Sergey Y. Yurish
Editor

Advances in Networks, Security and Communications

Volume 3



Sergey Y. Yurish
Editor

Advances in Networks, Security and Communications

Volume 3

International Frequency Sensor Association Publishing

Sergey Y. Yurish
Editor

Advances in Networks, Security and Communications
Volume 3

Copyright © 2022 by International Frequency Sensor Association Publishing, S. L.

E-mail (for orders and customer service enquires): ifsa.books@sensorsportal.com

Visit our Home Page on http://www.sensorsportal.com

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (IFSA Publishing, S. L., Barcelona, Spain).

Neither the authors nor International Frequency Sensor Association Publishing accept any responsibility or liability for loss or damage occasioned to any person or property through using the material, instructions, methods or ideas contained herein, or acting or refraining from acting as a result of such use.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identifies as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

e-ISBN: 978-84-09-47239-0

BN-20221221-XX

BIC: TJK

Contents

Contributors **11**

Preface **13**

1. Novel Results on Blind Symbol Rate Estimation in Digital Satellite Receivers **15**
1.1. Introduction 15
1.2. Signal Model 16
1.3. Cramer-Rao Lower Bound 17
1.3.1. Computation of the Modified CRLB 17
1.3.2. Computation of the Asymptotic CRLB 17
1.4. Estimator Algorithms 20
1.4.1. CCB and ML Approach 20
1.4.2. DF Concept 21
1.5. Numerical Results 23
1.6. Conclusions 26
Acknowledgements 26
References 26
Appendix 28

2. Confidentiality Concerns in Cloud Environments **29**
2.1. Introduction 29
2.2. Overview of Existing Attacks in Cloud Computing 30
2.2.1. Authentication Attacks 30
2.2.2. Man-in-the-middle Attacks 31
2.2.3. Denial of Service Attacks 32
2.2.4. Cloud Malware Injection Attacks 33
2.2.5. Side Channel Attacks 33
2.2.6. Collusion Attacks 33
2.3. Data Confidentiality in Cloud Computing 34
2.3.1. Classification of Solutions Ensuring Data Confidentiality in Cloud Computing 34
2.3.2. Confidentiality Through Biometric Encryption 36
2.3.2.1. Biometric Encryption in Mobile Cloud Computing *36*
2.3.2.2. Privacy-preserving Biometric Identification in Cloud Computing *36*
2.3.2.3. CloudBI *37*
2.3.2.4. BE to Enhance Confidentiality in Cloud Computing *38*
2.3.2.5. Privacy Preserving Security Using Biometrics *38*
2.3.3. Confidentiality Through Homomorphic Encryption 39
2.3.3.1. Cloud El Gamal and Fast Cloud RSA Schemes *39*
2.3.3.2. Fast-variant Cloud Paillier Scheme *40*
2.3.3.3. A Solution Based on Fully Homomorphic Encryption *40*
2.3.3.4. Speedy Cloud-RSA *41*
2.3.3.5. Solution Based on Intel SGX *41*
2.3.4. Confidentiality not by Encryption only 42
2.3.4.1. HPI-Secure *42*
2.3.4.2. K-NN Classifier for Data Confidentiality *43*
2.3.4.3. Steganography-based Solution *44*
2.3.4.4. Confidentiality Preserving Through Encryption and Obfuscation *44*

2.3.4.5. Attribute-based Proxy Re-encryption ... *45*
2.3.4.6. TR-MABE ... *46*
2.3.4.7. Privacy-manager ... *47*
2.3.4.8. Hierarchical Classes of Computing for Privacy Preserving ... *48*
2.3.4.9. Encryption and Trust-based Solution ... *49*
2.3.4.10. Blockchain for Cloud-based Data Privacy ... *50*
2.3.4.11. L-EncDB ... *52*
2.3.4.12. CS-IBE ... *53*
2.3.4.13. GRBAC ... *53*
2.3.4.14. ISRSAC ... *54*
2.3.4.15. Solution Using Modular Encryption Standard ... *55*
2.3.4.16. CPDA ... *56*
2.3.5. Confidentiality According to the Type of Storage ... 57
2.3.5.1. Secret Sharing Scheme ... *57*
2.3.5.2. Scheme for Image Data Privacy ... *57*
2.3.5.3. DCSSP ... *58*
2.3.5.4. CAFE ... *59*
2.3.5.5. ARIANNA ... *60*
2.3.6. Confidentiality Without Encryption ... 60
2.3.6.1. NetDB2-MS ... *60*
2.3.6.2. Multiple Storage Solution ... *61*
2.3.6.3. MONcrypt ... *62*
2.3.6.4. Matrix-based Solution ... *63*
2.4. General Assessment of the Studied Solutions ... 63
2.5. Conclusion ... 67
References ... 68

3. BAC-DC: Blockchain-based Access Control for Decentralized Cloud Data ... 71
3.1. Introduction ... 71
3.2. Background and Related Works ... 72
3.2.1. Meta-key ... 72
3.2.2. Storj ... 73
3.2.3. Blockchain technology for Data Privacy in cloud computing ... 75
3.2.4. BACC (Blockchain-based Access Control for Cloud Data) ... 75
3.2.5. FileCoin ... 76
3.2.6. SIA 77
3.2.7. Discussion ... 77
3.3. Overview of BACC (Blockchain-based Access Control for Cloud Data) ... 79
3.3.1. The Main Components of the System ... 79
3.3.2. System Configuration ... 80
3.3.2.1. Data Upload ... *81*
3.3.2.2. Data Download ... *81*
3.3.3. The Limits of the BACC Scheme ... 81
3.4. Transition from Centralized to Decentralized Storage ... 83
3.4.1. Basic Cloud Computing ... 83
3.4.2. Why Decentralizing the Cloud? ... 83
3.4.3. Decentralized Cloud ... 84
3.5. Proposed Model BAC-DC: Blockchain-based Access Control for Decentralized Cloud Data ... 85
3.5.1. The Main Components ... 85
3.5.1.1. The Nodes ... *85*
3.5.1.2. The Peer-to-peer Communication and Discovery System ... *85*
3.5.1.3. The Reference Table or Metadata ... *86*

