



Article

A Hybrid Multi-Cloud Framework Using the IBBE Key Management System for Securing Data Storage

Manreet Sohal ¹, Salil Bharany ^{2,*} , Sandeep Sharma ², Mashael S. Maashi ³  and Mohammed Aljebreen ⁴¹ Department of Computer Applications, Guru Nanak Dev Engineering College, Ludhiana 141006, India² Department of Computer Engineering & Technology, Guru Nanak Dev University, Amritsar 143005, India³ Software Engineering Department, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia⁴ Department of Computer Science, Community College, King Saud University, Riyadh 11437, Saudi Arabia

* Correspondence: salil.bharany@gmail.com

Abstract: Information storage and access in multi-cloud environments have become quite prevalent. In this paper, a multi-cloud framework is presented that secures users' data. The primary goal of this framework is to secure users' data from untrusted Cloud Service Providers (CSPs). They can collude with other malicious users and can hand over users' data to these malicious users for their beneficial interests. In order to achieve this goal, the data are split into parts, and then each part is encrypted and uploaded to a different cloud. Therefore, client-side cryptography is used in this framework. For encrypting users' data, the BDNA encryption technique is used. This framework presents a hybrid cryptographic approach that uses Identity-based Broadcast Encryption (IBBE) for managing the keys of the symmetric key algorithm (BDNA) by encrypting them with the particular version of IBBE. The work presented in this research paper is the first practical implementation of IBBE for securing encryption keys. Earlier, IBBE was only used for securely broadcasting data across many users over a network. The security of this hybrid scheme was proved through Indistinguishable Chosen-Ciphertext Attacks. This double encryption process makes the framework secure against all insiders and malicious users' attacks. The proposed framework was implemented as a web application, and real-time storage clouds were used for storing the data. The workflow of the proposed framework is presented through screenshots of different working modules.

Keywords: multi-clouds; storage security; client-side cryptography; key management



Citation: Sohal, M.; Bharany, S.; Sharma, S.; Maashi, M.S.; Aljebreen, M. A Hybrid Multi-Cloud Framework Using the IBBE Key Management System for Securing Data Storage. *Sustainability* **2022**, *14*, 13561. <https://doi.org/10.3390/su142013561>

Academic Editor: Zubair Baig

Received: 3 August 2022

Accepted: 17 October 2022

Published: 20 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Cloud computing has been used extensively over the last few years. It offers vigorously scalable resources accessible as a service over the internet [1–4]. Above and beyond various other services, cloud storage has charmed numerous users for saving their data on the clouds. By making use of this facility, the users or organizations can store their data on the cloud which can be made available at anytime and anywhere and in an economical manner. The attractiveness and demand of cloud storage has been proved through the widespread use of storage clouds such as Dropbox, Google drive, iCloud, etc. [5]. However, despite its various benefits, the cloud is still vulnerable to security and privacy issues. Once the data are housed on the third-party servers, there is no surety that is the data are secure. Although the CSPs assure it with their promises of cloud encryption, it is not certain that they will not hand over one's data to the government agencies without one's knowledge or approval. The cloud users cannot fully trust the third party CSPs as they may collude with some malicious parties or may exploit the customer's data for their profitable purposes [6]. To overcome these issues, in the last few years there has been a drift towards "multi-clouds" or "intercloud" or "cloud-of-clouds" [5]. As a corollary, there is no single trusted CSP that has control over the customer's data, rather there are multiple CSPs who can host only parts of user data, or we can say it is a technique by which users can smartly store their data

in more than one cloud so that the confidentiality and privacy of their data is maintained. The service providers are unaware of the storage of data on other clouds.

Multi-Cloud or “Cloud-of-Clouds” is the deployment of different services in a single heterogeneous architecture and the multi-cloud storage implies the exploitation of different cloud storage services by making use of a single web interface [7] instead of the default interface provided by each individual CSP [8]. The organizations should gradually make full use of plentiful contributions of the cloud to derive maximum benefits from it and to reduce the outflow in programming advancement. “Moreover, a year ago, in a futurescape report, the International Data Corporation projected multi-cloud appropriation will expand definitely and till 2020, over 85% of big business IT partnerships have migrated towards the multi-cloud environment” [9–11].

Undoubtedly, it is a more intricate system as compared to the hybrid cloud which is just a combination of public and private clouds. Multi-cloud on the other hand deals with various aspects such as the possibility of failures of service availability, loss and corruption of data, malicious insiders in a single cloud architecture, and the server collusion attacks. The security and privacy of users’ data is the topic of utmost concern. Over 80% of the companies [12] have the fear of losing control over their data and security threats on their data. Therefore, more focus should be laid upon storing the data on multiple clouds rather than on a single cloud. Applications should be developed that store only portions of data on a single cloud so that no cloud has access to the complete data. In addition to this, the data need to be encrypted before sending it to the cloud, so that even the CSPs cannot extract the original data from the outsourced encrypted data.

In this research work, a multi-cloud framework is proposed which not only saves data on multiple clouds but also implements security algorithms at its end and use identity-based broadcast key management system. The work was inspired by the techniques represented by [5,13,14] and it overcomes the shortcomings of these existing approaches. The major focus of this research paper is on the design and implementation of a client-side security framework that can preserve the confidentiality of data stored on the clouds. There are lots of highly secure cryptographic algorithms available for securing the cloud storages but besides security, time also plays a crucial role; therefore, for real time application uploading and securing the data should be considerably fast. Therefore, the cryptographic algorithm used in this framework is BDNA [15,16]. The authors of this algorithm [15,16] have already proved its security and performance with respect to other cryptographic algorithms.

Key Contributions: In this paper, we propose a multi-cloud cryptographic framework to secure users’ data saved on the cloud. It is a novel framework that uses the concept of IBBE for securing the cryptographic keys and controlling access to the data. The framework proposed in this paper is the first practical application of using IBBE to encrypt the private key of a symmetric key algorithm (BDNA in this paper). Till now, IBBE has been used for encrypting the data that have to be broadcasted to a group of users. However, in our paper, the private key of BDNA is encrypted with a group key of all the users who are granted access to a particular file. This group key is calculated using the identities of the users in the group. Instead of broadcasting the secret key to the users, it is saved on the security framework’s storage space and the users can extract the group key using their respective private key. The game theory is used to prove that the proposed system is secure against Indistinguishable Chosen-Ciphertext Attacks.

The rest of the paper is organized as: Section 2 reviews the related research papers, Section 3 presents the proposed security framework, in Section 4 the key management technique used in the security framework is discussed, in Section 5 the implementation of the security framework is presented, Section 6 presents the security analysis, in Section 7 results of the proposed framework has been discussed and compared with the existing approaches and the last Section 8 concludes the research work.

2. Related Work

This section discusses various research papers on multi-cloud frameworks and key management techniques. In the first subsection, research papers on multi-clouds are reviewed and compared and in the next subsection, papers on identity-based broadcast system are discussed.

2.1. Multi-Clouds

In this section, different multi-cloud frameworks are reviewed and their merits and demerits are lineated in Table 1.

Table 1. Advantages and limitations of various multi-cloud frameworks.

Reference	Cryptographic Technique	Advantage	Limitations
[17]	Attribute Based Encryption (ABE) and cryptographic secret sharing	<ul style="list-style-type: none"> Client-side encryption is used Encrypted records are split by multi-cloud proxy 	<ul style="list-style-type: none"> Huge computation overheads Long waiting time is required for sharing the records among the groups No file indexing High costs of implementation The whole scheme can be corrupted by uploading malicious files Not fully automated No key management
[13]	Advanced Encryption Standard (AES)	<ul style="list-style-type: none"> Cryptographic data splitting is used on client side 	<ul style="list-style-type: none"> No key management technique used No focus on insider attacks No prevention against malicious files No provision for solving same file name conflicts
[5]	Advanced Encryption Standard (AES) and device-based identity access techniques	<ul style="list-style-type: none"> Client-side cryptography is used Device based identity verification is completed to only allow authorized users access to data 	<ul style="list-style-type: none"> Data are split after encryption which results in longer waiting times No automation process file splitting, merging File merging conflicts are there at time of retrieval
[18]	Shamir secret sharing algorithm and base64 encoding scheme	<ul style="list-style-type: none"> Providing secure sharing of files in multi-cloud environments Prevents against attacks by malicious insiders 	<ul style="list-style-type: none"> No file indexing Complex file retrieval process All the tasks have not been automated No provision for solving same filename conflicts
[14]	Dynamic indexed based cryptographic data slicing	<ul style="list-style-type: none"> Appropriate for the process of decision making for organizations and individuals Suitable for different file formats Prevents against CSP collusion attacks for retrieving significant information Elimination of centralized allocation of storage 	<ul style="list-style-type: none"> Whole system is based upon an assumption that there are no malicious intentions of any of the CSPs No focus on non-repudiation as well as on dynamic symmetric encryption No provision for solving same filename conflicts No key management
[19]	Any standard encryption technique such as AES combined with cipher block chaining mode or cipher feedback mode and Erasure encoding	<ul style="list-style-type: none"> Provides intrusion tolerance Tolerant to the loss of multiple chunks of data 	<ul style="list-style-type: none"> No key management No file indexing No provision for solving same filename conflicts

Table 1. Cont.

