Authentication Framework for Kerberos Enabled Hadoop Clusters

M. Hena, N. Jeyanthi

Abstract*: The corporate world is intensively thinking of how to process vast datasets efficiently and securely. Apache Hadoop, an open source framework serves this requirement. Most of the current Hadoop technologies use Kerberos Authentication to incorporate security aspect to Hadoop, which suffers from numerous security and performance issues. The reliance on authentication credentials, a single point of failure as well as a single point of vulnerability, the insider threat and time synchronization problem adds to the list. A comprehensive review of various authentication issues in Kerberos-enabled Hadoop Clusters is provided in this paper. This paper proposes an authentication framework for Hadoop that uses an enhanced One-time Password (OTP) that can solve all the identified problems. The simulation results in Riverbed Modeler proves that the proposed model performs as good as traditional Kerberos Authentication Mechanism. A comparative analysis with existing mechanisms is also presented to strengthen the claims of proposed method

Keywords: Big Data, Security and Privacy, Hadoop, Kerberos Authentication, One-time Password

I. INTRODUCTION

Apache Hadoop is the answer to many of the leading industries in the corporate world today for their concern of data storage and processing. With the advent of the latest computing technologies, devices, and communication modes like social media, enormous multimedia data is getting generated due to actions ranging from human to IoTs (Internet of Things). Big data denotes the huge datasets that are complex and diverse in structure and formats. Organizations are very much interested in this data to gain various insights and harvest benefits out of this data. Social media websites, Financial Service Institutions, Wireless Sensors, and other web services contribute a lot of multimedia big data. Recent developments in the Big data environment like Apache Spark, MongoDB, Giraph, Mahout, Pig, Hive, Kafka and Strom encompasses a number of services. As the number of access points in the cloud increases, the surface area of attack also increased. This is evident from the recent reports on big data breach confirmed by leading restaurants group, Earl Enterprises [1]. As per the report, more than two million credit cards of customers were compromised between May 2018 and March 2019. Another report [2] reveals that over 87 GB of email addresses (772 million) and passwords (22 million) had been leaked in January 2019. According to [3], the challenges faced by the Big Data Ecosystem may be organized into four aspects viz., Infrastructure Security, Data Privacy, Data Management, Integrity, and Reactive Security. Fig.1 indicates the growth rate of data breaches happened over the past five years as per a study reported by Risk-based Security [4]. Apache Hadoop is a big data storage and processing platform developed in Java by Apache Foundation. It has got widespread acceptance where well-known enterprises like Facebook and Amazon have adopted Hadoop for business processing. Map-Reduce and Hadoop Distributed File System (HDFS) are the two major components of Hadoop. A Hadoop cluster follows a master-slave architecture with a Name Node as Master and one or more Data Node(s) as slaves. Name nodes contain all metadata and log files. It receives live information (by heartbeat mechanism) on the active data

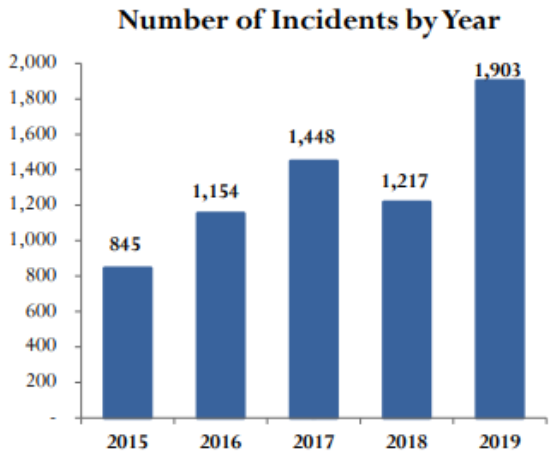

Fig. 1.Data Breaches Happened Over the Past Five Years

nodes and maintains a list of those nodes as well. The jobs submitted by users are received by name nodes and are distributed to data nodes for processing. Unfortunately, Hadoop by itself comes with no in-built security mechanisms. The security problems got more serious with the increasing popularity of Hadoop. Most of the Hadoop technologies support and uses Kerberos Authentication mechanism to ensure the credibility of users and Hadoop entities. Kerberos Protocol relies on third-party authentication. It uses the Key Distribution Centre (KDC) to store the user and other communicating entity information. A logical partition divides KDC into Authentication Server (AS) and Ticket Granting Server (TGS). Any user who wants to access a service from Kerberized Hadoop cluster first requests for a Ticket Granting Ticket (TGT) by sending the client ID. The Authentication Server (AS) verifies the user credentials in its local database and issues the

Reveised Manuscript Received October 05, 2019
***** Correspondence Author
Hena M*, School of Information Technology and Engineering, VIT University, Vellore, Tamilnadu, India.
Dr. N. Jeyanthi, School of Information Technology and Engineering, VIT University, Vellore, Tamilnadu, India.