3.5.2. BAC-DC: Operating Mode....86
3.5.2.1. Data Upload....86
3.5.2.2. Data Download....87
3.5.3. Model Architecture....88
3.6. Link between Our BAC-DC Scheme and the One Studied BACC....89
3.7. Security Analysis....90
3.7.1. File Partitioning Before Storage....90
3.7.2. End-to-end Encryption....91
3.7.3. Parity Fragments for Redundancy....92
3.7.4. File Verification....92
3.7.5. Data Loss....92
3.7.6. Access Control....92
3.8. Security against Attacks....94
3.8.1. The Google Attack....94
3.8.2. The Sybil Redundancy Attack....95
3.8.3. The Incorrect Distribution....95
3.8.4. Cheating Client Node....95
3.8.5. The Hostage Byte Attack....95
3.8.6. Geppetto's Honest Attack....96
3.9. Advantages and Disadvantages of the BAC-DC Model....96
3.9.1. Advantages....96
3.9.2. Disadvantages....97
3.10. Conclusion....98
References....98

4. Attack Detection in IoT Networks Using Intelligent Classification Mechanisms....101
4.1. Introduction....101
4.2. Security Issues in IoT Networks....102
4.2.1. Physical Threats....102
4.2.2. Cyber Threats....103
4.2.2.1. Denial of Service: DoS / DDoS....103
4.2.2.2. Mirai....103
4.2.2.3. R2L (Remote to Local)....103
4.2.2.4. U2R (User to Root)....104
4.2.2.5. Man-in-the-middle....104
4.2.2.6. Injection Attack....104
4.2.2.7. Password Attack....104
4.3. Defense Solutions Associated with an IoT System....105
4.3.1. Network Based Intrusion Detection System (NIDS)....105
4.3.2. Host Based Intrusion Detection System (HIDS)....106
4.4. Anomaly Detection by Classification....106
4.4.1. Data Classification in AI....107
4.4.2. Classification Types....107
4.4.3. Classification Algorithms....108
4.4.3.1. Linear Models....108
4.4.3.2. Nonlinear Models....109
4.4.4. Anomaly Detection by Binary Classification....109
4.4.5. Detection and Identification of Anomalies by Multiclass Classification....110
4.5. Presentation of Used Tools, Datasets and Methods....111
4.5.1. Tools and Libraries....111
4.5.2. Dataset 1: UNSW-NB15....113
4.5.3. Dataset 2: IoT Network Intrusion Dataset....116
4.5.4. Method....117

4.5.5. Data Pre-processing 118
4.5.5.1. Shape Analysis 118
4.5.5.2. Background Analysis 121
4.5.5.3. Data Preparation 124
4.6. Results and Discussion 125
4.6.1. Evaluation Metrics 125
4.6.1.1. Confusion Matrix 125
4.6.1.2. Metrics 126
4.6.2. Analysis and Interpretation of Results 126
4.6.2.1. Anomaly Detection 126
4.6.2.2. Identification of Attacks 132
4.7. Conclusion 137
References 139

5. Emerging Trends Using IoMT for Healthcare System 141
5.1. Introduction 141
5.2. Literature Review 142
5.3. General Framework of IOMT 142
5.4. IoMT Conceptual Frameworks and Architecture 143
5.5. Technologies & Platforms behind IoMT 144
5.5.1. Device Integration Layer 146
5.5.2. Information Integration Layer 146
5.5.3. Information Extraction or Information Representation Layer 146
5.6. IoMT Communication Technologies 146
5.7. Typical IoMT-based E-health System 147
5.8. Real Time Health Monitoring Systems 149
5.9. Cloud, Fog & Edge Computing 150
5.10. Security Aspects in IoMT Communications 151
References 153

Index 157

Contributors

Cherif Diallo
Laboratoire d'Algèbre, Cryptographie, Codes et Applications (LACCA),
Dept. Informatique, UFR SAT, Université Gaston Berger, BP 234, Saint-Louis, Sénégal
E-mail: cherif.diallo@ugb.edu.sn

Wilfried Gappmair
Institute of Communication Networks and Satellite Communications,
Graz University of Technology, Austria

P. Madhavasarma
SASTRA Deemed University, Thanjavur 613401, Tamilnadu, India
E-mail: spms1975@yahoo.co.in

T. N. Prabakar
Sri Sairam College of Engineering, Bangalore, India

Barbara Suesser-Rechberger
Institute of Communication Networks and Satellite Communications,
Graz University of Technology, Austria

P. Veeraragavan
University College of Engineering, Tindivanam 604001, Tamilnadu, India

Sarah Wayzani
Laboratoire d'Algèbre, Cryptographie, Codes et Applications (LACCA),
Dept. Informatique, UFR SAT, Université Gaston Berger, BP 234, Saint-Louis, Sénégal

Preface

After successful publication of the first two volumes of '*Advances in Networks, Security and Communications'* open access Book Series in 2017 (Volume 1) and in 2019 (Volume 2), and positive feedback from our authors and readers, we have decided to publish the 3rd volume of this Book Series in 2022.

This volume contains 5 chapters written by 7 authors from 3 countries: Austria, India and Senegal. But it is not a simple set of reviews. Each of the chapters contains the extended state-of-the art followed by new, obtained by the authors results, unpublished before.

Chapter 1 presents in details the asymptotic Cramer-Rao lower bound (ACRLB) for carrier-blind non-data-aided (NDA) estimation of the symbol rate. This figure of merit represents the theoretical limit of the jitter variance produced by any estimator in the lower signal-to-noise ratio (SNR) range. It could be observed that the SNR value characterizing the intersection of ACRLB and modified Cramer-Rao lower bound (MCRLB) might be helpful in practice. Furthermore, a novel estimator algorithm has been discussed, which suffers only marginally from error floor effects, although this solution requires a second filter operated in parallel to the receiver matched filter.

The confidentiality concerns in cloud environments are considered in the Chapter 2. The recent solutions concerning this subject have been described by the authors. This will allow to identify a better perspective that will help to develop in future an effective cloud strategy based on existing solutions in order to counter the security challenges regarding data confidentiality.

The blockchain-based access control for decentralized cloud data proposed in Chapter 3 brings many advantages in terms of security. Authors have integrated the blockchain technology into the cloud while emphasizing access control managed by specific nodes called master nodes and not by a central entity. Indeed, it responds to security vulnerabilities identified in existing decentralized cloud computing solutions. Among the many advantages, it is very profitable, emphasizes data confidentiality, offers good redundancy and unlimited bandwidth.