Reference	Cryptographic Technique	Advantage	Limitations
[20]	Advanced Encryption Standard (AES) and SHA-256	<ul style="list-style-type: none"> • High availability due to RAID-Z technology • Ensures confidentiality and integrity of data in multi-cloud environment with the usage of zeta-byte file system 	<ul style="list-style-type: none"> • Higher computation overheads • Performance degrades for files above 100 MB • No key management technique used
[21]	MD5 and base64 encoding	<ul style="list-style-type: none"> • Increases the trust on CSPs • Ensures reliability and transparency • Continuous auditing of cloud services • Proper indexing of files 	<ul style="list-style-type: none"> • Filename conflicts cannot be resolved
[22]	Advanced Encryption Algorithm combined with fiestal networks	<ul style="list-style-type: none"> • Secure against insider attacks • Provides indexing of data files • Client-side encryption mechanism 	<ul style="list-style-type: none"> • No key management techniques used • Framework has not been tested with real cloud service providers • No provision for solving same filename conflicts
[23]	Combination of AES, BLOWFISH and RC5	<ul style="list-style-type: none"> • A secure hybrid client-side encryption mechanism • Files are split before encryption 	<ul style="list-style-type: none"> • No key management technique used • File indexing is not completed • Consists of filename conflicts

The authors in [17] proposed an architecture that uses the concept of multi-cloud for sharing health care records. These records are encrypted using Attribute Based Encryption (ABE) and cryptographic secret sharing. After encryption the record is split by the multi-cloud proxy which is then stored on multiple clouds. There are several shortcomings of this technique such as a huge computation and long waiting time is required for sharing the records among the groups; moreover, the approach does not use file indexing which makes the file retrieval process ambiguous, and a highly configured machine is used for carrying out the experiments which makes this approach costly. Moreover, the whole scheme can be corrupted by uploading malicious files and it is not even fully automated.

For enhancing the secure sharing of data stored in multi-cloud environments, the researchers in [13] presented a multi-cloud framework that uses Advanced Encryption Standard Algorithm (AES) in order to endow the customers with better cloud storage decision making. However, the authors did not focus on the insider attacks, colluding attacks, and no measures were taken for enforcing data integrity and securing data from intruders and preventing malicious files.

In [5], the researchers presented a novel concept of using multi-cloud storage for enhancing cloud storage security together with the encryption algorithms as well as the device-based identity access techniques. In this approach, first of all, the data are encrypted using the standard AES algorithm which is then split into different chunks and these chunks are then stored on different clouds. In addition to that, this approach also uses a device-based identity verification security technique which is used to validate the users who can access the data.

In [18], the authors came forward with a new model that used Shamir secret sharing algorithm and base64 encoding scheme for providing secure sharing of files in multi-cloud environments. This scheme prevents against attacks by malicious insiders but does not involve file indexing which makes the file retrieval process difficult and complex. Further, all the tasks were not automated which decreases the overall efficiency of the proposed approach.

In [14], the researchers proposed a new framework that uses a standard algorithm for enhancing the secure data sharing by making use of dynamic indexed based cryptographic data slicing. The proposed framework is appropriate for the process of decision making for organizations and individuals by adopting multi-storage architecture for data storage based upon trust. The proposed system is suitable for different file formats, CSPs collusion attacks for retrieving significant information and elimination of centralized allocation of storage. However, the whole system is based upon an assumption that there are no malicious intentions of any of the CSPs. Moreover, the system does not focus on non-repudiation as well as on dynamic symmetric encryption.

The authors of the research work [19] presented a novel multi-cloud architecture that makes use of code erasure, data splitting and distribution of these data splits across different cloud storage systems in order to prevent attacks on data availability, integrity and confidentiality. The authors proved that the proposed system is tolerant to the loss of multiple chunks of data.

The researchers in [20] came forward with a new technique known as controller for storage as a service (CsaaS), which can help business organizations to stock up their data in multiple clouds by making use of file storage technology. The proposed technology consists of four units which aid the clients in attaining secure storage of data across different clouds. The performance analysis proved that for files of up to 90 MB, the completion latency and average rate of data transfer for reads and random reads were improved considerably as compared to single clouds.

In this research work [21], a novel multi-cloud architecture was proposed that facilitated the third-party auditors to constantly audit the services of each cloud in multi-cloud environments. This approach increases the trust on CSPs and ensures reliability and transparency and prevents against the hackers.

The authors in [22] presented a multi-cloud framework that uses hybrid encryption algorithm which is the combination of Advanced Encryption Standard (AES), BLOWFISH and RC5. The data is sliced and then each slice is encrypted with a different encryption algorithm and is then uploaded to different clouds. The authors claim that their system is highly secure against malicious attacks and provides high protection and the overall results are better when compared to the other architectures. However, the authors did not implement any key management techniques to secure the keys generated by the encryption algorithms and they did not provide any provision of resolving same filename conflicts.

In [23] the authors proposed a multi-cloud framework which performs the encryption of data, partitions the data, and imposes proper indexing on the data files in order to securely upload the data on different clouds. The authors presented a hybrid encryption algorithm by combining Advanced Encryption Standard (AES) with feistel networks to efficiently secure the data before uploading it to the multiple clouds. The proposed system provides security against insider attacks.

Lots of similar techniques have been proposed but none of them have succeeded in implementing an effective system for the secure sharing of data using multi-clouds. The techniques discussed above do not offer automated processes of file slicing, encryption, decryption, and retrieval. The existing work does not focus on resolving the file conflicts while merging the file parts for retrieval process, malicious files, colluding service provider attacks, insider attacks, removal of centralized distribution of data, and key management while sharing the data in multi-cloud storage. The existing techniques presented by [5,13,14] used AES encryption algorithm for securing the data. However, the authors in [16] already proved that BDNA is faster than AES and provides considerable security. Hence, it is suitable for real time data storage. Secondly, the techniques presented by [5,13] split the data after encryption, which leads to longer waiting times by the customers and it is also difficult to upload large files securely. Thirdly, in the approach presented by [14], there is an additional overhead on the data owners, since each time a user needs to access the data, he has to request the data owner for the keys. Lastly, these approaches have not provided any technique for the key management. However, practically, the security of a symmetric

key algorithm lies in its key. If keys are not managed properly, they can go in hands of malicious parties and the overall security of the algorithm will go in vain [24–26]. The framework presented in this research work overcomes all the issues present in the existing techniques.

2.2. Key Management

Key management is one of the most important concepts in the cloud environment for providing synchronized data flow in the networks. Encryption is one of the major techniques for securing users' data saved on the clouds but it involves a lot of computation power and the cryptographic keys generated as a part of encryption process can cause problems if not managed properly. These keys cannot be stored simply on the clouds along with the data because if these cryptographic keys become compromised, the whole of the encryption process will fail as the hackers can easily decrypt the data. Therefore, besides providing an efficient cryptographic technique for securing data stored on the clouds, a proper key management mechanism is also required to be implemented. While going through cryptographic techniques, we were inspired by the identity-based encryption techniques because they do not require the distribution of public key certificates and the identity of the user serves as the public key itself. Therefore, for the key management, we focused upon the identity-based cryptography [24] for securing the cryptographic keys of the symmetric key algorithms. This section discusses about some of the major research papers on identity-based cryptography referenced by us during the course of our research work.

In [27], the authors presented an IBBE scheme which can be used to efficiently encrypt a message broadcasted to a group of users without revealing their identities even to each other. The authors claimed that the proposed scheme can be used in those environments where the recipients have devices with a limited number of resources. This has been made possible by reducing the size of system's public parameters as well as the secret keys. Further, the size of user's private key has also been made constant which is far less as compared to the existing IBBE scheme.

The authors in [28] proposed an IBBE for open networks where the senders can send the messages to a subset of the system users and only those receivers will receive access to the message. The authors compared the proposed scheme with the already existing broadcast encryption scheme and proved that their scheme has lower costs and the number of system users can be altered efficiently.

In [29], the researchers proposed a generic construction for IBBE by exploiting the robust property of anonymous IBE. The proposed scheme achieves confidentiality and anonymity against Chosen-Ciphertext attacks in random oracle model. Further, the proposed scheme has a constant size of public parameter, private key, and constant cost of decryption.

The research work in [30] introduced the concept of anonymous certificate-based broadcast encryption [30] and made the decryption cost constant. As compared to the existing techniques, it is more computationally efficient; therefore, it can be used with computationally constrained users [31–38]. The authors proved that the proposed model is secure against Adaptive Chosen-Ciphertext attacks.

3. The Proposed System Model

In order to enhance the security of data stored on the clouds, the proposed system model adopts the concept of multi-clouds due to their advantages over the single cloud data storage. In this model, first of all the file is uploaded to the security framework which segregates the file into different parts and then, each part is encrypted separately using a secret key and stored on different clouds. A proper indexing of the file parts is completed and all the file-related metadata are stored by the security framework in its local database. Whenever, the receiver or the user wants to access the file, first of all he has to prove his authentication at the security framework's interface and after successful authentication,

the security framework downloads the various file parts from the different clouds. After downloading, each part is decrypted with the same secret key with which it has been encrypted earlier. After performing decryption, all the different parts of the file are merged to re-create the original file and then it is made available to the user. The framework offers the users a fair chance of choosing multi-cloud storage facilities in order to save and share their data securely over the clouds. The proposed framework provides assurance that the file is split up into at least three parts and each part is sent to a different cloud. We have taken at least three clouds in order to reduce server collusion attacks and to ensure the confidentiality of the data. Not even a single CSP can extract the original data from the data parts stored on its servers, without obtaining any additional data from other CSPs.

3.1. Overview of the Architecture

Figure 1 represents the overall architecture of the proposed security model. It consists of three levels which contain the various components involved in the security process. These levels are described as follows.

