BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **21**, No 4
Sofia • 2021 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2021-0046

A Three-Tier Authentication Scheme for Kerberized Hadoop Environment

M. Hena, N. Jeyanthi

School of Information Technology and Engineering, VIT Vellore, Tamilnadu, India
E-mails: henashabeebvit@gmail.com njeyanthi@vit.ac.in

***Abstract*:** *Apache Hadoop answers the quest of handling Bigdata for most organizations. It offers distributed storage and data analysis via Hadoop Distributed File System* (*HDFS*) *and Map-Reduce frameworks. Hadoop depends on third-party security providers like Kerberos for its security requirements. Kerberos by itself comes with many security loopholes like Single point of Failure* (*SoF*), *Dictionary Attacks, Time Synchronization and Insider Attacks. This paper suggests a solution that aims to eradicate the security issues in the Hadoop Cluster with a focus on Dictionary Attacks and Single Point of Failure. The scheme roots on Secure Remote Password Protocol, Blockchain Technology and Threshold Cryptography. Practical Byzantine Fault Tolerance mechanism* (*PBFT*) *is deployed at the blockchain as the consensus mechanism. The proposed scheme outperforms many of the existing schemes in terms of computational overhead and storage requirements without compromising the security level offered by the system. Riverbed Modeller* (*AE*) *Simulation results strengthen the aforesaid claims.*

Keywords: *Apache hadoop, authentication, bigdata, blockchain, Kerberos.*

1. Introduction

A tremendous increase in the generation of data has been witnessed in recent years and this data needs to be processed innovatively and efficiently to get useful information that can lead to strategic business decisions. Big data is the term used to indicate these voluminous, unstructured, heterogeneous data that cannot be stored and processed by traditional computing systems. Apache Hadoop is a big data platform based on Java that offers distributed storage and processing termed as Hadoop Distributed File System and MapReduce, respectively.

The security of Bigdata is a very serious issue to be taken care of to make the best use of big data analytics benefits. Most of the issues were caused due to unauthorized access and resultant manipulation or retrieval of data. Apache Hadoop came without any security features in its initial versions as it was designed to work

in an internal project environment. Most of the recent Hadoop technologies rely on third-party security systems like Kerberos Protocol for incorporating security into it. The user is issued with Ticket Granting Tickets and Service Tickets to get the session key for its communication with the secured Hadoop Cluster. However, the Kerberos Protocol itself has some in-built pitfalls, which include Single Point of Vulnerability or Failure (SoV/SoF), Password Guessing or Dictionary Attacks, Insider Attacks and Time Synchronization Problem. Active researches are happening across the globe to enhance the security issues in Kerberos Enabled Hadoop Clusters. For instance, Rahul and Kumar [1], put forward an authentication framework that combines cryptographic techniques, hashing and random number generation to get a unique key for clients. But this solution made the system slower and adds computational overhead.

In a Kerberos-enabled Hadoop Cluster, the Key Distribution Center (KDC) is a Single Point of Failure in the sense that any failure or attack on the KDC affects the entire authentication system. The KDC comprises an Authentication Server (AS), a Ticket Granting Server (TGS) and a local database. This paper proposes a three-tier authentication framework that has its roots in Secure Remote Password Protocol (SRP), One Time Passwords (OTP) and Threshold Cryptosystems. The focus of this work is to eradicate Password Guessing Attacks and Single Point of Failure Problem in the Kerberized Hadoop Clusters. For that, the local database at the KDC is replaced with a Blockchain network for distributed storage. This is tamper-proof storage and cannot be hence compromised. This eradicates the issues that arise when the local storage at the KDC is compromised. Next, as in Secure Remote Password (SRP) Protocol [2], the password or any details about it are not directly shared with the KDC. Instead, a salted hash of the password along with the user public key is shared. This is verified using the result mined from blockchain storage. Thus, password-guessing attacks can be avoided and the session key for the user to communicate with the Ticket Granting Server is securely shared. An enhanced One Time Password is used as proposed by Hena and Jeyanthi [3], to verify that the session key is computed correctly at both ends, that is, at the user and the AS. The user needs to respond correctly as per the pre-agreement which will be further elaborated in the coming sections. The threshold cryptography ensures the authentication system availability by deploying multiple Ticket Granting Servers. Thus, even if one of the Ticket Granting Server is failed or compromised, a prefixed threshold number of Ticket Granting Servers (TGSs) can collaborate and accomplish the authentication function. The proposed system achieves the desired level of security suitable for real-time big data systems with less communication and computational overhead. Moreover, the usage of tamper-proof blockchain technology pushed out storage management from the KDC. Practical Byzantine Fault Tolerance (PBFT) [4] Mechanism is used as the consensus mechanism at the blockchain network. It demands more than 2/3 of the total number of nodes need to honest in contributing the mined result. The nodes in the network are arranged such that each node can communicate with each other and there is no permanent leader. The nodes leadership comes in turn.

The remaining of this paper is organized in six sections. Section 2 explains the Preliminaries and Mathematical Intuitions. Section 3 discusses the related works. Section 4 presents the system being proposed. The implementation details and result analysis are described in Section 5. Section 6 concludes the paper.

2. Preliminaries and mathematical intuitions

2.1. Secure remote password protocol

Secure Remote Password Protocol (SRP), zero-knowledge proof protocol, in which the client/user demonstrates to the server that he/she knows the right password. The password or any other information from which the password can be deducted is not directly sent via the network. In other words, the password stays within the client system itself and the server or any other entity has no clue about what it is. The protocol is hence resilient to dictionary attacks and doesn't rely on any trusted third parties.

To register, the client submits username $uname$, a random salt, s and salted hash of client's password p, x as verifier, and verifier v:

(1) $$v = g^x,$$

and

(2) $$x = H(s, p).$$

Here, g is a generator of predetermined group G, which is an additive group with a multiplication operation. This information is stored by the server in its local database for future authentication.

To authenticate the user, the user chooses a random number a as its secret and generates the public key as

(3) $$A = g^a.$$

This value along with the username is sent to the server as an authentication request. The server looks up the corresponding verifier and salt values stored against the username in its database. Also, it generates its public key as

(4) $$B = v + g^b.$$

where b is a random number generated at the server side.

Hence, the server's public key is blinded with v, a value derived from the user's password. A random value u is also generated on the server-side. The server sends these values, that is, the salt s, its public key B and random u to the client. The client re-computes the value of x using hid/her password and salt s received from the server. A common secret, S is computed at both sides as

(5) $$S = \begin{cases} (B - g^x)^{a+ux} & \text{at client side,} \\ (A \times v^u)^b & \text{at server side.} \end{cases}$$

Both sides then hash the value of shared secret S to obtain the session key K as

(6) $$K = H(S).$$

To verify the key, both sides make use of this key to send messages M_1 and M_2:

(7) $$\text{Client to Server: } M_1 = H(A, B, K),$$

(8) $$\text{Server to Client: } M_2 = H(A, M_1, K).$$

Both sides re-compute the messages received and verify the messages received from the other end are the same as the computed one. If same, the user is authenticated.