Chapter 4 describes the models and techniques for intelligent classification of network traffic in order to increase accuracy in the anomaly detection and attack identification in IoT environments. The implementation of several artificial intelligence classification models namely a decision tree, random forest, AdaBoost and an artificial neural network for anomaly detection and attack identification in IoT networks have been analyzed and discussed.

Chapter 5 is focused on the emerging trends using Internet of Medical Things (IoMT) for healthcare system. The frameworks, architectures, technologies and platforms of IoMT on the different integration levels are described in the Chapter.

Every chapter in '*Advances in Networks, Security and Communications'* Vol. 3 is independent and self-contained. The book can be useful for post-graduate students, researchers, engineers and scientist working in the appropriate areas.

I shall gratefully receive any advices, comments, suggestions and notes from readers to make the next volumes of '*Advances in Networks, Security and Communications*'open access book Series very interesting and useful.

Dr. Sergey Y. Yurish

Editor
IFSA Publishing *Barcelona, Spain*

Chapter 3
BAC-DC: Blockchain-based Access Control for Decentralized Cloud Data

Sarah Wayzani and Chérif Diallo

3.1. Introduction

Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. It's an umbrella term for anything that involves using hosted services on the internet.

It is clear that cloud technologies have proven a major commercial success over recent years (since the appearance of products and cloud offerings like Amazon EC2 and Microsoft Azure) since it provides cost savings to enterprises thanks to virtualization technologies and opens gates for new business opportunities as well.

However, Cloud Computing has to face several challenges and issues. Storing and processing data out of the boundaries of a company raises security and privacy concerns by itself.

Protection and integrity of data is not a guarantee by any major cloud service provider. Major cloud service providers simply do not put an emphasis on privacy, protection, and integrity of user data. Traditional cloud providers use a Centralised technology where there has to be one trusted entity for data storage. As every trust-based model, it suffers from weaknesses that may harm the data. The centralized cloud is vulnerable to many potential security threats, such as malware, app-flaws, man-in-the-middle attacks that can affect confidentiality of data.

We have previously studied several solutions ensuring the confidentiality of data in the cloud [1]. In fact, we had classified these solutions according to their way of ensuring data

Sarah Wayzani
Laboratoire d'Algèbre, Cryptographie, Codes et Applications (LACCA), Dept. Informatique, UFR SAT, Université Gaston Berger, BP 234, Saint-Louis, Sénégal

protection. Blockchain technology had caught our attention and therefore we had focused our research on articles enforcing the cloud associated with blockchain technology.

We have consequently proposed in this paper a new Scheme that builds a fair ecosystem where everyone's right to privacy is fully protected and integrity is insured by the blockchain technology. The data is protected from abusive cloud provider's activities ensuring fair, decentralized and truly private operations. Indeed, files are not stored in centralized data center instead; they are encrypted, split into pieces, and distributed on a global cloud network. In other words, the technology introduced here breaks data into fragments and scatters them across a variety of cloud-based storage devices, rather than centrally storing in any one cloud. In some ways, decentralized cloud storage works similarly to bit torrents, which seed data on machines that participate in the torrent network. One key difference, however, is that decentralized cloud storage is based on the use of blockchain technology, which helps to ensure the integrity of the data.

This paper is organized as follows. In the next section, we have studied some solutions of decentralized cloud associated with blockchain technology and at the same time we have discussed the scheme chosen for the adaptation in our solution. In Section 3.3, we give an overview of the chosen solution BACC. This is followed by Section 3.4 that presents the transition from centralized to decentralized storage. Section 3.5 discusses the proposed model and its configuration and particularities. In Section 3.6, we expose the link between the studied scheme and the one we proposed. We show the security analysis of the proposed solution in Section 3.7, and then discuss the performance analysis against attacks in Section 3.8. Finally, before concluding, in Section 3.9 we have discussed the advantages and the disadvantages of the proposed model BAC-DC.

3.2. Background and Related Works

In this section, we will highlight some decentralized cloud solutions studied among which we will choose one in reference to our future work. We will also propose a comparison table according to specific criteria that could better explain our approach.

3.2.1. Meta-key

Existing data-sharing strategies are based on centralized cloud-service providers. Data-owners simply authorize the trusted service providers and let them carry out the sharing process. However, in blockchain storage where every node is untrustworthy, existing data-sharing schemes cannot be applied directly. We need a secure data-sharing scheme that can work under blockchain-based storage architecture. Furthermore, if data are shared in their encrypted form, owners must provide their decryption key to others, which either undermines the security of the owner's other data in an untrusted environment or we need a better key-management mechanism to encrypt each piece of data with a separate key. The authors of this solution [2] propose a series of policies as an attempt to solve these problems.

Secure data-sharing of a general cloud storage system is commonly performed in the following way. When any recipient sends a sharing-request, the data-owner encrypts his data and uploads the data cipher-text to cloud-storage. Then, he applies that recipient's public key to encrypt the decryption key and sends the key cipher-text to the recipient through some secure channels. The recipient decrypts the key ciphertext with his private key to get the decryption key and then download the data cipher-text to decrypt. In the untrusted environment we have, the only secure and reliable channel for decryption key transport will be the blockchain, therefore in Meta-key we put the key cipher-text.

Also, on-chain as a part of the meta-data, and use the blockchain for both key-management and key-distribution, and thus comes the name of the proposal. Since the meta-data on-chain are protected by the owner's private key, a secure data-sharing now turns into the generation of a new record of meta-data on chain owned by the recipient for his copy of the shared data.

In a blockchain-based cloud storage, the security of data is both protected by encryption and the conceal of the location. These locations are recorded into meta-data that can only be read by data-owners; therefore, we need to use new decryption key and location for the shared data different from the original copy and only known to the recipient to ensure the level of security for both parties after the sharing. Since the data to be shared are very likely already encrypted and stored in the cloud, asking the data-owner to download, decrypt, re-encrypt (with a new decryption key) and upload again (to a different location) is not only very inefficient, but also poses security risks because the plain-text needs to be recovered during the sharing process. To tackle these issues in this solution, proxy re-encryption is chosen as the foundation of key and cipher-text transformation mechanism.