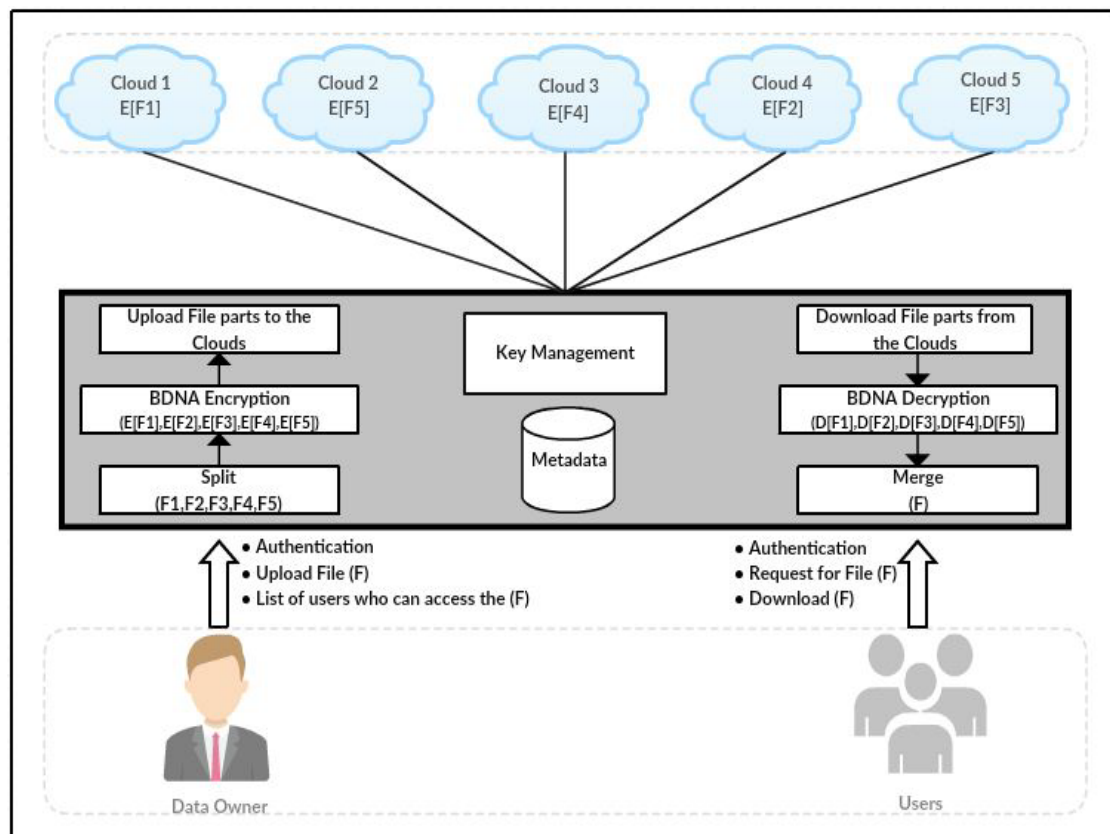


Figure 1. The Proposed Multi-Cloud Framework to Secure Cloud Data Storage.

3.1.1. The Cloud Level

The cloud level consists of the different storage cloud servers that are used for storing different parts of the same file. The minimum number of storage clouds used by our framework is three and maximum number is 5. The number of cloud servers can be increased as per the requirements and the availability of the storage clouds. It is not mandatory that a file part should be stored on each and every cloud. The file is split up into random parts and these parts are stored on the clouds randomly. The only constraints imposed are that the number of parts should always be three or more and a minimum of three clouds should always be involved. The storage clouds used by the proposed approach are Google Drive, Dropbox, Microsoft OneDrive, Mediafire, and Amazon Cloud Drive.

3.1.2. The Security Framework Level

The major component of the proposed security model is the security framework layer. It has been designed as a web application. It involves all the security mechanisms that have been implemented in order to secure users' data saved on the clouds. The various components involved in the security framework module are:

- i. Upload file: The first step involves the data owner uploading his file at the security framework's interface. Before actually uploading the file, the data owner is checked for his authenticity and after the successful authentication, he is allowed to upload the file which he wants to store onto the cloud;
- ii. Split: After the file has been uploaded successfully, it is split up into several parts by the split module of the security framework. The data owner is then asked to select the number of clouds to which he wants to upload his data and depending upon this number, the number of file parts is selected. The minimum number of file parts is always three;
- iii. Encryption: Once the file has been split into different parts, each file part is encrypted separately using BDNA symmetric-key algorithm [16]. BDNA uses a 256-bit key for encryption and here, for each file part a completely new encryption key is used;
- iv. Upload file parts to the clouds: After the successful encryption of the different file parts, each file part is separately uploaded to a different cloud. The parts are distributed randomly, and it is possible that one or more clouds may not receive any part of a given file. Different users can have a file with a same name, so while uploading the files with same names, it may be possible that a file with the same name may already be present on the cloud. This could lead to files being overwritten. So, in order to overcome this issue, we used a file naming scheme, where the filename is appended with the unique timestamp values, so that no two files can have the same name. This file naming process is transparent to the end users. The users only use the original filenames which they have provided at the time of uploading the file to the security framework. Besides these unique file names, a proper indexing of the file parts was completed, so that at times when the original file needs to be reconstructed, it is easier to identify all the file parts;
- v. Download file parts from the clouds: Whenever any user requests a particular file at the security framework's interface, he provides the name of the file. First of all, the security framework checks the metadata associated with that file and checks whether the user is authorized to access that file or not and then the unique filename to download the file parts from different CSPs is used. After downloading, the file parts are provided to the decryption algorithm for further processing;
- vi. Decryption: After downloading all the file parts associated with the requested file, each part is decrypted separately using the corresponding secret key used for encrypting that part. The proposed system model uses BDNA decryption algorithm for the decryption process;
- vii. Merge: After successful decryption of the various file parts, the file parts are provided to the merge module. This module combines all the parts of the file to reconstruct the original file. After merging the file, its file name is again changed to the name provided by the data owner at the time of uploading the file to the security framework;
- viii. Download file: After successful merging of all the file parts, the file is provided to the user. The user can then download the file and can access it;
- ix. Key management: This is one of the important components of the proposed security framework. This component is used for managing the secret keys which are used by the BDNA algorithm. These keys are stored on the local database of the security framework. The strength of any symmetric-key encryption algorithm lies in its secret key; therefore, securing and managing these keys requires the utmost attention. The best way to design any security model is to encrypt the data with a

symmetric-key algorithm and subsequently, encrypt the symmetric keys with an asymmetric algorithm; therefore, the security model proposed in this paper uses the similar approach. The symmetric keys used in BDNA are managed by storing them in encrypted format using Identity Based Broadcast Encryption (IBBE), in which the keys are encrypted with the public keys of all the users whom the access has been granted. The key management is discussed in detail in Section 4;

- x. Metadata: All the data related to the proposed model are stored in this database. It contains information related to all the files, their indexing, their unique filenames, and the clouds on which the parts of each file are stored. In addition to this, it contains the list of all the users who are given access to a particular file, list of data owners, access rights of different users, etc.

3.1.3. The User Level

This level consists of the customers who want to store their data onto the cloud or want to access the data stored on the clouds. Therefore, the user level involves two types of customers, data owners and users:

- i. Data owners: These are the customers who want to store their data onto the cloud servers. The data owners are responsible for providing the list of users whom they want to allow to access the files uploaded by them. They can add a new user to the access list as well as can revoke an already allowed user. The data owners can be any organization whether business or any other who wants to save its data on the cloud;
- ii. Data users: These are the users who want to access the data uploaded by some other users onto the clouds. Before obtaining access to the stored data, they are required to prove their authenticity and are asked to provide an access token with which the secret key is decrypted. If the user is a valid user, he will have the correct access token. The users can be the employees of the organization or the clients with whom the organization wants to share its data.

4. Key Management Using Identity-Based Broadcast Encryption

The strength of a symmetric-key cryptographic algorithm lies in the strength of its key. For BDNA encryption algorithm, we used a 256-bit secret key. It has already been proved that this key size is large enough to make it secure against the brute force attacks. However, a proper key management scheme is required to ensure that the keys are safe enough to be used for encrypting the valuable data [25]. So, the keys need to be protected from unauthorized and malicious users. Therefore, in this security model we exploited the fact from the literature that the best way to design a security framework is to encrypt the data with a symmetric-key algorithm and then encrypt the secret key used in symmetric algorithm with an asymmetric-key algorithm. This is so because the asymmetric algorithms are slower as compared to the symmetric ones due to the usage of intricate mathematical computations [26], and hence are not recommended to be used in situations where large amounts of data are to be protected, as they would take a lot of time to encrypt it. Time plays a crucial role in today's world; therefore, symmetric-key cryptographic algorithms are preferred over the asymmetric ones for encrypting huge amounts of data. However, in case of symmetric-key algorithms, the whole security lies in the key, so it should be protected and distributed with the utmost care. The key size is small as compared to the data size; it is preferable to encrypt them using an asymmetric algorithm, as they involve complex mathematical expressions and offers a higher security.

With these considerations in mind, a public key encryption scheme that uses Broadcast Encryption (BE) was used for encrypting the secret keys of BDNA algorithm. By using BE, senders can proficiently broadcast their ciphertext to multiple receivers in such a way that only those receivers who have been authorized can decrypt the ciphertext. Generally, BE approaches can be categorized into two groups: the symmetric-key BE and asymmetric-key BE. In the former technique, only a trusted center creates the private keys of all users

and using these keys it broadcasts the data to the authorized users [39,40]. This trusted center is a single point of failure for this technique as if this center fails the working of the whole broadcast system will collapse [28–30]. In contrast to this, the asymmetric-key or the public-key BE scheme does not incur this type of failure as in this scheme the public keys are used for encryption that are not required to be kept secret. However, the disadvantage of public key BE is that it incurs greater computational costs as compared to symmetric-key BE as it involves more complex mathematical expressions [28–30]. To balance the computational costs and the advantages of public-key BE, we have used it for the encryption of cryptographic keys and not the actual data as the size of the keys is comparatively smaller than the size of the actual data.

The public key BE is an ideal scheme for broadcasting the messages securely among a group of people, but since it is a public key cryptographic technique it requires the distribution of public key certificates for all the users. Therefore, to overcome this issue of distribution of certificates, the secret keys of the BDNA algorithm are encrypted using a variant of BE called Identity-Based Broadcast Encryption (IBBE) [31,32]. This encryption technique uses the identity of a user as its public key and there is no need for the distribution of public key certificates [31]. Therefore, it is easy to compute the public key of the users whose identity related information is known.

The public key BE must satisfy two vital security constraints. The first one is that it should be entirely collusion resistant which means that even if all the unauthorized users, who are not the part of chosen receiver set collude, they would not be able to generate the original message. The second constraint is of stateless receivers [41,42] which means that the existing authorized users are not required to update their secret keys when other users are revoked or added to the broadcast system. The IBBE technique [27,29] used in our framework satisfies both of these constraints.

