ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 12 N. 1 (2023), e19163
eISSN: 2255-2863
DOI: https://doi.org/10.14201/adcaij.19163

Beaf:BD – A Blockchain Enabled Authentication Framework for Big Data

Manish Kumar Gupta and Rajendra Kumar Dwivedi

Department of Information Technology & Computer Application, Madan Mohan Malaviya University of Technology, Gorakhpur, U.P., India, 273010.
manish.testing09@gmail.com, rajendra.gkp@gmail.com

KEYWORDS

Hadoop; authentication; big data; Kerberos; blockchain

ABSTRACT

The widespread utilization of Internet-based applications in our daily routines has resulted in enormous amounts of data being generated every minute. This data is not only produced by humans but also by various machines such as sensors, satellites, CCTV, etc. For many organizations, Apache Hadoop is the solution for handling big data. Big data refers to the extensive set of dissimilar data that can be processed to derive meaningful insights. For its security needs, Hadoop relies on trusted third-party security providers such as Kerberos. Kerberos has several security vulnerabilities. The focus of this paper is to eliminate security issues, particularly dictionary attacks and single points of failure, by proposing a model based on blockchain technology and threshold cryptography.In comparison to other existing schemes, the proposed approach offers superior computational overhead and storage requirements while maintaining the system's security level.

1. Introduction

Data is generated in large amounts from countless sources, including social networks, e-commerce, mobile, urban management, vehicle networks, medical sectors. These large amounts of data are known as big data (BD) (Chandra et al. 2017). There are numerous tools & frameworks available for BD management. Apache Hadoop is one of them; it is based on Java and works on the concept of streamline access patterns (Dean et al. 2008). Hadoop distributed file system (HDFS) and map reduce are two core components of the Hadoop framework that offer distributed storage and processing, respectively. Hadoop is a cluster of commodity computers containing a single name node, multiple data nodes, multiple resource managers, and a single secondary name node. Ensuring the security of BD is of utmost importance to fully harness the benefits of BD analytics. The majority of the security concerns stem from unauthorized access, leading to the manipulation or retrieval of data. Many of the latest Hadoop technologies rely on external security systems, such as the Kerberos protocol, to integrate security measures.However, Kerberos has several security vulnerabilities, such as replay attacks, secure time services, password guessing attacks, spoofing login, inter-session chosen plaintext attacks, exposure of session keys, dictionary attacks, and single point of vulnerabilities (Lingappa et al. 2021).

Initially designed to cater to the requirements of project Athena, the Kerberos authentication system was developed by MIT. Over time, numerous other organizations have adopted this system for their purposes, and it is currently being considered as a potential standard. Kerberos was specifically created for that particular environment, and if the fundamental assumptions differ, the authentication system might require modification. Additionally, some issues arise from the protocol design itself, some of which have been addressed in Version 5 of Kerberos, but not all. Table 1 contains the definitions of the acronyms and abbreviations that are used in the paper.

The KDC in Kerberos-enabled Hadoop, is known as a single point of failure (SPF). KDC encloses two servers: AS and TGS and one local database. This paper proposes a blockchain (BC)-enabled authentication framework for big data (baef:BD) that draws inspiration from SRP, OTP, and TC. The main objective of this research is to eliminate password guessing attacks, replay attacks, dictionary attacks, and the issue of SPF. For this, a BC network is used instead of the local database. The public key along with the salted hash of the password is shared between the user and server by using SRP protocol (Lingappa et al. 2021). An OTP mechanism is used to verify the correctness of the SKon both sides (user and server). To ensure the availability of the authentication system, the deployment of multiple TGS is made possible by implementing TC. The suggested system attains level of security that is appropriate for real-time BD systems while minimizing communication and computational overhead. The consensus mechanism employed in the BC network is practical byzantine fault tolerance (PBFT) (Castro et al. 1999), which requires more than two-thirds of the total number of nodes to be honest in contributing to the mined result. The nodes within the network are organized in a way that allows them to communicate with one another, and there is no permanent leader. The leadership role of the nodes rotates periodically.

The rest of the paper is organized as follows. Section 2 provides the background of SRP, OTP, and TC. Section 3 discusses related works. The proposed model is described in Section 4. Section 5 describes the performance evaluation. Finally, Section 6 concludes the paper.

Table 1. Acronyms/abbreviations and their definitions

Acronym/Abbr.	**Definition**	**Acronym/Abbr.**	**Definition**
SRP	Secure remote password protocol	S	Salt
BC	Blockchain	PSK	Pre shared key
TC	Threshold cryptography	KDC	Key distribution center
V	Verifier	AS	Authentication server
ASpuk/Spuk	AS public key	SK	Secret key
RK	Random key	h()	Hash function
TGS	Ticket granting server	usn	Username
pwd	Password	SSK	Session key
ST	Service ticket	SN	Sequence number
DK	Decryption key	TGS_{id}	Identity of ticket granting server

2. Background

2.1. Secure Remote Password Protocol (SRP)

SRP is a protocol for zero-knowledge proof, which enables clients to securely authenticate the server without the server needing to store any password equivalent information (such as a hashed version). SRP may involve complex mathematical concepts, but it can be broken down into three simple steps: registration, authentication, and verification.

Registration is the initial step of the process, the client provides specific information that is used for authentication in the future. Therefore, when signing up, the client generates a random salt (S) and a verifier (V). After computation of S and V, the client sends V, S and SRP group with username (usn) to the server, and the server stores V & S of usn in its local database. Figure 1 shows the registration process of SRP.

$$X = KDF(usn, pwd, S)$$

$$V = (SRP\ group, X)$$

Authentication as shown in Figure 2, is an interesting part of the SRP. To demonstrate that the user is aware of their password, the client and server exchange non-sensitive data to create a key independently. This key is then utilized for verification purposes. The client sends a request to the server for S, and SRP group for a particular usn. After receiving S and SRP group, the client computes their private key (Cprk) and public key (Cpuk). Cprk is kept with the client and Cpuk is sent to the server. The server also computes its private key (Sprk) and public key (Spuk) using the same SRP group. The server keeps Sprk in its temporary storage and sends Spuk to the client. After all, the client has (secret key, Cprk, Spuk) and the server has (V, Sprk, Cpuk).