Drawbacks of the solution: Even if the particularity of Meta-key is to store the decryption keys in the blockchain, we can show that it is a weak point of the solution. In addition, it is a vulnerable solution to brute force attack.

3.2.2. Storj

According to Wilkinson and Lowry [3], it is time for cloud storage systems to become a real cloud. And that's only possible when a massive number of resources are connected in order to form a storage system. With the advent of blockchain, implementation of this idea became possible. In this platform [5], ordinary people share their unused storage hardware space. Storj configures a trust-based cloud storage system between clients and hosts. All clients' data must be encrypted before transmitting on the network. Considering the fact that people may want to store massive files, storing all data on the blockchain can cause a problem called bloating. To solve this problem, only the metadata of each block is stored on blockchain. Metadata holds some essential information, such as a file's location and its hash. Like any other data on the blockchain, metadata must also be encrypted. Storj uses Ethereum blockchain and stores the metadata in Satoshi style, enabling users to retrieve their information in full whenever needed. An application called Metadisk runs on the Storj platform and checks the network periodically to make sure the files stored are

available and untampered. Each file is stored in at least three locations with the option of increasing this amount to meet the client's demands. As a result, even if the original file is destroyed, the data can still be retrieved through one of the copies. If a node or an audit is not available, another copy of the file would be saved on a new node. This process is called proof-of-redundancy [3]. In Storj all files are split into small fractions before being sent on the network. These small fractions are called shards and the process of splitting data into smaller parts is called sharding. With the help of sharding, no single node will have access to a complete copy of a file uploaded on the network. Shards can even be combined with each other when a client is sending smaller files as it helps having them verified at once. Moreover, when important data is being transmitted on the network, sharding can help conceal them by blending them with unnecessary information. Proof of Space (PoS), also known as Proof of Storage or Proof of Capacity (PoC) is, the consensus protocol used in Storj. The amount of dedicated free space of each miner, has a direct connection to their influence on the network (Wahab and Memood, 2018). As noted, before, farmers (node that store data on their hardware) are requested to prove that the stored data are available and untampered. The written contract between the client and the farmer states the fees agreed between the parties for the service. The client will then pay the fees in Storjcoin X (or SJCX) to the service provider according to the smart contract.

Drawbacks of the solution: Although well-known and among the most popular decentralized cloud solutions, Storj has a few limitations:

- Satellite plays a key role (designed to store metadata). If it crashes, it would render the whole system useless as users would no longer be able to access their data. This single point of failure within the system makes the satellite an ideal candidate for a (D) DoS attack.
- The problem with the farm node storage network is consistency and future scalability - not that the network is not economically scalable, but the point is, there will come a time when the profitability of being a storage node will no longer exist due to rising electricity costs. The point of network consistency calls for the fact that no one has control over the availability of the storage node - that it can be disabled or suspended at any time.
- Data is transferred via HTTP. Farmers expose endpoints where client applications can store or download fragments - Farmer nodes can be hacked by exposing their IP addresses.
- Spartacus attacks, or identity theft, are possible on Kademlia.
- Vulnerable to attack Honest Geppetto - The attacker exploits a large number of puppet nodes on the network, building up trust and contracts over time. Once it hits a certain threshold, it pulls the strings on each puppet to perform a hostage attack with the data involved, or simply removes each node from the network.
- Vulnerable to Eclipse Attack - An Eclipse attack attempts to isolate a node or set of nodes in the network graph, ensuring that all outgoing connections reach the malicious nodes.

3.2.3. Blockchain technology for Data Privacy in cloud computing

The authors in [6] studied the potential to use the blockchain technology to protect healthcare data hosted within the cloud.

There has been recent interest in utilizing blockchain in the provision of secure healthcare data management. Broadly speaking, blockchain is a technology able to build an open and distributed online database, which consists a list of data structures (also known as blocks) that are linked with each other. These blocks are distributed among multiple nodes of an infrastructure, and are not centrally stored. Each block contains a timestamp of its production, the hash of the previous block and the transaction data, and in this context, a patient's healthcare data and the healthcare provider information. Fig. 3.6 describes the authors conceptual blockchain-based EMR (Electronic Medical Records)/HER (Electronic Health Records)/PHR (Personal Health Records) ecosystem. Specifically, when new healthcare data for a particular patient is created a new block is instantiated and distributed to all peers in the patient network. After a majority of the peers have approved the new block, the system will insert it in the chain. This allows them to achieve a global view of the patient's medical history in an efficient, verifiable, and permanent way. If the agreement is not reached, then a fork in the chain is created and the block is defined as an orphan and does not belong to the main chain. Once the block has been inserted into the chain, the data in any given block cannot be modified without modifying all subsequent blocks.

Drawbacks of the solution: Securing EMR/EHR/PHR ecosystem and ensuring privacy and integrity of the data is an active research area. Many approaches include using cryptographic primitives, such as those based on public key infrastructure and public clouds. However, this limits the searchability of the data, in the sense that healthcare providers have to decrypt the data prior to searching on the decrypted data, resulting in increases in time and costs for the data retrieval and diagnosis.

The strong data integrity feature of blockchain results in immutability that any data, once stored in blockchain, cannot be altered or deleted. However, if the record were healthcare data, then such personal data would come under the protection of privacy laws, many of them would not allow personal data to be kept perpetually.

Another practical issue is on how fit it is for blockchain to store healthcare data. Blockchain was originally designed to record transaction data, which is relatively small and linear. Healthcare data, such as imaging and treatment plans, however, can be large and relational that requires searching. How well blockchain storage can cope with both requirements is currently unclear.

3.2.4. BACC (Blockchain-based Access Control for Cloud Data)

Traditional cloud-based access control solutions are based on a centralized approach (i.e., a cloud server becomes the central authority to control accesses to the data), which makes it difficult to prevent malicious cloud servers from disclosing user's data; and therefore,

compromising the privacy of the stored data. Additionally, the centralization of authority can cause a single point of failure. Furthermore, to provide confidentiality, which is one of the essential security requirements, user's data is encrypted before it is stored on the cloud. Most of the cloud servers store the decryption keys, after they encrypt the data, in their premises. This compromises data privacy. In [4], the authors propose a new model that addresses the aforementioned issues. To address the centralization problem, we distributed the access control tasks to smart contracts over a decentralized network, i.e., blockchain. To address the latter, we used Shamir secret sharing scheme to manage the encryption keys. Then we introduced a new type of node, called master node, to our blockchain platform, to store the decryption key parts.

