Attribute Based Honey Encryption Algorithm for Securing Big Data: Hadoop Distributed File System Perspective

Gayatri Kapil [Corresp., 1] , **Alka Agrawal** [1] , **Abdulaziz Attaallah** [2] , **Abdullah Algarni** [2] , **Rajeev Kumar** [1] , **Raees Ahmad Khan** [1]

[1] Information Technology, Babasaheb Bhimrao Ambedkar University, Lucknow, Uttar Pradesh, India

[2] Faculty of Computing and Information Technology, King Abdul Aziz University, Jeddah, Saudi Arabia

Corresponding Author: Gayatri Kapil
Email address: gayatri1258@gmail.com

Hadoop has become a promising platform to reliably process and store big data. It provides flexible and low cost services to huge data through Hadoop Distributed File System (HDFS) storage. Unfortunately, absence of any inherent security mechanism in Hadoop increases the possibility of malicious attacks on the data processed or stored through Hadoop. In this scenario, securing the data stored in HDFS becomes a challenging task. Hence, researchers and practitioners have intensified their efforts in working on mechanisms that would protect user's information collated in HDFS. This has led to the development of numerous encryption-decryption algorithms but their performance decreases as the file size increases. In the present study, the authors have enlisted a methodology to solve the issue of data security in Hadoop storage. The authors have integrated Attribute Based Encryption with the honey encryption on Hadoop, i.e., Attribute Based Honey Encryption (ABHE). This approach works on files that are encoded inside the HDFS and decoded inside the Mapper. In addition, the authors have evaluated the proposed ABHE algorithm by performing encryption-decryption on different sizes of files and have compared the same with existing ones including AES and AES with OTP algorithms. The ABHE algorithm shows considerable improvement in performance during the encryption-decryption of files.

1 # Attribute Based Honey Encryption Algorithm for
2 # Securing Big Data: Hadoop Distributed File System
3 # Perspective

5 **Gayatri Kapil[1], Alka Agrawal[1], Abdulaziz Attaallah[2], Abdullah Algarni[2], Rajeev Kumar[1],**
6 **Raees Ahmad Khan[1]**

8 [1]Information Technology, Babasaheb Bhimrao Ambedkar University, Lucknow, UP, India
9 [2]Faculty of Computing and Information Technology, King Abdul Aziz University, Jeddah, Saudi
10 Arabia
11 Corresponding Author: Gayatri Kapil
12 Email: gayatri1258@gmail.com

Abstract

16 Hadoop has become a promising platform to reliably process and store big data. It provides
17 flexible and low cost services to huge data through Hadoop Distributed File System (HDFS)
18 storage. Unfortunately, absence of any inherent security mechanism in Hadoop increases the
19 possibility of malicious attacks on the data processed or stored through Hadoop. In this scenario,
20 securing the data stored in HDFS becomes a challenging task. Hence, researchers and
21 practitioners have intensified their efforts in working on mechanisms that would protect user's
22 information collated in HDFS. This has led to the development of numerous encryption-decryption
23 algorithms but their performance decreases as the file size increases. In the present study, the
24 authors have enlisted a methodology to solve the issue of data security in Hadoop storage. The
25 authors have integrated Attribute Based Encryption with the honey encryption on Hadoop, i.e.,
26 Attribute Based Honey Encryption (ABHE). This approach works on files that are encoded inside
27 the HDFS and decoded inside the Mapper. In addition, the authors have evaluated the proposed
28 ABHE algorithm by performing encryption-decryption on different sizes of files and have
29 compared the same with existing ones including AES and AES with OTP algorithms. The ABHE
30 algorithm shows considerable improvement in performance during the encryption-decryption of
31 files.

33 **Keywords:** Big Data; Cloud Storage; Hadoop; HDFS; Data Security; Encryption; Decryption.

Introduction

37 Data security has now become one of the top most concerns for any individual or organization. Day
38 by day substantial amount of information is transferred through digital applications which require

39 heaps of extra storage space, processing assets and dynamic framework execution. The
40 exponential use of smart phones, social networking sites, downloaded apps, web sensor are
41 generating huge amount of data. This has led to several issues in big data including storage
42 customization, security, cost-effectiveness, smooth performance, vendor lock-in, and compliance.
43 All these issues have their importance in Hadoop. But, big data security and privacy has become
44 the burning issue for Hadoop HDFS data storage and distributed computing. This study essentially
45 focuses on ensuring security and privacy for big data at the storage level.

46

47 When utilizing the Hadoop HDFS data storage service, clients have no compelling reason to store
48 information locally and thus convey it constantly. As is the usual norm, the information is kept on
49 Hadoop HDFS storage server to ensure that clients can access a given information as per their
50 convenience, irrespective of the time and place they choose to avail it from. Hadoop HDFS storage
51 server provides both hardware allocation and information security assurance. As long as the clients
52 are connected with the internet, they get their information easily. Hadoop is an on-going innovation
53 which is utilized as a system for the huge information storage. It is an open source execution of the
54 structure dependent on java. Hadoop is utilized in a substantial bunch or as an open cloud
55 administration. This process is termed as the standard conveyance parallel processing framework
56 [28]. The versatility of Hadoop is evident by its ubiquitious use, yet Hadoop is devoid of effective
57 mechanisms to ward off security breaches of the data stored in HDFS

58

59 As Hadoop provides no inherent security for the information stored in it, numerous methods and
60 approaches for securing the stored HDFS files have been explored by various researchers and
61 practitioners. Among all these efforts, encryption seems to be the most promising answer for
62 securing information in HDFS that is kept in DataNodes as well as for securely exchanging datafrom
63 one DataNode to another DataNode while executing MapReduce jobs. Encryption techniques can
64 considerably reduce the security breaches and data infringement in Hadoop environment. However,
65 the results obtained through various encryption algorithms have demonstrated that the document
66 sizes of the original files can be extended to about one and a half. Further, the uploading as well as
67 the downloading time of a given file can also be increased. Hence, to adress these concerns, the
68 researchers of this study have propositioned a new encryption-decryption algorithm, i.e., the ABHE.
69 As per the simulation results, this technique has shown marked improvements over encryption-
70 decryption time in comparison with the already available algorithms including the Advanced
71 Encryption Standard (AES) and AES with OTP [27].
72 The main contributions of paper are :
73 • To carry out the in-depth study of big data processor, i.e., Hadoop and to assess its strength
74   and weakness in terms of security and privacy.
75 • To propose an ABHE, a secure and efficient algorithm executed on single and two
76   DataNodes in Hadoop. Also, it ensures the full security against all side channel attacks,
77   brute force attack and collusion attack.
78 • To conduct experiments on test data to prove the efficacy of our proposed approach ABHE
79   vs. other secure approches i.e. AES and AES-OTP.
80 • The performance of proposed ABHE has been calculated in terms of File size, Encryption
81   Time, Decryption Time, Throughput and Power Consumption.

82   • The result shows that ABHE improves the execution time (total time taken for encryption
83   and decryption of data)without affecting the size of original file.
84   The rest of the paper has been divided into the following sections:
85   • Section 2- enlists the pertinent research done in the domain of Big Data storage.
86   • Section 3- enunciates the suggested data encryption algorithm formulated on ABHE.
87   • Section 4- presents the integration of the suggested algorithm with Hadoop environment.
88   Furthermore, this section also provides a comparison between the efficacy of the suggested
89   encryption approach vis-a-vis the two already available encryption algorithms namely; AES
90   and AES with OTP with different sizes of text files ranging from MBs to GBs (64MB, 128MB,
91   512MB, and 1GB).
92   • Section 5- underlines the significance of this research study.
93   • Section 6- concludes the study.
94
95   **Related Work**
96
97   **a) Hadoop Security**
98   Hadoop was created in 2008 with the intention to manage only huge amount of data confined to a
99   specific condition. Thus, security issues weren't the topmost preference [5]. For any data storage,
100   Hadoop employs the user's name. In the default node, there is no encryption among Hadoop, the
101   client host as well as the HDFS. All the records are feeded into and constrained by a central server
102   which is known as NameNode. Thus, HDFS lacks in security system against capacity servers.
103   Hence, all information stored in this process is prone to be breached. Besides, a strong security
104   model is also lacking between Hadoop and HDFS. Matter-of-fact is that the correspondence among
105   DataNodes and among the end users and DataNodes remains encoded. It has no validation of
106   clients or administration. Even after Yahoo concentrated on including authentications in 2009,
107   Hadoop still had constrained approved abilities [5]. In 2013, the Apache Software Foundation
108   defined venture Rhino to include security highlights [5].
109
110   Hadoop has the facility of data management that is scalable, rich in features and cost-effective for
111   the masses. It has been a data platform of storing secret information for many organizations. The
112   data stored in slots is saved but once it is brought together and made accessible for organizations
113   over the masses, new security challenges arise. Big data in a Hadoop contains sensitive information
114   related to financial data, corporate information, personal information and many such confidential
115   data of clients, customers and employees. Hence, optimum protection of such data and ensuring
116   that it remains free from any encroachment or tampering is of utmost significance [6-11].
117
118   **b) Big *Data: Hadoop Security Solutions***
119   To elucidate the mentioned problems, a few activities have been appended to Hadoop to keep up
120   with the equivalent [12, 13]:
121   *Perimeter Security:* Network Security firewalls, Apache Knox gateway
122   *Authentication:* Kerberos
123   *Authorization***:** E.g. HDFS permissions, HDFS ACL3s, MR ACLs
124   *Data Protection:* Encryption at rest and encryption in motion. To provide security for data in
125   HDFS, few available mechanisms are:
126

127   ***c)***   **Authentication**
128   Authentication implies user's identification. Authenticators are answerable for gathering
129   testimonials by the API (Application Programming Interface) consumers, authenticating them and
130   publicizing the success or failure status to the clients or chain providers. Because of this primary
131   check, uncertain users won't be able to access the cluster network and trusted network.
132   Identification is regulated by the client host. For strong authentication, Hadoop uses Kerberos [12,
133   13], and LDAP (Lightweight Directory Access), AD (Active Directory) integrated with Kerberos,
134   establishing a single point of truth.
135   Kerberos is a computer grid authentication protocol which generates "tickets" to allow the nodes
136   communicating over an unprotected network to prove their identity to one another. The reliable
137   server authentication key is placed in each node of the array to achieve authenticity of the Hadoop
138   cluster node communication which will develop the HDFS array. It can effectively prevent non-
139   trusted machines posing as internal nodes registered to the NameNode and then process data
140   on HDFS. These components are used throughout the cluster. Hence, from the storage point of
141   view, the legitimacy of the nodes in HDFS cluster could be guaranteed by Kerberos. It is
142   completely entrusted by Hadoop for authentication between the client and server. Hadoop 1.0.0
143   version includes the Kerberos mechanism. Client requests an encrypted token of the
144   authentication agent. A particular service can be requested from the server by using this.
145   Password guessing attacks remains inoperative in Kerberos and thus multipart authentication is
146   not provided [15].
147
148   ***d)***   **Authorization**
149   Authorization or restrict access is the method of securing the access within the data by the users
150   as per the corporate policies or service provider. Authorization provider may also use an ACL
151   (Access Control List) based authorization access called the Knox gateway [12, 13] which is based
152   on the evaluation of rules that comprises username, groups and IP (Internet Protocol) addresses.
153   The aim of Hadoop's developer is to design an authorization plan for the Hadoop platform to
154   manage the authorization structure for Hadoop components.
155
156   **e)   Data Protection**
157   Data protection is a process to protect the data at rest or store and during transmission with the
158   help of encryption and masking [12, 13]. Encryption is a technique which acts as an added layer in
159   security in which data is encrypted (unreadable) during transmission or at rest. Hadoop employs
160   the existing capabilities of data protection by providing the solution for data at rest and data
161   discovery and masking. However, Hadoop's security still needs some improvement. The work that
162   has already been done by the researchers and practitioners on Hadoop is highly commendable.
163   Several research studies have focussed on techniques to improvie the security of the data at rest
164   as well as during transmission. Some of the relevant approaches have been discussed below:
165
166   ***Achieving Secure, Scalable and Fine-grained Data Access Control***
167   The work combines techniques of Attribute-Based Encryption (ABE), proxy re-encryption, and lazy
168   re-encryption [16]. This integrated method accomplishes fine grainedness, scalability, and data
169   confidentiality during data access control in cloud computing. In this work, data files are encrypted
170   using symmetric DEKs (symmetric data encryption key of a data files) and later, encrypted DEKs
171   with KP-ABE (public key cryptography primitive for one-to-many communications). Such a dual
172   encryption technique is called hybrid encryption.