TGT. The user uses this TGT to request for a Service Granting Ticket (ST). The TGS verifies the user TGT and issues the ST. The requested Hadoop service can be accessed using this Ticket. However, the use of Kerberos Protocol in Hadoop Clusters elevates many security challenges and concerns which includes the dependence on password leading to password guessing attack or dictionary attack, Single point of Failure, Insider threats and Time Synchronization problem of communicating entities. In this paper, we proposed a framework based on an Enhanced Variant of One-Time Password (OTP) to authenticate users. Users have to register themselves with a valid mobile number first for accessing the secured Hadoop Cluster. An image file with some (to be downloaded and saved in the local machine) random numbers are sent to users after a successful registration. Upon sign-on request, a verification code is sent to the registered mobile number of user and user is prompted to enter the authentication code which are the digits of the random number in the received file at places represented by the digits of OTP. The AS compares this OTP received with the image file stored in its local database. If digits matches, the user is authenticated to access the Hadoop services.The remainder of this paper is organized as follows: Section 2 puts light on Kerberos Protocol in Hadoop Clusters. Section 3 presents a comparative study of different recent data protection methodologies. Section 4 discusses the proposed authentication framework. Section 5 presents a experimental analysis using Riverbed Simulation of the proposed solution. Section 6 concludes the paper with hints on future research directions.

II. KERBEROS PROTOCOL IN HADOOP CLUSTERS

Hadoop was initially developed to work in a trusted and secure environment. Security features were not at all focussed at that time. With the widespread usage of Hadoop applications, the security problems popped up and designers adopted third-party authentication mechanisms to ensure the credibility of users and Hadoop entities. Kerberos Authentication mechanism as shown in Fig.2, is supported and adopted by most of the Hadoop technologies today. A Key Distribution Centre (KDC) is a central database that keeps all usernames, passwords and all other information of the clients as well as the Hadoop daemons. KDC is composed of Authentication Server (AS) and Ticket Granting Server (TGS). A user who wants to access a service of Kerberized cluster first asks the Ticket Granting Ticket (TGT) from the AS. With TGT, the client can ask for Service Ticket (ST) from the TGS. Then, the client directly communicates with the namenode in the secured Hadoop cluster using the ST.

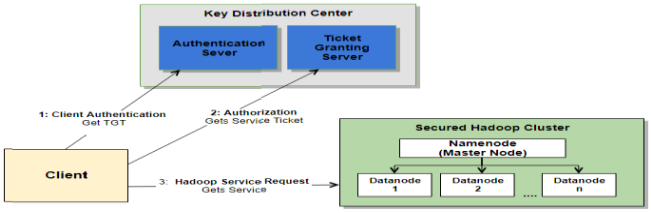

Fig. 2. Kerberos Enabled Hadoop Cluster

Even though the introduction of Kerberos Authentication in Hadoop solves the problem of masquerade attacks and replay attacks to an extent, it still has limitations discussed as follows:

A. *Single Point of Failure:*

The entire authentication system relies on the trusted third party server called Key Distribution Server for mutual authentication of the client and Hadoop cluster. The KDC in the Kerberos system stores all the user name, password and other information related to the communicating entities. If the KDC is compromised by malicious users or overloaded with high-velocity incoming authentication requests, the entire system fails. Nobody will be able to access the secured Hadoop clusters. Even though delegation tokens and block access tokens are introduced in Hadoop, it intensifies the complexity of token management and expands the attack surface.

B. *Password Attacks:*

Even though the password is not directly sent over the network, the authentication system is highly vulnerable to password guessing attack. The Key Distribution Centre (KDC) has the password of all principals (client, TGS, Hadoop Name Node) stored in its local database. When clients send a request for TGT, the Authentication server (AS) in KDC sends the ticket encrypted using the secret key of the TGS and the session key to be used by the client and TGS is encrypted using the secret key (generated from the password) of the client. This session key is used for further communication between the client and the TGS. An attacker can perform offline guessing of this password if he is able to capture all the messages from AS to the client by brute force methods.

C. *Time Synchronization Problem:*

The system heavily depends on time synchronization between communicating entities. This is not an easily achievable task in a distributed environment like Hadoop. The internal network of Hadoop cluster that composes of cheap commercial computers fails to achieve time synchronization and the data nodes in a cluster are widely distributed and different systems may show different timings.

D. *Insider Attacks:*

The trusted data nodes in a cluster itself can turn hostile in a later phase. They can add or remove any data in the HDFS or can even tamper the metadata.

III. RELATED WORKS

Kerberos Protocol depends on Symmetric Cryptography and as a result, there are a number of apparent weaknesses like vulnerability to password guessing attack, replay attacks, and complex key management. Researches have been conducted from the time since Kerberos came into being to overcome these problems. E. El-Emam et al., [5] proposed to change the Kerberos database to defend against password guessing attack. Instead of generating a private key from a user password, the method exploits SHA-256 hashing and 3-DES algorithm. Another method [6]used Dynamic Passwords and Diffie –Hellman

Algorithm to prevent dictionary attacks and to exchange passwords. The security of Kerberos Server (KDC) and replay attack is not addressed here. Z. Hu et al.,[7] also suggested the usage of Diffie Hellman DSA mechanism to improve Kerberos performance. As discussed before, the Kerberos Authentication mechanism in Hadoop Environment suffers from various limitations including the Single Point of Failure, Password Attacks, Time Synchronisation Problem and insider attacks. Various researches are being conducted across the globe by industrial as well as the academic world. P. K. Rahul and T. Gireeshkumar [8] proposed a novel authentication framework for Hadoop which is an amalgam of