2.2. The e-OTP mechanism

The e-OTP or Enhanced OTP mechanism works as follows.

Instead of just entering the OTP received by the user in this mobile number or email as such, the user should reply with a code as per the pre-agreement. The user and the Authentication Server shares a Pre-Shared Key (PSK) during the registration process. This will be some random number. During authentication process the authentication server sends an OTP code to the user, which is another random number. User has to respond back with the digits in the PSK located at positions denoted by digits in the OTP. That is, for example, assume the digits in the PSK is "81 52 13 74 65 26 37 98 49" and the OTP send by the AS is "2 1 4 7". The user has to reply with digits at 2nd, 1st, 4th and 7th positions in the received file. That is, "5 8 7 3" in this case.

2.3. Threshold cryptography

Threshold Cryptography is a technique that encrypts the information and splits it into parts to store in different fault tolerant systems. Asymmetric Cryptosystem is used here. The information is encrypted using public key and the private key to decrypt it is distributed among the shareholders. A prefixed threshold number of shareholders need to collaborate and contribute their shares to get the decryption key to decrypt the information. If n be the total number of shareholders or participants, and t be the pre-fixed threshold number, then at least t number of participants should contribute their shares to decrypt the message correctly.

3. Related works

Many satisfactory proposals have been put forward by researchers across the globe in the field of big data security. Li et al. [5] have proposed Distributed Authentication and Authorization Scheme (DAAS) to solve the problem of Authentication, Authorization and Auditing (AAA) in Bigdata. It also safeguards Bigdata veracity, secure key exchange, and confirmation of user identity. The scheme deploys Identity-based Signature for user authentication and Ciphertext Policy Attribute-Based Encryption (CP-ABE) for authorization. The problem here CP-ABE is less efficient for its difficulty to manage users and specify policies when the size of the universe attribute increases. Moreover, the Identity-based Signature scheme has the inherent problem of key escrow property. Wang et al. [6] have proposed a pre-authentication approach wherein the data at cloud is shared with others with full knowledge of users. The users who satisfy certain conditions are only given access to the secured data. However, the method involves complex computations. Wang et al. [6] have proposed a Software-defined architecture to improve the security and performance of the Industrial Internet of Things (IIoT). However, it's difficult to

standardize Software Defined Networking (SDN). Also, centralized control system leads to delay in data forwarding. To address the issue of latency, A a z a m, Z e a d a l l y and H a r r a s [8] have proposed to deploy Fog Computing between the IoT devices and the cloud. O m o n i w a et al. [9] also have recommended to introduce Fog computing. But, both the works have given the least priority to security.

A One-Time Pad (OTP) based method has been proposed by S o m u, G a n g a a and S r i r a m [10]. The method uses two servers – the registration server and the backend server. The authors recommend encrypting the user's password with a OTP and are further secured with modular operations before storing it in the registration server. This is again encrypted with the users' password and stored along with username in the backend server. During the authentication phase, the user provides only the username. The backend server sends a key encrypted with the user's password and if the user successfully decrypts it user is authenticated. This method incurs avoidable communication overhead and the method is later proved to be vulnerable to offline password guessing attack as investigated by S a r v a b h a t l a e t, C h a n d r a, and V o r u g u n t i [11]. The authors proposed to hash values of passwords and usernames before transmitting over the internet. Though it adds security, additional computational and communicational overhead and consequent latency should be expected. This is not an acceptable feature for a big data platform security system.

E s f a h a n i et al. [12] have proposed a lightweight authentication for machine-to-machine message exchange in the IoT Environment. The method relies on simple XOR operations and Hashing. Secure Elements (SE) in the sensor devices and Trusted Platform Modules (TPM) in the network devices like routers are used for authenticating. The reliability of the Trusted Platform Module in terms of bugs and other online attacks is a concern here. A privacy-preserving authentication protocol based on biometrics using Elliptic Curve Cryptography (ECC) is proposed by L i et al. [13]. Complex computations at sensor nodes and gateway nodes for authentication are not an ideal deal as sensors in most cases have less computational power. A blockchain-based approach is proposed by L i n et al. [14] using Attribute-Based Signature (ABS) and Certificateless Multi-Receivers Encryption (CL-MRE). The performance is not optimized and hence causes considerable degrading of system performance. K a r a t i, I s l a m and K a r u p p i a h [15] have proposed a secure scheme based on Certificateless Signatures using bilinear pairing. Although the authors claim the scheme to be computationally efficient, the execution cost stands high if some pairing computations are not discarded. Z h a n g et al. [16] later have proved the failure of K a r a t i, I s l a m and K a r u p p i a h [15] scheme against some signature falsification attacks and proposed a robust Certificateless Signature Scheme for data authentication. The scheme introduces partial private key generation. The problem with this scheme is the complex computations involved. A blockchain-based scheme with a deep reinforcement scheme is proposed by L i u, L i n and W e n [17] and a credit-based consensus mechanism is proposed by H u a n g et al. [18]. These methods have storage overhead and computational complexity. The schemes fail to

control the quality of collected data as well and hence are not practical in a big data scenario.

As per the above study, it is understood that each method has got one or other setback. Decentralizing the authentication task while guaranteeing the security of the user and/or data is a challenging chore. The network and processing delay also need to be taken care of.

4. Proposed system

The proposed system has its roots in Secure Remote Password (SRP) Protocol [22], threshold cryptography and blockchain technology. The existing Kerberos enabled Hadoop Cluster environment is modified as follows:

1. The user demonstrates to the KDC that he knows the password without plainly sharing it. A salted hash of it is shared (as exponent of generator of a pre-determined cyclic group G).
2. Blockchain network stores user details instead of local storage.
3. Single Ticket Granting Server (TGS) at KDC is substituted with many Ticket Granting Servers (TGSs) as in [23], wherein a pre-determined threshold number of TGSs should work together to get the decryption key to decrypt the Ticket Granting Ticket (TGT).

The client's details is kept in the blockchain as quadruplets $\{\langle \text{username} \rangle, \langle \text{verifier} \rangle, \langle \text{salt} \rangle, \langle \text{PSK} \rangle\}$ [where salt is a random number, x is salted hash of the password and verifier v is calculated by exponentiation of g the generator, of the pre-determined Group with x]. The Client submits a request for a authentication at the Key Distribution Center (KDC). The Authentication Server (AS) posts the user information to the Blockchain. The miners in the get the user information and send corresponding salt and verifier to the AS. The user obtains its $salt$ stored at the KDC during registration (at blockchain) under his username, along with the public key of the AS and an unsigned integer u. Both user and the AS computes a common secret S using their own formula. The session key K is computed as hash of the value of S.