Basically, all the broadcast encryption algorithms came into existence for broadcasting the encrypted data among group of users, but we exploited this concept to be used with the secret keys instead of actual data. In our proposed security model, the secret keys used by the BDNA algorithm are saved by the security framework in its local database. These keys are encrypted with the public keys of all the users who need to access the data and whose information is provided by the data owner at the time of file uploading. To make it more realistic, the concept of identity-based encryption [24] was used as it uses the identity of a user as its public key and the public key certificates distribution is not required.

In this research work, a special version of IBBE was used which is based upon the techniques proposed by [27–30]. The 256-bit secret key and the value of N are encrypted using a group key which is extracted from the public keys, i.e., the identities of all the users who are allowed to access a particular file. Instead of broadcasting this key to the users, the key is saved on the security framework's storage space and the users are instead provided with a group key.

4.1. Preliminaries

Before discussing the actual technique, first of all the basic preliminaries involved in these techniques are discussed.

- i. The BDNA secret key is denoted by K_s , random number is N , the overall key that needs to be encrypted is $K_{BDNA} = \{K_s, N\}$;
- ii. Bilinear Pairing: Suppose G_1 , G_2 and G_T are three cyclic groups having same prime order p . Then, the bilinear map can be defined as a mapping $e: G_1 \times G_2 \rightarrow G_T$ where $G_1 = G_2$. The bilinear maps have the following properties:
 - Bilinearity: For every $P \in G_1$ and $Q \in G_2$ and $x, y \in \mathbb{Z}_p^*$ $e(P^x, Q^y) = e(P, Q)^{xy}$;
 - Non degeneracy: $e(P, Q) \neq 1$ for P and Q that belongs to G_1 and G_2 , respectively;
 - Computability: For given $P \in G_1$ and $Q \in G_2$, it is computationally efficient to obtain $e(P, Q)$.

We assumed that the bilinear pairings are symmetrical, i.e., $G_1 = G_2$; therefore, in the rest of the paper we denote G_1 and G_2 with G ;

- iii. Bilinear Diffie-Hellman Assumption: By making use of the aforementioned notations, the bilinear Diffie Hellman problem can be defined as computing $e(P, Q)^{xyz} \in G_T$ given a generator P of G and components P^x, P^y, P^z for $x, y, z \in Z_p^*$. A correspondent formulation of this problem is computing $e(X, Y)^z$ provided a generator P of G and elements X, Y, P^z in G . There is an advantage $e(k)$ for an algorithm \tilde{A} to solve a bilinear group $\langle a, G, G_T, e \rangle$, Diffie-Hellman Problem where k has been defined as the security parameter, if $\Pr \left[\tilde{A} \left(k, G, G_T, X, Y, P^z \right) = e(X, Y)^z \right] \geq e(k)$. If there is a negligible advantage for any polynomial-time algorithm to decipher the Diffie-Hellman assumption having the security parameter k , the bilinear group $\langle a, G, G_T, e \rangle$ is said to fulfill the BDH assumption;
- iv. Identity-based Broadcast Encryption with Privacy Preserving: In this scheme, the identities of the recipients are preserved. This means that the group of receivers are unaware of the identities of other receivers who are also receiving the same message. It is basically a four-tuple algorithm which is used for encrypting the message and broadcasting it to a group of users. The various algorithms involved in this scheme are described as follows:
 - $\text{Setup}(k)$: This algorithm takes the security parameter (k) as the input and the outputs MK and PK as Master Secret Key and Master Public Key, respectively. MK is provided to PKG as input and PK is kept public;
 - $\text{Extract}(ID_i, MK)$: It takes user's identity ID_i and Master Secret Key MK as inputs and produce the user's Private key Pr_i as the output;
 - $\text{Encrypt}(I, PK)$: This algorithm takes the Master Public key PK and a group of receivers' identities, denoted by a set $U = \{ID_1, \dots, ID_n\}$ as input and outputs SK , i.e., the encryption of K_{BDNA} . For this, first of all, the algorithm yields a pair (HDR, Gr_{key}) where $Gr_{key} \in G_T$ and is a group key for encrypting the broadcast message. Whenever there is K_{BDNA} that needs to be shared among a group of users in I , the algorithm produces (HDR, Gr_{key}) using PK and U and then encrypts the K_{BDNA} with this group key Gr_{key} to obtain ciphertext CK and broadcasts it by encapsulating it in the body of broadcast message $SK = (HDR, CK)$ where HDR is the header of SK . This approach preserves the privacy of the recipients; therefore, unlike previous approaches, the identity set U is not included in the broadcast message which would otherwise reveal the list of receivers to the public;
 - $\text{Decrypt}(ID_i, Pr_i, SK)$: The decryption algorithm takes the user's identity ID_i and associated private key Pr_i and the header HDR from SK as the input and generates group key Gr_{key} , from which the broadcast message is decrypted.

4.2. Privacy Preserving IBBE for Securing BDNA Keys

In the proposed security model, the data are encrypted using BDNA symmetric key algorithm. The keys used by this algorithm are secured using the special version of Privacy Preserving IBBE encryption. Since in the proposed approach, the BDNA secret keys are stored by the security framework on its local database, these keys are stored in encrypted format so that no one can gain access to these keys. The keys are encrypted using privacy preserving IBBE encryption. A special version of IBBE was used in this approach because generally IBBE has been designed for encrypting the messages that are to be broadcast to a group of users, but in the proposed approach we are not broadcasting our keys to any group of users. Rather, IBBE is being used to encrypt the BDNA key (K_{BDNA}) with the group key of all the users who are allowed to access a particular file that has been encrypted using this is K_{BDNA} . This group key (Gr_{key}) has been calculated using the identities of all the users in the group. Therefore, the concept of broadcast encryption is used for sharing the data among a group of users, without actually broadcasting the data or the keys.

Whenever a user needs to access any file, after authentication he needs to provide a private key Pr_i corresponding to his identity ID_i . Then, the security framework's key management module uses HDR along with these credentials, to generate the Group key Gr_{key} which is used to obtain the BDNA secret keys, which can then finally decrypt the original data which has been requested by the particular user.

Construction

In this section, we provide the construction for IBBE that can be used to secure the BDNA keys. The encryption scheme ensures the user's privacy as no user in the group is aware of any other user in the group. The detailed construction of all the algorithms described in the preliminaries' section is as follows:

- i. Setup(k): From a known security parameter k , a bilinear map group scheme $S = (a, G, G_T, e)$ is created such that $|a| = k$. In addition to this, a random choice is made for a generator $P \in G$ and a secret value $s \in Z_p^*$. After that, the selection is made for the two cryptographic hash functions $H_1: \{0, 1\}^* \rightarrow G$ and $H_2: G_T \rightarrow Z_p^*$. The public parameters of the system are (P, S, H_1, H_2) . The master secret key $MK = s$, and master public key $PK = P^s$;
- ii. Extract (ID_i, MK): This algorithm takes as the input the master secret key MK and the user's public key, i.e., is the identity of the user ID_i . and yields the private key of the user:

$$Pr_i = H_1(ID_i)^s = \theta_i^s \quad (1)$$

- iii. Encrypt (I, PK): Given the System Public Parameters $PK = P^s$ and set of users $U = \{ID_1, \dots, ID_n\}$, the IBBE runs encryption algorithm with the following steps:

- $\forall ID_i \in U$, calculate:

$$\theta_i = H_1(ID_i) \quad (2)$$

- Chose a random $p \in Z_p^*$ and $\forall ID_i \in U$ calculate:

$$d_i = H_2(e(\theta_i^p, P^s)) \quad (3)$$

- Chose a random $k \in Z_p^*$ and compute a group key:

$$Gr_{key} = e(P, P)^k \quad (4)$$

- Select a random $\delta \in Z_p^*$.
- Calculate $HDR = (C_1, C_2, C_3)$ where:

$$C_1 = P^p, C_2 = (P^\delta)^k, C_3 = \left\{ c_i = \left(P^{1-\frac{1}{\delta}} \right)^{\frac{1}{d_i}} \right\}_{ID_i \in U} \quad (5)$$

Finally, this algorithm generates (HDR, Gr_{key}) and this Gr_{key} is used to encrypt K_{BDNA} and generates $CK = Gr_{key} \oplus K_{BDNA}$ which is then saved in the database encapsulated in HDR as Secret Key $SK = [HDR, CK]$. Therefore, each entry in the key management's database is stored as SK.

It should be noted that whenever there is change in the list of users U that are allowed to access the file, the value of k and δ are changed to maintain forward and backward secrecy of the messages. This would change the values of HDR as well as Gr_{key} .

- i. Decrypt (ID_i, Pr_i, SK): whenever the user needs to access a file, after authentication, he is prompted to enter his private key Pr_i corresponding to his identity ID_i . This Pr_i and $SK = [HDR, CK]$ act as input to the decryption algorithm. Using this Pr_i and HDR, Gr_{key} is recomputed, which in turn decrypts the K_{BDNA} encapsulated in HDR. The decryption process for K_{BDNA} involves the following steps:

- Calculate:

$$d_i = H_2(e(Pr_i, C_i)) = H_2(e(\theta_i^s, C_1)) \quad (6)$$

- Recover c_i from C_3 and calculate:

$$e(C_2^{-1}, c_i^{d_i}) \times e(P \times C_2) = e\left(\left((P^\delta)^k\right)^{-1}, \left(\left(P^{1-\frac{1}{\delta}}\right)^{\frac{1}{d_i}}\right)^{d_i}\right) \times e\left(P, (P^\delta)^k\right) \quad (7)$$

$$= e(P, P)^{-k(\delta-1)} \times e(P, P)^{k\delta} = Gr_{key}$$

$$Gr_{key} \oplus CK = K_{BDNA} \quad (8)$$

This K_{BDNA} can be used further for decrypting the requested file which has been encrypted using BDNA cryptography. The decryption process ensures that only the users in the access list U can successfully compute Gr_{key} and decrypt K_{BDNA} successfully.