173   The KP-ABE technique is used for basic fuctions like the creation or deletion of files and user
174   allocation with fine-grained data access control mechanism. User allocation is a big issue in this
175   process and to achieve this, the author has combined proxy re-encryption with KP-ABE and
176   distributed some tedious computational tasks to cloud servers. The cloud server stores secret key
177   components and one dummy attribute corresponding to each user. When data owner does some
178   modifications in the set of attributes while user allocation, the proxy re-encryption keys are
179   generated and transferred to cloud servers. Later, cloud servers update their secret key on the
180   basis of new re-encryption keys and re-encrypt the data files accordingly. Due to this, data owner
181   is free from computation load during user allocation and do not need to stay online, since the cloud
182   servers have already taken over this task after having the pre keys. Moreover, the burden of secret
183   key updating and re-encryption of data file tasks are merged as single task using lazy re-encryption
184   technique to save computation overhead of cloud servers during user revocation.
185
186   ***Secure Data Sharing by Using Certificate-Less Proxy Re-encryption Scheme***
187   This study stated that by using a public cloud, data can be shared securely. The research work
188   presented a concept wherein a Certificate-Less Proxy Re-Encryption scheme (CL-PRE) is
189   introduced [21]. According to this concept, an identity based public key is added to the proxy re-
190   encryption technique. This removes the traditional identity problem of key escrow. This scheme
191   requires no certificates for the authenticity of the public key. This scheme (CL-PRE) is used to
192   decrease the figuring and correspondence cost for information proprietor.
193
194   ***Fully Homomorphic Encryption***
195   This research [13] proposed a design of trusted file system by combining the authentication agent
196   technology with the cryptography fully homomorphic encryption technology.This is used for Hadoop
197   which provides reliability and security from data, hardware, users and operations. This enables the
198   user to prevent data breach along with enhanced efficiency of the application which is possible due
199   to the encrypted data in the homomorphic encryption technology. Authentic agent technology also
200   provides a range of techniques which are an integration of different mechanisms such as privilege
201   separation and security audit that provides security of data in Hadoop system.
202   Fully homomorphic encryption technique gives the ability to various users to carry out any operation
203   on encrypted data with same results, provided the nature of the data remains same, i.e., encrypted
204   form throughout the operation. The data remains in encrypted form when processed with map
205   reduce technique and stored safely in HDFS.
206
207   ***A Novel Data Encryption in HDFS***
208   A new method for encrypting a file while uploading in HDFS has been proposed in this research
209   work [17]. The upload process is done along with the encryption process before uploading data on
210   HDFS. In this method [17], the user selects a file to upload and provides a unique secret key for
211   encryption of selected file. In this approach, user can feel the same experience when uploading a
212   normal (without encryption) file to HDFS since the encryption is done in a fair manner. Also, this
213   method utilises the characteristics of read/write operation to reduce the total time in HDFS. As an
214   experiment, the author applied this technique on 32 MB file and observed that the encrypting upload
215   and decrypting download process is usually 1.3 to 1.4 times faster than the conventional method.
216   The major drawback of this approach is the key management because the keys are increased with
217   respect to the users and to deal with them is quite challenging. Additionally, encrypting file sharing

218   issue is also not possible with this approach. This proposed approach is lagging due to these two
219   major issues and needs the dedicated attention of researchers and practitioners.
220
221   ***Secure Hadoop with Encrypted HDFS using AES Encryption/Decryption***
222   Security in Hadoop architecture is proposed in this paper by applying encryption and decryption
223   techniques in HDFS [18]. In Hadoop, it is achieved by adding AES encrypt/decrypt function to
224   Compression Codec. Experiments on Hadoop proved that the computation overhead is reduced by
225   less than 7% when representative MapReduce job is done on encrypted HDFS.
226
227   ***Triple Encryption Scheme for Hadoop-Based Data***
228   Cloud computing has the distinctive ability to provide users with customised, adaptable and
229   trustworthy services at feasible costs. Hence, more and more users are availing of cloud computing.
230   Given the rising demand of cloud appications, the protection of the cloud storage of data has
231   become imperative. A method called novel triple encryption has been introduced in this paper [19]
232   to achieve data protection at cloud storage. The Triple Encryption approach uses DEA (Data
233   Encryption Algorithm) for HDFS files encryption, RSA for the data key encryption and finally IDEA
234   for encrypting the user's RSA private key. In this approach, DES and RSA based hybrid encryption
235   technique is used to encrypt HDFS files and IDEA (International Data Encryption Algorithm) to
236   encrypt the RSA based user key. In Hybrid encryption, DES algorithm is used to encrypt the files
237   and get the Data key. This Data key is again encrypted by using RSA algorithm to provide additional
238   security. The Data key can be decrypted by using private key only, therefore, it is always with the
239   user. This method uses both symmetrical and asymmetrical encryption techniques, so called as
240   hybrid encryption. This approach is tested and implemented in Hadoop based cloud data storage.
241
242   ***Attribute-Group based Data Security Access***
243   Due to various security issues, development and use of cloud storage has been decreased. To gain
244   the confidence of user, author has defined an Attribute Group based data security access scheme
245   to protect the data during network and data sharing features in cloud storage services. In this
246   scheme [20], the data owner has limited user rights and re-encryption on the data node reduces
247   the computational cost along with the management of the clients. It also reduces the complexity of
248   property and rights management. Also, the author uses cipher text CP-ABE encryption algorithm to
249   secure the data at cloud storage. The centralised management of key distribution and Name Node
250   based CP-ABE algorithms have advantages like more transparency for the user and easy
251   managemnt of the user key as compared to the data owner key distribution technique.
252
253   ***Towards a trusted HDFS storage platform***
254   The mechanism for the protection of a Hadoop infrastructure has been explained in this research
255   [22] to deal with the concept of creating a reliable HDFS and safety hazards. Also, the researchers
256   of this paper figure out the relation between security mechanisms and their effect on its performance
257   [22]. In the discussion, the authors implemented trusted computing concepts on a Hadoop
258   considering a threat based environment. This framework is based on the Trusted Platform Module
259   (TPG) and implemented into a base environment. Furthermore, the authors have utilized hardware
260   key protections encryption scheme for Hadoop and AES-NI for accelerating the encryption and
261   compared results after their implementation. In addition, the authors have claimed that there is 16%

262 of the overhead reduction on encryption and 11% overhead reduction while decryption during
263 experiment on simulated 128 MB block data with the AES-NI instructions. This approach provides
264 a concrete layered security design in Hadoop.
265
266 ***Security Framework in G-Hadoop***
267 An approach has been introduced where Hadoop's MapReduce task runs simultaneously on
268 multiple clusters in a Grid environment called G- Hadoop [13, 14]. G-Hadoop reuses user
269 authentication and job submission mechanism of Hadoop in a Grid environment. But initially,
270 Hadoop's user authentication and job submission mechanism have been designed for a single
271 cluster in a non-Grid environment. Therefore, G-Hadoop is an extended version of Hadoop
272 MapReduce task. It established a secure link for user and target cluster by following the secure
273 shell protocol. A single dedicated connection is allotted to each participating user in a cluster and
274 each user has to log on to only those clusters for which they are authenticated. Unlike Hadoop, they
275 have to design a new security framework for G-Hadoop with various security solutions like public
276 key cryptography and Globus Security Infrastructure.
277
278 Concepts of proxy identity, user interface, and user instance, are embedded in this security
279 framework to give better functions in a Grid environment [13, 14]. This security framework
280 introduced a single-sign on approach during user authentication and job submission process of the
281 G-Hadoop. Also, this security approach protects the G-Hadoop system during threat environment,
282 i.e., traditional attacks, abusing and misusing. The model of security framework is based on Globus
283 Security Infrastructure (GSI). The utilization of SSL protocol for communication between the master
284 node and the CA (Certification Authority) is also the key factor in security. GSI is based on single
285 sign on process and uses asymmetric cryptography to provide a secure communication. GSI is a
286 standard grid security which adapts to various techniques to provide necessary requirements in a
287 Grid environment.This includes authentication, integrity of the messages and delegation of the
288 authority from one entity to another in a grid environment. The user can only log-in into the master
289 node after providing his authentication in the form of user name and password and submit jobs to
290 the cluster. SSL handshaking is used in the security framework to establish a secure connection
291 between DataNode and a NameNode.
292
293 ***Elliptic Curve Cryptography based Security Scheme for Hadoop distributed file system***
294 This paper [3] introduces a token based authentication scheme to protect HDFS stored data from
295 security threats like breach and impersonation attacks. In this scheme, HDFS client authentication
296 is done by the Data Node through block access token and functions as an extra layer of security
297 along with the existing security, i.e., symmetric key HDFS authentication. Also, ECC encryption
298 method is used for for authentication of anonymous keys and provides protection against external
299 threats like security breaches or accidental exposures. This scheme adopts the hash chain of keys
300 approach instead of a public key exchange approach which is a very common HDFS authentication
301 protocol. Apart from providing protection to the sensitive HDFS data, it also improves the
302 performance as compared to the public key-based HDFS authentication protocols.
303
304
305

306 ***Secure Multi sharing in Big Data Storage***
307 A method of privacy preserving security by using different mechanisms, i.e., anonymity, multiple
308 receiver and conditional sharing is explained in this paper [24]. In this approach, to get the maximum
309 security, Advanced Encryption Standard (AES) with Message Digest (MD5) & Data Encryption
310 Standard (DES) have been employed to encrypt the data and authentication of data has been done
311 using the DSA. Also, security and privacy preserving approaches have been used for the big data
312 processing in the proposed framework. In this approach, owner uploads the data in cloud storage
313 and after encryption, data is stored in HDFS. Thereafter, the data is shared among the multiple
314 receivers. Cipher text is used to hide the identity of the sender and receiver whereas Anonymization
315 mechanism is used to hide information of a particular receiver. A mechanism based on user and
316 their received data category called conditional sharing starts working after receiving the receiver's
317 details. And, if the user's category is matched with receiver's data category, then the receiver gets
318 authenticated and the transmission is started. Once the conditional sharing is complete, receiver
319 retrieves the cipher text. The big data is shared with the cloud only if the result is secured. This
320 proposed algorithm is verified for small data sets only.
321
322 ***Towards Best Data Security***
323 In this paper, the author has described about the enormous information and its safety issues [25].
324 Also, he has described about the existing ways to improve the security of enormous information like
325 security hardening methodology with attributes relation graph, attribute selection methodology,
326 content based access control model and a scalable multidimensional anonymization approach. The
327 author of this paper [25] has proposed an intelligent security approach based on real time data
328 collection and threat analytics to detect the threat before the security breach takes place.
329
330 ***HDFS Data Encryption based on ARIA Algorithm***
331 In this paper [2], the author has presented an encryption scheme based on South Korea's ARIA
332 encryption scheme to protect the HDFS data in Hadoop. The ARIA algorithm uses 128-bit block for
333 data encryption. In this approach, variable length data (not necessarily the 128-bit data) is divided
334 into HDFS blocks. The proposed ARIA based algorithm provides the same level of data security at
335 cost of only 23 % performance degradation (during query processing) compared to AES algorithm.
336 In addition, the researchers explained the future of ARIA based encryption scheme in genuine word
337 applications like area based administrations and financial related data handling.
338
339 ***Chaos-Based Simultaneous Compression and Encryption for Hadoop***
340 This paper [26] introduced a framework based on a masking pseudorandom key stream to increase
341 the encryption quality and provide robust encryption security & compression during read and write
342 operation when integrated in HDFS. Also, the researchers have proposed a scheme for Hadoop
343 using simultaneous compression and encryption to solve the implementation issues. The
344 enhancement consequently improves the speed and efficiency of the algorithm. The proposed
345 algorithm is highly compatible with Hadoop and provides efficient encryption security and
346 compression during storage of data. Various experimental results concluded that the performance
347 of the cluster in Hadoop gets reduced when compression and encryption operations are done
348 separately because they need a significant volume of data for both the operations. This proposed
349 algorithm can compress and encrypt the data simultaneously during MapReduce which reduces

350 the required data space with minimum network resources. The proposed algorithm has passed
351 edits security analysis test with a 99% confidence interval. Further, all NIST SP800-22 assays are
352 successfully passed on cipher text generated from the plaintext.
353