To confirm the correctness of the generated session key, the AS in the proposed system, a a random key keyRand is sent to the user. The user encrypts the reply with the computed session key K. The response is decided as per the pre-agreement. The session key K is considered to be vaild if and only if:

- the user's reply can be successfully decrypted by the Authentication Server (AS), as only the AS and the user know the key K
- the user responded appropriately as per the pre-agreement; it indicates that the client is authenticated as only the client has the Pre-shared key (obtained from AS during registration) to figure out the e-OTP (enhanced – One Time Password) as per the pre-agreement

As illustrated in Fig. 1, the scheme consists of three entities:

a. The user.
b. The KDC with blockchain storage.
c. The secured Hadoop Cluster which user wants to access.

Table 1. Summary of related existing authentication mechanisms in the Big Data Environment

Scheme from	Problem Addressed	Methodology	Comments
[5]	Single Point of Failure	Identity-based cryptography – signature verification Ciphertext-Policy Attribute-based encryption (CP-ABE) – for authorization	- CP-ABE scheme – difficulty in managing users and specifying policies, overhead increases with increase in the size of the universe attribute set - IBS is key escrow property
[8]	Need for middleware in IIoT 4.0 environment	Introduce fog as middleware	-Least priority to security
[11]	Password Guessing attack	- Hash values of password and usernames before transmitting over the internet - OTP	Communicational and computational overhead
[12]	Password guessing attack	- Only hash and XOR operations - Secure Elements (SE)in the sensor devices and Trusted Platform Modules (TPM) in the network devices like routers are used for authenticating - The sensors communicate using the alias identity to prevent eavesdropping	- Infrastructural changes need to be made in the sensor devices for incorporating SEs - Reliability of TPMs to be considered
[13]	User anonymity	- Elliptic Curve Cryptography - Biometrics	Complex computations at sensor nodes and gateway nodes
[14]	Mutual authentication with fine-grained access control	- Blockchain - Attribute signature - Multi-receivers encryption - Message authentication code	Performance not optimized – degrades system performance
[15]	Data authentication and untrustworthiness of third parties	Certificateless signature scheme using bilinear pairing	- High execution cost - Vulnerable to signature falsification attacks - Not robust as secure channel needed between the third party and DOs
[16]	Data authentication in IIoT	Partial private key generation	- Complex computations
[17]	Secure sharing and exchanging data	- Deep reinforcement learning - Blockchain	- Storage overhead - Computational complexity
[18]	Single Point of Failure and other malicious attacks	A credit-based Proof-of-Work (PoW) consensus mechanism	- Sensor data quality control - Storage limitations
[19]	Bigdata storage privacy	Only users who satisfy certain attributes are given access to data	- Computational complexity
[20]	Security challenges in Industrial IoT and information-based interaction for the industrial environments in Industry 4.0	Software-defined IIoT architecture to regulate network resource provisioning and speed up information exchange mechanisms by an effortlessly customizable networking protocol	- Difficult to standardize Software Defined Networking (SDN) - Centralized control system leads to delay in data forwarding
[21]	Password Guessing attack	-Communication of password over the network is avoided by using OTP based authentication - User's password is encrypted using an OTP and stored in the registration server - The backend server further encrypts it with user password and store for future authentication purposes	- Avoidable communication overhead - Offline Password Guessing Attack

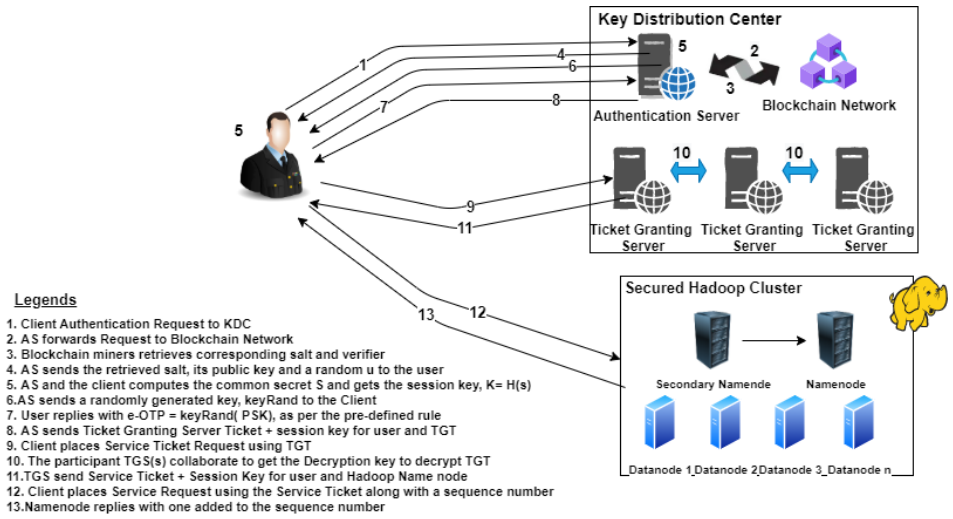

Fig. 1. Proposed system architecture

The AS grants the Ticket Granting Server Ticket which comprises the session key for the user to communicate with the Ticket Granting Server (TGS). The user submits the Service Ticket request by providing the Ticket Granting Ticket (TGT). The shareholders at the TGS pool their resources to get the decryption key to decrypt the Ticket Granting Server Ticket (TGT). TGS then sends the Service Ticket and a Session Key for the user to encrypt its communication with Hadoop Name node and vice-versa. The user places then access to secured Service by providing the Service Ticket along with a sequence number. The Namenode in the Hadoop Cluster adds one to the sequence number and replies to the user. This is for server's identity verification.

The proposed user authentication framework comprises the following steps:

a. User registration,
b. User Authentication.

4.1. User registration phase

- To register the user sends her identity $\mathrm{ID_u}$, a random salt s, and a salted hash of the password x to the KDC where

(9) $$x = H(s, \mathrm{Pw}),$$

$$\textbf{User} \longrightarrow \textbf{KDC}: \mathrm{ID_u} \parallel s \parallel v = g^x \bmod N.$$

- The KDC checks if these user ID details already exist in the blockchain storage.
- If no, the KDC sends back a PSK to the user for safe storage

$$\textbf{KDC} \longrightarrow \textbf{User}: \mathrm{PSK}.$$

- KDC posts this user info $(\mathrm{ID}, s, v = g^x \bmod N, \mathrm{PSK})$ to the blockchain for future authentication purposes by calling the smart contract of the blockchain,

$$\textbf{KDC} \longrightarrow \textbf{Blockchian}: \mathrm{ID_u} \parallel s \parallel v \parallel \mathrm{PSK}.$$

Table 2 defines various variables used in the proposed algorithm.