Verification is the last part of SRP. In the verification phase, the client encrypts the message using session encryption key (SEK) and sends it to the server.The server decryptsthe message and verifies it. After successful verification, the server encrypts the message using its SEK and sendsit back to the client. The client again decryptsthe message and verifies it.

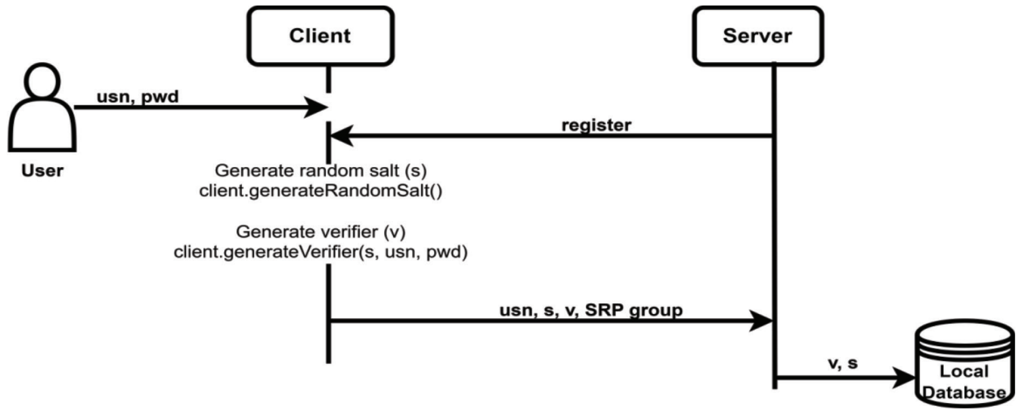

Figure 1. Registration process of SRP

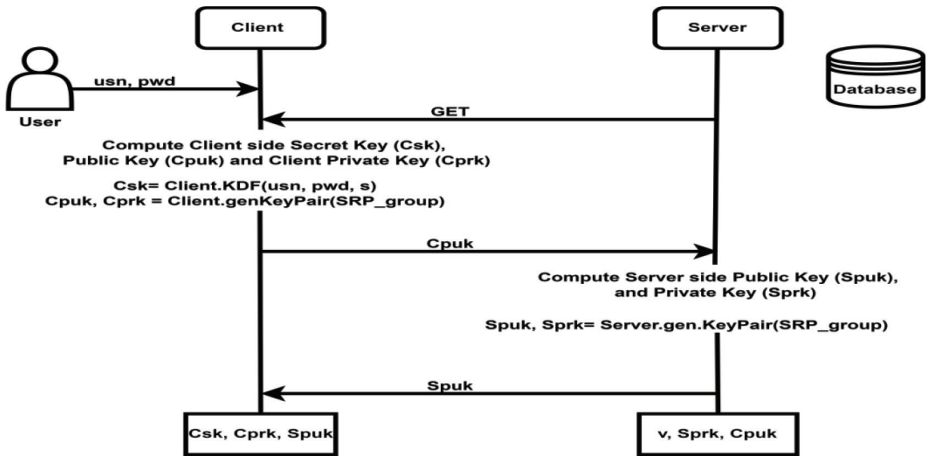
Client
Server
Database
User
usn, pwd
GET
Compute Client side Secret Key (Csk),
Public Key (Cpuk) and Client Private Key (Cprk)
Csk= Client.KDF(usn, pwd, s)
Cpuk, Cprk = Client.genKeyPair(SRP_group)
Cpuk
Compute Server side Public Key (Spuk),
and Private Key (Sprk)
Spuk, Sprk= Server.gen.KeyPair(SRP_group)
Spuk
Csk, Cprk, Spuk
v, Sprk, Cpuk

Figure 2. Authentication process of SRP

2.2. The One Time Password (OTP)

In the registration process, the user and the AS establish a PSK, which is a randomly generated number. During authentication, the AS sends anOTP code with a shuffle bitto the user. The user is required to reply with the PSK digits at the positions specified by the OTP with a shuffle bit.For example, suppose that the digit in PSK is “152254359” and OTP is sent by AS is “32521”. The rightmost digit of OTP code is a shuffle bit. So, the user replies with a digit at 4th (3rd +1), 3rd (2nd +1), 6th (5th +1), and 3rd (2nd +1) positions, that is 2242.

2.3. Threshold Cryptography

TC is a security technique that utilizes asymmetric cryptosystems to encrypt information and store it in multiple fault-tolerant systems. The data is encrypted using a public key, while the corresponding private key is distributed among the shareholders. To decrypt the information, a predefined threshold number of shareholders must collaborate and contribute their shares, which will allow for the reconstruction of the private key necessary for decryption. It enhances security by requiring a minimum threshold of participants to perform cryptographic operations. Individual participants cannot perform operations alone, preventing single points of failure. Applications include secure key management, digital signatures, and encryption.