Public-key cryptography, Private key cryptography, Hashing, and Random number generation. The usage of public-key cryptography makes the system slower and added the computational overhead. This is not a favorable thing for big data scenario. Y. Jeong and S. S. K. Han [9] used a high dimensional data authentication protocol based on hash chains. The problem with this approach is either long hash chain need to be maintained in the memory leading to storage overhead or the hash chains need to be reconstructed very often. A Trusted Platform Module (TPM) that Combines hardware and software security solutions is proposed by Z. Dou et.al., [10] applies the RSA Encryption technique. This method didn't address the attack against the TPM and the Hadoop Name nodes are assumed to be partially secure. Moreover, the endorsement key is not made available to end users. A token-based authentication security scheme for the Hadoop distributed file system using elliptic curve cryptography is recommended by Y. J. Y. Kim [11]. The method used Block Access Token and Hash Chain of Keys. As the algorithm is very complex there is a high chance of implementation error and hence compromise on security. D. Chattaraj et.al., [12] proposed an Efficient and Fault-tolerant Authentication and Key Exchange Protocol for Hadoop-assisted Big Data Platform based on the digital signature, Advanced Encryption Standard and Elliptic Curve Cryptography. Time-consuming and complex computations are the problems of this scheme. K. A. Shakil et. al., [13]came up with biometric Signature and Resilient Back Propagation. The hardware cost and signature inconsistency are the hindering factors for adoption of this method. Wang et.al., [14] proposed a Token Push Mechanism where Agents generates crossnode tokens after uses' first successful login through KDC. These are pushed to all datanodes involved. Datanode uses these cross node tokens to to authenticate users. T-nonce parameter generated by hash processing of Timestamp and IP to prevent replay attacks. The trustworthiness and security of agents need to be ensured here. Time Synchronization in the distributive environment also is a concern hereTable I is a comparative study on the various existing mechanisms to solve the authentication issues in Kerberos enabled Hadoop Clusters. From the study, it is observed that each method has one or the other pitfalls and there does not exist a complete solution for the authentication issues in Hadoop clusters. This paper proposed an Authentication Mechanism for Kerberized Hadoop Clusters which washes out all the limitations of the methods described above.

IV. PROPOSED AUTHENTICATION MECHANISM FOR KERBERIZED HADOOP CLUSTERS WITH ENHANCED OTP VERIFICATION METHOD

A. *Overview*

The proposed system as shown in Fig. 3, authenticates the users or Hadoop clusters using an enhanced variant of One-Time Password. The user has to register for Hadoop services. A third-party authentication server, called Key Distribution Centre, KDC, maintains a local database to store user credentials. This server is logically divided into two servers - Ticket Granting Server (TGS) and Authentication Server (AS). Upon registration, the AS sends an image file to the client which they have to download and store in their respective machines. This file consists of some nine-digit number called Pre-Shared Key (PSK).When the user wants to access any of the Hadoop services, the user has to log on to the system. This request consists of the identity of the user, address of the work station and identity of Ticket Granting Server (TGS). The Authentication Server (AS) checks for the client details in its local database and sends a random (5 digit) OTP to the registered mobile number of the user. The user has to look into the image file received during registration and has to reply with digits from the random number in the received file at the places represented by the OTP. That is, for example, suppose the digits in the received file is "4 5 1 9 6 2 3 7 4" and the OTP sent by the AS is "2 8 4 1". The user has to reply with digits at 2nd, 8th, 4th and 1st positions in the received file. That is, "1 7 9 5" in this case. The AS compares this OTP received with the image file stored in its local database. If both matches, the user is authenticated to access the Hadoop services.

Table I. Comparative Analysis of some Existing Solutions

Title	Methodology	Introduced Idea	Drawbacks

A Novel Authentication framework for Hadoop	• Public key cryptography • Private key cryptography • Hashing • Random number generation.	• Two servers – User Server (for user authentication) and Data Server (for secure communication) • A system-generated random number [SHA-2(username & this random number)] is used for all client authentication. The block IDs of secured data is transferred in encrypted form (Symmetric Key).	• Public key cryptography makes the system slower • Computational overhead • Certification problem of the public key, not mentioned
High-dimensional data authentication protocol based on hash chain for Hadoop systems	• Hash chain-based Authentication	• The namenode generates a secret key to be shared with each client. A Delegation Token is generated from a seed value using hash chain mechanism and is shared with clients. The clients who submit correct Delegation Tokens are given Block Access Tokens to access the data blocks in the data node.	• Either long hash chain need to be maintained => memory • Or hash chain need to be reconstructed very often • Whole load on Namenode
Robust Insider Attacks Countermeasure for Hadoop: Design and Implementation	Trusted Platform Module (TPM) • Combines hardware and software security solutions • RSA Encryption	• Session keys are encrypted using Authentications Keys in TPM and values in Platform Configuration Registers (PCRs). The entities are sealed and bound to specific trusted platform configurations. Remote platform attestation services are also enabled through periodic fingerprint checking mechanism (Heartbeat Mechanism)	• The attack against TPM not addressed • Hadoop Name nodes are assumed to be partially secure • The endorsement key is not made available to end-users. Only the manufacturer knows it
A token-based authentication security scheme for Hadoop distributed file system using elliptic curve cryptography	Elliptic Curve Cryptography (ECC) • Block Access Token • Hash Chain of keys	• Two tokens are used – Delegation Token (DT) and Block Access Token (BAT) generated using ECC. Clients authenticated using DT are given BAT to access secure data blocks in the data node.	• Complex Algorithm • High chance of implementation error and hence can compromise security.
HEAP: An Efficient and Fault-tolerant Authentication and Key Exchange Protocol for Hadoop-assisted Big Data Platform	• Digital Signature • Advanced Encryption Standard (AES) • Elliptic Curve Cryptography (ECC)	• HEAP-KDC has introduced three servers – two public (Client Management Server & Namenode Management Server) and one private (Enrollment Server). The entities first register themselves the ES and obtain dummy one-time credentials. Then they enroll with the corresponding public servers. For mutual authentication, digital signatures are used.	• Time-consuming • Complex computations • Additional Server management requirements and increased area of attacks.