Drawbacks of the solution:

- Privileges granted by the Data Owner (DO): If the DO gives privileges to a malicious user DU, the latter will be able to access the files stored in the cloud, download them and even receive the decryption key kept in the Blockchain network;
- Identity theft: If a user usurps the identity of any Data User, they could impersonate the latter and recover the data stored in the cloud. It should therefore be ensured that data spoofing is not effective in this model and that a given user cannot enter the user group;
- The *t* and *n* in the equation of Shamir's algorithm: The more the *t* tends to *n* the more robust the solution is. If the *t* is small, the solution may be weak. Therefore, we should make sure that *t* tends to *n* and that *t* nodes are always available;
- Vulnerabilities in SSL secure channels: There are common attacks encountered in SSL secure channels that are also not discussed in [4];
- Smart contracts: The authors of the BACC solution, also did not take into account attacks on smart contracts which are generally used as a gateway to the blockchain network.

3.2.5. FileCoin

FileCoin represents another decentralized storage network. It runs on top of IPFS. The InterPlanetary File System (IPFS) [7] was created to connect all computing devices in a distributed peer-to-peer file system, in order to avoid a single point of failure in the network. With the help of FileCoin, clients can store data on space offered by storage providers who are not their trusted parties (FileCoin: A Decentralized Storage Network, 2017). Like other decentralized storage networks, clients need to be assured that their data are safe throughout time. To fulfil this demand two methods are proposed by FileCoin:

- Proof-of-Spacetime: Through this method, clients will be assured that during a specific period of time their data is being stored;
- Proof-of-Replication: Proof-of-Replication shows that data are safe in its dedicated physical storage locations and not a single node in the network has duplicated files on their own hardware. The token used in FileCoin transactions is Filecoin (FIL).

Drawbacks of the solution: Some potential issues with Filecoin are:

- Speed: Instead of accessing one server, it needs to access multiple servers to piece together documents when required. This depends on servers being online and Internet speeds;
- Dapp: By processing transactions using the Filecoin cryptocurrency, it creates a barrier to entry, as someone has to buy another coin and exchange it to use the service.

3.2.6. SIA

SIA is another blockchain-based storage platform [8]. Like Storj, in SIA peers rent out their hardware capacity. Storage providers are known as hosts in this network. Storage proofs, containing a list of hashes from the file and a fraction of the original file, are publicly available and verifiable therefor the client does not need to verify them personally. Similar to Storj, data in SIA must be encrypted and digitally signed before submitting in the network. Also, a contract has to set up between the client and hosts, in which the duration of data storage and the reward for each valid or invalid proof must be declared. In addition, maximum number of accepted invalid proofs will be mentioned. At the core of each contract, there is a Merkle root (Merkle, 1980) of each file utilized to provide frequent proofs by the hosts. SIA runs on its own open source blockchain and uses SIACoin (SC) as its native cryptocurrency. However, SIA's developers are working on making it possible to use other cryptocurrencies on the network.

Drawbacks of the solution: SIA is a memory and resource hog when performing several tasks such as synchronizing with the network, completing file contracts, or other background tasks, making a system nearly unresponsive and showing a lack of consideration for the fact that a user may be running SIA alongside other apps. In addition, we can note several other inconvenient of the solution:

- SIA Isn't Easy to Use, and the UI is All but Abandoned;
- Renting and Hosting Aren't Reliable and Break Often;
- Documentation is a mess;
- Hosts Have Liability Questions.

3.2.7. Discussion

Li Dagang & al. have presented in [2] a data-sharing mechanism where data decryption keys are stored in a blockchain as part of the metadata and protected by user’s private key. This efficiently realizes an easy and secure key-management mechanism in a decentralized fashion. In addition, encrypted data are stored in dedicated storage nodes and proxy re-encryption mechanism is used to ensure secure data-sharing in the untrusted environment. However, even if the particularity of Meta-key is to store the decryption keys in the blockchain, we can show that it is a weak point of the solution. Also, in this diagram, the authors privilege the confidentiality of the keys and data and forget about the criterion of integrity which is very important.

We have presented in Table 3.1 a comparative study of the decentralized cloud solutions.

Table 3.1. Comparative table of the decentralized cloud solutions.

	Meta-key	**Storj**	**Blockchain for cloud-based data privacy**	**BACC**	**Filecoin**	**SIA**
Blockchain	Yes	Yes	Yes	Yes	Yes	Yes
Cryptocurrency	No cryptocurrency	Storj (STORJ)	No cryptocurrency	No cryptocurrency	Filecoin (FIL)	SIAcoin (SC)
Particularity of the solution	Two layers of security	Encryption and file sharding	Global agreement before inserting data	Access control and smart contracts	Proof of spacetime Proof of replication	File contracts, SIAcoin's form of smart contracts
Eliminate single point of failure	Yes	Yes, but in terms of storage of data and not of metadata (storj bridge)	Yes	Yes	Yes	Yes
Smart contracts	No	No	No	Yes	No	Yes (called file contracts)
Fee payment method	No payment	One-time payment	No payment	No payment	One-time payment	Payment after proof
Access control	No	No	No	Yes	No	No
Key management	Decentralized / blockchain	Managed by User	No key management	No key management	No key management	No key management

Figueiredo & al. demonstrate in [5] the particularity of storj network: file sharding. Indeed, files aren't stored in centralized data centers, instead, they're encrypted, split into pieces, and distributed on a global cloud network. However, this solution, despite the fact that it is decentralized and secure, contains a major problem: the centralization of the management of metadata by the satellite. This can cause a single point of failure, which could create serious problems.

Esposito & al. studied in [6] the potential to use the Blockchain technology to protect healthcare data hosted within the cloud. We remained skeptical about this solution which is designated for a specific type of data (healthcare data).