354 ***Data Encryption Based on AES and OTP***
355 This research paper [27] has highlighted a method to improve the upload and download time with
356 reduction of encrypted file size by AES and OTP algorithms in HDFS. The authors performed
357 encryption and decryption by two different ways which are based on AES and AES-OTP algorithms.
358 The researchers chose cipher block chaining with the ECB mode of AES algorithm for handling
359 HDFS blocks and OTP algorithm is used as a stream cipher. This keeps length of the plaintext
360 same. For decryption, a private key is required which is always in the custody of user. In this
361 method, when client has mentioned to transfer a record to HDFS, the application server creates an
362 arbitrary key which is then separated into 2 keys for doing multi encoding and unscrambling by
363 utilizing AES-OTP algorithm.Moreover, the authors have compared the file encryption time among
364 Generic HDFS, encrypted HDFS by AES and HDFS encrypted file by AES with OTP. The results
365 show that the AES with OTP algorithm increased the encrypted file size by 20% of the original file.
366 The researchers also executed parallel decryption processing in Map Task to improve performance.
367

368 ***Two-layer Selective Encryption using Key Derivation Method***
369 In this paper [44], the authors have explained the use of two-layer selective encryption technique
370 based on key derivation method to implement the authorization policy [45]. In this method, the user
371 assigns a secret key corresponding to each file which is encrypted using a symmetric key. The
372 owner creates public tokens by using his secret key to allow any user further. Later, these public
373 tokens along with token distribution task are transferred to the semi-trusted server. To derive the
374 decryption key for a file, a minimal number of secret key per user and a minimal number of
375 encryption key are required since the server cannot derive decryption key of any file with the
376 available public tokens. The file creation and user grant/revocation operation gets complex as the
377 number of users increases. This makes the suggested method unscalable [46]. Also, the user
378 access privilege accountability is not supported in this method.
379

380 ***Security and Privacy Aspects in MapReduce on Clouds***
381 Hadoop uses the filters in Vigiles [48] for a fine grained access control framework. These filters are
382 coded in Java by security administrators and handled authorization by means of per-user
383 assignment lists. On the other hand, in GuardMR, filters are allocated with limited roles on the basis
384 of subject and a formal specification approach for the definition of filters is proposed. GuardMR and
385 Vigiles rely on platform specific features for regulating the execution of a MapReduce task such as
386 the Hadoop APIs and the Hadoop control flow and do not need the Hadoop source code
387 customization. Vigiles and GuardMR have observed apractically low implementation overhead
388 which means that they do not provide any support for context aware access control policies [49]. In
389 [10] authors considered security and privacy challenges and urgent requirements in the scope of
390 MapReduce and reviewed the existing security and privacy protocols for MapReduce including
391 AccountableMR and TrustMR. The study also provides a comparison of several security algorithms,
392 protocols and frameworks for MapReduce framework.
393

394 ***Hybrid Storage Architecture and Efficient MapReduce Processing for Unstructured Data***
395 In this paper [50], a technique called Hybrid Storage Architecture is proposed. With this technique,
396 different kinds of data stores are integrated to the model and it also enables the strorage and
397 process of the unstructured data.To execute MapReduce-based batch-processing jobs, various
398 partitioning techniques are applied which are based on the said technique. The paper also
399 demonstrates the utilization of the characteristics of different data stores for building a smart and
400 an efficient system. The partitioning techniques leverages the unified storage system thus reducing
401 the I/O cost and improves the large-scale data processing efficiency marginally.
402
403 ***Towards Privacy for MapReduce on Hybrid Clouds using Information Dispersal Algorithm***
404 In [51], to ensure privacy for MR in a hybrid computing environment based on the Grid'5000
405 platform, an algorithm known as information dispersal algorithm is required which comprises both
406 untrusted infrastructures (such as, desktop grids and public clouds) and trusted infrastructures
407 (such as, private clouds).
408
409 ***SEMROD: Secure and Efficient MapReduce over Hybrid Clouds***
410 SEMROD [52] firstly segregate the data into sensitive and non-sensitive data groups and then send
411 the non-sensitive data to public clouds. Private and public clouds execute the map phase. However,
412 the private cloud pulls all the outputs includng outputs of the map phase containing sensitive keys.
413 Also, it executes the reduce phase operation only on record associated with sensitive keys
414 andignores the non-sensitive keys. On the other hand, a public cloud execute the reduce phase on
415 all outputs without knowing the sensitive keys. Finally, a sensitive key is generated by removing the
416 duplicate entries with the help of filtering step.
417
418 ***MtMR: Ensuring MapReduce Computation Integrity with Merkle Tree-based Verification***
419 Proposed MtMR [53] is a method based on Merkle tree based verification to ensure the high integrity
420 of the MapReduce tasks. It performs two rounds of Merkle tree based verification for the pre-
421 reduction and restoration phases and covers MapReduce in a hybrid cloud environment. In each
422 round, MtMR samples a small portion of reduces task input/output records on the private cloud and
423 then applies the Merkle tree-based verification. The authors believe that MtMr can significantly
424 improve the results while producing moderate performance overhead.
425
426 ***Security Threats to Hadoop: Data Leakage Attacks and Investigation***
427 This article [54] presents an automatic analysis method to find any data leakage attacks in Hadoop.
428 It also presents a forensic framework including an on-demand data collection method in which it
429 collects data from the machines in the Hadoop cluster on the forensic server and then analyzes the
430 same. It can detect suspicious data leakage behaviors and give warnings and evidence to users
431 using its automatic detection algorithm. And, collected evidences can help to find out the attackers
432 and reconstruct the attack scenarios. The authors of the paper have also talked about the security
433 concerns of HDFS (or Hadoop) and presented some possible data leakage attacks in it.
434
435 ***VC3 and M2R in MapReduce Computation***
436 VC3 [55] uses SGX to achieve confidentiality and integrity as part of the MapReduce
437 programming model and requires a trusted hardware to perform computation. VC3 is not allowed

438   to perform system calls but works and follows the executor interface of Hadoop. On the other
439   hand, M2R [56] offer mechanisms for dropping network traffic analysis leakage for MapReduce
440   jobs.
441
442   ***Preserving Secret Shared Computations using Mapreduce***
443   The main reason of cloud insecurity is the loss of control over the data which can cause serious
444   harm to the confidentiality of customer using cloud-computing. This problem can be overcome by
445   providing secure computing environment and data storage [58]. Also, techniques like encrypted
446   representation and secret sharing techniques have emerged that offer verified security along with
447   relatively efficient processing but are limited to only computing selection queries [57].
448
449   ***Privacy Preservation Techniques in Big Data Analytics***
450   In this paper [1], authors have described about the various privacy threats and preservation
451   techniques and models along with their limitations. The authors also proposed a Data lake based
452   method for privacy preservation of unstructured data. Data lake is a repository to store raw format
453   of the data either structured or unstructured, coming from different sources. Apache Flume is used
454   for data ingestion from different sources and for their processing; data is transformed to HIVE
455   tables. Also, Hadoop MapReduce using machine learning or vertically distributed can be applied to
456   classify sensitive attributes of data whereas tokenization is used to map the vertically distributed
457   data.
458
459   **f)   Major Findings from the Literature**
460   After a cautious and focused study of various methodologies/approaches on big data and Hadoop
461   security, the following observations have been made:
462   •   Hadoop stores the data on multiple nodes in a distributed manner while metadata and edit logs
463   are stored on name nodes. The communication of data happens between client node and data
464   node. Hence, multiple nodes are used in the Hadoop framework. The data is vulnerable to the
465   hacks during the transmission, as it is not encrypted by default. Various communication
466   protocols are used for internode communication. The available approaches or solutions for
467   securing the data transmission include Kerberos, Simple Authentication and Security Layer
468   (SASL), etc. However, these traditional approaches are not effective and sufficient enough to
469   secure big data.
470   •   Data that is stored in fixed storage is known as data at rest or at storage level. Initially the
471   stored data is prone to security attacks being not encrypted. Since, Hadoop works on the
472   principal of spreading data across multiple nodes, consequently it is exposed to all insecure
473   entry points. There are numerous methods available for data encryption in Hadoop. As Hadoop
474   deals with large volume of data, it takes time in the encryption/decryption process. In order to
475   maintain the performance, it is important to use an encryption technique that is fast enough to
476   encrypt/decrypt. According to the studies, the encrypted data increases in the size almost by
477   one and half time of the original data so the file upload time also gets affected.
478   •   Cloud providers need to design a cost-effective infrastructure that understands customers'
479   needs at all levels. To meet the requirements, it is needed to share the storage devices
480   amongst the multiple users, which is known as multi-tendency. But sharing of resources results
481   in security vulnerability. If proper security measures are not implemented, then the attacker is
482   able to get easy access to the customer's data, more so in the case of using the same physical
483   device.

484   •   Companies would never know if the data is being used by someone else or not, because they
485   don't have direct control over their data. The lack of resource monitoring mechanisms creates
486   many security issues.
487   •   Customers have to rely upon trust mechanism as an alternate security measure in which they
488   have to control data and their resources. Cloud providers also provide certificates of operations
489   of a secure provider to the customers. The certificates are well authenticated with established
490   standards.
491   •   The security capabilities which are for "non big data" are needed for big data also to ensure
492   client verification, management of data masking and encryption.
493

Materials & Methods

496 **Data Encryption based on Attribute Based Honey Encryption (ABHE)**

497 In Hadoop, the inherent security feature is simple file permission and access control mechanisms.
498 In such context, encryption is the best technology applied for securing HDFS files that are stored
499 in DataNodes. Further, while processing MapReduce transferring files among DataNodes,
500 encryption is the best solution. We can use cryptography for data protection in Hadoop, solution to
501 data confidentiality and data integrity can be achieved using encryption technique. Cryptography
502 keys can be categorised into: secret key cryptography and public key cryptography. Public key is
503 known as asymmetric key cryptography [33] while secret key is symmetric secret key cryptography
504 which is used in stream ciphers for generation of password based encryption [34].

505 Encryption is mainly used to ensure secrecy. Encryption actually means secret writing which was
506 initially used by ancient humans desiring to store secrets. In the past, encryption was available only
507 to Generals and Emperors, but today it is used by nearly everyone, every day, every time whenever
508 a credit card transaction, data storage and node to node communication is done, phone call is
509 made, secure website is used; encryption techniques are used. Efficacy of an encryption algorithm
510 depends on the key length [29]. However, the available encryption algorithms are considered to be
511 secure. But depending on the time and computing power, they are also susceptible to intrusions
512 [30]. The present encryption techniques are also beset with vulnerabilities, for instance, when
513 decrypting with a wrongly guessed key, they yield an invalid looking plaintext message, while
514 decrypting with the right key, they give a valid-looking plaintext message, confirming that the cipher-
515 text message is correctly decrypted [30].

516 In the same row, the honey encryption has been proposed by Jules and Ristenpart [31]. It is a
517 concept which addresses vulnerability discussed in the previous paragraph and makes the
518 password based encryption (PBE) more difficult to be broken by brute-force. Traditional encryption
519 methods would show random text with no meaning at all when decrypting is done with wrong key
520 and hence confirming its invalidity. On the contrary, honey encryption shows plausible looking text
521 even when the key is wrong so the attacker won't know if the guessed key is the right one. This
522 unique approach slows down the attacker by fooling him and increases the complexity of password
523 guessing as well as cracking process. There are few other technologies that share same term
524 "honey". For example, Honeywords [31] are a password that are used as decoy and generates an
525 alert and notifies the administrator if used. Honeypots [37], Honeynet [38], and Honeyfarm [39] are
526 some other examples of luring systems. Honey encryption is related to Format-Preserving-