5. Implementation

The very backbone of our entire research work is securing the data which is required to be stored on the clouds. So, the most important thing was setting up the clouds for our experimentation and the proposed framework exploits the concept of storing data on multiple clouds instead of using just one cloud; therefore, we used Dropbox, Mediafire, Google Cloud, One drive, and Amazon Cloud Drive for the implementation of our proposed security model. The API's and the access tokens were taken from these clouds in order to access these clouds in an authentic manner. The entire interaction mechanism with the clouds was completed using a web service. The web service was made and configured in ASP. Net using Visual Studio 2017. The entire security framework including the encryption/decryption algorithm as well as the key management component was implemented as a web application in Java using Netbeans. Both Data Owner and Data users were operated on a Windows 7, 64-bit machine. The machine uses an Intel® core i5 processor 3210M @ 2.67 Ghz, 4 GB RAM. The following sub-section explains the overall working of our proposed security model with the help of the Algorithms 1 and 2 and screenshots.

5.1. Algorithm for File Uploading by the Data Owner

Algorithm 1 explains the overall working of the file upload process. After successful authentication, the data owner is required to upload the file as well as the list of users who can access the file, at the security framework's (web application) interface. After that, the framework calculates the size of the file and splits the file into different parts depending upon the number of clouds to be used. Then, the file parts are assigned with the appropriate indexes for the easy reconstruction of file from these parts at the times of file downloading. Then, the file parts are encrypted using the BDNA encryption algorithm. The cryptographic keys used for encrypting each file part are further encrypted using the IBBE technique. Then, the filename is modified by appending the timestamp values at time of uploading the file parts to the clouds. In the final step, the file parts are uploaded to different clouds and the file-related indexes and encrypted keys are stored on the local database of the security framework.

Algorithm 1 File Uploading

Input: Any File F

Output: Encrypted File parts $E(F_1), E(F_2), E(F_3), E(F_4), E(F_5)$

-
- Step 1: Upload a File (F) to the security framework and provide list of users set $U = \{ID_1, \dots, ID_n\}$ who can access the file
- Step 2: Calculate the File Size F_{size}
- Step 3: Split or divide the F_{size} with minimum of three parts
- Step 4: Assign the proper indexes to the file parts (F0, F1, F2, F3, F4, ... , Fn)
- Step 5: Generate a different 256-bit Key K_s and N for each file part.
- Step 6: Encrypt each file part of the sliced file with BDNA algorithm to obtain: $E(F_0), E(F_1), E(F_2), E(F_3), E(F_4), \dots, E(F_n)$ And Encrypt $K_{BDNA} = (K_s, N)$ with public keys or identities of all the users in U, using Identity Based Broadcast Encryption to obtain $E(K_{BDNA})$.
- Step 7: Associate with each file part the unique timestamp $E(F_0).T, E(F_1).T, E(F_2).T, E(F_3).T, E(F_4).T, \dots, E(F_n).T$ And Upload the encrypted file parts to the multiple cloud Servers
- Step 8: Save the entire file indexing information as well as the encrypted keys on the local database of the Security Framework.
- Step 9: End
-

5.2. Algorithm for File Downloading by the Data Users

The stepwise procedure involved in downloading the file by the users is presented in Algorithm 2. For downloading the file, first of all, there is an authentication check for the user. After its successful verification, the user is required to provide his private key along with the name of the file which he wants to download. From metadata, all the file-related information is collected, and the file is searched in different clouds. Then, the encrypted K_{BDNA} keys associated with the various file parts are fetched and decrypted using the private key of the user using the IBBE approach. Then, using these K_{BDNA} keys the file parts are decrypted. Later on, the various file parts are merged, and the file is provided to the user for downloading. If the user has provided the wrong private key or he is not authorized to access the file, in these cases also he will be provided with a file with the same name which he can download, but the file will contain garbage information.

Algorithm 2 File DownloadingInput: File (F) and User's Private key Pr_i

Output: File (F) after Decryption and merging

-
- Step 1: Enter the File name (F) and User's Private Key Pr_i
- Step 2: From the metadata obtain name of File parts corresponding to the File (F) and check the access information of that File
- Step 3: Perform a search with the Filename in each of the corresponding multi-cloud storage service provider directory (F0.T), (F1.T), (F2.T), (F3.T), (F4.T), ... , (Fn.T) and acquire the encrypted file parts $E(F_0), E(F_1), E(F_2), E(F_3), E(F_4), \dots, E(F_n)$
- Step 4: Fetch the $E(K_{BDNA})$ related to these file parts
- Step 5: Use the Private key Pr_i to decrypt the K_{BDNA}
- Step 6: Using K_{BDNA} , decrypt all the file parts $D[E(F_0)], D[E(F_1)], D[E(F_2)], D[E(F_3)], \dots, D[E(F_n)]$ and obtain (F0, F1, F2, F3, F4, ... , Fn)
- Step 7: Merge all the File parts to form File (F)
- Step 8: Download the File F
- Step 9: End
-

5.3. Working

In this section, we present the working of the proposed security model using various screenshots. The proposed model is divided into three levels: the cloud level, the security framework level and the user level. First of all, we show the working of the security framework whose various components are maintained by the admin. Admin is responsible for managing all the customers whether data owner or users. Then, the working of the user

level has been discussed which consists of access rights and privileges of these customers. In the end, screenshots of storage clouds have been shared, which show the file parts saved on them:

- i. **Admin:** Admin is responsible for managing the overall security framework level. Admin is responsible for granting privileges to all the clients whether they are data owners or users. Admin allows the data owners to upload as well as delete the files. Besides this, the data owners are granted permission to manage the users who can access the files uploaded by them. On the other hand, users can only view the files. The admin can revoke these permissions from anyone at any time. Figures 2 and 3 represents the functioning of the admin;

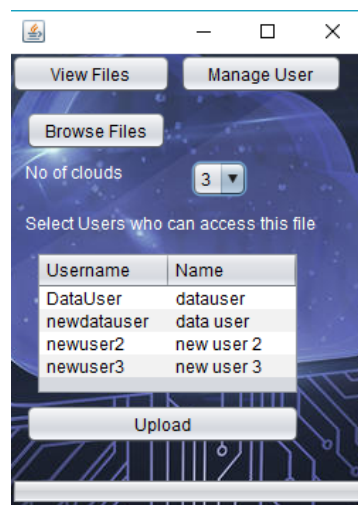


Figure 2. Admin homepage.

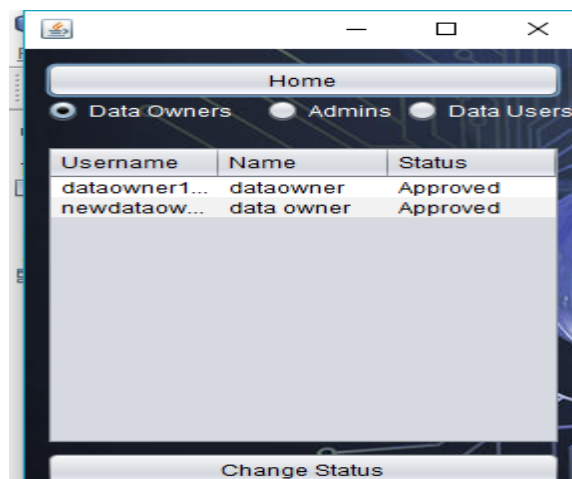


Figure 3. Manage users by admin.

- ii. **Data Owner:** Data owner can upload the files to the security framework's interface as well as manage the access of his files to various users. He can grant the access as well as revoke it from the other users. He can further delete the file whenever required. The various privileges granted to the data owners have been show in the Figures 4–9;

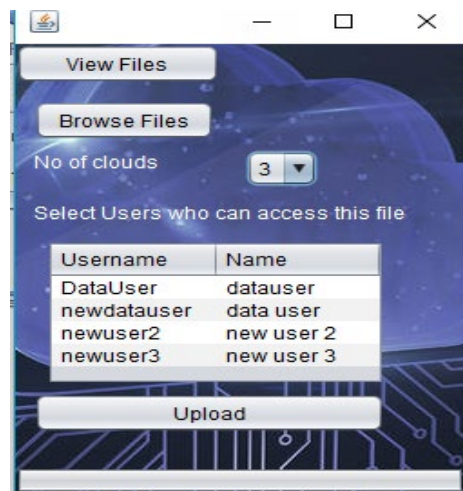


Figure 4. Data owner homepage.

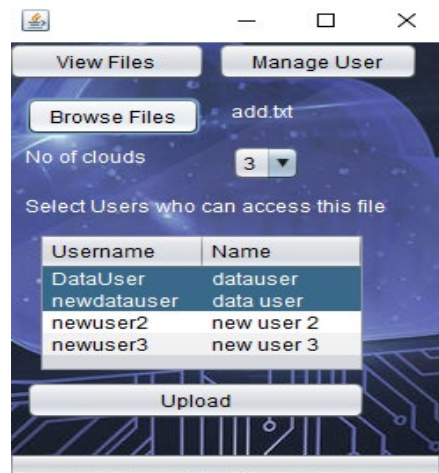


Figure 5. Selecting file and users for access control.

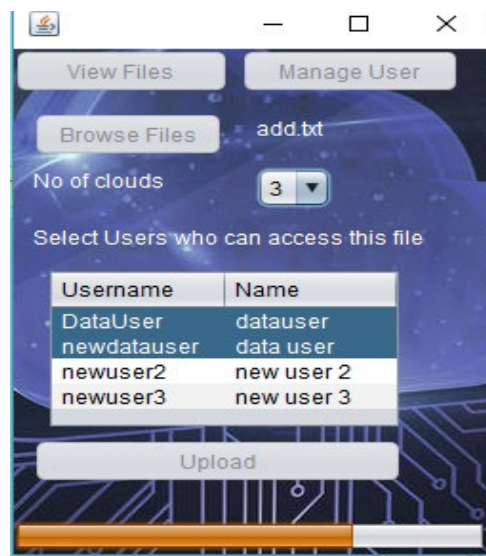


Figure 6. Uploading file.