Filecoin [7] is a unique file storage marketplace that is not controlled by one entity; instead, it enables independent storage miners to set their fees in a hypercompetitive open market, a process that is bolstered by its specialized blockchain. Particularity of filecoin is that it uses two types of proofs to prove the work done: proof-of-replication (PoRep) and proof-of-spacetime (PoSt). Proof-of-replication allows the network to confirm replication of data to a unique location. Whereas, Proof-of-spacetime allows the network to verify that the data is stored for a specific duration of time. Combining the two, Filecoin manages large scale storage networks with multiple independent parties. Therefore, he removes the possibility of forgery of data storage records to increase mining rewards. Filecoin's competitors like Storj ($STORJ) and SIAcoin ($SC) lack this functionality. Despite the fact that data are more secure due to replication and storage mechanisms used on this network, this solution does not take into account the access control mechanisms and the encryption of data before storage. Also, in order to participate in the network miners, face the high cost of equipment, the added cost of staking FIL tokens, and not

getting the block rewards upfront is a real concern that needs to be addressed soon by the team to prevent fork this early in this network's life.

The authors in [8] introduce SIA, which is the combination of a peer-to-peer network with blockchain technology to create a decentralized storage platform. SIA enables the formation of storage contracts between peers. Contracts are agreements between a storage provider and their client, defining what data will be stored and at what price. They require the storage provider to prove, at regular intervals, that they are still storing their client's data. Contracts are stored in a blockchain, making them publicly auditable. In this respect, SIA can be viewed as a Bitcoin derivative that includes support for such contracts. This protocol essentially revolves around contracts by forgetting several other important details such as data access, encryption, integrity etc. In addition, the product is known to have many technical issues, which make it unstable and unusable.

Finally, in [4], the authors presented a solution called BACC which contains several aspects sought in the work of our thesis. They take the access control out of the cloud environments and port it into a blockchain network. This resolves the centralization issue of the current access control models. Hence, there is no single point of failure. Also, they take the decryption key out of the cloud servers while still enable the users to use the cloud services, such as data sharing service. To design the decentralized access control model, the authors employed blockchain, and distributed the access control tasks, i.e., authentication, authorization, and auditing, to smart contracts over the blockchain network. Smart contracts permit trusted transactions and agreements to be carried out among disparate, anonymous parties without the need for a central authority, legal system, or external enforcement mechanism.

Of course, we do not fully follow the solution of Nasrin Sohrabi & al. but we take the essentials in order to make a solution well suited to our expectations. BACC take into consideration access control, confidentiality, data integrity, availability, and data security. Everything you need, to offer a solution ensuring both confidentiality and integrity but above all, general security of data storage.

3.3. Overview of BACC (Blockchain-based Access Control for Cloud Data)

3.3.1. The Main Components of the System

The system model comprises of Blockchain, Cloud Storage Server, and User group (which constitutes of Data owner and Data user) components, all connected together through a peer-to-peer (P2P) network.

The blockchain: which contains the blocks of transactions and smart contracts. Blockchain is used here to provide access control tasks which are defined as smart contracts and deployed onto the blockchain. Each of these smart contracts performs a specific task. The system consists of four smart contracts, namely, AUD-Contract, ACPContract, LKP-Contract, and CLU-Contract. These contracts are created and deployed onto the

blockchain by data owners and the cloud storage server. The execution of these contracts will perform the access control tasks. This model introduces also a new node, called master node, to the blockchain network. The master node named as M-MN, is the miner node who is also a security server. The security server in the proposed model, is the data decryption key holder. The decryption key pieces, which are generated from Shamir secret sharing algorithm by data owner, are distributed to these nodes through a secure channel.

The cloud storage server: which is the resource provider that allocates storage resources to data owners to store their data. As mentioned, data owners encrypt their data before uploading them onto the cloud. This ensures that confidentiality, one of the fundamental security requirements, is provided. The decryption keys, however, are not stored in the cloud server, as we assume, in this model, that the cloud server is untrustworthy.

The user group:

– Data owner is the owner of the data who could request to upload their data onto the cloud storage. Therefore, s/he is required to, first, encrypt the data and then upload it onto cloud. After uploading the encrypted data on cloud, data owner with the use of Shamir secret sharing algorithm, divides the decryption key into n pieces. These decryption key pieces are then distributed to n master nodes, and each master node keeps only one piece of the decryption key. To find the master nodes, the data owner sends a transaction to the LKP-Contract, which will be executed upon receiving a transaction and returns the address of n available master nodes to the data owner. In addition, the data owner is responsible for creating and deploying AUD-Contract and ACP-contract onto blockchain. ACP-contract is the access control policy contract which is executed every time the cloud server requires to verify a data user's access rights.

– Data user is the user of the cloud data. To download the data from the cloud server, data user sends a download request to the cloud storage server. Cloud server verifies the data user's access rights by sending a transaction to the blockchain to execute the ACP-Contract. After receiving the response from the Blockchain, it will send the encrypted data to the data user, if the user has access right. Upon receiving the encrypted data, data user sends decryption key request to t master nodes. Once it receives the decryption key pieces, it reconstructs the decryption key. To find the master nodes, data user sends a transaction to the LKP-Contract, which will be executed by the miner nodes and return the address of t master nodes to the data user.

All of these components are connected together via a peer-to-peer (P2P) network.

3.3.2. System Configuration

The BACC model can be summed up in two main phases:

– Data upload;

– Data download.

3.3.2.1. Data Upload

The data storage process is as follows:

- Creation and deployment of AUD and ACP contracts in the blockchain by the DO Data Owner);
- DO requests cloud storage;
- DO encrypts the data;
- DO sends data for storage;
- DO sends a transaction to the LKP-contract to find the available master nodes of the blockchain;
- DO divides the decryption key into *n* pieces with the SSS algorithm;
- These n pieces are distributed to the *t* master nodes of the blockchain via secure SSL channels (each node takes only one coin).

3.3.2.2. Data Download

The data download process is as follows:

- DU (Data User) sends a request to the cloud server to download the data;
- The cloud server verifies the user's access rights by sending a transaction to the blockchain to execute the ACP-contract;
- After receiving the response from the blockchain, the cloud server sends the encrypted data to the user if they are authenticated;
- DU sends a transaction to the LKP-contract, which will be executed by the single nodes and return the address of the *t* master nodes;
- DU then sends a request to retrieve the decryption key to the *t* master nodes of the blockchain;
- The master nodes send the pieces of the decryption key to the DU which reconstructs the initial key.