527 Encryption (FPE) [40] and Format-Transforming-Encryption (FTE) [33]. In the FPE, both the
528 plaintext and cipher-text message are same whereas it is not the case in FTE.
529 In Honey Encryption, the messages are stored to a seed range in the seed space. Seed space and
530 message space are different, so the cipher-text message space is different from the message
531 space. Vinayak and Nahala [34] used the HE scheme in MANETs to secure Ad-hoc networks
532 against brute force attacks. Tyagi et al. [41] applied HE technique to protect simplified text
533 messages and credit card number that are susceptible to brute force attacks. Hoyul Choi et al. [42]
534 proposed schemes to solve human typo problems with message recovery security. Legitimate user
535 may get confused seeing the different result than expected if there was some mistake in typing the
536 password correctly. Edwin Mok et al. [43] came up with an eXtended Honey Encryption (XHE) by
537 adding additional security measures on the encrypted data. However, Honey Encryption is still
538 difficult to be applied in certain applications. For example, if the attacker has some clue about the
539 data which is encrypted, suppose he has a part of the original data, he can easily tell which result
540 is bogus and which is the correct data by matching the data with the decrypted result. However, it
541 is possible to brute force honey encryption if the attacker has crib that must match with it to confirm
542 its legitimacy [32]. It is still vulnerable and susceptible and further researches are going on. To
543 overcome its limitations it must be expanded further by bringing out new security methods.
544 This persuades the authors of this paper to develop an in-depth understanding of data security and
545 privacy to solve issues related to Honey encryption. This paper aims to focus on fixing the
546 vulnerabilities in Honey encryption and making it more secure. The authors have designed and
547 implemented the attributed based Honey encryption as an extension of the public key encryption.
548 This would enable the users to encrypt and decrypt messages based on users' attributes. Only if
549 the user matches the predefined attributes will the user be able to decrypt the message. It will help
550 to keep the attacker away by blacklisting them.
551
552 **Proposed Encryption Algorithm**
553 The proposed encryption algorithm is a more secure version of honey encryption. The encryption
554 algorithm provides two tier securities so that it can overcome the limitations prevailing in existing
555 encryption techniques. The proposed algorithm is termed as Attribute-Based Honey Encryption
556 (ABHE). Its 128/256 bits encryption algorithm will perform two layers of encryption in order to
557 enhance security and effectiveness. The use of Cipher text Policy- Attribute Based Encryption (CP-
558 ABE) [35, 36] has been proposed in the algorithm. In the algorithm user's private-key is
559 superimposed with an access policy over a set of attributes within the system. A user can decrypt
560 a cipher text only if his attributes satisfy the set of rules of the respective cipher-text.
561 Firstly, a set of attributes are chosen from the file to be encrypted; then a set of rules/policies are
562 created for these attributes. On the basis of these rules, the given file is encrypted. Further, for more
563 security the encrypted file is again protected by password. As this password is based on honey
564 encryption, it creates a set of honey words. The encrypted file is passed on and may be received
565 by different users. Now according to the proposed algorithm, only the user having the desired set
566 of attributes or the password would be able to decrypt the data. If someone wants to decrypt the
567 encrypted file, he/she will have to enter the correct password. If password does not match, the user
568 will be treated as intruder and previously set honey words will be displayed to him. If the password
569 matches, the genuine user has to enter the private key which has been already created while
570 encrypting the file. Again, if private key does not match, the user will not be allowed to access the

571 file. On matching, the user will be able to successfully decrypt the file. The overall process will
572 provide better security for files.
573
574
575 **ABHE Algorithm for Data Security**

Input:	Plain Text file
Output:	Encrypted file
Step 01:	**Generate Private Key**
Step 01.a:	Set of attributes is specified that describe the key.
Step 01.b:	Output private key 'q'
Step 02:	**Encryption:**(The algorithm encrypts File 'F' with policy 'P' and outputs the cipher-text)
Step 02.a	Selects the file to be encrypted and set of attributes.
Step 02.b:	Encrypt a file F using a set of attributes occurring in the policy 'P'
Step 03.c:	Generate cipher-text CT
Step03:	**Encrypted file (in step-02) is protected again by the password**
Step 04:	**Generate honey words and present it to user.**
Step 05:	**Decryption:** (Decryption algorithm gets as input an encrypted file which is protected by the password. Cipher-text CT is produced by the encrypted algorithm, an access policy 'P' under which CT was encrypted.)
Step 05.a:	Input is encrypted file
Step 05.b:	Enter the password; if password matches, the cipher text CT is decrypted, otherwise intruder is detected.
Step 05.c:	User applies y number of attributes to compute private key
Step 05.d:	If key matches, file is decrypted and output the corresponding original file 'F', otherwise it outputs NULL.

576
577 Authors have introduced a method to enhance the security level in which the data encryption and
578 key management server are put together and provided a unique key for each application or cluster
579 with HDFS encryption. When HDFS encrypted file entering into the Apache Hadoop environment,
580 it remains in encrypted form at storage after processing. The results including intermediate results
581 are always stored in encrypted form within the cluster in a file system having non HDFS form. At
582 client level, data has been divided into smaller chunks by using parallel computing technique and
583 stored at HDFS in encrypted form. Also, the Map Reduce tasks can be done on encrypted data
584 directly and decrypt before processing after importing the corresponding decryption library. Input to
585 a MapReduce job is divided into fixed-size pieces. These are the input splits which is a chunk of
586 the input that is consumed by a single map. At Map function, the input data is processed in
587 decrypted form and stored output data in encrypted form into the HDFS. The Reduce function is
588 executed on the intermediate results of Map function after decryption and the stored final output
589 data in again encrypted form into the HDFS and provided access to the authorized clients only.
590 Decryption process is replica of encryption process and both these methods are simple, cost-
591 effective and scalable without deteriorating the performance, scalability or functionality. So, they
592 are easy to recommend and effectively address the security deficiencies with big data clusters.
593
594 Evaluation of performance of the proposed algorithm has also been done. The performance
595 parameter includes encryption–decryption time (rate of encryption is given by encryption time and

596 rate of decryption is given by decryption time), throughput of encryption-decryption where
597 throughput is calculated as the size of plain text (in MB or GB) is divided by the total time during
598 encryption-decryption. The speed and power conumption of encryption-decryption process are
599 mainly dependent on the throughput of the encryption-decryption scheme, as it defines the speed
600 of encryption-decryption. In case of encryption-decryption, as the throughput increases, power
601 consumption decreases. Also, authors have compared results with the exsisting HDFS encryption
602 algorithms namely AES and AES-OTP with different file sizes (varies from MB to GB). The
603 performance parameters results have shown that the proposed ABHE scheme with Hadoop
604 environment is a considerable improvement over AES, AES with OTP (Integrating with Hadoop).
605 Also, the proposed algorithm provides the security for data stored at HDFS and distributed
606 computing against all side channel attacks, brute force attacks and collusion attacks. Detailed
607 description is given in the next section.
608

609 ## Results

610

611 **Implementation**
612 AES is propositioned to be better than the other secure approaches that address the secure data
613 processing using Hadoop HDFS and MapReduce job in context of data encryption. To support this
614 claim, the performance of proposed ABHE algorithm has been evaluated in this section and
615 performance of the proposed ABHE algorithm has been compared with the existing algorithms, i.e.,
616 AES and AES-OTP, while doing the same experiment in a standard Hadoop setup. The
617 performance is evaluated in terms of throughput and power ponsumption by doing the encryption-
618 decryption techniques on different sizes of files (size varies from MB to GB).
619

620 **Implementation Environment**
621 The implementations and experiments are based on Hadoop cluster. The Hadoop cluster consists
622 of one host which runs on laptop with Intel core i3-2330M processor 2.20GHz with Turbo Boost
623 upto 2.93GHz and 4GB RAM. In this, one of the host is tagged as NameNode and other is used as
624 a DataNode. NameNode playsa role of centre point in cluster and all information about the stored
625 data is saved on it. For simplicity, there is only one NameNode and one DataNode in a Hadoop
626 cluster with one run. DataNode provides the physical space for storing data. The operating system
627 of the host is Linux with CentOS 6.4, Ubuntu-14.04 (GNU/ Linux 3.13.0-24-generic x86-64). On top
628 of the operating system is Hadoop with version 2.7.3. Single node architecture of Hadoop is used
629 in which all components of the Hadoop are on the same node. Implementation and stand-alone
630 applications are written in Java. As Hadoop requires JDK for its working so, Java Open Java
631 Development Kit (JDK) is installed on the system using the <apt-get>command. The running
632 component of the Hadoop can be checked using the <jps> command. HDFS distribution process
633 overcomes outages, acts as a backup while maximising the availability of the services.
634

635 ***Results of the Experiment***
636

637 In this section, we present the results and analysis of our proposed algorithm versus the available
638 securing approaches. In the proposed encryption technique, at first, we apply the attribute based
639 encryption which is based on cipher text policy based attribute encryption. The proposed approach

640   uses a specific type of encrypted access control wherein user's private-key is super imposed with
641   an access policy over a set of attributes within the system. A user can decrypt a cipher text only if
642   his attributes satisfy the set of rules of the respective cipher-text. An enhance security to ensure full
643   safety against all side channel & brute force attack, the proposed algorithm is combined with Honey
644   encryption algorithm. The combination of these two algorithms, i.e., the ABHE provides a stronger
645   security against confidentially & brute force attack and all side channel as well as collusion attacks
646   as encryption is not easy to break and get the actual data.
647   The performance of ABHE has been calculated in terms of file size, encryption time, decryption
648   time and power consumption and compared with two existing encryption algorithms namely; AES
649   and AES with OTP applying on different sizes of text files. The working of the proposed algorithm
650   has been demonstrated in figure 1 and figure 2.
651

652   **Figure 1: Working of the Proposed ABHE encryption Algorithm when Attribute and**
653   **Password are entered for 64 MB file size. (For both encryption and decryption)**
654

655   **Figure 2: Working of Proposed ABHE Encryption Algorithm when Right and Wrong**
656   **Password are entered for 64 MB File Size.**
657   *File Size*
658   The proposed encryption algorithm reduces the encryption-decryption time without affecting the
659   size of original file. Here, the file named 2048MB.txt is of size 1.12GB and contains 9 blocks
660   starting from 0 to 8. The size of block 0 to block 7 is remains unchanged after encryption while
661   size of block 8 is changed from 125449781 bytes to 125449792 byte which is insignificant and
662   shown in figure 3 to figure 6. Also, the output file size when encrypted a file of size 1 GB using
663   AES, AES-OTP and the proposed approach is compared which is shown in table 1.
664

665   **Figure 3: Size Block one before Encryption**
666

667   **Figure 4: Size of Block one after Encryption**
668

669   **Figure 5: Size of Block eight before Encryption**
670

671   **Figure 6: Size of Block eight after Encryption**
672

673   **Table 1: Data Size Comparison among AES and AES with OTP algorithms and the**
674   **Proposed ABHE Algorithm**
675

676

677   *Encryption Time using ABHE*
678   This is the time taken by encryption algorithm to produce a cipher-text from plain-text. Encryption
679   is performed while writing the file on Hadoop so that the stored data can be saved from various
680   attacks. This process involves a number of steps which has been shown as follows:
681   i   HDFS client interacts with NameNode by calling the create() function on Distributed file
682   system (DFS).
683   ii   DFS sends a request to NameNode to create a new file.

684   iii   NameNodes provide address of the DataNode, i.e., Slave which is based on the availability
685   of space and capacity in DataNode on which HDFS client is going to start writing encrypted
686   data.
687   iv   The HDFS client starts entering the attributes to encrypt the file. After that for more security,
688   it applies the password which is based on Honey encryption. Now the HDFS client starts
689   writing data though FS Data OutputStream to specific slave for a specified block.
690   v   The slave starts copying the block to another slave when HDFS client has finished writing the
691   blocks.
692   vi   During the block copying and replication process, metadata of the file is updated in the
693   NameNode by Datanode. (DataNode provides the periodically heartbeat signal to the
694   NameNode).
695   vii   After the successful completion of write operation,DataNode sends the acknowledgement to
696   HDFS client through DFS.
697   viii   After that HDFS client closes the process.
698   ix   Write operation is closed after receiving the acknowledgement from HDFS client.

699   The complete operation (with above steps i.e. i, ii, iii…) is explained in figure 7. As shown in table
700   1, it took 12.9751 minutes for the encrypted HDFS using AES algorithm, whereas it took 11.2511
701   minutes for the encrypted HDFS using AES with OTP algorithm. On the other hand, the proposed
702   approach took only 6.08 minutes to encrypt 1GB file in HDFS as shown in table 2.
703

704   **Figure 7: Writing a File with Encryption in HDFS**
705

706   **Table 2: File Encryption Performance Comparison among AES and AES with OTP**
707   **Algorithms and the Proposed ABHE Algorithm**
708

709   ***Data Decryption Time using ABHE***
710   It is the time taken by decryption approach ABHE to produce the plain-text from cipher-text. With
711   our proposed cryptographic scheme, whenever a node will try to read a file on HDFS it will first have
712   to decrypt the file. Then only it will be allowed to perform reads operation. This has been done in
713   the proposed approach to filter out the intruders or unauthorized access. Following is the step-by-
714   step process on how Read operation is performed in HDFS with the proposed approach:
715   i   First of all HDFS client interacts with NameNode by calling the read function on Distributed
716   File System (DFS).
717   ii   DFS sends a request to NameNode for reading a file.
718   iii   NameNode provides address of the DataNode, i.e., Slave on which HDFS client will start
719   reading the data.
720   iv   For HDFS client to start reading data through FS Data InputStream from specified slave and
721   from a specified block, firstly it has to enter the correct password. If password does not match,
722   the user will be treated as an intruder. If the password matches, the genuine user has to enter
723   the private key which has already been created while encrypting the file. Again, if private key
724   does not match, the user will not be allowed to access the file. On matching, the user will be
725   able to successfully decrypt the file.
726   v   After a successful completion of read operation, HDFS client terminates read operation.
727   vi   Read operation is closed after receiving the acknowledgement from HDFS client.
728   vii   As it has been shown in step 4, the proposed approach provides dual layer security to the
729   data stored in HDFS. Step-wise demonstration of decryption operation is shown in figure 8.