3. Related Works

Rahul et al. (2015) used symmetric key cryptography, PKI, and Kerberos authentication. A central authentication server managed the symmetric keys and authenticated users. The proposed framework offered improved security and dynamic group creation for varying access levels. Li et al. (2017) used a hierarchical group-based approach to manage access control. It employed a combination of symmetric and asymmetric cryptography techniques for security. Wang et al. (2017) utilized a hybrid cryptosystem combining symmetric and asymmetric encryption. It enhanced the security and efficiency of re-encryption operations in BD scenarios. The proposed approach offered improved performance and lower computation costs compared to traditional re-encryption methods. Abdullah et al. (2017) utilized a distributed ledger to store and verify data transactions and user identities. Somu et al. (2014) utilized a random key generator to create one-time keys for encrypting and decrypting data. Sarvabhatla et al. (2015) used a random key generator to create one-time keys for secure communication between Hadoop nodes. It provided a low-overhead solution for user authentication in Hadoop clusters. Lin et al. (2018) proposed an authentication system that utilizes smart contracts to enforce access control policies and authenticate users. It offered improved security and privacy for industrial IoT systems by preventing unauthorized access and data tampering. Wang et al. (2017) utilized a hybrid cryptosystem combining symmetric and asymmetric encryption. In their study, Algaradi et al. (2019) proposed an approach that involves a centralized authentication server responsible for handling user credentials and issuing secure communication tickets. Tsu Yang et al. (2021) ensure secure communication among SIoV nodes and fog nodes by establishing an authenticated key agreement through mutual authentication and key generation. Hena et al. (2022) proposed a distributed authentication framework for securing Hadoop-based BD environments. The framework used Kerberos-based authentication and access control policies to manage user authentication and authorization. The proposed framework is evaluated using a set of experiments and shows improved security and performance compared to existing authentication mechanisms. Honor et al. (2021) proposed a scheme that addresses the problem of data trustworthiness and provenance in IoT systems by providing an immutable and tamper-proof record of data transactions.Marco et al. (2023) used a set of metrics and quality attributes to assess the level of trustworthiness of BD. Tall et al. (2023) proposed a framework that allows for fine-grained control over data access based on a set of attributes associated with the data. Jeong et al. (2015) proposed a model aiming to enhance the security of HDFS by improving the authentication process and preventing unauthorized access to the data. Algardi et al. (2022) proposed a framework that focuses on managing authentication keys, which are crucial for ensuring secure access to and communication within a big data environment. Zahednejad et al. (2023) proposed a scheme that addressed the challenges of authentication and key agreement in IoT systems, with a particular focus on revocability. The scheme aimed to provide efficient and secure authentication and key management for IoT devices in a big data environment. A comparative analysis in terms of method used, features and challanges associated with various authors, shown in Table 2.

Table 2. Features and challenges of existing work

Citation & Year	Method Used	Features	Challenges
Rahul et al. [2015]	HMAC-based authentication and verification mechanism.	Protects sensitive data and unauthorized access.	Does not provide complete security against man-in-the-middle attacks.
Li et al. [2017]	Distributed authentication and authorization scheme.	Ensures secure sharing of BD among multiple nodes.	Limited scalability due to high overheads in generating and verifying signatures.

(continued)

Table 2. Features and challenges of existing work (continued)

Citation & Year	Method Used	Features	Challenges
Wang et al. [2017]	Pre-authentication and proxy re-encryption scheme.	Provides secure and efficient data sharing in a BD context.	Limited applicability to specific use cases due to high computational overhead.
Abdullah et al. [2017]	BC-based authentication and access control mechanism.	Ensures transparency and tamper-proof BD management.	High computational overhead and potential security risks due to BC-based approach.
Somu et al. [2014]	One time pad-based authentication mechanism.	Provides secure and efficient authentication in Hadoop-based systems.	Limited applicability due to high overhead in key management.
Sarvabhatla et al. [2015]	Lightweight authentication service based on one time pad.	Provides secure and efficient authentication in Hadoop-based systems.	Limited applicability due to high overhead in key management.
Lin et al. [2018]	BC-based mutual authentication and fine-grained access control mechanism.	Ensures secure and efficient data sharing and management in Industry 4.0 contexts.	High computational overhead and potential security risks due to BC-based approach.
Wang et al. [2017]	Pre-authentication approach to proxy re-encryption.	BD context, fine-grained access control, scalable re-encryption, hybrid cloud environment.	Ensuring data confidentiality, scalability, re-encryption efficiency.
Algaradi et al. [2019]	Static knowledge-based authentication mechanism using Kerberos.	Hadoop Distributed Platform, single sign-on authentication, role-based access control, secure communication.	Security vulnerabilities, limitations of Kerberos, complexity in managing roles and permissions.
Tsu Yang et al. [2021]	Lightweight authenticated key agreement protocol using fog nodes.	Social Internet of Vehicles, secure communication, efficient key exchange, lightweight computation.	Ensuring privacy, scalability, reliability of Fog nodes, resistance to attacks.
Hena et al. [2022]	Distributed authentication framework for Hadoop-based BD environment.	Hadoop ecosystem, distributed authentication, single sign-on, multi-factor authentication.	Ensuring data privacy, scalability, secure authentication.
Honor et al. [2021]	IoT BD provenance scheme using BC on Hadoop ecosystem.	BC technology, Hadoop ecosystem, provenance, data integrity, traceability.	Ensuring data confidentiality, scalability, reliability of BC, efficient data processing.
Marco et al. [2023]	Assurance process for BD trustworthiness.	BD, data quality, data provenance, data security, trustworthiness.	Ensuring data quality, data provenance, data security, trustworthiness, scalability.

(continued)

Table 2. Features and challenges of existing work (continued)

Citation & Year	Method Used	Features	Challenges
Tall et al. [2023]	Framework for attribute-based access control in processing BD with multiple sensitivities.	BD, attribute-based access control, fine-grained access control, sensitive data protection.	Ensuring data privacy, scalability, efficient policy management, and resistance to attacks.
Jeong et al. [2015]	Token-based authentication security scheme using elliptic curve cryptography.	Secure authentication, token-based approach for better scalability, efficient communication between nodes, and support for distributed file system Hadoop.	Implementation and management of token-based approach, potential vulnerabilities in elliptic curve cryptography, and scalability issues in large-scale systems.
Algaradi, et al. [2022]	Authentication key management scheme.	Securing big data environments.	Scalability Efficiency
Zahednejad et al. [2023]	Lightweight, secure big data-based authentication and key-agreement scheme.	Authentication and key agreement for IoT.	Revocability

4. Proposed Model

SRP, TC, and BC are the main root of beaf:BD.The working process of Kerberos is modified as:

1. A salted hash value of the password(pwd) is shared to KDC by the user/client, instead of plain pwd.
2. BC is used to store user/client details instead of the local database.
3. Single TGS is replaced with multiple TGS.

BC store client details in the form of {usn,V,S, PSK}. When a client requests authentication to the KDC, AS sends client details to BC. Miners in BC get client details and send corresponding S and V to AS. Client gets its S, Spuk, and an integer u. Client and AS compute common SK individually. SK is obtained by applying the hash function of S. For the authenticity of SK, a random key (RK) is sent to the client by AS. Client encrypts SK. SK is considered valid when, AS decrypt the client's reply, means the client, and AS is only aware of SK.

SK= h(S).