Table 2. Symbols Used

Symbol	Definition	Symbol	Definition
G	An additive group with multiplicative operation	T_x	Ticket to access entity *x*
N, g	Group Parameters (Prime & Generator of group G)	SHA3()	Secure Hash Function
ID_x	Identity of entity *x*	s	Random salt
KDC	Key Distribution Center	x	Salted hash of password
AS	Authentication Server	K_{xy}	Session Key for *x* & *y*
Pw	Password	e_OTP	Enhanced OTP
PSK	Pre-Shared Key	KEY_RAND	Randomly Generated Key
K_x^{pub}	Public key of *x*	K_x^{pvt}	Private key of *x*
$E()$	Encryption function	α, β	Public parameters for ElGamal Encryption

4.2. Authentication Step

- The user places an authentication request to the KDC with his $\mathrm{ID_u}$ and the identity of the Ticket Granting Server (TGS) $\mathrm{ID_{TGS}}$, it wants to get access into, together with his public key

(10) $$K_{\mathrm{u}}^{\mathrm{pub}} = g^{K_{\mathrm{u}}^{\mathrm{pvt}}},$$

where $K_{\mathrm{u}}^{\mathrm{pvt}}$ is the secret key of the user.

$$\mathbf{User} \longrightarrow \mathbf{KDC}: \mathrm{ID_u} \parallel \mathrm{ID_{TGS}} \parallel K_{\mathrm{u}}^{\mathrm{pub}}.$$

- The Authentication Server (AS) publishes this to the Blockchain Network so that the blockchain miners retrieve the user's verifier v and salt s from the BC,

$$\mathbf{AS} \longrightarrow \mathbf{Blockchain}: \mathrm{ID_u} \parallel \mathrm{ID_{TGS}} \parallel K_{\mathrm{u}}^{\mathrm{pub}},$$
$$\mathbf{Blockchain} \longrightarrow \mathbf{AS}: v \parallel s.$$

- If user details already exist, the Authentication Server chooses a random $K_{\mathrm{AS}}^{\mathrm{pvt}}$ as the secret key and compute analogous public key masked with the verifier v of the user as per following equation:

(11) $$K_{\mathrm{AS}}^{\mathrm{pub}} = v + g^{K_{\mathrm{AS}}^{\mathrm{pvt}}}.$$

- A 32-bit value u is calculated as follows:

(12) $$u = H\left(K_{\mathrm{u}}^{\mathrm{pub}}, K_{\mathrm{AS}}^{\mathrm{pub}}\right).$$

- Then, the salt s, public − key $K_{\mathrm{AS}}^{\mathrm{pub}}$ and u are shared with the user.

$$\mathbf{AS} \longrightarrow \mathbf{User}: s \parallel K_{\mathrm{AS}}^{\mathrm{pub}} \parallel u.$$

- The shared secret is calculated at the user side as follows:

(13) $$S = (K_{\mathrm{AS}}^{\mathrm{pub}} - g^x)^{K_{\mathrm{u}}^{\mathrm{pvt}} + ux}.$$

- At the KDC side, the AS computes the shared secret as

(14) $$S = (K_{\mathrm{u}}^{\mathrm{pub}} \times v^{\mathrm{u}})^{K_{\mathrm{AS}}^{\mathrm{pvt}}}.$$

- Then, the shared secret S are hashed at both sides and get the session key, K_{ua} for use in further communication between the user and the Authentication Server as

(15) $$K_{\mathrm{ua}} = H(S).$$

- The AS then sends a random key to the user,

$$\textbf{AS} \longrightarrow \textbf{User}: \text{keyRand}.$$

- The user needs to submit the correct response as per the pre-agreement and revert to the AS. That is, the user should submit the numerals in the Pre-Shared Key, PSK, that are at places indicated by the numerals in the keyRand. This is considered as an enhanced type of OTP (e_OTP). This e_OTP is encrypted using the session key K_{ua},

$$\textbf{User} \longrightarrow \textbf{AS}: E_{K_{\mathrm{ua}}}(\text{e\_OTP}).$$

- If the AS is successful decrypting the above message and confirms the correctness of e_OTP then a Ticket Granting Server Ticket (TGT) is allotted to the user.
- The Ticket Granting Ticket (TGT) is then encrypted with Ticket Granting Server's (TGS's) the public key as

(16) $$T_{\mathrm{t}} = E_{k_{\mathrm{TGS}}^{\mathrm{pub}}}(K_{\mathrm{tu}} \parallel \mathrm{ID}_{\mathrm{u}}).$$

- Elliptic Curve ElGamal Encryption is deployed here. That is,

(17) $$\text{Encrypt}\,(T_{\mathrm{t}}) = \langle c_1, c_2 \rangle = \langle \alpha^k, T_{\mathrm{t}}.\beta^k \rangle,$$

where $k \in \mathbb{Z}_p$ is a arbitrarily selected integer by the TGS.

$$\textbf{AS} \longrightarrow \textbf{User}: \{E_{K_{\mathrm{u}}^{\mathrm{pub}}}(K_{\mathrm{tu}}) \| \ T_{\mathrm{t}}\}.$$

- The user places a request to the Ticket Granting Server (TGS) to access the secured Hadoop Server Service by presenting the Ticket Granting Server Ticket (TGT),

$$\textbf{User} \rightarrow \textbf{TGS}: \mathrm{ID}_{\mathrm{u}} \parallel \ T_{\mathrm{t}} \parallel \mathrm{ID}_{\mathrm{h}}.$$

- TGS is arranged as several TGSs so that a pre-determined threshold number of TGSs should collaborate to compute the secret key's shares to decrypt the TGT:
  - The TGS has divided it into *n* shareholders to allow multi-party authentication.
  - A predetermined threshold *k* number of shares of the secret key contributed by participant TGS is required here to decrypt the Ticket Granting Server Ticket (TGT). This ensures that the TGS is continuously accessible. The decryption key share of *i*-th shareholder is calculated as:

(18) $$d_i = (c_1)^{K_i^{\mathrm{pvt}}}.$$

  - The decryption key is then calculated using the shares as follows:

(19) $$d \equiv \prod_{i \in I} {d_i}^{\Lambda_i},$$

where Λ_i is calculated using LaGrange's Construction Method and I is the set of contributors. Then, the Ticket Granting Ticket (TGT) is decrypted as