The improvement of HDFS Authentication Model Based on Token Push Mechanism	• Agent-based Token Push Mechanism	• Agents generate cross node tokens after uses' first successful login through KDC. These are pushed to all data nodes involved. Datanode uses these cross node tokens to authenticate users. • T-nonce parameter generated by the hash processing of Timestamp and IP to prevent replay attacks.	• Security of agent needs to be ensured • Time synchronization problem in a distributed environment

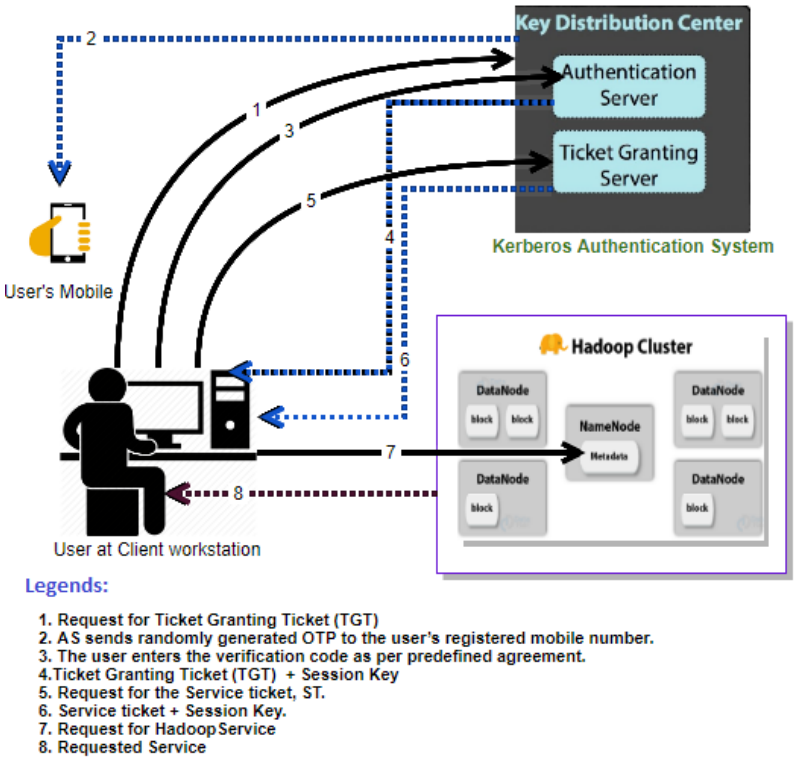

Fig. 3.Proposed Authentication Framework

Algorithm: User Registration Process
Input: User registration request with *X*, user_mob
Output: User registered, obtained *PSK*
Step 1: X → KDC: ***{"X", user_mob}***
Step 2: At KDC
a. ***PSK=rand.nextDouble() * 10000000000L;***
// generates 10-digit random number
b. KDC → X: {file containing PSK}
c. KDC →local_db : ***{"X", user_mob, PSK}***
Step 3: X downloads and saves PSK file safely
Step 4: End.

A. Notations

The variables used in the proposed algorithm are listed in Table II.

B. Working Mechanism

The steps in the proposed authentication system are as per the following algorithms:

Table II Variable Used in the Proposed Algorithm

Variables used	
X	Identity of user
KDC	Key Distribution Center
AD_x	Address of user X
TGS	Ticket Granting Server (TGS).
TGS _ID	Identity of Ticket Granting Server (TGS)
user_mob	Registered Mobile Number of user
AS	Authentication Server
KEY_RAND	Randomly generated key by AS; temporarily stored in "**verifi_code"**
PSK	Pre-shared Key
K_{xp}	Public key of User X
K_{xt}	Session Key for User X and the Ticket Granting Server (TGS);
T_t	Ticket to be used by the user to access the Ticket Granting Server (TGS)
K_{tp}	Public Key of the Ticket Granting Server (TGS)
E(K(M))	Message M encrypted with Key K
$scrt\#_x$	randomly generated secret number generated by X
H	Identity of Hadoop Server
T_H	Ticket to be used by the user to access the Kerberos Enabled Hadoop Server, H
seq#	starting sequence number to be used by the server for messages to the client during this session
k, j, x	Array variables
flag	Flag variable
i	Loop variable