As shown in Figure 3, beaf:BD consists of three entities, client, KDC with BC, and Hadoop. The client requests the AS for its authentication. AS sends this request to BC. BC miners mine S, V and send it to the client with ASpuk. Common SK is computed by both client and AS and gets SSK=h(SK). After that, AS sends RK to the client. Using this RK, the client replies with OTP= RK(PSK). AS sends TGT and SSK. TGS sends ST, SSK for client, and Hadoop. The client makes a request using ST with SN. Hadoop replies with SN+1.

Proposed beaf:BD model comprises two steps. Registration and Authentication. Let us discuss them one by one.

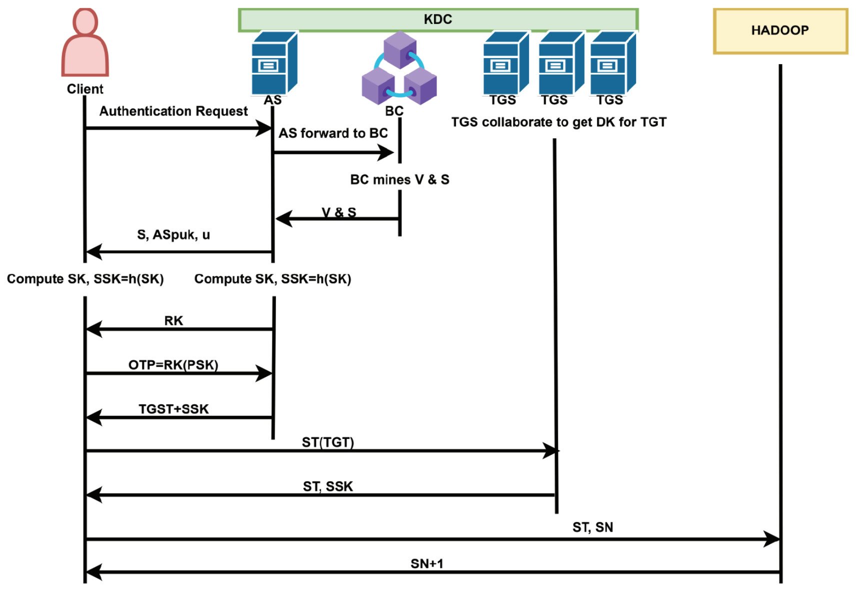

Figure 3. Architecture of beaf:BD

4.1. Registration Phase

In this phase, the client registers itself with KDC and KDC sends all details of the client such as usn, S, V, PSK to BC for future use. The working process of the registration phase can be easily understood in Figure 4.

Algorithm: Client Registration
Input: usn, pwd
Output: usn, S(pwd), V, PSK
Start

1. Client calculates salted hash value of pwd, S(pwd) and V

S(pwd)= h(S, pwd)

V= g^x |N|

2. After computing S(pwd) & V, Client sends usn, S(pwd) and V to KDC

Client → KDC: usn, S(pwd), V

3. Usn details checked by KDC in BC, and if found usn details do not exist in BC, thenPSK sends to the client by KDC for storage

KDC→Client: PSK

4. KDC sends usn, S(pwd), V, PSK to BC by calling smart contract.

KDC→BC: usn, S(pwd), V, PSK.

End

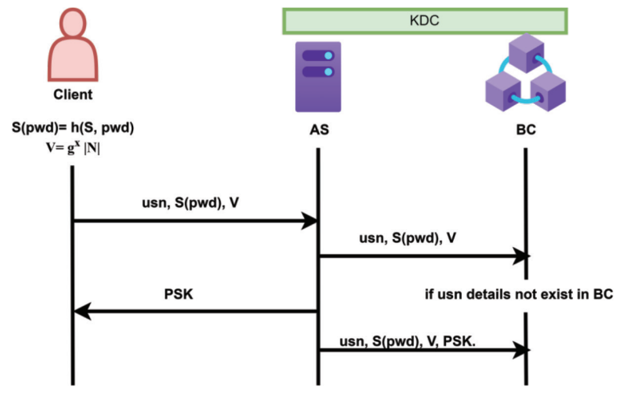

Figure 4. Registration process of baef:BD

4.2. Authentication Phase

This is the second phase of baef:BD. Before accessing Hadoop, the client must authenticate their identity. Figure 5 shows the working process of the authentication phase.

Algorithm: Client Authentication
Input: usn, TGS_{id} and C_{puk}
Output: Encrypt($SSK_{c,h}$ (SN + 1))
Start

1. Client makes an authentication request to KDC with their usn, TGS_{id} and C_{puk}
Client → KDC: usn, TGS_{id} and C_{puk}
2. AS sends usn, TGS_{id} and $C_{puk}$to the BCfor the retrieval of V and S
AS → BC: usn, TGS_{id} and C_{puk}
BC → AS: V, S
3. If client details exist, AS computesAS_{puk} by masking V and AS_{prk}.
AS_{puk}= V masked $g^{(ASprk)}$
4. Computation of hash value of $AS_{puk}$with AS_{prk}
U= h($AS_{puk,}$ AS_{prk})
5. AS sends S, $AS_{puk,}$ and U to the client.
AS →Client: S, AS_{puk} and U
6. Client calculates SSK.
SSK= $(AS_{puk} - g^x)^{Cprk + Ux}$
7. AS calculates SSK
SSK= $(C_{puk} \text{ X } V^u)^{ASprk}$
8. Computation of SSK for communication between Client and AS
$SSK_{c,AS}$ = h(SSK)
9. AS sends RK to the client

10. The client submits the number in the PSK at that place, indicated by the numerals of RK. This process is considered as OTP mechanism. Now, OTP is encrypted by $SSK_{u,a}$

Client → AS: $SSK_{c,AS}$(OTP)

11. A TGT is allotted to the client if AS confirms the correctness of OTP.

12. The ticket granting ticket (TGT) is then encrypted with ticket granting server's (TGS's) public key ast = Encrypt($SSK_{TGS',pub}$($SSK_{t,c}$, usn)).