3.3.3. The Limits of the BACC Scheme

After a study of the BACC model, some critical points were identified. These points can paralyze the proper functioning but especially the security and confidentiality of data in this system:

- Privileges granted by the Data Owner (DO): If the DO gives privileges to a malicious user DU, the latter will be able to access the files stored in the cloud, download them and even receive the decryption key kept in the Blockchain network;

- Identity theft: If a user usurps the identity of any Data User, they could impersonate the latter and recover the data stored in the cloud. It should therefore be ensured that data spoofing is not effective in this model and that a given user cannot enter the user group;
- The *t* and *n* in the equation of Shamir's algorithm: The more the *t* tends to *n* the more robust the solution is. If the *t* is small, the solution may be weak. Therefore, we should make sure that *t* tends to *n* and that *t* nodes are always available;
- Attacks in Blockchain environments: There are some attacks not studied in [4] for which the solution is not robust and which could hinder its growth.
- Vulnerabilities in SSL secure channels: There are common attacks encountered in SSL secure channels that are also not discussed in [4];
- Vulnerabilities in cloud environments: Attacks in cloud environments are becoming more and more common. It would be important to cite in [4], that the solution is resilient against the most common cloud attacks;
- Smart contracts: The authors of the BACC solution also did not take into account attacks on smart contracts, which are generally used as a gateway to the blockchain network.

We have summarized in Figure 3.1 the different steps mentioned above concerning the storage and downloading of data.

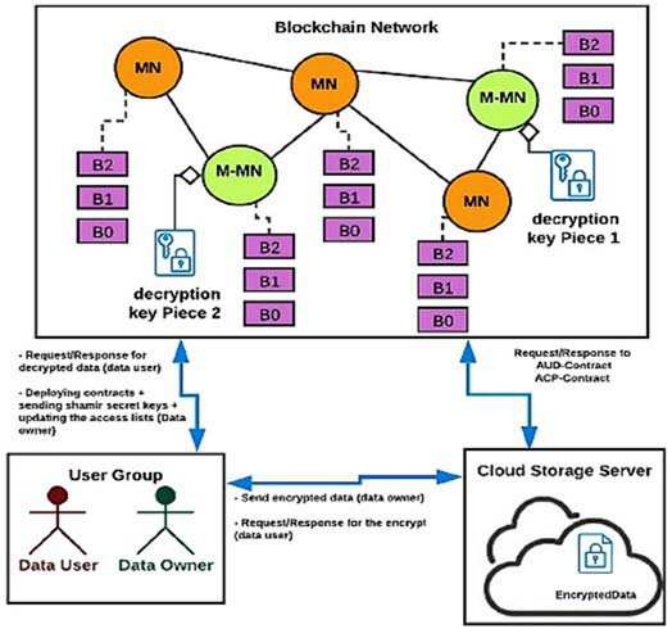


Fig. 3.1. An overview of the proposed model.
M-MN stands for Master Miner Node. MN stands for Miner Node.

3.4. Transition from Centralized to Decentralized Storage

3.4.1. Basic Cloud Computing

Cloud computing is the development and use of computer technology ("computing") based on the Internet ("cloud"). It is a style of computing in which dynamically scalable and often virtualized resources are delivered as a service over the Internet. Users do not need to have knowledge, expertise or control over the “cloud” technology infrastructure that supports them. The concept integrates infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) as well as Web 2.0 and other recent technology trends.

The term cloud is used as a metaphor for the Internet, based on how the Internet is depicted in computer network diagrams, and is an abstraction of the complex infrastructure it conceals [9] (example of cloud computing network in figure 3.2).

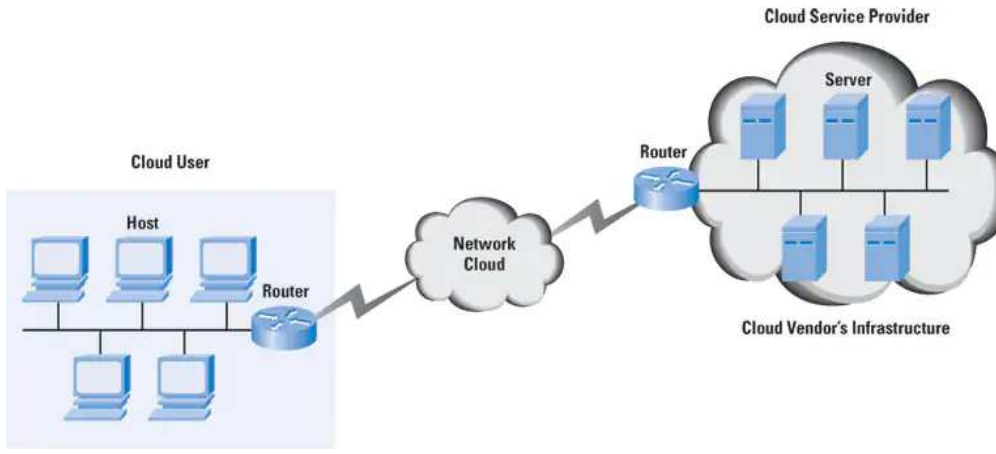


Fig. 3.2. Basic architecture of a cloud solution.

3.4.2. Why Decentralizing the Cloud?

Centralized cloud storage service providers charge significant fees for their services. While these fees don't mean much to large businesses, small businesses can feel the pinch. Amazon, for example, charges $23 per month for the storage service they provide [10]. While they don't handle data unethically, they still have the option of using data analytics to drive data. In some cases, cloud storage providers may also use the data stored on their servers to direct personalized advertisements to the data owner, which many businesses find intrusive. Worse yet, cloud platforms currently operate a server model that has a single point of failure, making data breaches much more likely.

Decentralized cloud storage is a service that has already started, just under two years ago, to consolidate itself in the daily process of business and enterprises. This service ensures

that only customers have access to their data and no one else. In this case, the service provider cannot display the stored content. And who says that the service provider does not access the data also says that any attacker will find it difficult to recover a file scattered in a network of nodes and that only a key held by the owner can reconstruct it.

Service providers like Storj and SIAcoin have already built a credible market presence by providing a level of security that server-client cloud providers cannot match. These new vendors are using blockchain technology to implement even more secure cloud storage solutions.