User Registration Process

The user has to register initially with the authentication system. The user submits her ID and mobile number to the system and the Authentication Server (AS) generates a Pre-Shared Key (PSK) which is a nine-digit random number to send back to the user. The user keeps this PSK safe with them to use in all future communication with the secured system. The algorithm for the user registration process is as follows:

Authentication Service Process to obtain TGT

User X, at the workstation, sends a message to the Key Distribution Center (KDC). This message requests a session key for her communication with the Ticket Granting Server (TGS). The Authentication Server (AS) at Key Distribution Center (KDC) sends a randomly generated key, KEY_RAND to the registered mobile number of the user. The user enters the verification code as per the Pre-shared Key (PSK) and sends to the Authentication Server (AS). The Authentication Server verifies the verification code by retrieving the value stored in the local database of the Key Distribution Center (KDC). If the verification code matches, the Authentication Server (AS) generates a session key for communication between the User X and the Ticket Granting Server (TGS). This is encrypted with the public key of the user and is sent to the user. If the verification code mismatches for 3 consecutive times the user is blocked with a message "Access Denied". The user decrypts the message (4) using her private key and gets the session key Kxt. The algorithm works as follows:

Algorithm: Authentication Service Process to obtain TGT
Input: Authentication Request with X, AD_x, TGS_ID
Output: User Authenticated, obtained Tt, K_{xt}

Step 1: X → KDC : ***{"X", AD_x, TGS_ID}*** ------- (1)
Step 2: At AS,
a. ***KEY_RAND= (int)(100000.0 * Math.random());***
// generates 5-digit random key
b. AS → user_mob : ***{KEY_RAND}*** ------- (2)
Step 3: At User X
a. verifi_code = *{KEY_RAND(PSK)}*
b. X → AS: verifi_code -------(3)
Step 4: At AS

```
a. for(int i=0; i<PSK.length(); i++)
        {
            k[i] = Character.digit(PSK.charAt(i), 10);
        }
```

```
b. for(int i=0; i<userKeyDB.length(); i++) /* Extracts digits of KEY_RAND and stores in array j*/
        {
            j[i] = Character.digit(userKeyDB.charAt(i), 10);
        }
c. for(i=0;i < verfi_code.length();i++) /* Checks if the user has entered the correct key as per pre-agreement */
        {
          If (k[j[i]]==x[i])      flag=1;
          Else                    flag = 0;
        }
If (Flag==1)
        i) Tt = E(Ktp(Kxt, "X", ADc))
        ii) AS→ X: {E (Kxp(Kxt)), Tt} -------(4)
        iii) Goto Step 5
        else if : the verification code mismatches
```

for 3 consecutive times the user is blocked with a message "**Access Denied**".

Step 5: The user decrypts the message (4) using her private key and gets the session key $K_{xt.}$

Step 6: End.

Ticket Granting Process to obtain Service Ticket

The user sends a request to the Ticket Granting Server (TGS) asking for access permission to the Secured Hadoop Server. The Ticket Granting Server (TGS) decrypts the ticket Tt, received in the message (5) using its private key and obtains the session key Kxt.; secret number scrt#x using the session key Kxt . (Note: This confirms the identity of user X, as only those who know user X's private key could have decrypted message (4) and hence obtain the session key Kxt.). The Ticket Granting Server (TGS) generates a session key, Kxh for communication between user X and the Hadoop Server H. It is encrypted using the session key Kxt and is sent to the user. The algorithm works as follows:

Algorithm: Ticket Granting Process to obtain Service Ticket
Input: Service Ticket request with T_t, *scrt#x* (encrypted), "H"
Output: Service Ticket T_h , K_{xh}

Step 1: At user X
X → TGS : $\{T_t, E(K_{xt}(scrt\#_x), "H")\}$ -------(5)
Step 2: The Ticket Granting Server (TGS) decrypts
a. the ticket T_t, received in the message (5) using its private key and obtains the session key $K_{xt.}$
b. secret number $scrt\#_x$ using the session key K_{xt} obtained in the above step (2a).

(Note: *This confirms the identity of user X, as only those who know user X's private key could have decrypted message (4) and hence obtain the session key* $K_{xt.}$*).*
Step 3: At TGS:
$T_H = E(K_{hp}(K_{xh}, "X", AD_c))$
TGS → X : $\{T_H, E(K_{xt}(K_{xh}), scrt\#_x)\}$ -------(6)
where
T_H : Ticket to be used by the user to access the Kerberos Enabled Hadoop Server, H
Step 4: The user X decrypts the message and obtains the session key K_{xh} , to communicate with the Hadoop Server.
(Note: The $scrt\#_x$ *assures the freshness of the message and is not a replay by some attacker)*
Step 5: End.