(20) $$T_{\mathrm{t}} = c_2 d^{-1}.$$

  - The client's authenticity is verified and the TGS issues the Hadoop Service Ticket (T_{h}) to the user.

$$\textbf{TGS} \longrightarrow \textbf{User}: \{T_{\mathrm{h}}, E_{k_{\mathrm{u}}^{\mathrm{pub}}}(K_{\mathrm{uh}})\},$$

where

(21) $$T_{\mathrm{h}} = E_{k_{\mathrm{h}}^{\mathrm{pub}}}(K_{\mathrm{uh}}, \mathrm{ID}_{\mathrm{u}}).$$

- With Service Ticket, user accesses the Hadoop Server as follows:

$$\textbf{User} \longrightarrow \textbf{Hadoop}: \{\mathrm{ID_u} \parallel T_\mathrm{h} \parallel E_{k_{\mathrm{uh}}}(\mathrm{seq\#})\}.$$

- Then, the Hadoop server answers, and hence the user can confirm the authenticity of the Hadoop server as follows:

$$\textbf{Hadoop} \longrightarrow \textbf{User:} \quad \{E_{k_{\mathrm{uh}}}(\mathrm{seq\#}+1)\}.$$

The kerberized Hadoop Server Identity should also be confirmed. As a last step in the proposed method, the user sends a sequence number encrypted with the session key shared between the client and secured Hadoop Server. So, only the secured Hadoop server can decrypt it and it returns a one- added value of the sequence number encrypted with their session key. This endorses that the user got a response from the same Hadoop server from which it requested the service and is thus mutually authenticated.

4.3. Blockchain-assisted consensus mechanism

The proposed scheme deploys the Practical Byzantine Fault Tolerance (PBFT) Mechanism to reach at an agreement on the data being mined from the blockchain storage for registration/authentication purposes. When the user places a registration request, the Authentication Server forwards the details to the blockchain as follows:

$$\textbf{KDC} \longrightarrow \textbf{Blockchian}: \mathrm{ID_u} \parallel s \parallel v \parallel \mathrm{PSK}.$$

The consensus process occurs as per the following steps:

Step 1. Generate. The leader node amongst the endorsement nodes receives the above transaction when its turn comes, and a candidate block is created to add to the blockchain network.

Step 2. Pre-prepare. The leader node then broadcasts the candidate node just created to all other endorsement nodes in the network.

Step 3. Prepare. The endorsement nodes check whether this block data is already existing one and hashes the block data otherwise. This is then broadcasted to other endorsement nodes in the network.

Step 4. Commit. According to PBFT, every node must receive prepare messages from more than 2/3 of the total number of nodes to reach on a decision (consensus). Upon reaching the consensus, every node broadcasts a commit message to every other nodes.

Step 5. Import. This new block is then added to the chain if consensus is reached and the Authentication server is notified.

5. Implementation and result analysis

Riverbed Modeler (AE) Simulator [24] is opted here to simulate the Kerberos enabled Hadoop Environment. As depicted in Fig. 2, it consists of the User Workstations, the Key Distribution Center (KDC) with an Authentication Server (ASr), and numerous Ticket Granting Servers (TGSs) and secured Hadoop Cluster with Namenode and Datanodes. The ppp_wkstn_adv node object is deployed as a user workstation and is accepted as the originator of all the communications. ppp_server_adv node object is deployed as the Authentication Server, Ticket Granting Server and as Namenode. The Internet node discards 0.0% of inward traffic and augments 100 ms delay to the

network packets. The authentication request size is presumed to be 2 KB and the user devotes 4 s to prepare this message. The Blockchain network required 0.5 s to retrieve the user's salt and verifier. Next phases, namely SRP Verification and PSK validation, together needs 4.5 s. The tickets in this model are presumed to be of size 1 KB and the ticket encryption takes 5 s. The size of encrypted exchanges between the user and the Hadoop Server is homogeneously distributed between 1 KB and 10 KB.

5.1. Security analysis

The proposed approach has dealt with most of the security challenges faced by Kerberized Hadoop Clusters. The following are the details.

5.1.1. Password guessing attacks

Assumption 1. An intruder guesstimates the password of a genuine user, he used to log into the secured system during previous communications.

P r o o f: A zero-knowledge-proof security is guaranteed in the proposed scheme as password or any information about it isn't openly shared with the KDC during registration or authentication processes. Adversary $\mathcal{A}$ won't thus get any chance to guesstimate the password and enter them into the system. Again, the session key is computed from a common shared secret at the user (13) and server (14) sides separately.

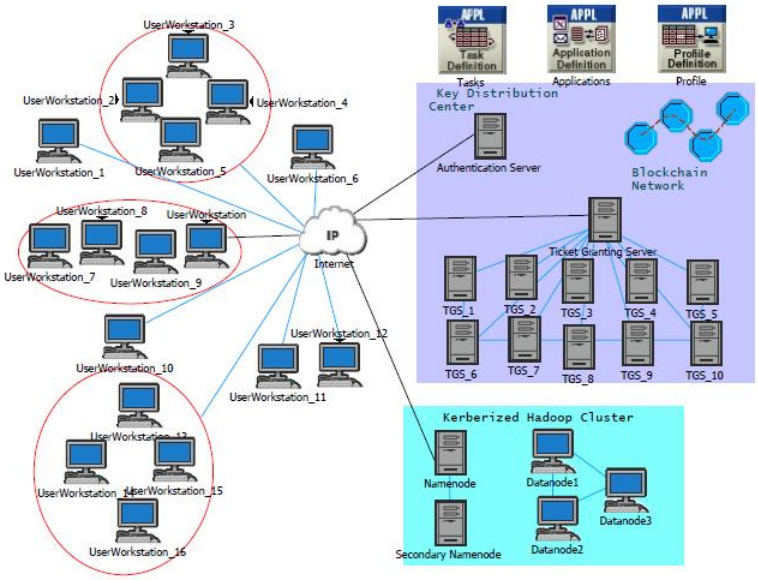

Fig. 2. Simulation environment

The passwords are stored in a manner that is not directly usable to an attacker. Even if the password database is hacked, the adversary still needed an expensive dictionary search to get the correct password. The computations which involves exponentiation operations to validate the guess are further time-consuming and

difficult to solve. The SRP protocol mentions the client terminate the communication when $K_{\mathrm{AS}}^{\mathrm{pub}} = 0$ in order to protect the system from offline guessing threats.

The values salt s, $K_{\mathrm{AS}}^{\mathrm{pub}}$, $K_{\mathrm{u}}^{\mathrm{pub}}$, u and keyRand are public parameters, computed according to (12), (13), (14), and (15). Conversely, x is private parameter computed as

(22) $$x = \mathrm{SHA3}(s, \mathrm{Pw}).$$

Suppose adversary $\mathcal{A}$ gets a dictionary from (12), (13), (14), and (15).