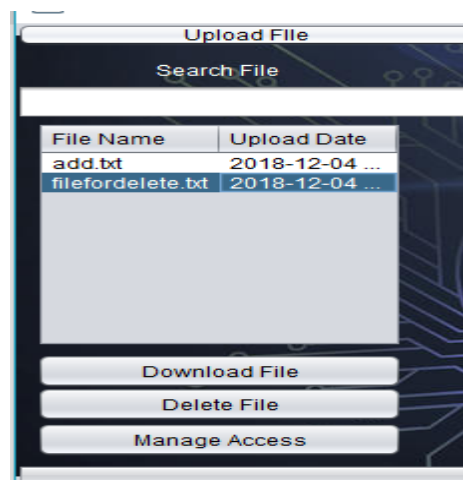


Figure 7. Deleting file.

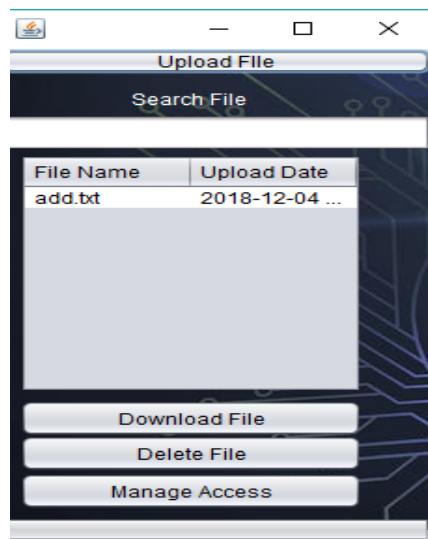


Figure 8. View file by data owner.

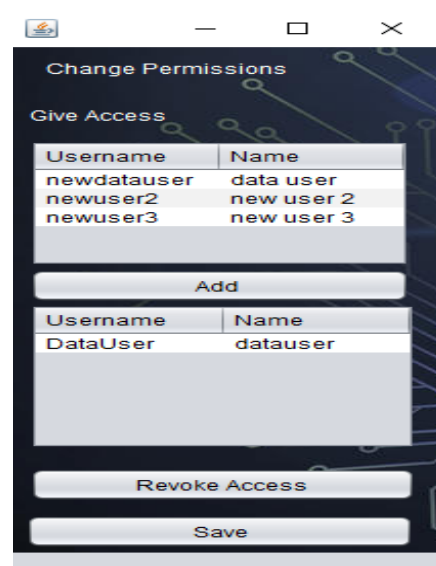


Figure 9. Grant/revoke privileges.

- iii. Users: Data users can only view and download the file whose access is provided to them by the data owners. He cannot delete or modify the file. The users cannot view the other users who are authorized to access the same file. The various privileges granted to the users are shown in Figures 10–12;

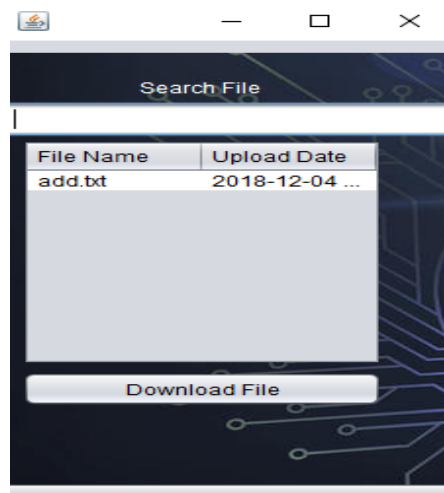


Figure 10. View file by user.

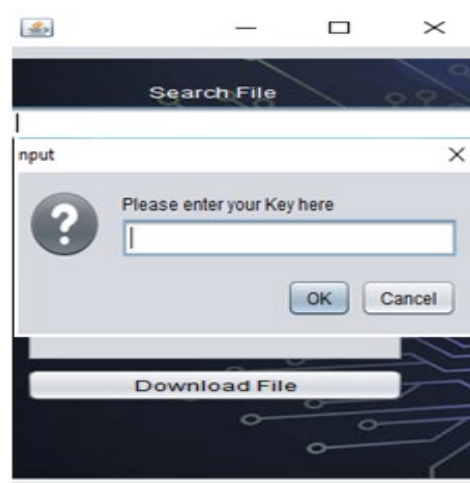


Figure 11. Enter the private key.

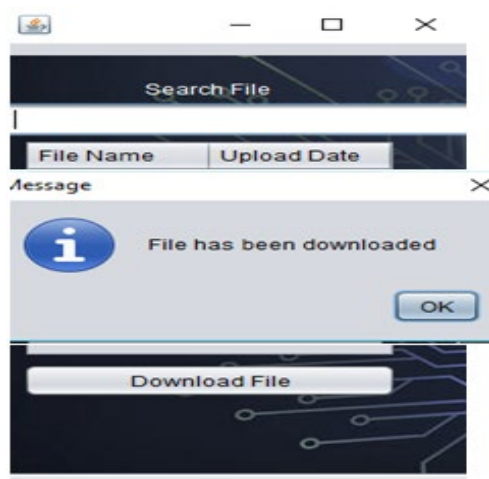


Figure 12. Download file.

- iv. **Cloud Level:** The security model proposed in this Chapter involves-multi cloud storage i.e., multiple clouds have been used for storing the data. In this Section, we have shared the screenshots of two cloud storages (Figures 13 and 14) among the five clouds we have used for our approach [33–38,43,44]. The screenshots depict the various file parts of same file stored on these clouds and the filenames have unique timestamp values attached to them. These screenshots confirm the splitting of the file and their storage on different clouds.

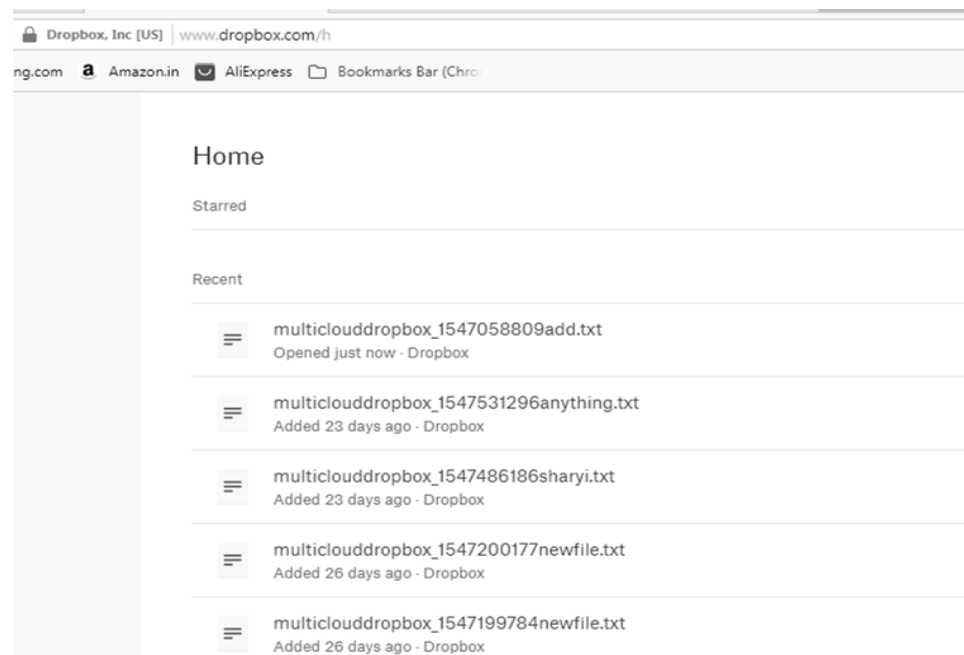


Figure 13. File parts stored in Dropbox.

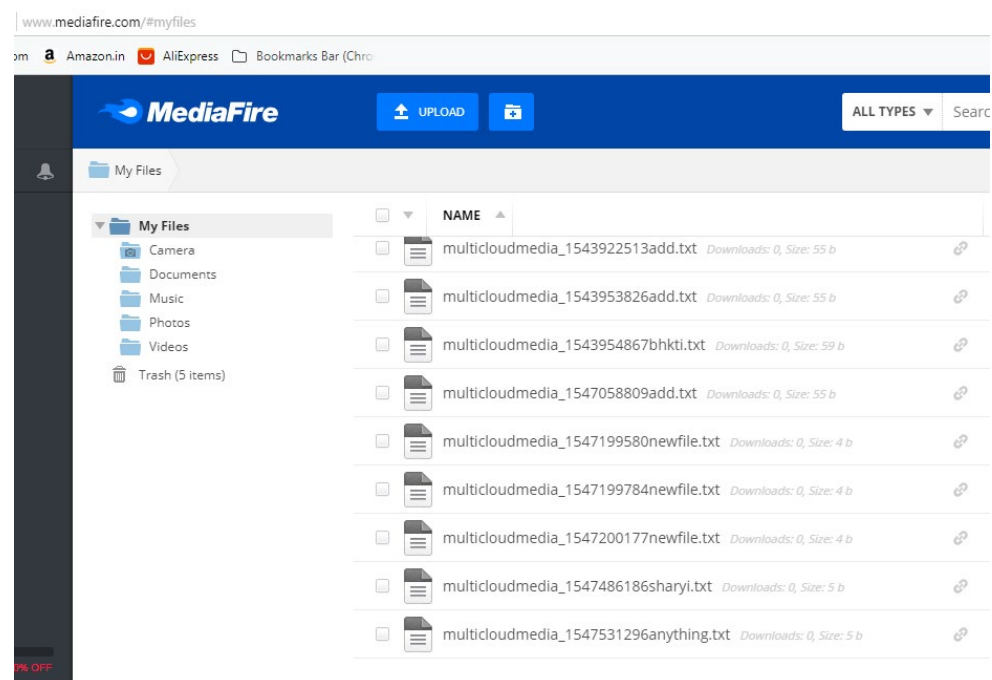


Figure 14. File parts stored in MediaFire.