730 When using the proposed approach for decryption of 1GB file on Mapper job, it took 6.73 minutes.
731 On the other hand, the existing algorithms, i.e., AES and AES with OTP respectively to 14.0841
732 and 12.2115, respectively for decrypting 1 GB file as shown in table 3.
733
734 **Figure 8: Reading a File with Decryption in HDFS**
735
736 **Table 3: File Decryption Performance Comparison among AES and AES with OTP**
737 **Algorithm and the Proposed ABHE Algorithm**
738
739 The values for each criterion was logged and graphically plotted to represent the results as shown
740 in figure 9 and figure 10. Further, these figures show the comparative time taken (in minutes)
741 during the encryption and decryption process by different algorithms i.e. AES, AES with OTP and
742 Proposed Algorithm (ABHE). From figure 9 and figure 10, it is clear that the proposed algorithm
743 is taking less time for encryption and decryption as compared to other existing algorithms in
744 Hadoop environment.
745
746 **Figure 9: Encryption Time (minutes) of AES, AES with OTP and Proposed ABHE**
747 **Algorithm**
748
749 **Figure 10: Decryption Time (minutes) of AES, AES with OTP and Proposed ABHE**
750 **Algorithm**
751
752 When the proposed ABHE algorithm is integrated with Hadoop, it showed better performance
753 than the previously available cryptographic algorithm. From the results of table 1, table 2 and
754 table 3, it is clear that:
755
756 • The proposed algorithm ABHE is taking less time to encrypt and decrypt text files than the
757   AES, AES with OTP algorithms.
758 • The throughput of ABHE is very high as compared to the AES, AES with OTP algorithms.
759 • As the throughput increases, the power consumption decreases, hence the power
760   consumption of ABHE is low than that of the AES, AES with OTP.
761
762 Furthermore, for analyzing the performance of the proposed encryption technique with sharing
763 data between two different DataNodes in Hadoop environment, the same has been simulated
764 with random text file size 712 MB (in terms of block size) before and after encryption shown in
765 figure 11 and figure 12. Also, the Browse directory show that encrypted file and abc.txt non-
766 encrypted file HDFS in figure 13.
767
768 **Figure 11: Size of Block five before Encryption**
769
770 **Figure 12: Size of Block five after Encryption**
771
772 **Figure 13: Browse directory show that encrypted file and abc.txt non-encrypted file**
773 **HDFS**
774
775

Discussion

776

777 The proposed study has been able to successfully solve the weaknesses present in the security
778 approaches available for big data security. The significance of the proposed work is as follows:

779 • Proposed encryption technique which uses the concept of Attributes Based Honey Encryption
780   (ABHE) may help to securing sensitive information stored at HDFS in insecure environment
781   such as the internet and cloud storages.
782 • Proposed technique provides both HDFS and Map Reduce computation in the Hadoop as
783   well as cloud environment to secure and preserve the integrity of data during execution or at
784   rest. Therefore, we have directed our efforts in securing the data transfer and computation
785   paradigm in Hadoop environment by using chipper text policy attributes based honey
786   encryption and Honey encryption for secret share of tuple of data and sent them to the cloud
787   in a secure manner.
788 • The chipper text policy attributes based encryption makes the application secure and has a
789   high performance when compared with the rest of the encryption techniques. Also, it provides
790   the secure data transfer to all cloud applications.
791 • In the proposed algorithm, we have assured the data security by using simplified chipper text
792   policy attribute based encryption with Honey encryption which is difficult to decrypt by any
793   unauthorized access.
794 • The user authorization access is based on the user define policy which reflects the overall
795   organizational structure and also, depends upon a set of attributes within the system.
796 • With the proposed algorithm, the security of data is not only dependent on the secrecy of
797   encryption algorithm but also on the security of the key. This provides dual layer security for
798   the data.

Conclusion

800

801 In this proposed approach, we mainly concentrated on protection of big data stored in HDFS by
802 integrating the proposed ABHE algorithm with Hadoop key management server. In a nutshell, for
803 ensuring data security in Hadoop environment through the proposed encryption technique, HDFS
804 files are encrypted by using attribute based honey encryption through the proposed ABHE
805 algorithm. For evaluating the suggested technique, we carried out some experiments using two
806 data nodes. Our objective was to experiment and gauge the effectiveness of ABHE algorithm. For
807 accuracy in sharing secret key, data sharing between different clients and speed with which each
808 file stored in HDFS. As the proposed ABHE algorithm, execution time (a function of encryption
809 time) is less as compared to the other available approaches. This proves that the proposed
810 technique is fast enough to secure the data without adding delay. Also, the proposed ABHE
811 algorithm has a higher throughput which proves its applicability on big data. It provides a feasible
812 solution for secure communication between one DataNode to other DataNode. The proposed
813 encryption technique does not increase the file size therefore it saves the memory and bandwidth,
814 and hence reduces traffic in a network. Also, it has an ability to encrypt structured as well as
815 unstructured data under a single platform. Only HDFS client can encrypt or decrypt data with
816 accurate attributes and password. The Proposed technique provides a dual layer security for all
817 DataNode as data is not confined to a specific device and clients can access the system and data
818 from anywhere. This encryption approach may be reckoned as a premise for visualizing and
819 designing even more robust approaches to ensure optimum security of big data.

820

821

References

1) P. R. Mohan Rao, S. M. Krishna, and A. P. S. Kumar, "Privacy preservation techniques in big data analytics: a survey", Journal of Big Data, 2018, DOI:https://doi.org/10.1186/s40537-018-0141-8.

2) Y. Song, Y. S. Shin, M. Jang, J. W. Chang, "Design and implementation of HDFS data encryption scheme using ARIA algorithm on Hadoop",   IEEE International Conference on Big Data and Smart Computing (BigComp), 2017, doi:10.1109/BIGCOMP.2017.7881720.

3) Y. S. Jeong, and Y. T. Kim, "A Token-Based Authentication Security Scheme for Hadoop Distributed File System Using Elliptic Curve Cryptography", Journal of Computer Virology and Hacking Techniques, vol. 11, 2015, pp. 137-142.

4) K. Shobha, and S. Nickolas, "Time domain attribute based encryption for big data access control in cloud environment", ACCENTS Transactions on Information Security, vol. 2, no. 7 2017, doi: http://dx.doi.org/10.19101/TIS.2017.27003.

5) C. Yalla, A. Gill, M. Gupta, H. Mohankumar, T. McCloskey, L. Minas, N. Ngo, S. Tolentino, and D. Watson, "Big Data: Security Intel IT's Apache Hadoop Platform", white paper, Intel, 2016.

6) D. S. Rerzi, R. Terzi and S. Sagiroglu, "A Survey on Security and Privacy Issues in Big Data", 10th International Conference for Internet Technology and Secured Transactions (ICITST), 2015, pp. 202-207, doi:10.1109/jicitst.2015.7412089.

7) A. Mehmood, I. Natgunanathan, and Y. Xiang, "Protection of big data privacy", In IEEE translations and content mining are permitted for academic research, 2016.

8) M. Bardi, Z. Xianwei, L. Shuai, L. and Fuhong, "Big Data security and privacy: A review", China Communication, vol.11, no.14, 2014, pp. 135-145, doi:10.1109/cc.2014.7085614.

9) M. K. Scrinivasan and P. Revthy, "State-of-the-art Big Data Security Taxonomies", Proceeding of the 11th Innovation in Software Engineering Conference-ISEC, 2018, doi:10.1145/31728771.317288

10) P. Derbeko, S. Dolev, E. Gudes, and S. Sharm, "Security and Privacy aspects in MapReduce on Clouds: A Survey", Journal Computer Science Review, vol.20, no. c, 2016, pp.1-28, doi:10.1016/j.cosrev.2016.05.001.

11) M. Gupta, F. Patwa and R. Sandhu, "Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem", Processing's of the Third ACM Workshop on Attribute-Based Access Control-ABAC-18, 2018, pp.13-24, doi: 10.1145/3180457.3180463.

12) Vormetric Data Security, "Securing Hadoop: Security Recommendation for Hadoop Environments".   Available   at https://securosis.com/assets/library/reports/Securing_Hadoop_Final_V2.pdf

13) M. R. Jam, M. K. Akbari, and L. M. Khanli, "A Survey on Security of Hadoop", 4th International Conference on Computer and Knowledge Engineering (ICCKE), 2014, pp.716-721.

14) J. Zhao, L. Wang, J. Tao, J. Chen, W. Sun, R. Ranjan, J. Kolodziej, A. Streit, and D. Georgakopoulos, "A security framework in G-Hadoop for big data computing across distributed Cloud data centres", Journal of Computer and System Sciences, vol.80, 2014, pp. 994-1007.

15) Zettaset, "The Big Data Security Gap: Protecting the Hadoop Cluster", White Paper, at https://www.zettaset.com/wp-content/uploads/2014/.../zettaset_wp_security_0413.pdf, Zettaset, 2014.

16) S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing", Communications Society IEEE INFOCOM, doi:978-1-4244-5837-0/10, 2010.

17) T. C. Nguyen, W. Shen, J. Jiang, and W. Xu , "A Novel Data Encryption in HDFS" , 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of

872   Things and IEEE Cyber, Physical and Social Computing, 2013, pp. 2183-2187, doi:
873   10.1109/GreenCom-iThings-CPSCom.2013.4.

874 18) S. Park and Y. Lee "Secure Hadoop with Encrypted HDFS", J.J. Park et al. (Eds.): GPC 2013,
875   LNCS 7861, 2013, pp. 134–141.

876 19) C. Yang, W. Lin, and M. Liu, "A Novel Triple Encryption Scheme for Hadoop-Based Cloud
877   Data Security," in Emerging Intelligent Data and Web Technologies (EIDWT), Fourth
878   International Conference, 2013, pp. 437-442.

879 20) H. Zhou, and Q. Wen, "Data Security Accessing for HDFS Based on Attribute-Group in Cloud
880   Computing", International Conference on Logistics Engineering, Management and Computer
881   Science, 2014, pp-1142-1145.

882 21) L. Xu, X. Wu and X. Zhang, "CL-PRE: A certificate less proxy re-encryption scheme for secure
883   data sharing with public cloud", in ACM Symposium on Information, Computer and
884   Communications Security (ASIACCS'12), 2012, pp. 87-88.

885 22) J. C. Cohen, and S. Acharya, Journal of Information Security and Applications, "Toward a
886   trusted HDFS storage platform: Mitigating threats to Hadoop infrastructure using hardware-
887   accelerated encryption with TPM-rooted key protection", 2014, pp. 224-244.

888 23) P. Zerfos, H. Yeo, B. D. Paulovicks and V.Sheinin, "SDFS: Secure Distributed File System for
889   Data-at-Rest Security for Hadoop-as–a-Service", IEEE International Conference on Big Data
890   (Big Data), 2015, pp-1262-1271, 2015, doi:978-1-4799-9926-2/15/\$31.00.

891 24) M. I. Maheswari, S. Revathy, and R. Tamilarasi, "Secure Data Transmission for Multi sharing
892   in Big Data Storage", Indian Journal of Science and Technology, vol. 9 no. 21, 2016,
893   doi:10.17485/ijst/2016/v9i21/95164.

894 25) Y. Tian, "Towards the Development of Best Data Security for Big Data", Communication and
895   Network, Scientific Research Publishing Inc. vol.9, 2017, pp-291-301.

896 26) M. Usama, and N. Zakari, "Chaos-Based Simultaneous Compression and Encryption for
897   Hadoop", 2017, doi:10.1371/journal.pone.0168207.

898 27) H. Mahmoud, A. Hegazy, and M. H. Khafagy, "An Approach for Big Data Security Based on
899   Hadoop Distributed File System", International Conference on Innovative Trends in Computer
900   Engineering (ITCE), 2018, doi:10.1109/ITCE.2018.8316608.