Adversary $\mathcal{A}$ can't even guess x, because the key K_{ua} is verified by e_OTP scheme where the user's reply to KDC is encrypted with K_{ua}. Only the AS and the user know it and can decode the e_OTP, which is a innovative feature of the proposed system.

5.1.2. Compromise on Key Distribution Center

Assumption 2. The adversary $\mathcal{A}$ hacks the KDC to get access to user credentials.

Proof: The deployment of a distributed tamper-free blockchain storage as storage at the Key Distribution Centre and the threshold cryptography which demands shares from prefixed threshold number of TGS for the authentication process to proceed ahead makes sure that there is no effect of any single node shutdown or compromise. The system is protected from the DoS attacks and henceforth the Single Point of Vulnerability. The Key Distribution Center is freed from storage and management of the credentials and hence the security is enhanced to a great extent. There is no probability that the adversaries target on the Key Distribution Centre as nothing is stored there.

For example, consider the following scenario, where user submits the request for service ticket by providing the Ticket Granting Ticket he/she has. Upon receipt of this request, the TGS asks the participant TGSs to contribute their shares to decrypt the TGT. That is,

$$\textbf{User} \rightarrow \textbf{TGS}: \mathrm{ID_u} \parallel T_{\mathrm{t}} \parallel \mathrm{ID_h},$$

$$d_i = (c_1)^{K_i^{\mathrm{pvt}}},$$

$$d \equiv \prod_{i \in I} d_i^{\Lambda_i}.$$

Here, d_i is the decryption key share hold by TGS_i and a threshold number of these share are sufficient to compute *d.* If any of the participants contribute wrong shares, that won't affect the authentication process.

5.1.3. Replay attacks

Assumption 3. An adversary $\mathcal{A}$ gets Authentication Dialogue

$$\text{User} \rightarrow \text{TGS}: \mathrm{ID_u} \parallel T_{\mathrm{t}} \parallel \mathrm{ID_h},$$

and retransmit it to get access to the secured Hadoop Server.

Proof: An adversary $\mathcal{A}$ fails replay the messages at any stage of the authentication process as the secrets are not ever communicated over the network. It is tremendously tough to figure the discrete log and if at all be successful in that, the credentials are kept in a manner that is not valuable to the adversary as it is.

Here, the Ticket Granting Server replies with the Authentication Dialogue: TGS$\longrightarrow$ User: $\{T_{\mathrm{h}}, E_{k_{\mathrm{u}}^{\mathrm{pub}}}(K_{\mathrm{uh}})\}$; where $T_{\mathrm{h}} = E_{k_{\mathrm{h}}^{\mathrm{pub}}}(K_{\mathrm{uh}}, \mathrm{ID}_{\mathrm{u}})$. The key K_{uh} is encrypted with the user's public key and thus only the user can decode it. Hence, adversary cannot use it for reply attacks.

A comparative security analysis of the proposed scheme with that of related schemes in [1] and [25] is presented in Table 3.

Table 3. Comparative analysis of security features

Security feature	Rahul and Girishkumar [1] scheme	Algaradi and Rama [25] scheme	Proposed scheme
No single point of failure	✓	☒	✓
Resist the insider attack	☒	✓	✓
Resist dictionary attacks	☒	✓	✓
Resist replay attack	☒	☒	✓

5.2. Key sensitivity analysis

Any deliberate or accidental modification of a single bit of the private key creates a severe consequence on the whole authentication mechanism. The computation of secure hash function separately at the server and client side senses any change in the value of shared secret as described in Table 4. The e_OTP scheme make sure that the key generated at both the sides are same. If the keys are different on both sides, the AS fails to decrypt the reply message from the user that is encrypted with the derived session key.

Table 4. The value of Secret *S* computed same at both sides

User	KDC
$S = (K_{\mathrm{AS}}^{\mathrm{pub}} - g^{x})^{K_{\mathrm{u}}^{\mathrm{pvt}}+ux}$	$S = (K_{\mathrm{u}}^{\mathrm{pub}} \times v^{\mathrm{u}})^{K_{\mathrm{AS}}^{\mathrm{pvt}}}$
$S = (v + g^{K_{\mathrm{AS}}^{\mathrm{pvt}}} - g^{x})^{K_{\mathrm{u}}^{\mathrm{pvt}}+ux}$	$S = (g^{K_{\mathrm{u}}^{\mathrm{pvt}}} \times g^{x^{\mathrm{u}}})^{K_{\mathrm{AS}}^{\mathrm{pvt}}}$
$S = (g^{x} + K_{\mathrm{AS}}^{\mathrm{pvt}} - g^{x})^{K_{\mathrm{u}}^{\mathrm{pvt}}+ux}$	$S = (g^{K_{\mathrm{u}}^{\mathrm{pvt}}+ux})^{K_{\mathrm{AS}}^{\mathrm{pvt}}}$
$\boldsymbol{S = (g^{K_{\mathrm{AS}}^{\mathrm{pvt}}})^{K_{\mathrm{u}}^{\mathrm{pvt}}+ux}}$	$\boldsymbol{S = (g^{K_{\mathrm{AS}}^{\mathrm{pvt}}})^{K_{\mathrm{u}}^{\mathrm{pvt}}+ux}}$

5.3. Key space analysis

A perfect key for any security mechanism should not be too long or too short. The Greater key sizes result in slowing down the encryption. This is not a suitable option for real-time big data systems. If the key sizes are small, it is susceptible to easy cracking also. As per reports in [26], the keys space should be greater than or equal to 2^{100} to attain high level security. The proposed scheme a key of size 40 bytes which is equal to 320 bits. Thus with 320 bits the key space is 2^{320}. This key space is sufficiently large for preventing any brute force attacks or password guessing attacks.

5.4. Performance analysis

The performance of the proposed scheme is compared with the traditional Kerberos enabled Hadoop systems that rely on RSA (Rivest-Shamir-Adleman) cryptosystem.

As shown in Fig. 3, the duration for completion of all tasks in the Initial Login Process to get the TGT in RSA-based traditional system is h compared to the

proposed system. The difficult computations in RSA and larger key sizes can be a cause for this.