6. Security Analysis

In this section we prove the security of this hybrid approach. The following proofs illustrate that the proposed system offers semantic security and confidentiality (Indistinguishable Chosen-Ciphertext Attacks secure) against a static adversary provided that the BDH assumption is true. For proving the security, a game between the adversary AD and Challenger CH was conducted.

Confidentiality: Suppose there is a security parameter k , and there exists a polynomial time adversary AD , who has an advantage $E(k)$ against the proposed system. Assume that AD queries for the private key extraction Q_p times and for the hash value H_2 , Q_{H_2} times. Then there exists an algorithm CH which resolves the BDH problem with advantages

$$Adv_{CH}^{ind-smid-cpa}(k) \geq \frac{E(k)}{Q_{H_2}}.$$

Proof: Let $(P, P^x, P^y, P^z) \in G$ be any random instance of BDH algorithm which has been distributed uniformly and has been provided to algorithm CH as an input. In order to come across the solution $(P, P)^{xyz}$, CH simulates the challenge for AD as follows:

Init: CH runs AD and obtains set $U^* = \{ID_1, \dots, ID_n\}$ as a target which AD wants to attack.

Setup: CH sets $PK = P^z$ the public key of the system and provides system parameters $\langle a, G, G_T, e(\cdot, \cdot), P, PK, H_1, H_2 \rangle$ where H_1 and H_2 are the random oracles controlled by CH .

H_1 -query: AD can query the random oracle H_1 at any time. Initially CH maintains an empty list H_1^{list} of quadruples $(ID_i, h_i, \theta_i, Pr_i)$, for responding to the queries. Whenever there is a query from AD , for a hash value of any ID_i , CH responds as follows:

1. If there is list about ID_i in H_1^{list} , then CH replies with $\theta_i \in G$;
2. Or else, CH verifies whether $ID_i \in U^*$ or not and releases a hash query on an identity, according to the following situations:
 - If $ID_i \notin U^*$, then CH selects a random h_i and sets $\theta_i = P^{h_i}$.
 - Otherwise, if $ID_i \in U^*$, then CH selects a random h_i and sets $\theta_i = (P^y)^{h_i}$.

CH adds the tuple $(ID_i, h_i, \theta_i, *)$ to the list H_1^{list} and replies with θ_i .

Phase 1: AD can then submit queries to the random oracle H_2 . It is also be capable of send extraction queries related to private key.

Extraction query: AD can also query for the extraction identity strings. For an input string ID_i for private key extraction, CH answers as follows:

1. If an extraction query on ID_i has already been issued by AD , then CH responds with the consequent Pr_i in H_1^{list} ;
3. Otherwise, if a hash query for H_1 on ID_i has been issued by AD , CH calculates the private key using the corresponding h_i using the following procedure:
 - If $ID_i \notin U^*$, then CH calculates $Pr_i = (P^z)^{h_i} = \theta_i^z$. CH , then includes the private key Pr_i in the related tuple on the list H_1^{list} as the $\langle ID_i, h_i, \theta_i, Pr_i \rangle$ and replies with the Pr_i .
 - If $ID_i \in U^*$, then CH reports abort.

H_2 -queries: The H_2 oracle can be queried at any time by the adversary AD . Again CH , maintains an empty list initially for responding the queries. The empty list is defined as $H_2^{list} = \langle ID_i, X_i, d_i \rangle$. Whenever AD makes a query request for the hash value of any Y_i , which is the pre-image of d_i , CH replies as follows:

1. If a query X_i is already present in H_2^{list} , then CH replies with d_i as a hash value;
4. Else, CH selects a random value $d_i \in Z_p^*$ and includes the tuple $\langle ID_i, X_i, d_i \rangle$ into the hash list H_2^{list} . Afterwards, CH responds by sending d_i as a hash result of X_i , i.e., $H_2(X_i) = d_i$.

Challenge: Following the completion of Phase 1, CH randomly selects p, k, δ . It also randomly selects $d_1^*, \dots, d_n^* \in Z_p^*$ and defines ciphertext as follows:

1. CH calculates the group key $Gr_{key} = e(P, P)^k$;
5. CH also calculates header as $HDR^* = \langle C_1 = P^{xp}, C_2 = P^{\delta k}, C_3 = \left\{ \left(P^{1-\frac{1}{\delta}} \right)^{\frac{1}{d_i}} \right\}_{ID_i \in U^*} \rangle$.
6. CH then randomly chooses $b \leftarrow \{0, 1\}$ and sets $Gr_{key_b} = Gr_{key}$. It then allocates a random value to $Gr_{key_{1-b}}$ and provides $(HDR^*, Gr_{key_0}, Gr_{key_1})$ as a challenge to the adversary AD.

Phase 2. AD adaptively requests for the extraction of private key and hash queries, and the challenger CH replies in the same way as in phase 1.

Guess. As a final point, AD guesses $b' \in \{0, 1\}$. CH, at this moment selects a random tuple $\langle ID_j, X_j, d_j \rangle$ from the list H_2^{list} and calculates $T_j = X_j^{p^{-h_j}}$ with a corresponding value h_j and p . CH then produces T_j as the solution to the given instance of BDH problem.

In the aforementioned game, the real attack environment is simulated by CH.AD and is not able to figure out that it is not the real environment but its simulation. In the real system, AD is required to guess the correct $X_j = e(\theta_j^s, C_1)$ to retrieve the key d_j because d_j is just calculated using the hash function H_2 . Therefore, AD will also query the exact X_j with the probability $\in \binom{k}{k}$, in the simulation environment. Subsequently, towards the end of the simulation, X_j will emerge in some tuple H_2^{list} with the probability $\in \binom{k}{k}$. Hence, CH can yield the correct solution T_j with at least $\in \binom{k}{k} / q_{H_2}$.

7. Discussion of Results

The security analysis presented in the previous section proves that the proposed framework is secure against Indistinguishable Chosen-Ciphertext Attacks. This is proved through game theory in which adversary is given access to the decryption system where he can submit any ciphertext of his choice to attain its plaintext. The adversary tries to compute the private key using this information. However, the results prove that in our framework the adversary fails to gain access to the secret keys. Hence, the proposed framework is immune against malicious users' attacks.

The existing algorithms with similar architecture discussed in this paper do not resolve the filename conflicts, that is if the name of the file being uploaded is same as some other file saved earlier over the cloud, it would result in over writing of the file. The implementation section presents snapshots of files saved on the cloud (Figures 13 and 14). It is evident from those snapshots that each file has timestamp associated with it, which make each filename unique. Thus, the proposed framework resolves these filename conflicts.

Most of the similar architectures have not presented any key management techniques for securing the encryption keys. In this framework, in addition to the encryption of data, a novel key management technique has also been presented to secure the encryption keys.

Lastly, all of these techniques are not fully automated as they require a response from the data owner or admin in one way or another. The framework presented in this paper is fully automated (presented in the implementation section) as all the authorized users can access the data without any further approvals by the admin or data owner. Thus, reducing the additional overheads on data owners.

8. Conclusions

In this paper, a multi-cloud security framework was proposed and implemented. The major goal of this framework is to prevent users' data from untrusted CSPs who can collude with other malicious users and can hand over users' data to these malicious users for their profitable interests. For achieving this goal, the security framework splits the user's data into several parts, encrypts these parts using BDNA and then, uploads these encrypted data parts to different CSPs so that no single cloud has the entire user's data available with them. Secondly, this framework provides an efficient mechanism for securing the secret

keys generated by BDNA. Since, the strength of the symmetric key algorithm depends upon its key; therefore, the secret keys should also be secured. For this, IBBE scheme was used and the results proved that the proposed scheme is secure against Indistinguishable Chosen-Ciphertext Attacks. Further, the working of the proposed security framework was explained with the help of algorithms and snapshots. The proposed framework can be used in all application areas (such as business organization, finance companies, academic organizations, etc.) that use the cloud for the storage of their personal information and also for the storage of pervasive information that is generated from diverse everyday devices. Since research is a never-ending process, the completion of one research work opens up doors for future enhancements of the completed work. In the proposed multi-cloud security framework, the maximum cloud limit was set to five due to the availability of access tokens from the clouds, and therefore in the future the number of clouds could be increased to any number. Further, the parameter of data availability could also be included in the multi-cloud framework, though most of the cloud providers claim the availability of eleven nines (99.99999999%).

Author Contributions: Conceptualization: M.S., S.B. and S.S.; methodology: M.S., S.B., S.S. and M.S.M.; validation: M.S., S.B., S.S., M.S.M. and M.A.; formal analysis: M.S., S.B., S.S., M.S.M. and M.A.; investigation: M.S., S.B., S.S. and M.S.M.; resources: S.B., M.S.M. and M.A.; data curation: S.B., S.S., M.S.M. and M.A.; writing—original draft preparation: M.S., S.B. and S.S.; writing—review and editing: M.S., S.B., S.S., M.S.M. and M.A.; visualization: M.S., S.B., S.S., M.S.M. and M.A.; supervision: S.S. and S.B.; project administration: S.B.; funding acquisition: M.S.M. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by King Saud University, RSP2022R459.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The study did not report any data.

