017. doi: 10.3390/s17092024.
- [6] N. C. Luong, D. T. Hoang, P. Wang, D. Niyato, and Z. Han, "Applications of Economic and Pricing Models for Wireless Network Security: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2735-2767, Fourthquarter 2017, doi: 10.1109/COMST.2017.2732462.
- [7] J. Baron, K. Eldefrawy, J. Lampkins, and R. Ostrovsky, "Communication-Optimal Proactive Secret Sharing for Dynamic Groups, Cryptology," *13th International Conference, Applied Cryptography and Network Security*, 2015, doi: 10.1007/978-3-319-28166-7_2.
- [8] J. Brendel and D. Demirel, "Efficient proactive secret sharing," *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, 2016, pp. 543-550, doi: 10.1109/PST.2016.7907013.
- [9] Y. N. Seitkulov, S. N. Boranbayev, G. B. Ulyukova, B. B. Yergaliyeva, and D. Satybaldina, "Methods for secure cloud processing of big data," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 3, pp. 1650-1658, 2021, doi: 10.11591/ijeecs.v22.i3.pp1650-1658.
- [10] I. Ahmed, "A brief review: Security issues in cloud computing and their solutions," *Telkonnika (Telecommunication Computing Electronics and Control)*, vol. 17, no. 6, pp. 2812-2817, 2019, doi: 10.12928/telkonnika.v17i6.12490.
- [11] R. M. Ospanov, Y. N. Seitkulov, N. M. Sissenov, and B. B. Yergaliyeva, "An example of an internal function for the SPONGE scheme," *Vestnik Sankt-Peterburgskogo Universiteta, Prikladnaya Matematika, Informatika, Protsessy Upravleniya*, vol. 17, no. 3, pp. 287-293, 2021, doi: 10.21638/11701/spbu10.2021.306.
- [12] Y. S. Abdulsalam, M. Hedabou, "Security and Privacy in Cloud Computing: Technical Review," *Future Internet*, 2022, 14(1), 11, doi: 10.3390/fi14010011.
- [13] T. B. Seong, V. Ponnusamy, N. Z. Jhanjhi, R. Annur, and M. N. Talib, "A comparative analysis on traditional wired datasets and the need for wireless datasets for IoT wireless intrusion detection," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 2, pp. 1165-1176, 2021, doi: 10.11591/ijeecs.v22.i2.pp1165-1176.
- [14] H. Tabrizchi, M.K. Rafsanjani, "A survey on security challenges in cloud computing: Issues, threats, and solutions," *The Journal of Supercomputing*, vol. 76, pp. 9493-9532, 2020, doi: 10.1007/s11227-020-03213-1.
- [15] A. H. Ali, M. N. Abbod, M. K. Khaleel, M. A. Mohammed, and T. Sutikno, "Large scale data analysis using MLlib," *Telkonnika (Telecommunication Computing Electronics and Control)*, vol. 19, no. 5, pp. 1735-1746, 2021, doi: 10.12928/TELKOMNIKA.v19i5.21059.
- [16] Pronika, S. and S. Tyagi, "Performance analysis of encryption and decryption algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 2, pp. 1030-1038, doi: 10.11591/ijeecs.v23.i2.pp1030-1038.
- [17] M. F. Falah et al., "Comparison of cloud computing providers for development of big data and internet of things application," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 3, pp. 1723-1730, 2021, doi: 10.11591/ijeecs.v22.i3.pp1723-1730.
- [18] S. Zaineldeen and A. Ate, "Improved cloud data transfer security using hybrid encryption algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, 20(1), pp. 521-527. doi: 10.11591/ijeecs.v20.i1.pp521-527.
- [19] Y. S. Abdulsalam and M. Hedabou, "Decentralized Data Integrity Scheme for Preserving Privacy in Cloud Computing," *2021 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, 2021, pp. 607-612, doi: 10.1109/SPAC53836.2021.9539946.
- [20] S. Villamil, C. Hernández, and G. Tarazona, "An overview of internet of things," *Telkonnika (Telecommunication Computing*