13. Elliptic Curve ElGamal Encryption is deployed

Encrypt(T_t) = <C_1, C_2> = <δ^k, T_t. $£^k$> , where δ and £ are arbitrary integer selected by TGS

14. AS → Client: *E*ncryption($SSK_{u',pub}$($SSK_{t,c}$, T_t)).

15. The client sendsusn, $T_{t,}$and hash(usn) to TGS to access the Hadoop server.

Client → TGS: usn, T_t, hash(usn)

16. Hadoop service ticket (T_h) is issued by TGS to access the server

TGS → Client:T_h, Encrypt($SSK_{c, pub}$ ($SSK_{c,h}$))

T_h= Encrypt($SSK_{h,pub}$, usn)

Client →Haddop:usn, T_h, Encrypt($SSK_{c,h}$ (SN))

17. The client confirms the authenticity

Hadoop →Client: Encrypt($SSK_{c,h}$ (SN + 1))

End

4.3. Consensus Mechanism

PBFT mechanism is deployed for the proposed model. When the client makes a request, AS sends this request to BC

AS → BC: usn, S, V, PSK

The whole mechanism is summarized in the following six steps:

1. **Leader Election:** The nodes in the network elect a primary node, known as the "leader," to manage the consensus process. This is done using a traditional Byzantine Fault Tolerance algorithm, such as a coin-flipping protocol, to select a leader node that is trusted by all nodes.
2. **Transaction Propagation:** The leader receives a new transaction from a client and broadcasts it to all other nodes in the network.
3. **Validation and Preparation:** Each node in the network receives the transaction and validates it to ensure that it meets the required criteria, such as proper format and digital signature. Once a node validates the transaction, it creates a message containing the transaction and sends it to the other nodes in the network, including the leader.
4. **Pre-Commit:** The leader receives messages from the other nodes, containing their validations of the transaction. Once the leader has received messages from a sufficient number of nodes to reach a consensus, it creates a "pre-commit" message, which includes the validated transaction and the nodes that agreed to it. The leader then sends the pre-commit message to all the nodes in the network.
5. **Commit:** Each node in the network receives the pre-commit message and validates it to ensure that it includes the required number of agreements. Once a node validates the pre-commit message, it adds the transaction to its BC and sends a "commit" message to the other nodes.

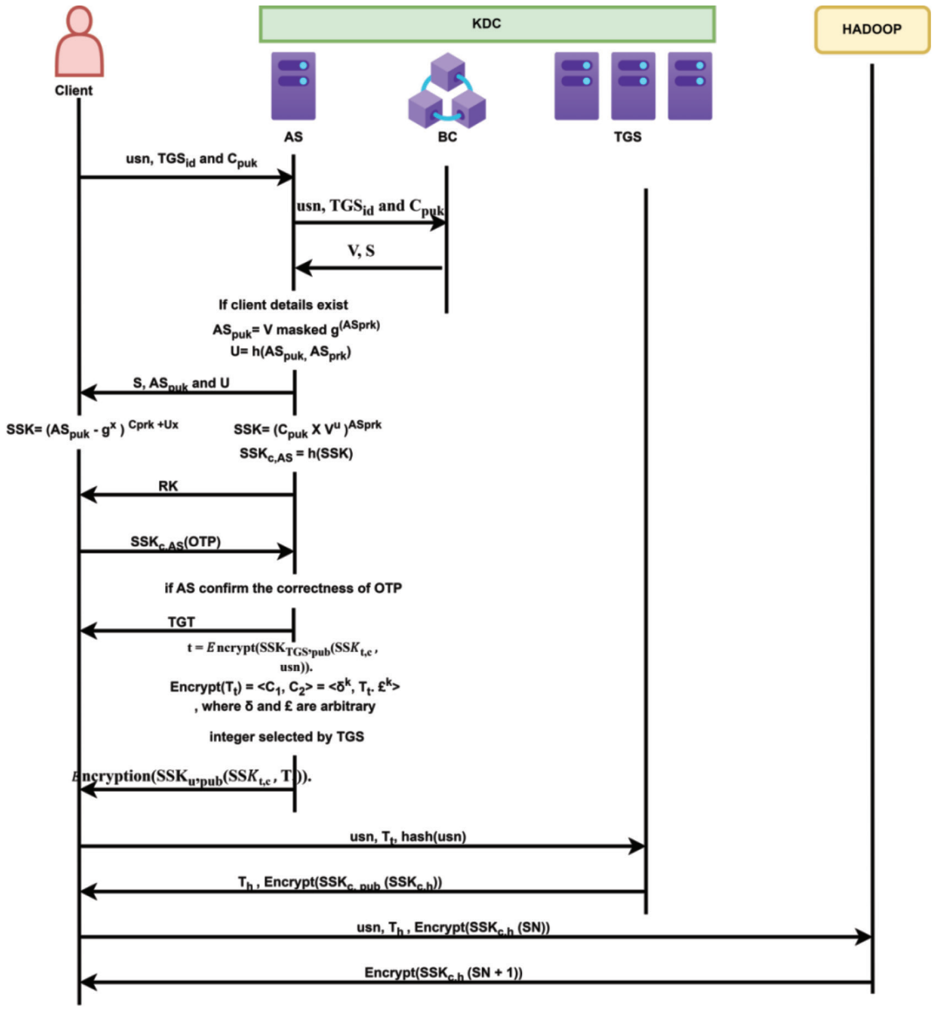

Figure 5. Authentication process of baef:BD

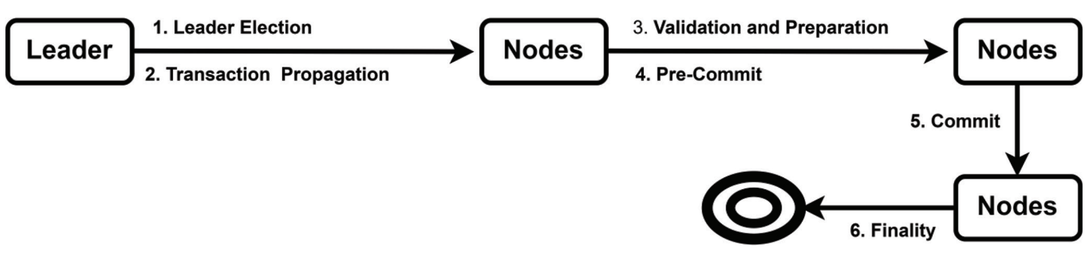

Figure 6. Consensus Mechanism

6. Finality: Once a node receives a sufficient number of commit messages to reach a consensus, it considers the transaction finalized and adds it to its BC permanently.