901 28) I. Polato, R. Re, A. Goldmn, and F. Kon, "A Comprehensive view of HadoopReseach –A
902   Systematic Literature Review", Journal of Network and Computer Applications, vol. 46, 2014,
903   pp. 1-25, doi:http://dx.doi.org/10.1016/j.jnca.2014.07.022.

904 29) M. Ebrahim, S. Khan, and U. B. Khalid, "Symmetric Algorithm Survey: A Comparative
905   Analysis", International Journal of Computer Applications (0975 – 8887), vol. 61, no. 20, 2013,
906   pp. 12-19.

907 30) W. Yin, J. I. Iska, and H. Zhou, "Protecting Private Data by Honey Encryption", Hindawi
908   Security and Communication Networks vol. 2017, Article ID 6760532, 9 pages. Available at:
909   https://doi.org/10.1155/2017/6760532.

910 31) A. Juels and T. Ristenpart, "Honey Encryption: Security beyond the Brute-Force Bound," in
911   Advances in Cryptology-EUROCRYPT2014, 2014, pp.293–310.

912 32) Honey encryption, Available at: https://en.m.wikipedia.org/wiki/Honey_encryption

913 33) K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Protocol misidentification made easy
914   with format-transforming encryption," in Proceedings of the ACM SIGSAC Conference on
915   Computer and Communications Security, ACM, 2013, pp. 61-72.

916 34) P. P. Vinayakand, and M.A. Nahala, "Avoiding Brute Force Attack in Manet Using Honey
917   Encryption," International Journal of Science and Research, vol.4, no.3, 2015, pp.83-85.

918 35) T. Zhao, L. Wei, and C. Zhang, "Attribute- Based Encryption Scheme Based on SIFF", IEEE
919   ICC 2016 Communication and Information System Security Symposium.

920 36) B. S. Varsha, and P. S. Suryateja, "Using Attribute- Based Encryption with Advanced
921   Encryption Standard for Secure and Scalable Sharing of Personal Health Records in Cloud",

922 International Journal of Computer Science and Information Technologies, vol.5, no. 5, 2014,
923 pp. 6395-6399.
924 37) P. Owezarski, "A near real-time algorithm for autonomous identification and characterization
925 of honey pot attacks", In Proceedings of the 10th ACM Symposium on Information, Computer
926 and Communications Security, 2015, 531-542.
927 38) I. S. Kim, and M. H. Kim, "Agent-based honey net framework for protecting servers in campus
928 networks", IET Information Security 6 (3) (2012) 202-211.
929 39) P. Jain, and Sardana, "A. Defending against internet worms using honey farm", ACM in
930 Proceedings of the CUBE International Information Technology Conference, 2012, 795-800.
931 40) M. Bellare, T. Ristenpart, T. Rogaway, and P. Stegers, "Format-preserving encryption", In
932 International Workshop on Selected Areas in Cryptography, 2009, 295-312.
933 41) N. Tyagi, J. Wang, K. Wen, and D. Zuo, "Honey encryption applications", Network Security,
934 2015, 1–16.
935 42) H. Choi, H. Nam, and J. Hur, "Password typos resilience in honey encryption", IEEE
936 International Conference in Information Networking (ICOIN), 2017, 593-598.
937 43) S. F. Tan, and A. Samsudin, "Enhanced Security of Internet Banking Authentication with
938 Extended Honey Encryption (XHE) Scheme" In Innovative Computing, Optimization and Its
939 Applications, 2018, 201-216.
940 44) S. D. C. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption:
941 Management of access control evolution on outsourced data," in Proc. of VLDB'07, 2007.
942 45) M. Atallah, K. Frikken, and M. Blanton, "Dynamic and efficient key management for access
943 hierarchies," in Proc. of CCS'05, 2005.
944 46) S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained access
945 control in cloud computing," in Proc. of IEEE INFOCOM'10, San Diego, CA, USA, March 2010.
946 47) H. Ulusoy, P. Colombo, E. Ferrari, M. Kantarcioglu, and E. Pattuk, "GuardMR: Fine-grained
947 Security Policy Enforcement for MapReduce Systems", In ACM ASIACCS, 2015.
948 48) H. Ulusoy, M. Kantarcioglu, K. Hamlen, and E. Pattuk, "Vigiles: Fine-grained Access Control
949 for MapReduce Systems", In IEEE BigData, 2014.
950 49) P. Colombo, and E. Ferrari, "Access Control in the Era of Big Data: State of the Art and
951 Research Directions", Blue Sky Session: Innovation in Access Control and Privacy-aware
952 Data Management for Big Data and IoT, SACMAT'18, June 13-15, 2018.
953 50) W. Lu, Y. Wang, J. Juang, J. Liu, Y. Shen, B. Wei, "Hybrid storage architecture and efficient
954 MapReduce processing for unstructured data", Parallel Comput, **69**, 63–77
955 (2017). https://doi.org/10.1016/j.parco.2017.08.008
956 51) A. B. Cheikh, H. Abbes, G. Fedak , " Towards privacy for MapReduce on hybrid clouds using
957 information dispersal algorithm", Data Management in Cloud, Grid and P2P Systems - 7th
958 International Conference, Globe 2014, Munich, Germany, September 2-3, 2014. Proceedings,
959 Munich, Germany, 2014; 37–48.
960 52) Y. Oktay, S. Mehrotra, V. Khadilkar, and M. Kantarcioglu. SEMROD: secure and efficient
961 MapReduce over hybrid clouds. In Proceedings of the 2015 ACM SIGMOD International
962 Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015,
963 pages 153–166, 2015.
964 53) Y. Wang, Y. Shen, H. Wang, J. Cao, X. Jiang, "MtMR: Ensuring MapReduce Computation
965 Integrity with Merkle Tree-based Verifications", Journal of L atex Class Files, Vol. 14, No. 8,
966 August 2015.
967 54) X. Fu, Y. Gao, B. Luo, X. Du, and M. Guizani, "Security Threats to Hadoop: Data Leakage
968 Attacks and Investigation", IEEE Network, 2017, 31(2):67-71.
969 55) F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M.
970 Russinovich, "VC3: Trustworthy Data Analytics in the Cloud using SGX", In Proceedings of
971 the 36th IEEE Symposium on Security and Privacy (Oakland), pp. 38–54, 2015.

972   56) T. T. A. Dinh, P. Saxena, E.-C. Cang, B. C. Ooi, and C. Zhang, "M2R: Enabling stronger
973   privacy in MapReduce computation," in Proceedings of the 24th USENIX Security Symposium
974   (Security), Washington, DC, Aug. 2015.
975   57) S. Dolev, p. Gupta, Y. Li, S. Mehrotra, S. Sharma, "Privacy-Preserving Secret Shared
976   Computations using Mapreduce", 2018, Available at: https://arxiv.org/abs/1801.10323.
977   58) K. Sudhakar , M.A.H. Farquad , G. Narshimha, "Effective convolution method for privacy
978   preserving in cloud over big data using map reduce framework", IET Software, vol.13, pp.187-
979   194, 2019. doi: 10.1049/iet-sen.2018.5258.

Figure 1

Working of the Proposed ABHE encryption Algorithm when Attribute and Password are entered for 64 MB file size. (For both encryption and decryption)

Figure 2

Working of Proposed ABHE Encryption Algorithm when Right and Wrong Password are entered for 64 MB File Size.

Figure 3

Size Block one before Encryption

Figure 4

Size of Block one after Encryption
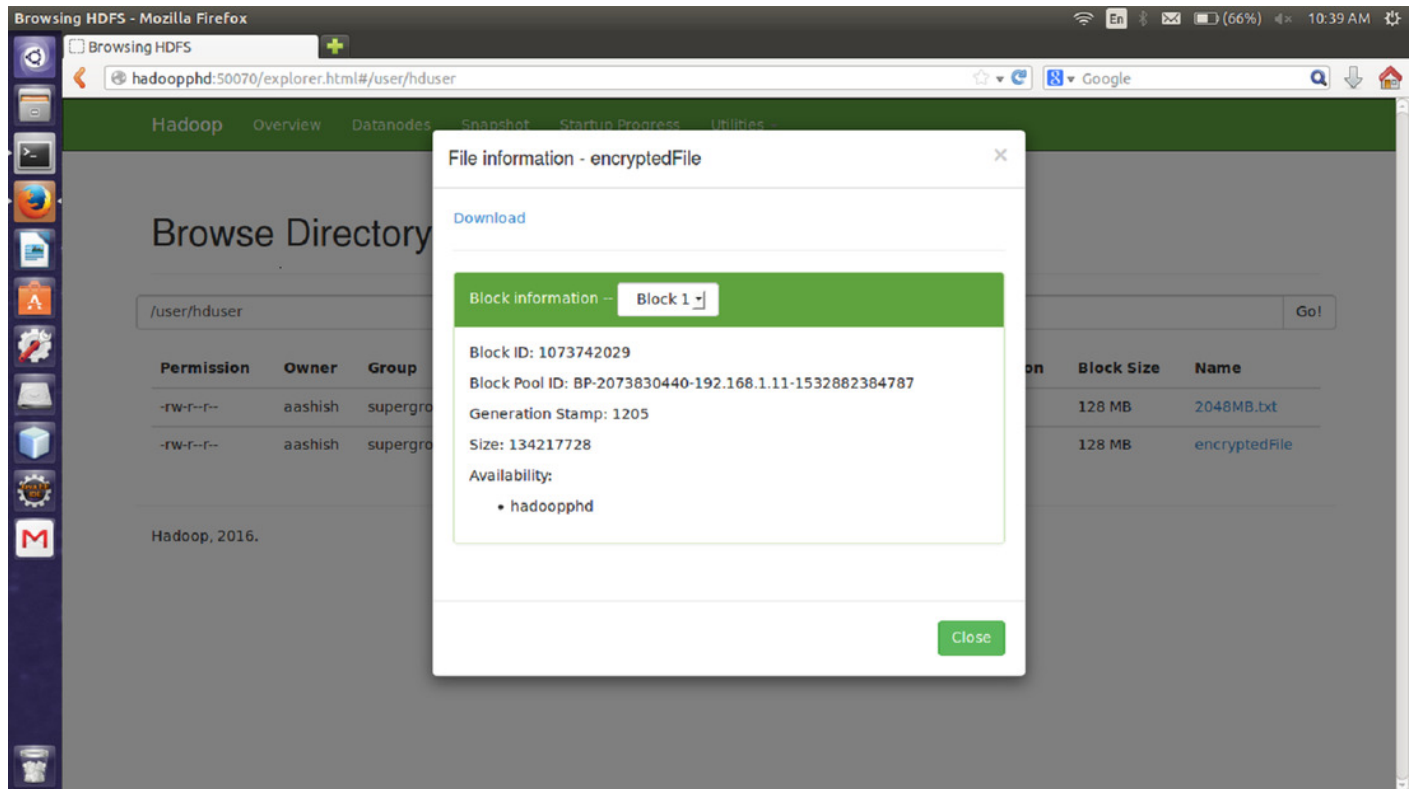
Figure 5

Size of Block eight before Encryption
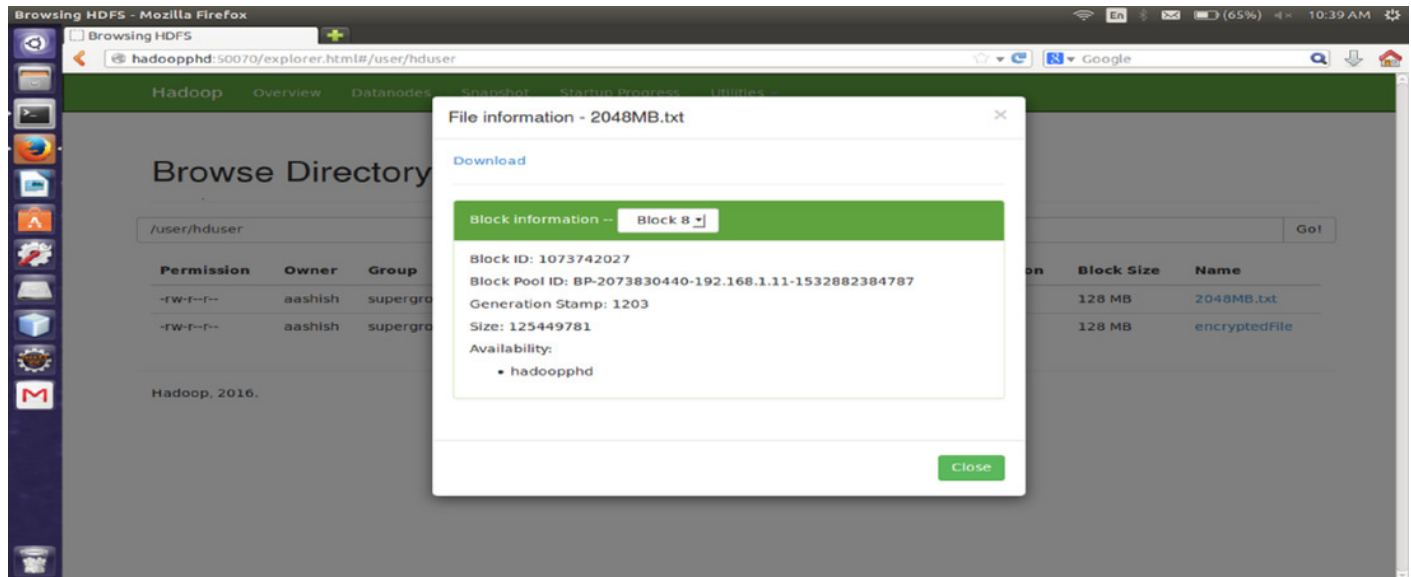
Figure 6