On some methods of storing data in the cloud for a given time (Banu Yergaliyeva)




- Electronics and Control*), vol. 18, no. 5, pp. 2320-2327, 2020, doi: 10.12928/TELKOMNIKA.v18i5.15911.
- [21] J. H. Yeh, "A PASS scheme in cloud computing – protecting data privacy by authentication and secret sharing," *Proc. of International Conference on Security and Management*, 2011. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.218.719&rep=rep1&type=pdf>
- [22] A. Bhawiyuga, D. P. Kartikasari, K. Amron, O. B. Pratama, and M. W. Habibi, "Architectural design of IoT-cloud computing integration platform," *Telkommika (Telecommunication Computing Electronics and Control)*, vol. 17, no. 3, pp. 1399-1408, 2019, doi: 10.12928/TELKOMNIKA.V17I3.11786.
- [23] J. Pan and J. McElhannon, "Future Edge Cloud and Edge Computing for Internet of Things Applications," in *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439-449, Feb. 2018, doi: 10.1109/JIOT.2017.2767608.0
- [24] A. Boranbayev, S. Boranbayev, Y. Seitkulov, and A. Nurbekov, "Proposing Recommendations for Improving the Reliability and Security of Information Systems in Governmental Organizations in the Republic of Kazakhstan," *Advances in Intelligent Systems and Computing*, vol. 3, pp. 854-868, Jan. 2021, doi: 10.1007/978-3-030-63092-8_57.
- [25] Y. N. Seitkulov, R. M. Ospanov, and B. B. Yergaliyeva "On one method of storing information for a specified time," *Vestnik KazNRTU*, vol. 143, no. 3, 2021, pp. 167-174. doi: 10.51301/vest.su.2021.i3.22.

BIOGRAPHIES OF AUTHORS






Banu Yergaliyeva    was born in the Arkalyk city, Republic of Kazakhstan. She graduated from the department of mathematics, ENU, Nur-Sultan, in 2005. She is a specialist in cloud computing, secure cloud storage, cryptography, Internet of Things. She has over 5 articles in peer-reviewed journals indexed in Scopus and she is currently a doctoral student on the specialty "Information Security". Also she is a researcher at the Institute of Information Security and Cryptology. She can be contacted at email: banu.yergaliyeva@gmail.com.






Yerzhan Seitkulov    was born in the Karakastek village, Almaty region, Republic of Kazakhstan. Graduated from the Department of Mechanics and Mathematics, MSU. He received his PhD in 2006, and he is a specialist in cryptography, cloud computing, secure cloud storage, Internet of Things. He has over 25 articles in peer-reviewed journals indexed in Scopus. Currently, he is the director of the Institute of Information Security and Cryptology, as well as a professor at the Department of Information Security, ENU. He is also the winner of 8 scientific projects under the MESRK. He can be contacted at email: yerzhan.seitkulov@gmail.com.



Dina Satyaldina    was born in the Republic of Kazakhstan. She received her PhD in physics in 2000 and her PhD in cryptography in 2011. She is a specialist in coding theory, applied cryptography, quantum cryptography. She has over 20 articles in peer-reviewed journals indexed in Scopus. Currently, she is head of the Department of Information Security, ENU. She is also the winner of 2 scientific projects under the MESRK. She can be contacted at email: dinasaty@gmail.com.



Ruslan Ospanov    was born in Karaganda city, Republic of Kazakhstan. He is a specialist in applied cryptography. Has over 2 articles in peer-reviewed journals indexed in Scopus. He is currently a research fellow at the Institute of ISC, ENU. He is also the Deputy Dean for educational work of the Faculty of Information Technology. He teaches and works with students specialized courses on modern cryptography. He can be contacted at email: ospanovrm@gmail.com.