This diagram represents the sequence of steps involved in the mechanism described. The leader node is responsible for managing the consensus process. The transactions are propagated from the client to the leader and then broadcasted to all other nodes in the network. Each node validates the transaction, and once a consensus is reached, the transaction is committed and considered finalized.

5. Performance Evaluation

5.1. Experimental Setup

Riverbed Modeler simulator is used to simulate baef:BD. As shown in Figure 3, baef:BD consistent client's workstation, KDC with AS and TGS and Hadoop cluster consist of the name node and data node. The node object named ppp_wkstn_adv is utilized as a client workstation and is recognized as the initiator of all communication. The node object ppp_server_adv is deployed as the AS, TGS, and name node. The Internet node experiences a packet loss rate of 0.0% for inbound traffic and introduces a 100 ms delay to network packets. It is assumed that the size of the authentication request is 4KB and that the user spends 8 seconds on composing this message. It took the BC network 1.0 seconds to fetch the user's salt and verifier. The subsequent phases, namely SRP verification and PSK validation require a total of 9.0 seconds. Ticket encryption in this model is assumed to take 10 seconds, and the size of tickets is assumed to be 2 KB. The encrypted exchanges between the user and the Hadoop server have a uniform distribution of sizes ranging from 2 KB to 20 KB.

5.2. Security Analysis

In this section, we will analyze the security of the baef:BD by identifying and addressing several common attacks powerfully and strictly. The goal is to enable users to access Hadoop more robustly and securely, even over an untrusted channel.

5.2.1. Guessing Attack

Assumption 1: During previous communications, a genuine user logged into a secured system using a password that an intruder is now attempting to guess.

Proof: The baef:BD ensures zero-knowledge-proof security, as no information about the password or related details is shared with the KDC during the registration or authentication processes. The employed password storage mechanism ensures that the passwords cannot be readily utilized by an attacker. Suppose an attacker gets S, Spuk, Cpuk, U, and RK as a public parameter, cannot even guess S(pwd), because SSK c,a is verified by OTP.

5.2.2. Replay Attack

Assumption 2: The attacker acquires the authentication dialogue and proceeds to resend it in order to gain entry to the secure Hadoop server.

Proof: A replay attack is characterized by a third-party intercepting and recording a command during transmission, and later replaying it to perform unauthorized actions. Replaying messages during the authentication process is impossible for an attacker, as the secrets are never transmitted over the network. TGS replied with Th, Encrypt(SSKc, pub (SSKc,h)) to the client, where SSKc,h is encrypted with the client's public key, so only the client can decode it.

Table 3. Comparative security analysis of the proposed scheme with that of the related schemes

Security Features	**Rahul et al. [1]**	**Algaradi et al. [9]**	**Jeong et al. [15]**	**Marco et al. [13]**	**Baef:BD**
Single point of failure	□	×	×	×	□
Guessing attack	×	□	□	□	□
Replay Attack	×	×	□	□	□
Brute-force and dictionary attacks:	×	□	×	□	□

5.2.3. Brute-Force and Dictionary Attacks

The attacker attempts to obtain the client's password or other personal information to access their information on the channel, stored in the KDC server database. By using a distributed tamper-free BC storage as the storage medium at the KDC and implementing TC that requires shares from a fixed threshold number of TGS nodes for the authentication process to proceed, baef:BD ensures that there is no impact from a single node shutdown or compromise. The system is safeguarded against denial of service (DoS) attacks, ensuring that there is no single point of vulnerability. The security of the system is greatly enhanced by freeing the KDC from the storage and management of the credentials. There is no possibility of an attacker targeting the KDC as no credentials are stored there. Table 3 provides a comparative security analysis of the proposed scheme with that of the related schemes.

5.3 Performance Analysis

Comparative analysis is conducted between the baef:BD and related systems to evaluate the computational cost. Let CCh represent the computational cost of hash operations, CCe represent the cost of exponentiation functions, and CCc represent the cost of cryptographic functions. During the registration step, the computational cost of the baef:BD is CCh + CCe, while during the user authentication step, it is 4CCc + 3 CCh + 4CCe. During the registration step, the scheme presented by Rahul et al. (2015) costs 4 CCc + CCh, while during the user authentication step it costs 21 CCc. The registration step in the scheme presented by Algaradi et al. (2022) incurs a computational cost of 2 CCc + 3 CCh + CCe, while the authentication step costs 8 CCc + 2 CCh. The registration step in the scheme presented by Jeong et al. (2015) incurs a computational cost of CCh + CCe, while the authentication step costs 6 CCc + 3 CCh + 6CCe. The costs mentioned above were calculated assuming CCh, CCe, and CCc to be approximately equal to 0.0023 ms, 0.0046 ms, and 2.226 ms, respectively. The total computation cost (CCt) =computation cost in registration step (CCr) +computation cost in the authentication step(CCa).

For scheme Rahul et al. (2015): CCtis

$$\begin{aligned} CCt[1] &= 4\ CCc + CCh + 21\ CCc \\ &= 25\ CCc + CCh \\ &= 25(2.226) + 0.0023 \\ &= \mathbf{55.6523\ ms} \end{aligned}$$

For scheme Algaradi et al. (2022): CCtis

CCt[9]= 2 CCc + 3 CCh + CCe + 8 CCc + 2 CCh
= 10CCc + 5CCh + CCe
= 10(2.226) + 5(0.0023) + 0.0046
= 22.26 + 0.0115 + 0.0046
= **22.2761ms**

For scheme Jeong et al. (2015): CCtis

CCt[15]= CCh + CCe + 6 CCc + 3 CCh + 6CCe
= 4CCh +6CCc + 7CCe
= 4(0.0023) + 6(2.226) + 7 (0.0046)
= 0.0092 + 13.356 + 0.0322
= **13.3974ms**

For baef:BD, CCtis

CCt [baef:BD] = CCh + CCe + 4 CCc + 3 CCh + 4 CCe
= 4CCh + 5 CCe + 4CCc
= 4(0.0023) +5(0.0046) + 4(2.226)
=0.0092 + 0.023 + 8.904
= **8.9362 ms**

From Table 4 and Figure 8, it can be easily concluded that the baef:bd model has less computation cost comparedto previous models.