The computational cost of the proposed system is compared with some of the related systems [1] and [25]. T_h indicates computational cost for hash operations T_e for exponentiation functions and T_s for cryptographic functions. A comparative analysis of these costs are shown in Table 5. The computational cost of the proposed scheme is $T_h + T_e$ during the registration step and $6T_s + 3T_h + 6T_e$ during the user authentication step. The scheme in [1] costs $4T_s + T_h$ during registration step and $21T_s$ during user authentication step. The scheme in [25] costs $2T_s + 3T_h + T_e$ for registration step and $8T_s + 2T_h$ for authentication step. To account the above costs, the values of T_h, T_e and T_s are considered as approximately equal to 0.0023 ms, 0.0046 ms and 2.226 ms, respectively. These values are taken as reported by authors in [27]. Then, Rahul and Girish Kumar [1] scheme incurs

$$4T_s + T_h + 21T_s \approx 25\times(2.226) + 0.0023 \approx 55.65 \text{ ms.}$$

For Algaradi and Rama [25] scheme, it is

$$10T_s + 5T_h + T_e \approx 10\times(2.226) + 5\times(0.0023) + 0.0046 \approx 22.28 \text{ ms.}$$

At the same time, in the proposed system this is

$$6T_s + 4T_h + 7T_e \approx 6\times(2.226) + 4\times(0.0023) + 7\times(0.0046) \approx 13.40 \text{ ms.}$$

Table 5. Comparative analysis of computational cost

Phase	Rahul and Girish Kumar [1] scheme	Algaradi and Rama [25] scheme	Proposed scheme
Registration	$4T_s + T_h$	$2T_s + 3T_h + T_e$	$T_h + T_e$
Authentication	$21T_s$	$8T_s + 2T_h$	$6T_s + 3T_h + 6T_e$
Estimated cost (ms)	$\approx$ 55.65	$\approx$ 22.28	$\approx$ 13.40

The graphical representation of the same is illustrated in Fig. 4. Thus, the proposed scheme is more efficient than the existing schemes in terms of computational cost as well.

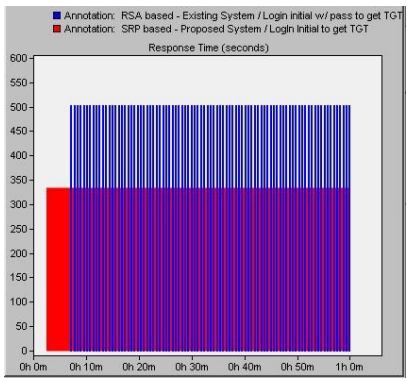

Fig. 3. Initial login task – response time

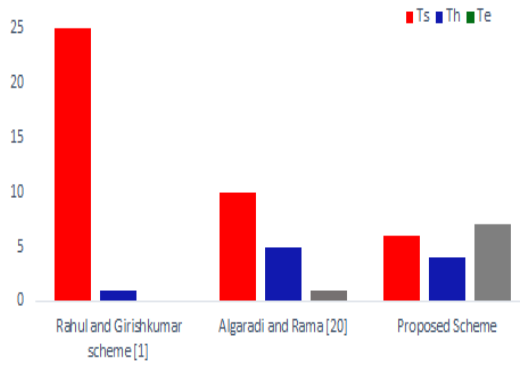

Fig. 4. Comparative analysis of computational cost

5.5. Time complexity

As seen in Table 5, the operations in the proposed system involve the following operations.

The registration phase involves a hashing operation and an exponentiation operation. Similarly, the authentication phase involves six encryption/decryption operations, three hashing operations, and six exponentiation operations. It is observed that hashing is involved in both phases. Hence, the time complexity of the SHA-256 hashing algorithm is taken into account which is $O(M \times n)$ where M is the message-length to be hashed. The Elliptic Curve ElGamal Cryptosystem has a time complexity of $O(n)$ whih includes the time for exponentiation and modular inverse operation. Hence, the total time complexity is $O(M \times n) + \ O(n) \ + O(n) + \ O(n)$ which approximates to $O(n)$.

The traditional system under study that is RSA-based. It has an overall time complexity of $O(n^2)$ for key generation and $O(n^3)$ for encryption and decryption. Thus, the proposed system is efficient in terms of time complexity.

6. Conclusion

With the increased popularity of Big Data Analytics and consequent industrial benefits, security issues have also popped up. Many researchers and experts have already proposed various solutions to address these issues. In this paper, a novel three-tier authentication framework is proposed to secure Kerberos-enabled Hadoop clusters. The proposed scheme depends on the Secure Remote Password Protocol, enhanced One Time password and threshold cryptosystem. The Blockchain network replaces the local database at the KDC. The proposed authentication system which mainly aims to eliminates single point of failure and password guessing attacks secures the Hadoop clusters from other main attacks like replay and insider attacks. The blockchain which is a tamper-proof storage also avoids the issues that can happen due to the compromise on local database at the KDC. The Riverbed Modeller (AE) Simulation evaluated the performance of the proposed system and found it be highly efficient. In the future, the time synchronization problem in Kerberized Hadoop Clusters will be addressed and real-time implementation results will be presented to prove the validity of the above claims.

References

1. Rahul, P. K., T. Gireesh Kumar. A Novel Authentication Framework for Hadoop. – Advances in Intelligent Systems and Computing, Vol. **324**, 2015, pp. 333-340.
2. Lingappa, R. What Is Secure Remote Password (SRP) Protocol and How to Use It? The Startup, Medium. 2019. Accessed 15 March 2021. **https://medium.com/swlh/what-is-secure-remote-password-srp-protocol-and-how-to-use-it-70e415b94a76**
3. Hena, M., N. Jeyanthi. Authentication Framework for Kerberos Enabled Hadoop Clusters. – Int. J. Eng. Adv. Technol., Vol. **9**, 2019, No 1, pp. 510-519.
4. Castro, M., B. Liskov. Practical Byzantine Fault Tolerance. – In: Proc. of 3rd Symposium on Operating Systems Design and Implementation, New Orleans, USA, February 1999, pp. 1-14.