Size of Block eight after Encryption
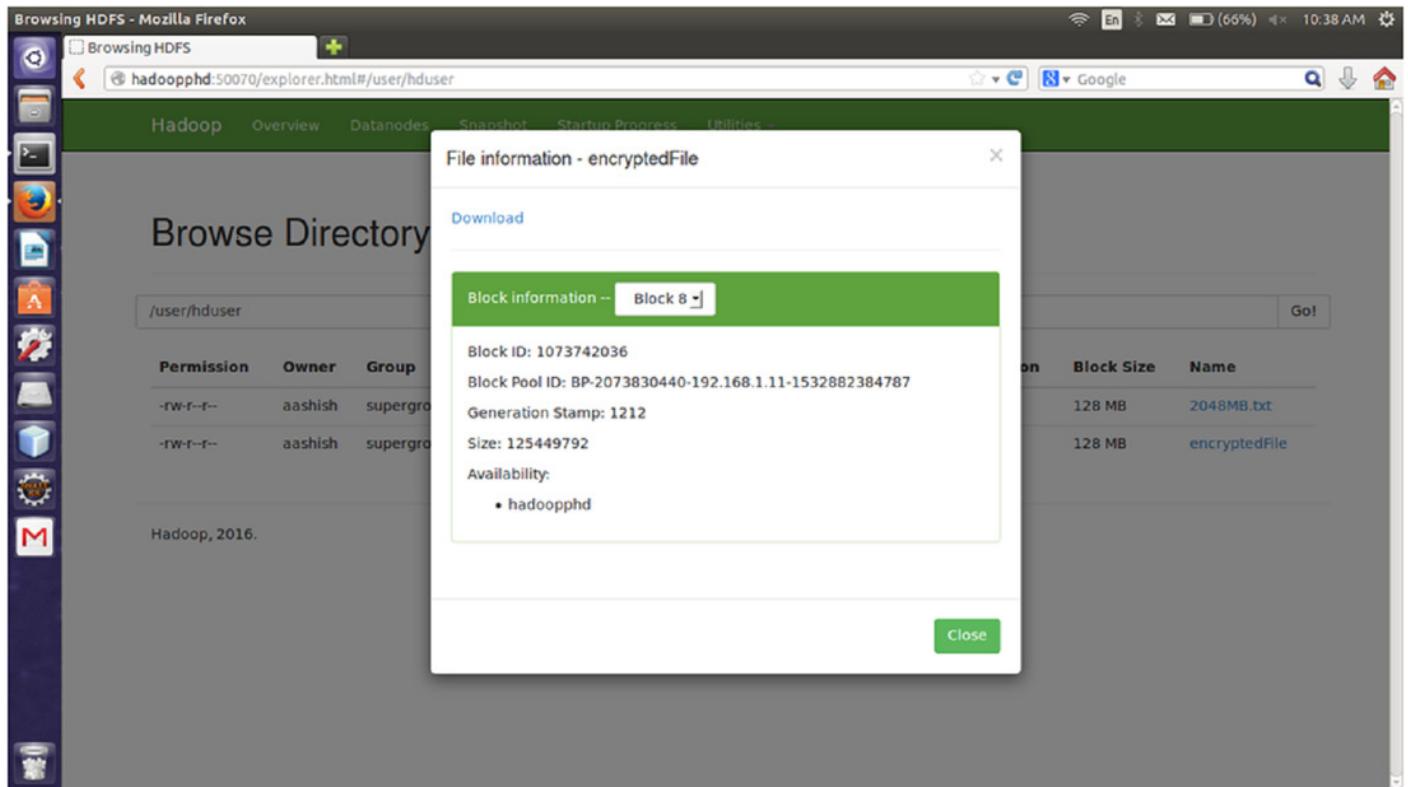
Figure 7

Writing a File with Encryption in HDFS
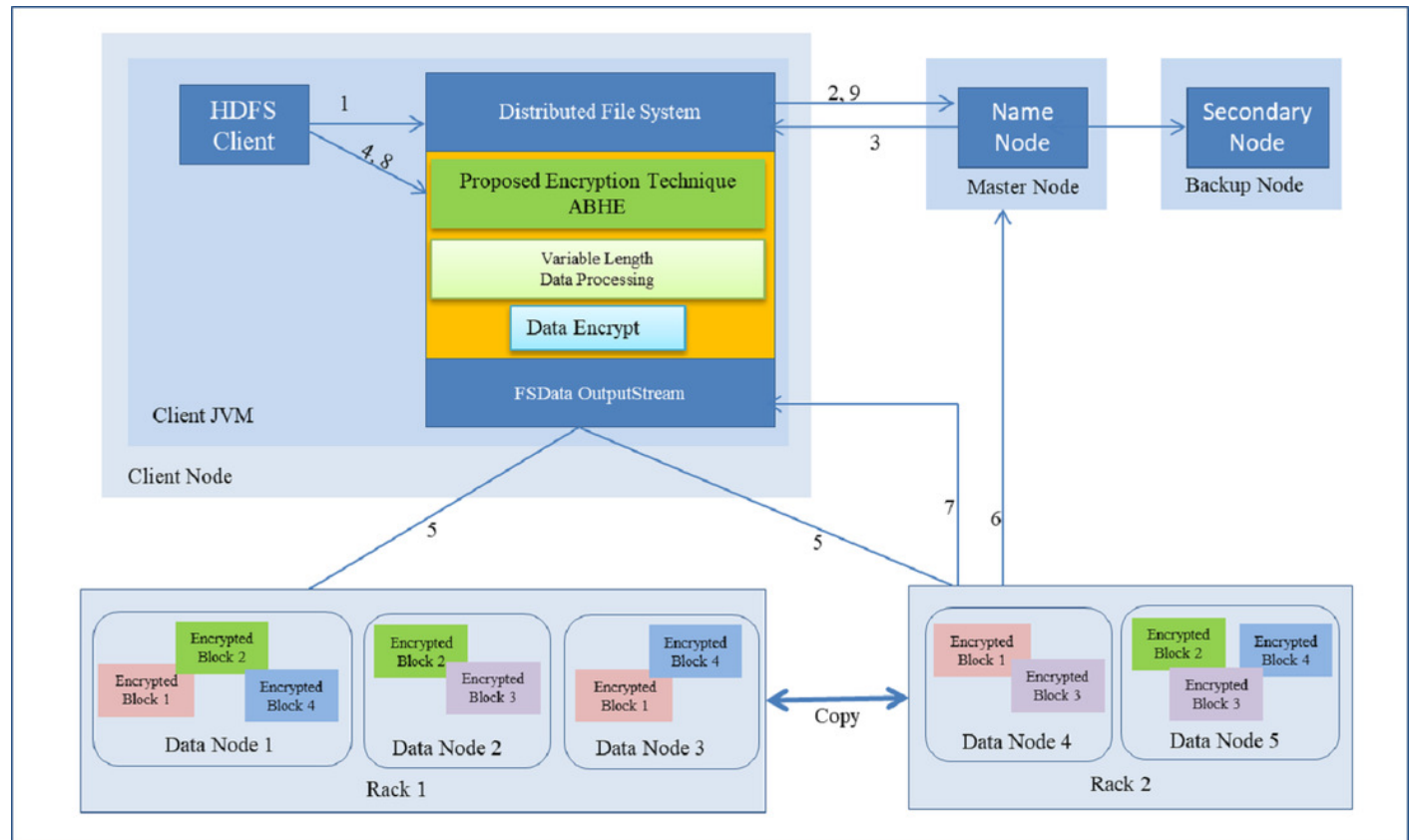
Figure 8

Reading a File with Decryption in HDFS
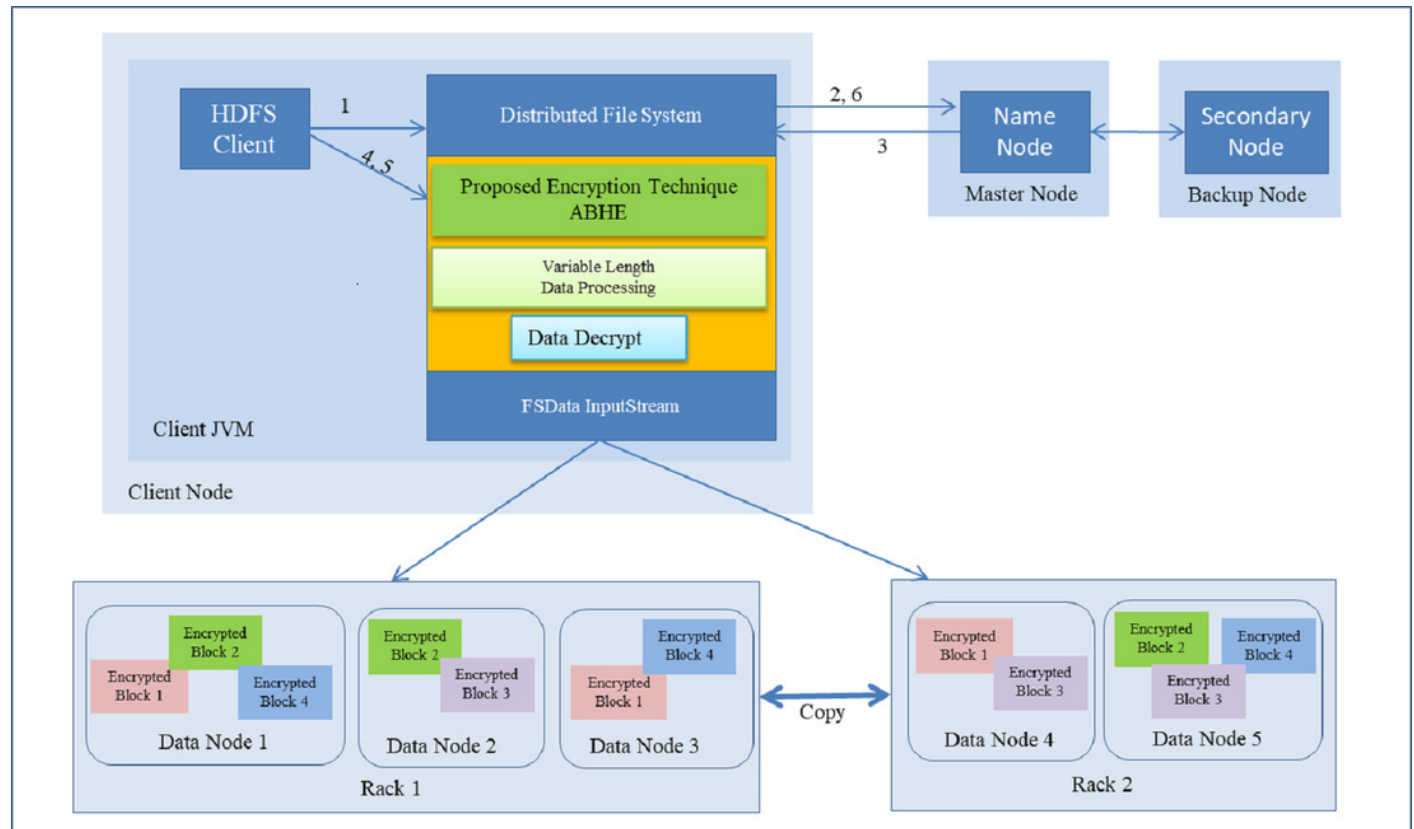
Figure 9