Acknowledgments: We deeply acknowledge King Saud University for supporting this study Research Supporting Project number (RSP2022R459), King Saud University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bohli, J.-M.; Gruschka, N.; Jensen, M.; Iacono, L.L.; Marnau, N. Security and Privacy-Enhancing Multicloud Architectures. *IEEE Trans. Dependable Secur. Comput.* **2013**, *10*, 212–224. [[CrossRef](#)]
2. Sathyanarayana, T.V.; Sheela, L.M.I. Data Security in Cloud Computing. In Proceedings of the 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), Chennai, India, 12–14 December 2013; pp. 822–827. [[CrossRef](#)]
3. Khan, A.M.; Ahmad, S.; Haroon, M. A Comparative Study of Trends in Security in Cloud Computing. In Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, India, 4–6 April 2015; pp. 586–590. [[CrossRef](#)]
4. Sirohi, P.; Agarwal, A. Cloud Computing Data Storage Security Framework Relating to Data Integrity, Privacy and Trust. In Proceedings of the 2015 1st International Conference on Next Generation Computing Technologies (NGCT), Kobe, Japan, 18–20 November 2015; pp. 115–118. [[CrossRef](#)]
5. Warhade, R.G.; Vankudothu, B. Enhancing Cloud Security Using Multicloud Architecture and Device Based Identity. In Proceedings of the 2015 7th International Conference on Emerging Trends in Engineering & Technology (ICETET), Dehradun, India, 4–5 September 2015; pp. 34–39. [[CrossRef](#)]
6. Kshetri, N. Privacy and security issues in cloud computing: The role of institutions and institutional evolution. *Telecommun. Policy* **2013**, *37*, 372–386. [[CrossRef](#)]
7. Cachin, C.; Keidar, I.; Shraer, A. Trusting the cloud. *ACM SIGACT News* **2009**, *40*, 81–86. [[CrossRef](#)]
8. AlZain, M.A.; Pardede, E.; Soh, B.; Thom, J.A. Cloud Computing Security: From Single to Multi-Clouds. In Proceedings of the 2012 45th Hawaii International Conference on System Sciences, Maui, HI, USA, 4–7 January 2012; pp. 5490–5499. [[CrossRef](#)]
9. Kritikos, K.; Zeginis, C.; Iranzo, J.; Gonzalez, R.S.; Seybold, D.; Griesinger, F.; Domaschka, J. Multi-cloud provisioning of business processes. *J. Cloud Comput. Adv. Syst. Appl.* **2019**, *8*, 18. [[CrossRef](#)]

10. Haqiq, A.; Talbi, J. A cloud broker architecture for cloud service selection based on multi-criteria decision making and rough set theory. *Int. J. Comput. Aided Eng. Technol.* **2020**, *13*, 448. [CrossRef]
11. Software Development Company in USA, Simform. 2022. Available online: <https://www.simform.com/> (accessed on 7 July 2022).
12. Bowers, K.D.; Juels, A.; Oprea, A. HAIL: A High-Availability and Integrity Layer for Cloud Storage. In Proceedings of the 16th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 9–13 November 2009; pp. 187–198. [CrossRef]
13. Balasaraswathi, V.R.; Manikandan, S. Enhanced Security for Multi-Cloud Storage Using Cryptographic Data Splitting with Dynamic Approach. In Proceedings of the 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, India, 8–10 May 2014; pp. 1190–1194. [CrossRef]
14. Subramanian, D.; John, F. Enhanced Security for Data Sharing in Multi Cloud Storage (SDSMC). *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 176–185. [CrossRef]
15. Sohal, M.; Sharma, S. Enhancement of Cloud Security using DNA Inspired Multifold Cryptographic Technique. *Int. J. Secur. Its Appl.* **2017**, *11*, 15–26. [CrossRef]
16. Sohal, M.; Sharma, S. BDNA-A DNA inspired symmetric key cryptographic technique to secure cloud computing. *J. King Saud Univ. Comput. Inf. Sci.* **2018**, *34*, 1417–1425. [CrossRef]
17. Fabian, B.; Ermakova, T.; Junghanns, P. Collaborative and secure sharing of healthcare data in multi-clouds. *Inf. Syst.* **2015**, *48*, 132–150. [CrossRef]
18. Althamary, I.A.; Alkharobi, T.M. Secure File Sharing in Multi-clouds using Shamir’s Secret Sharing Scheme. *Trans. Netw. Commun.* **2016**, *4*, 53–67. [CrossRef]
19. Madan, B.B.; Banik, M.; Wu, B.C.; Bein, D. Intrusion Tolerant Multi-cloud Storage. In Proceedings of the 2016 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 18–20 November 2016; pp. 262–268. [CrossRef]
20. Jogdand, R.; Batlad, S.; Gangodkar, D. CSaaS—A multi-cloud framework for secure file storage technology using open ZFS. *Int. J. High Perform. Comput. Netw.* **2016**, *9*, 230. [CrossRef]
21. Indhumathil, T.; Aarthy, N.; Devi, V.D.; Samyuktha, V.N. Third-Party Auditing for Cloud Service Providers in Multicloud Environment. In Proceedings of the 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM), Chennai, India, 23–24 March 2017; pp. 347–352. [CrossRef]
22. Kanna, G.P.; Vasudevan, V. A New Approach in Multi Cloud Environment to Improve Data Security. In Proceedings of the 2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS), Jammu, India, 11–12 December 2017; pp. 7–12. [CrossRef]
23. Viswanath, G.; Krishna, V. Hybrid encryption framework for securing big data storage in multi-cloud environment. *Evol. Intell.* **2020**, *14*, 691–698. [CrossRef]
24. Boneh, D.; Franklin, M. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.* **2003**, *32*, 586–615. [CrossRef]
25. 6 Security Risks of Enterprises Using Cloud Storage and File Sharing Apps, Digital Guardian. 2019. Available online: <https://digitalguardian.com/blog/6-security-risks-enterprises-using-cloud-storage-and-file-sharing-apps> (accessed on 7 July 2022).
26. Derfouf, M.; Mimouni, A.; Eleuldj, M. Vulnerabilities and Storage Security in Cloud Computing. In Proceedings of the 2015 International Conference on Cloud Technologies and Applications (CloudTech), Marrakech, Morocco, 2–4 June 2015. [CrossRef]
27. Hur, J.; Park, C.; Hwang, S.O. Privacy-preserving identity-based broadcast encryption. *Inf. Fusion* **2012**, *13*, 296–303. [CrossRef]
28. Li, M.; Xu, X.; Zhuang, R.; Guo, C.; Tan, X. Identity-Based Broadcast Encryption Schemes for Open Networks. In Proceedings of the 2015 Ninth International Conference on Frontier of Computer Science and Technology, Dalian, China, 26–28 August 2015; pp. 104–109. [CrossRef]
29. He, K.; Weng, J.; Liu, J.-N.; Liu, J.K.; Liu, W.; Deng, R.H. Anonymous Identity-Based Broadcast Encryption with Chosen-Ciphertext Security. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, Xi’an, China, 30 May–3 June 2016; pp. 247–255. [CrossRef]
30. Li, J.; Chen, L.; Lu, Y.; Zhang, Y. Anonymous certificate-based broadcast encryption with constant decryption cost. *Inf. Sci.* **2018**, *454–455*, 110–127. [CrossRef]
31. Sakai, R.; Furukawa, J. Identity-Based Broadcast Encryption; IACR Cryptology ePrint Archive, Paper 2007/217. 2007.
32. Delerablée, C. Identity-Based Broadcast Encryption with Constant Size Ciphertexts and Private Keys. In Proceedings of the 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, 2–6 December 2007; pp. 200–215. [CrossRef]
33. Bharany, S.; Sharma, S.; Badotra, S.; Khalaf, O.I.; Alotaibi, Y.; Alghamdi, S.; Alassery, F. Energy-Efficient Clustering Scheme for Flying Ad-Hoc Networks Using an Optimized LEACH Protocol. *Energies* **2021**, *14*, 6016. [CrossRef]
34. Ünsal, A.; Önen, M. Calibrating the Attack to Sensitivity in Differentially Private Mechanisms. *J. Cybersecur. Priv.* **2022**, *2*, 830–852. [CrossRef]
35. Bharany, S.; Sharma, S.; Bhatia, S.; Rahmani, M.K.I.; Shuaib, M.; Lashari, S.A. Energy Efficient Clustering Protocol for FANETS Using Moth Flame Optimization. *Sustainability* **2022**, *14*, 6159. [CrossRef]
36. Bharany, S.; Sharma, S.; Khalaf, O.I.; Abdulsahib, G.M.; Al Humaimeedy, A.S.; Aldhyani, T.H.H.; Maashi, M.; Alkahtani, H. A Systematic Survey on Energy-Efficient Techniques in Sustainable Cloud Computing. *Sustainability* **2022**, *14*, 6256. [CrossRef]
37. Bharany, S.; Kaur, K.; Badotra, S.; Rani, S.; Kavita; Wozniak, M.; Shafi, J.; Ijaz, M.F. Efficient Middleware for the Portability of PaaS Services Consuming Applications among Heterogeneous Clouds. *Sensors* **2022**, *22*, 5013. [CrossRef]

38. Shuaib, M.; Badotra, S.; Khalid, M.I.; Algarni, A.D.; Ullah, S.S.; Bourouis, S.; Iqbal, J.; Bharany, S.; Gundaboina, L. A Novel Optimization for GPU Mining Using Overclocking and Undervolting. *Sustainability* **2022**, *14*, 8708. [[CrossRef](#)]
39. Halevy, D.; Shamir, A. The LSD Broadcast Encryption Scheme. In Proceedings of the 22nd Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2002; pp. 47–60.
40. Lotspiech, J.B. Broadcast Encryption Versus Public Key Cryptography in Content Protection Systems. In Proceedings of the ninth ACM workshop on Digital rights management, Chicago, IL, USA, 9 November 2009; pp. 39–46. [[CrossRef](#)]
41. Naor, D.; Naor, M.; Lotspiech, J. Revocation and Tracing Schemes for Stateless Receivers. In Proceedings of the 21st Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; pp. 41–62. [[CrossRef](#)]
42. Mihaljevic, M. Reconfigurable Key Management for Broadcast Encryption. *IEEE Commun. Lett.* **2004**, *8*, 440–442. [[CrossRef](#)]
43. Bharany, S.; Sharma, S. Intelligent Green Internet of Things: An Investigation. In *Machine Learning, Blockchain, and Cyber Security in Smart Environments*; Chapman and Hall/CRC: New York, NY, USA, 2022; pp. 1–15. [[CrossRef](#)]
44. Talwar, B.; Arora, A.; Bharany, S. An Energy Efficient Agent Aware Proactive Fault Tolerance for Preventing Deterioration of Virtual Machines Within Cloud Environment. In Proceedings of the 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 3–4 September 2021.