User / Hadoop Server Mutual Authentication Process to obtain Service

The user X decrypts the message and obtains the session key Kxh , to communicate with the Hadoop Server. (Note: The scrt#x, assures the freshness of the message and is not a replay by some attacker). The user X sends a service request message to the Hadoop Server by producing the ticket it received. It includes a newly generated secret number along with the request message to ensure there is no replay. The Hadoop Server decrypts the ticket TH, using its private key and gets session key Kxh; the seq# using the session key Kxh.. The Hadoop Server encrypts the seq# obtained in step 2b using the session key Kxh and sends to the user X. The user decrypts the message to detect the replays.
The algorithm works as follows:

Algorithm: User / Hadoop Server Mutual Authentication Process to obtain Service
Input: Service request with *Th, seq#* (encrypted)
Output: Mutually Authenticated to provide requested Service

Step 1: At user X
X→ H : $\{T_H, E(K_{xh}(seq\#))\}$ -------(7)
Step 2: The Hadoop Server decrypts:
a. the ticket T_H, using its private key and gets session key $K_{xh;}$
b. the *seq#* using the session key K_{xh} obtained in the above step (2a).
Step 3: At Hadoop Server
H→ X: $\{E(K_{xh}(seq\#))\}$ -------(8)
Step 4: The user decrypts the message (8) to detect the replays.
Step 5: End.

Fig.4 illustrates the visual representation of the proposed algorithm.

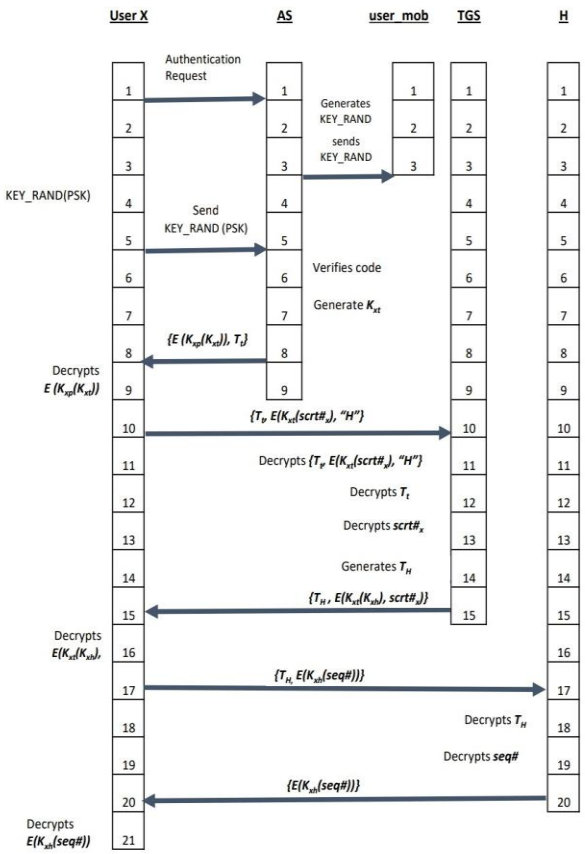

Fig. 4.Visual Summary of Proposed Algorithm

V. EXPERIMENTAL ANALYSIS OF PROPOSED SOLUTION

The proposed method is expected to overcome all the identified problems and is a viable solution with less communicational and computational overhead. In order to validate and analyze the virtues of the proposed authentication method, Riverbed Modeler (AE) Simulation [15] platform is used in this research.

A Hadoop cluster with a Namenode (Master) and three Data nodes (Slaves) are created. A Key Distribution Centre with Authentication Server (AS) and a Ticket Granting Server is also simulated as shown in Fig. 5.

In the simulation model, ppp_wkstn_adv node object is used as a user workstation. ppp_server node object is used as the Key Distribution Centre entities and for Namenode. The Internet cloud is configured to discard 0.0% of arriving packets and to add 100-millisecond delay to the traffic. The authentication request message size is assumed to be 2 KB and the user doesn't spend any time to initialize this message. The Pre-shared Key (PSK) is of 1 KB size and tickets are of size 5 KB each. It takes 0.2 seconds for encryption of the packets and inter-packet arrival time is 1.2 seconds. The size of encrypted messages is uniformly distributed between 1 KB and 10 KB. 1000 messages are generated throughout the communication session.

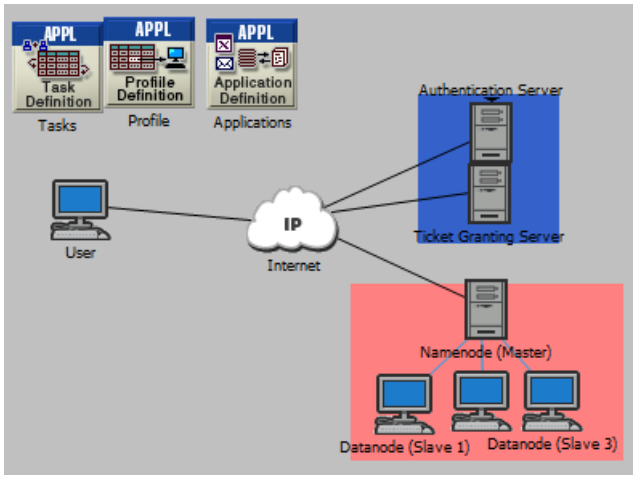

Fig. 5.Network Topology of Kerberos enabled Hadoop Cluster Simulation Model

A. Security Analysis

The security evaluation of the proposed authentication system for Hadoop clusters is done in this section. The potential security concerns are dealt as follows:

Replay Attacks

Assumption 1:

The attacker manages to get the KEY_RAND used during any previous transactions between the secured Hadoop cluster and an authenticated user and retransmit it to gain access to the system

Proof: The intruder fails to log in into the system with that KEY_RAND since it is no longer valid. Moreover, only a legitimate user can only produce the response KEY_RAND(PSK) Hence, the attackers' intention to create replays fails.