5. L i, R., H. A s a e d a, J. L i, X. F u. A Distributed Authentication and Authorization Scheme for In-Network Big Data Sharing. – Digit. Commun. Networks, Vol. **3**, November 2017, No 4, pp. 226-235.
6. W a n g, K., J. Y u, X. L i u, S. G u o. A Pre-Authentication Approach to Proxy Re-Encryption in Big Data Context. – IEEE Trans. Big Data, May 2017, p. 1.
7. A b d u l l a h, N., A. H a k a n s s o n, E. M o r a d i a n. Blockchain Based Approach to Enhance Big Data Authentication in Distributed Environment. – In: Proc. of 9th International Conference on Ubiquitous and Future Networks (ICUFN'17), 2017, pp. 887-892.
8. A a z a m, M., S. Z e a d a l l y, K. A. H a r r a s. Deploying Fog Computing in Industrial Internet of Things and Industry 4.0. – IEEE Trans. Ind. Informatics, Vol. **14**, October 2018, No 10, pp. 4674-4682.
9. O m o n i w a, B., R. H u s s a i n, M. A. J a v e d, S. H. B o u k, S. A. M a l i k. Fog/Edge Computing-Based IoT (FECIoT): Architecture, Applications, and Research Issues. – IEEE Internet Things J., Vol. **6**, Jun 2019, No 3, pp. 4118-4149.
10. S o m u, N., A. G a n g a a, V. S. S h a n k a r S r i r a m. Authentication Service in Hadoop Using One Time Pad. – Indian J. Sci. Technol., Vol. **7**, 2014, No April, pp. 56-62.
11. S a r v a b h a t l a, M., M. R. M. C h a n d r a, C. S. V o r u g u n t i. A Secure and Light Weight Authentication Service in Hadoop Using One Time Pad. – Procedia Computer Science, Vol. **50**, 2015, pp. 81-86.
12. E s f a h a n i, A., et al. A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment. – IEEE Internet Things J., Vol. **6**, February 2019, No 1, pp. 288-296.
13. L i, X., J. N i u, M. Z. A. B h u i y a n, F. W u, M. K a r u p p i a h, S. K u m a r i. A Robust ECC-Based Provable Secure Authentication Protocol with Privacy Preserving for Industrial Internet of Things. – IEEE Trans. Ind. Informatics, Vol. **14**, August 2018, No 8, pp. 3599-3609.
14. L i n, C., D. H e, X. H u a n g, K. K. R. C h o o, A. V. V a s i l a k o s. BSeIn: A Blockchain-Based Secure Mutual Authentication with Fine-Grained Access Control System for Industry 4.0. – J. Netw. Comput. Appl., Vol. **116**, 2018, No February, pp. 42-52.
15. K a r a t i, A., S. K. H. I s l a m, M. K a r u p p i a h. Provably Secure and Lightweight Certificateless Signature Scheme for IIoT Environments. – IEEE Trans. Ind. Informatics, Vol. **14**, August 2018, No 8, pp. 3701-3711.
16. Z h a n g, Y., R. H. D e n g, D. Z h e n g, J. L i, P. W u, J. C a o. Efficient and Robust Certificateless Signature for Data Crowdsensing in Cloud-Assisted Industrial IoT. – IEEE Trans. Ind. Informatics, Vol. **15**, January 2019, No 9, pp. 5099-5108.
17. L i u, C. H., Q. L i n, S. W e n. Blockchain-Enabled Data Collection and Sharing for Industrial IoT with Deep Reinforcement Learning. – IEEE Trans. Ind. Informatics, Vol. **15**, Jun 2019, No 6, pp. 3516-3526.
18. H u a n g, J., L. K o n g, G. C h e n, M. Y. W u, X. L i u, P. Z e n g. Towards Secure Industrial IoT: Blockchain System with Credit-Based Consensus Mechanism. – IEEE Trans. Ind. Informatics, Vol. **15**, Jun 2019, No 6, pp. 3680-3689.
19. W a n g, K., J. Y u, X. L i u, S. G u o. A Pre-Authentication Approach to Proxy Re-Encryption in Big Data Context. – IEEE Trans. Big Data, May 2017, pp. 1-11.
20. W a n, J., et al. Software-Defined Industrial Internet of Things in the Context of Industry 4.0. – IEEE Sens. J., Vol. **16**, October 2016, No 20, pp. 7373-7380.
21. S o m u, N., A. G a n g a a, V. S. S h a n k a r S r i r a m. Authentication Service in Hadoop Using One Time Pad. – Indian J. Sci. Technol., Vol. **7**, May 2014, No Supplementary 4, pp. 56-62.
22. T a y l o r, D., T. W u, N. M a v r o g i a n n o p o u l o s. Using the Secure Remote Password (SRP) Protocol for TLS Authentication. 2007.
23. H e n a, M., N. J e y a n t h i. Blockchain Based Authentication Framework for Kerberos Enabled Hadoop Clusters. – In: 10th International Conference on Soft Computing for Problem Solving (SocProS'20), 18-20 December 2020.
24. S e t h i, A. S. The Practical OPNET User Guide for Computer Network Simulation. Chapman and Hall/CRC, 2012.
25. A l g a r a d i, T. S., B. R a m a. Static Knowledge-Based Authentication Mechanism for Hadoop Distributed Platform Using Kerberos. – Int. J. Adv. Sci. Eng. Inf. Technol., Vol. **9**, 2019, No 3, pp. 772-780.

26. S c h n e i e r, B. Applied Cryptography : Protocols, Algorithms and Source Code in C. 2nd Ed. John Wiley & Sons, Inc., 1996.
27. K i l i n c, H. H., T. Y a n i k. A Survey of SIP Authentication and Key Agreement Schemes. – IEEE Commun. Surv. Tutorials, Vol. **16**, 2014, No 2, pp. 1005-1023.
28. I v a n o v a-R o h l i n g, V. N., N. R o h l i n g. Evaluating Machine Learning Approaches for Discovering Optimal Sets of Projection Operators for Quantum State Tomography of Qubit Systems. – Cybernetics and Information Technologies, Vol. **20**, 2020, No 6 pp. 61-73.
29. P r a b a d e v i, B., N. J e y a n t h i. TSCBA-A Mitigation System for ARP Cache Poisoning Attacks. – Cybernetics and Information Technologies, Vol. **18**, 2018, No 4, pp. 75-93.
30. P e n c h e v a, E. N., I. I. A t a n a s o v, V. G. V l a d i s l a v o v. Mission Critical Messaging Using Multi-Access Edge Computing. – Cybernetics and Information Technologies, Vol. **19**, 2019, No 4, pp. 73- 89.
31. B r i n d h a, K., N. J e y a n t h i. Secured Document Sharing Using Visual Cryptography in Cloud Data Storage. – Cybernetics and Information Technologies, Vol. **15**, 2015, No 4, pp. 111-123.
32. S r i v a s t a v a, M., J. S i d d i q u i, M. A. A l i. A Review of Hashing Based Image Copy Detection Techniques. – Cybernetics and Information Technologies, Vol. **19**, 2019, No 2, pp. 1-27.
33. P r a b a d e v i, B., N. J e y a n t h i. Security Solution for ARP Cache Poisoning Attacks in Large Data Center Networks. – Cybernetics and Information Technologies, Vol. **17**, 2017, No 4, pp. 69-86.
34. U s h a, S., S. K u p p u s w a m i, M. K a r t h i k. A New Enhanced Authentication Mechanism Using Session Key Agreement Protocol. – Cybernetics and Information Technologies, Vol. **18**, 2018, No 4, pp. 61-74.

Received: 14.06.2021; Second Version: 27.10.2021; Accepted: 08.11.2021