5.4. Time Complexity

During the registration phase, there are two operations: hashing and exponentiation. Both phases use hashing, which has a time complexity of O(M × n) where M is the message length to be hashed, according to the SHA-256 hashing algorithm. The Elliptic Curve ElGamal Cryptosystem, which includes exponentiation and modular inverse operations, has a time complexity of O(n). Therefore, the total time complexity can be expressed as O(M × n) + O(n) + O(n) + O(n), which approximates to O(n).

5.5. Key Sensitivity Analysis

The authentication mechanism is greatly impacted by any intentional or unintentional alteration of even a single bit within the private key. Table 5 describes how the computation of a secure hash

Table 4. Comparative Analysis of Computation Cost

CC in Phases	**Rahul et al. (2015)**	**Algaradi et al. (2022)**	**Jeong et al. (2015)**	**Baef:BD**
CC in Registration (CCr)	4 CCc + CCh =8.9063ms	2 CCc + 3 CCh + CCe =4.4635ms	CCh + CCe =0.0069ms	**CCh + CCe =0.0069ms**
CC in Authentication (CCa)	21 CCc =46.746ms	8 CCc + 2 CCh =17.8126ms	6 CCc + 3 CCh + 6CCe =13.3905ms	**4CCc + 3 CCh + 4CCe =8.9293ms**
Total CC (CCt)	55.6523 ms	22.2761ms	13.3974ms	**8.9362 ms**

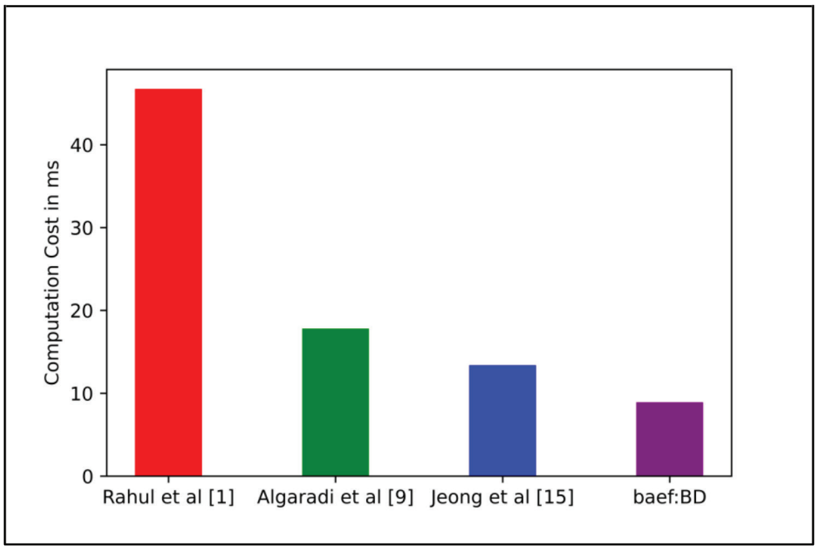

Figure 7. Computation Cost in authentication phase

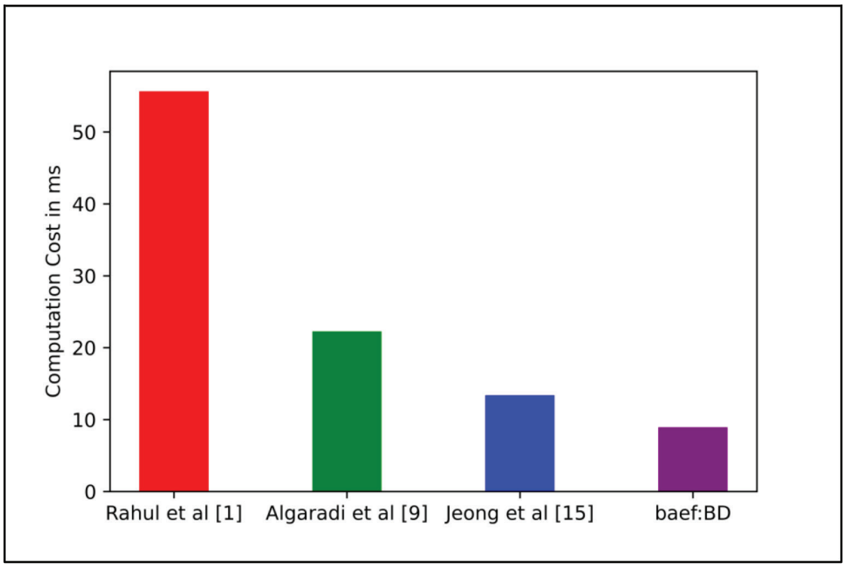

Figure 8. Total Computation Cost

functions on both the server and client sides can detect any alterations made to the shared secret value. The OTP scheme ensures that the key generated by both parties is identical. In the event that the keys on both sides are not identical, the AS is unable to decrypt the reply message sent by the user, as it is encrypted using a session key that was derived from the non-matching keys.

Table 5. SK at client and server side

Client	AS
SK= $(AS_{pub}-g^{x})^{Cpvt+cx}$	SK= $(C_{pub}\ X\ V^{u})^{ASpvt}$
SK= $(V+ g^{ASpvt} - g^{x})^{Cpvt+cx}$	SK= $(g^{Cpvt}\ X\ g^{xu})^{ASpvt}$
SK= $(g^{x} + AS_{pvt} - g^{x})^{Cpvt+cx}$	SK= $(g^{Cpvt} + ux)^{ASpvt}$
SK= $(g^{ASpvt})^{Cpvt+cx}$	SK= $(g^{ASpvt})^{Cpvt+ux}$

Key sensitivity analysis plays a crucial role in cryptography by evaluating the impact of key variations on the security of cryptographic algorithms. It involves analyzing the sensitivity of cryptographic systems to changes in their keys or key components. The key used in a cryptographic algorithm is a crucial factor in ensuring the confidentiality, integrity, and authenticity of data. Key sensitivity analysis helps in understanding how changes or compromises in the key could affect the security of the cryptographic system. It involves examining the relationship between changes in the key and the resulting impact on the algorithm's security properties.