Assumption 2:

The attacker captures {Tt, E(Kxt(scrt#x), "H"} send from the user X to the Ticket Granting Server and retransmits it get access to the system.

Proof: The attacker who pretends as a legitimate user sends {Tt, E(Kxt(scrt#x), "H"} to the Ticket Granting Server to gain access to the secured Hadoop Cluster. The attacker receives back {TH , E(Kxt(Kxh), scrt#x)} in response to the message. As the attacker does not know Kxt, he cannot decrypt the message. Only those users who have passed the initial authentication test by submitting the correct value of KEY_RAND(PSK) knows Kxh and hence Kxt. Thus, the attacker fails to replay.

Dictionary Attacks

Assumption 3:

An intruder guesses the password used by a legitimate user to log into the system during previous transactions.

Proof: Dictionary attacks or password guessing attacks are completely defended as the verification code entered when prompted is not exactly what received over the mobile number or email. It is a code generated from received KEY_RAND and previously-stored verification code, Pre-Shared Key (PSK).

Compromise on KDC

Assumption 4:

An adversary hacks Key Distribution Centre (KDC) to get the user credentials.

Proof: The proposed system uses an enhanced variant of One Time Password generated from a verification code, KEY_RAND and a Pre-Shared Key (PSK) to authenticate the registered users. The KDC is relieved from saving the passwords and as a result, the security of KDC is improved to a great extent. There is no possible threat of attackers compromising the KDC for attaining the password.

Time synchronization problems

Time synchronization problems are also abridged in the new system. As the dynamic passcodes are used (based on SMS), the dependence on ticket validity is just a supporting feature to prevent replay attacks.

B. Performance Analysis

The verification code generated at the time of registration needs to be stored in the local database of KDC. The AS has to take care of the key generation, random number (OTP) generation and verification of code for authentication. This may cause a bottleneck on KDC in large clusters. However, the proposed system doesn't require too many information to be transmitted between the client and the KDC. This reduces the time and computational complexity of encryption and decryption steps required in the authentication process.

The Packet Network Delay experienced in the deployed scenarios are depicted in Fig. 6. It is very evident that the proposed authentication method doesn't introduce any significant delay in the network performance when compared with traditional Kerberos enabled Hadoop clusters.

Fig. 7 shows the response time of the three deployed applications. The illustrations proves that the response time for Hash chain based method is very high due to complex computations at the Namenode.

The proposed method has doesn't introduce any significant delay in order to complete the various phases of the tasks in the simulated environment. Hence, the proposed authentication method performs

as good as traditional Kerberos Authentication Systems with enhanced security features. It is observed that the method doesn't introduce any significant computational and communicational overhead to the system.

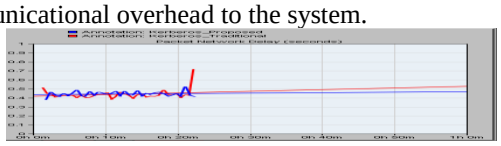

Fig. 6.Packet Network Delay

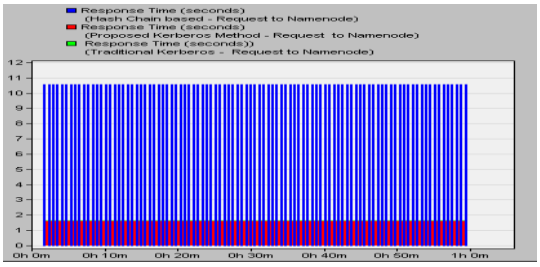

Fig. 7.Response Time

Table III summarizes the security analysis of the proposed system in comparison with existing systems without any authentication, traditional Kerberos-enabled Hadoop Systems and some existing improved mechanisms.

The performance of the proposed authentication protocol is compared with traditional Kerberos- enabled Hadoop systems in terms of Storage and Computational Overhead at Hadoop Master (Namenode). The results are summarized in Table IV.

Table III Security Evaluation

Security Analysis	Hadoop Cluster With Traditional Kerberos Authentication	[8]	[9]	[10]	[11]	[12]	[14]	Proposed System
Replay Attacks	☒	☒	☒	☒	☒	☒	☒	☒
Dictionary Attacks	✓	☒	☒	☒	☒	☒	✓	☒
KDC Compromise	✓	☒	☒	☒	☒	✓	✓	☒
Time Synchronisation	✓	✓	✓	☒	✓	☒	☒	☒

Table IV Performance Evaluation

Performance Analysis	Hadoop Cluster With Traditional Kerberos Authentication	Proposed System
Stored Values At Namenode	K_{xh}, K_{hp}	K_{xh}, K_{hp}
Stored Values At Client	K_{xp}, K_{xt}, K_{xh}	K_{xp}, K_{xt}, K_{xh} , PSK
Computation Overhead At Namenode	Low	Low

VI. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a new authentication framework for Kerberized Hadoop Clusters with Enhanced OTP Verification method. It provides strong authentication for communicating entities in a Kerberos enabled Hadoop Cluster. The OTP used in this approach improves the security of the Key Distribution Centre as no password is saved as such. The method prevents password guessing attacks and dictionary attacks. The problem of time synchronization is also solved in the proposed system. Moreover, the need to transmit only less information between communicating entities and KDC reduces the time and computational complexity of encryption and decryption processes. The proposed method is expected to overcome all the identified problems and is a viable solution with less communicational and computational overhead. In the future, we will come with the real-world implementation details and performance evaluation of the proposed framework in a real-time scenario to prove the strength of our method.