Encryption Time (minutes) of AES, AES with OTP and Proposed ABHE Algorithm
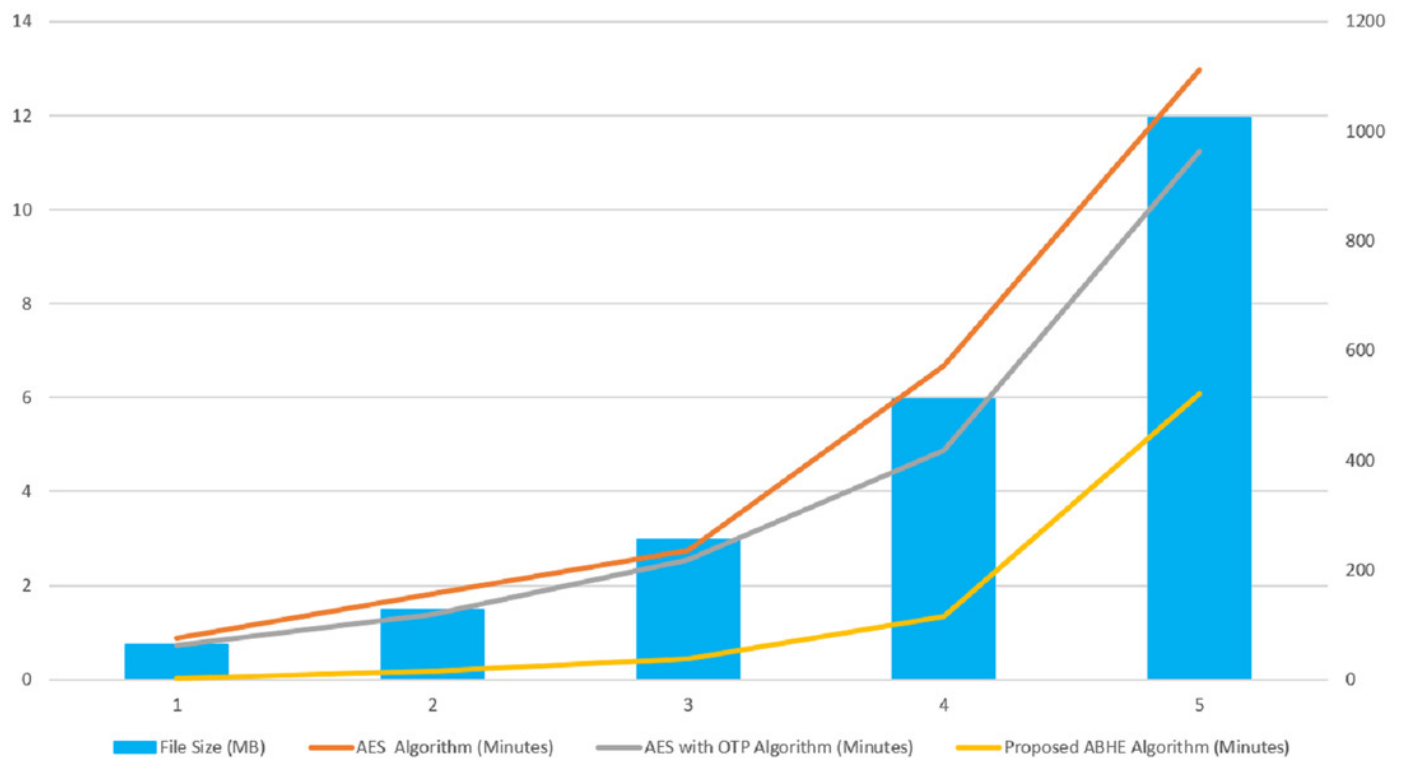
Figure 10

Decryption Time (minutes) of AES, AES with OTP and Proposed ABHE Algorithm
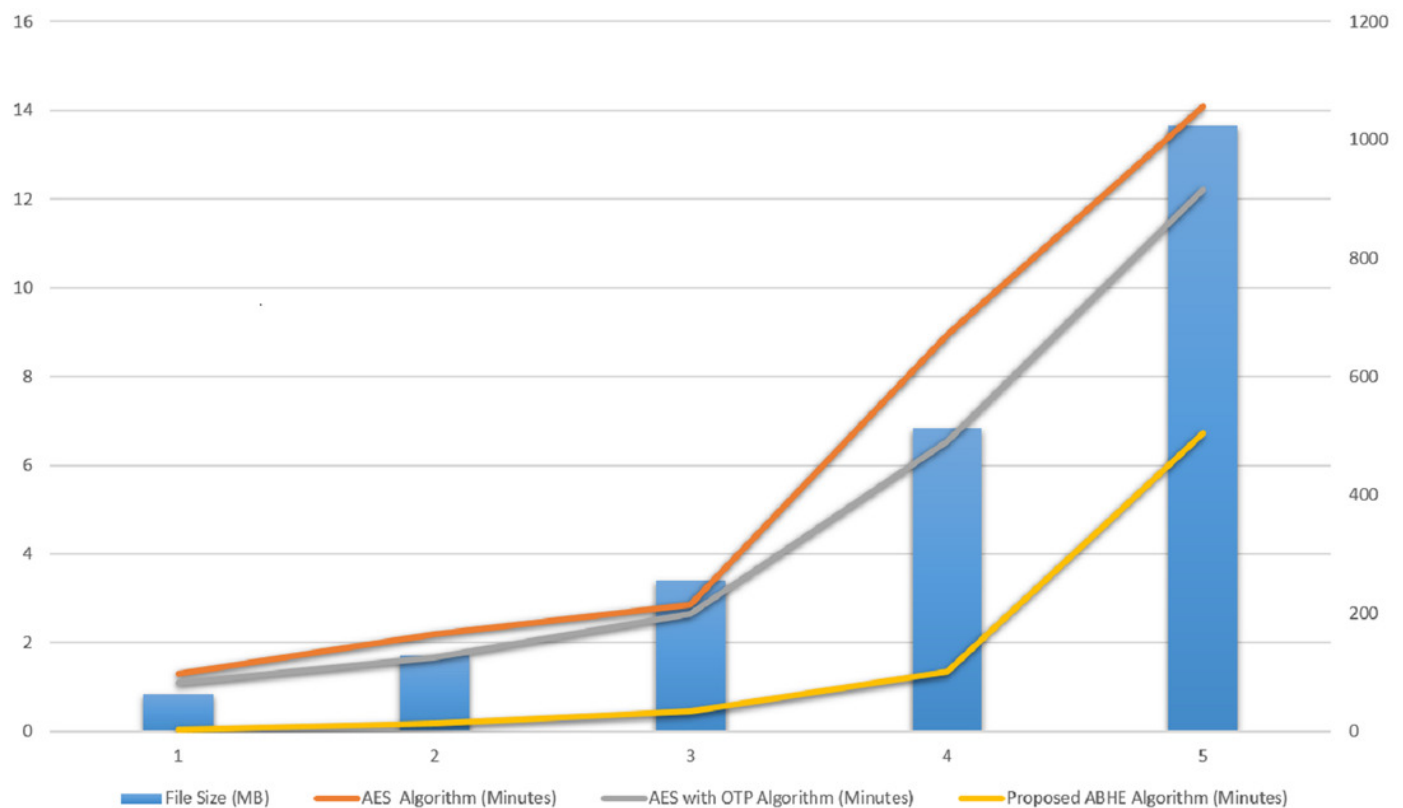
Figure 11

Size of Block five before Encryption
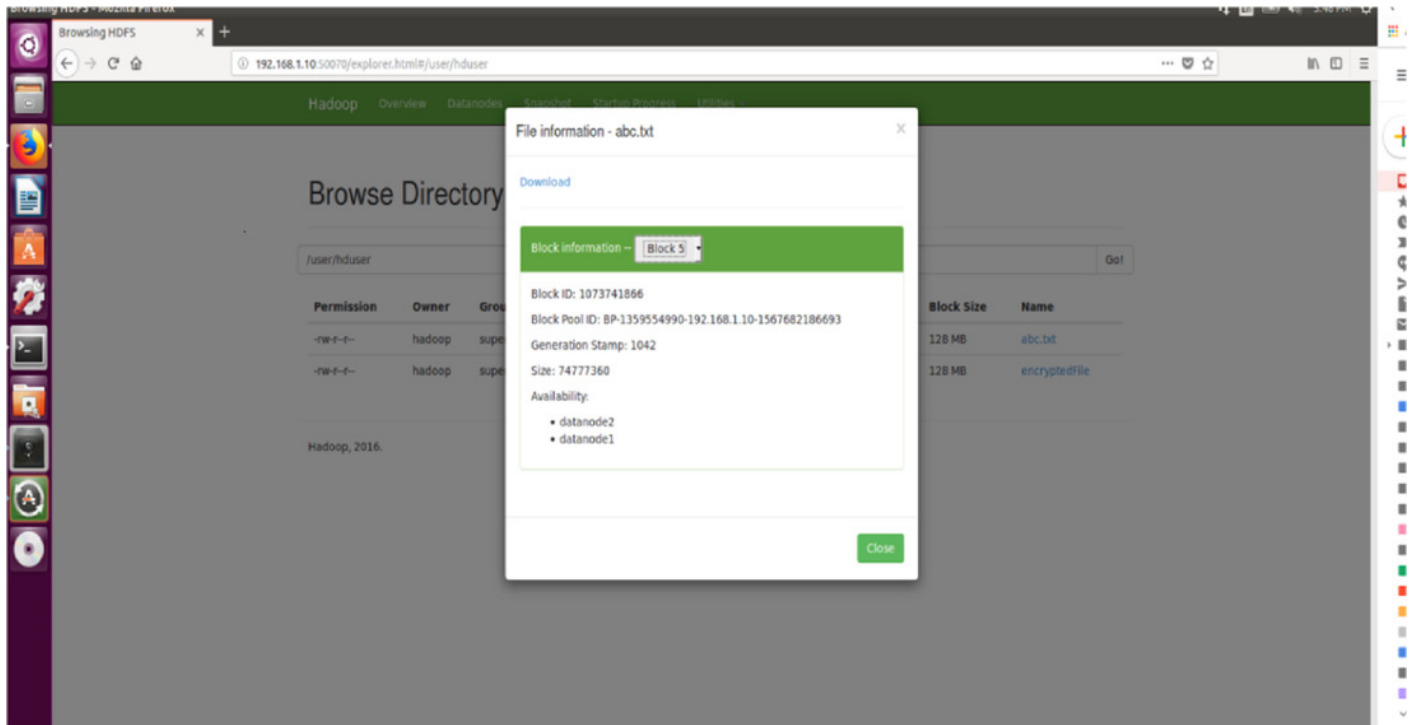
Figure 12

Size of Block five after Encryption

Figure 13

Browse directory show that encrypted file and abc.txt non-encrypted file HDFS

Data Size Comparison among AES and AES with OTP algorithms and the Proposed ABHE Algorithm

1 **Table 1: File Size Comparison among AES and AES with OTP algorithms and the Proposed**
2 **ABHE Algorithm**

File Size (MB)	AES Algorithm (MB)	AES with OTP Algorithm (MB)	Proposed ABHE Algorithm (MB)
64	96.0	74.7	64
128	192.8	149.3	128
256	384.0	298.7	256
512	768.0	597.3	512
1024	1536	1228.3	1024

File Encryption Performance Comparison among AES and AES with OTP Algorithms and the Proposed ABHE Algorithm

1 **Table 2: File Encryption Performance Comparison among AES and AES with OTP Algorithms**
2 **and the Proposed ABHE Algorithm**

File Size (MB)	AES Algorithm (Minutes)	AES with OTP Algorithm (Minutes)	Proposed ABHE Algorithm (Minutes)
64	0.8704	0.7311	0.026
128	1.8216	1.3820	0.168
256	2.7396	2.5484	0.45
512	6.6682	4.8780	1.35
1024	12.9751	11.2511	6.08
Total Encryption Time	25.0749	20.7906	8.074
Throughput of Encryption (MB/Minutes)	79.12	95.42	245.72

File Decryption Performance Comparison among AES and AES with OTP Algorithm and the Proposed ABHE Algorithm

1   **Table 3: File Decryption Performance Comparison among AES and AES with OTP Algorithm**
2   **and the Proposed ABHE Algorithm**

File Size (MB)	AES Algorithm (Minutes)	AES with OTP Algorithm (Minutes)	Proposed ABHE Algorithm (Minutes)
64	1.3056	1.0950	0.03065
128	2.1859	1.6560	0.168
256	2.8641	2.6554	0.45
512	8.9494	6.5361	1.35
1024	14.0841	12.2115	6.73
Total Decryption Time	29.3891	24.154	8.72865
Throughput of Decryption (MB/Minutes)	67.50	82.13	227.29