5.6. Key Space Analysis

For any security mechanism, an ideal key should not be excessively short or long. Larger key sizes can result in a decrease in encryption speed. Using small key sizes can make a security mechanism vulnerable to easy cracking. To achieve a high level of security, the key space should be equal to or greater than 2100. Baef:BD employs a key size of 320 bits, key space 2320. The key space is of a sufficient size to prevent any brute force attacks or password guessing attacks.

6. Conclusion and Future Work

With the growing fame of BD analytics and its wide-ranging benefits across industries, the emergence of security concerns has become inevitable. To tackle these challenges, numerous researchers have put forth various solutions. This paper introduces an innovative three-tier authentication framework. The framework relies on three key components: SRP, OTP, and TC. In this proposed framework, the local database at KDC is replaced by a BC network, serving as tamper-proof storage. This modification eliminates single points of failure and mitigates password guessing attacks, effectively safeguarding the Hadoop clusters against major threats such as replay and insider attacks. The system's efficacy was evaluated using the Riverbed Modeller (AE) simulation, demonstrating high efficiency in its performance.

Moving forward, the paper aims to address the time synchronization problem and provide real-time implementation results to substantiate the validity of the proposed claims. By tackling these additional aspects, the research intends to further strengthen the overall security and reliability of the system.

7. References

Abdullah, N., Hakansson A., & Moradian. E. (2017). Blockchain Based Approach to Enhance Big Data Authentication in Distributed Environment. *9th International Conference on Ubiquitous and Future Networks (ICUFN)*, 887-892. https://doi.org/10.1109/ICUFN.2017.7993927

Algaradi, T. S., & Rama. B. (2019). Static Knowledge-Based Authentication Mechanism for Hadoop Distributed Platform Using Kerberos. *Int. J. Adv. Sci. Eng. Inf. Technol., Vol. 9*(3), 772-780. https://doi.org/10.18517/ijaseit.9.3.5721

Algaradi, T. S., & Rama. B. (2022). An authentication key management scheme for securing big data environment. *Electrical and Computer Engineering*, *12*(3), 3238-3248. https://doi.org/10.11591/ijece.v12i3.pp3238-3248

Anisetti, M., Ardagna, C. A., & Berto, F. (2023). An assurance process for Big Data trust worthiness. *Future Generation Computer Systems*, *146*, 34-46, ISSN 0167-739X

Castro, M., & Liskov. B. (1999). Practical Byzantine Fault Tolerance. *3rd Symposium on Operating Systems Design and Implementation*, New Orleans, USA, 1-14.

Chandra, S., Ray, S., & Goswami R. T. (2017). Big Data Security: Survey on Frameworks and Algorithms. *IEEE 7th* International Advance Computing *Conference (IACC)*, 48-54. https://doi.org/10.1109/IACC.2017.0025

Dean, J., & Ghemawat. S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM 51*(1), 107-113. https://doi.org/10.1145/1327452.1327492

Hena, M., & Jeyanthi, N. (2022). Distributed authentication framework for Hadoop based bigdata environment. *J Ambient Intell Human Comput 13*, 4397–4414. https://doi.org/10.1007/s12652-021-03522-0

Honar Pajooh, H., Rashid, M.A., Alam, F., et al. (2021). IoT Big Data provenance scheme using blockchain on Hadoop ecosystem. *J Big Data 8*, 114. https://doi.org/10.1186/s40537-021-00505

Jeong, YS., & Kim, YT. (2015). A token-based authentication security scheme for Hadoop distributed file system using elliptic curve cryptography. *J ComputVirol Hack Tech 11*, 137–142. https://doi.org/10.1007/s11416-014-0236-5

Kilinc, H. H., & Yanik. T. (2014). A Survey of SIP Authentication and Key Agreement Schemes. *IEEE Commun. Surv. Tutorials*, *16*(2), 1005-1023.

Li, R., Asaeda, H., Li, J., & Fu. X. (2017). A Distributed Authentication and Authorization Scheme for In-Network Big Data Sharing. *Digit. Commun. Networks*, *3*(4), 226-235.

Lin, C., He, D., Huang, X., Choo, K. K. R., & Vasilakos. A. V. (2018). BSeIn: A Blockchain-Based Secure Mutual Authentication with Fine-Grained Access Control System for Industry 4.0. J. *Netw. Comput. Appl*., *116*, 42-52.

Lingappa, R. (2019). What Is Secure Remote Password (SRP) Protocol and How to Use It? The Startup, Medium. Accessed 15 March 2021. https://medium.com/swlh/what-is-secure-remote-password-srp-protocol-and-how-touse-it-70e415b94a76

Rahul, P. K., & GireeshKumar. T. (2015). A Novel Authentication Framework for Hadoop. *Advances in Intelligent Systems and Computing*, *324*, 333-340

Sarvabhatla, M., Chandra, M. R. M., & Vorugunti. C. S. (2015). A Secure and Light Weight Authentication Service in Hadoop Using One Time Pad. *Procedia Computer Science*, *50*, 81-86.

Somu, N., Gangaa, A., & Shankar Sriram. V. S. (2014). Authentication Service in Hadoop Using One Time Pad. *Indian J. Sci. Technol., 7*, 56-62

Tall, A.M., & Zou, C.C. (2023). A Framework for Attribute-Based Access Control in Processing Big Data with Multiple Sensitivities. *Appl. Sci*, *13*, 1183. https://doi.org/10.3390/app13021183

Wang, K., Yu, J., Liu, X., & Guo. S. (2017). A Pre-Authentication Approach to Proxy Re-Encryption in Big Data Context. *IEEE Trans. Big Data*, 1-11.

Wu, T-Y., Guo, X., Yang, L., Meng, Q., & Chen, C-M. (2021). A Lightweight Authenticated Key Agreement Protocol Using Fog Nodes in Social Internet of Vehicles. *Mobile Information Systems*, Article ID 3277113, 14 pages. https://doi.org/10.1155/2021/3277113

Zahednejad B., Teng H., Kosari S., & Xiaojun R. (2023). A Lightweight, Secure Big Data-Based Authentication and Key-Agreement Scheme for IoT with Revocability. *International Journal of Intelligent Systems*, 48-54. https://doi.org/10.1155/2023/9731239