REFERENCES

1. "The Latest Big Data Breach Should Make You Rethink How You Pay For Everything." [Online]. Available: https://www.forbes.com/sites/suzannerowankelleher/2019/04/04/the-latest-big-data-breach-should-make-you-rethink-how-you-pay-for-everything/#21069d3f4e4b. [Accessed: 18-Jun-2019].
2. In 2019's first big data breach, over 772 mn email addresses leaked | Business Standard News." [Online]. Available: https://www.business-standard.com/article/current-affairs/in-2019-s-first-big-data-breach-over-772-mn-email-addresses-leaked-119011700324_1.html. [Accessed: 18-Jun-2019].
3. D. Security, *The permanent and official location for Cloud Security Alliance Big Data research is*. 2016.
4. I. Risk Based Security, "Data Breach QuickView Report First Quarter 2019 - Data Breach Trends," 2019.
5. E. El-emam, M. Koutb, H. Kelash, and O. F. Allah, "An Optimized Kerberos Authentication Protocol," pp. 1–6, 2009.
6. C. Wang, "Security Analysis and Improvement for Kerberos Based on Dynamic Password and Diffile-Hellman Algorithm," vol. 1, no. 1, pp. 2–6, 2013.
7. Z. Hu, Y. Zhu, and L. Ma, "An Improved Kerberos Protocol based on Diffie-Hellman-DSA Key Exchange," pp. 400–404, 2012.
8. P. K. Rahul and T. Gireeshkumar, "A Novel Authentication Framework for Hadoop," *Artif. Intell. Evol. Algorithms Eng. Syst. Adv. Intell. Syst. Comput.*, vol. 324, pp. 333–340, 2015.
9. Y. Jeong and S. S. K. Han, "High-dimentional data authentication protocol based on hash chain for Hadoop systems," *Clust. Comput. J. Networks, Softw. Tools Appl.*, vol. 19, no. 1, pp. 475–484, 2016.
10. Z. Dou, I. Khalil, A. Khreishah, and A. Al-Fuqaha, "Robust insider attacks countermeasure for hadoop: Design and implementation," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1874–1885, 2018.
11. Y. J. Y. Kim, "A token-based

authentication security scheme for Hadoop distributed file system using elliptic curve cryptography,"
12. *Comput. Virol. Hacking Tech.*, vol. 11, no. 3, pp. 137–142, 2015.
13. D. Chattaraj, S. Member, and M. Sarma, "HEAP : An Efficient
14. and Fault-tolerant Authentication and Key Exchange Protocol for Hadoop-assisted Big Data Platform," *IEEE Access*, vol. PP, no. c, p. 1, 2018.
15. K. A. Shakil, F. J. Zareen, M. Alam, and S. Jabin, "BAMHealthCloud : A biometric authentication and data management system for healthcare data in cloud," *J. King Saud Univ. - Comput. Inf. Sci.*, pp. 0–7, 2017.
16. Wang Guiyuan; Ning Hongyun, "The improvement of HDFS Authetnication Model Based on Token Push Mechanism," in *ICBDC '18 Proceedings of the 2018 International Conference on Big Data and Computing*, pp. 53–58.
17. A. S. Sethi; and V. Y. Hnatyshin, *The Practical OPNET® User Guide for Computer Network Simulation*. 2013.

AUTHORS PROFILE

Hena M. received Masters in Technology in software Technology from VIT University, Vellore, Tamilnadu, India. Currently, she is a research scholar at School of Information Science and Engineering, VIT Vellore. She worked as Assistant Professor Eranad Knowledge City Technical Campus, Manjeri, Malappuram, Kerala, India and Lecturer in Computer Science at University of Kerala, Trivandrum, India. She has authored many publications in peer-reviewed reputed journals and conference proceedings. Her areas of interest include Cloud Computing and Big Data security.

Dr. N. Jeyanthi received her Ph.D. degree in Cloud Security from VIT University, Vellore, Tamilnadu, India. She is an Associate Professor in VIT, Vellore for School of Information Science and Engineering. Her research work was funded by Department of Science and Technology, Govt. of India. She has authored and co-authored over 62 research publications in peer-reviewed reputed journals and 30 conference proceedings. Her entire publications have been cited over 265 times (Google Scholar). The latest Google h-index of his publications is 10 and i10 index is 13. Books and book chapters were also added to her research contribution. She has served as the program committee member of various international conferences and reviewer for various international journals. She has been honoured by VIT as an active researcher for four consecutive years. Her current areas of interest include IoT, Cloud, and Big Data security.