3.4.3. Decentralized Cloud

Decentralized cloud storage is a rapidly growing use case aimed at solving one of today's biggest online challenges. A few very large providers (Google, Microsoft, Dropbox, Amazon, etc.) control cloud storage. Resolving questions about data protection, privacy, licensing, control and ownership of data are the primary goals of this new system.

A decentralized cloud service is a network storage model in which data is stored on multiple machines called "nodes" and hosted by the cooperating participants in the cloud. To be viable, such a model must provide storage space at least equal to the storage space required by users. However, some nodes may make more storage space available than others. Rewards can be set up based on the contribution of nodes in the system.

Unlike the traditional centralized model, the decentralized cloud service does not directly use dedicated servers for data storage, thereby eliminating costly hardware investments. Each node uses software to make storage space available.

Such services may nevertheless resort to a certain level of centralization for supervising their proper functioning.

Figs. 3.3 and 3.4 show us the difference between a centralized cloud architecture and a decentralized one.

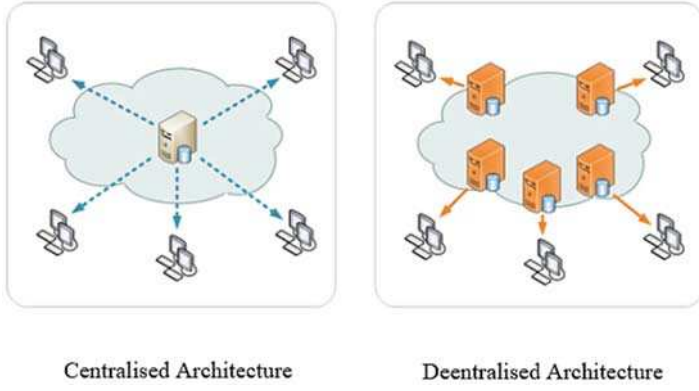
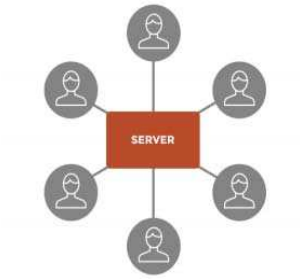


Fig. 3.3. Cloud Architecture centralized vs decentralized.





Centralized Architecture

- Central authority;
- Single point of failure;
- Opaque;
- Security by people;
- Client and server are two different entities;
- Slow;
- Expensive.

Decentralized Architecture

- No Central authority;
- No single point of failure;
- Transparent – open source;
- Safety through math;
- All data downloaders are also storers;
- Fast;
- Less expensive.

Fig. 3.4. Comparison between centralized and decentralized architecture.

3.5. Proposed Model BAC-DC: Blockchain-based Access Control for Decentralized Cloud Data

3.5.1. The Main Components

3.5.1.1. The Nodes

There are two types of nodes: Client or “tenant” nodes: are those who ask to store their data; Storage nodes or "farmers": are those that provide storage space. Their role is to store and return data. In addition to reliable data storage, nodes should provide adequate network bandwidth and responsiveness. Storage nodes are selected based on various criteria: ping time, latency, throughput, bandwidth, caps, sufficient disk space, geographic location, uptime, accurate audit response history, and so on. In exchange for their service, the nodes are paid.

These nodes work and communicate with each other via a Peer-to-Peer network. They perform authentication when they want to communicate with each other. This authentication is done twice to avoid attacks like man-in-the-middle.

3.5.1.2. The Peer-to-peer Communication and Discovery System

All the peers of the network communicate via a standardized protocol. This protocol should:

- Provide accessibility to peers;

- Provide authentication where each participant cryptographically proves the identity of the peer they are talking to, to avoid middleman attacks;
- Offer complete privacy. In cases such as bandwidth measurement the client and the storage node must be able to communicate without any risk of spying. The protocol should ensure that all communications are private by default.

In addition, the protocol establishes a way to search the network addresses of peers by a unique identifier so that, given the unique identifier of a peer, any other peer node can connect to it. This responsibility is similar to the standard Internet Domain Name System (DNS) [11], which involves mapping an identifier to an ephemeral login address, but unlike DNS, there can be no centralized registration process. To achieve this, a network overlay, such as Chord [12], Pastry [13] or Kademlia [14], can be built on top of our chosen peer-to-peer communication protocol.

3.5.1.3. The Reference Table or Metadata

Once we have split an object and selected storage nodes on which to store the new parts, we now need to keep track of the storage nodes that we have selected. We allow users to choose storage based on geographic location, performance characteristics, available space, and other features. Therefore, instead of an implicit node selection like a scheme using a consistent hash such as Dynamo [15], we need to use an explicit node selection scheme like directory-based searches [16]. Additionally, the user should be able to choose an arbitrary key, often treated as a path, to identify this mapping of data elements to the node. These characteristics imply the need for a metadata storage system.

3.5.2. BAC-DC: Operating Mode

3.5.2.1. Data Upload

Our model is a distributed storage service based on a blockchain network. The basic principle is to share a file on a peer-to-peer (P2P) network and to follow the encryption, storage, retrieval and administration of contracts using a blockchain.

To store user data this model works as follows:

- Clients (tenants) encrypt their file and control the encryption key;
- The file is then divided into several pieces and duplicates of each piece are created to provide redundancy according to the level of security required for the file;
- The fragments, i.e., smaller pieces of the file, are then stored in several computers in the P2P network (at the “farmers”);
- Each farmer only gets the shards that are already encrypted and therefore cannot access the content of the shard. Even if hackers attack a node, they will only get the encrypted fragments, which they cannot read, meaning they cannot rebuild the entire file from it.

3.5.2.2. Data Download

When downloading data, tenants proceed as follows:

- They will have to use a blockchain-based hash table with the encryption key, so that the entire file is reconstructed from the individual fragments. The file will be rebuilt after the farmers return the individual fragments they hold;
- The tenant then uses the encryption key to decrypt the file;
- Smart contracts in the blockchain govern the terms and conditions, tenants pay for the file they had stored while farmers are paid for their storage volume in a cryptocurrency that will be specified later;
- There is an audit process to ensure that farmers hold the shard they agreed to hold. When farmers successfully meet their obligation, they accumulate a good reputation as well as cryptocurrency. If the farmers don't meet the conditions, they don't get paid. Repeated instances of default may result in disqualification.

Figures 3.5 and 3.6 summarize by precise flowcharts the process of storing and downloading data

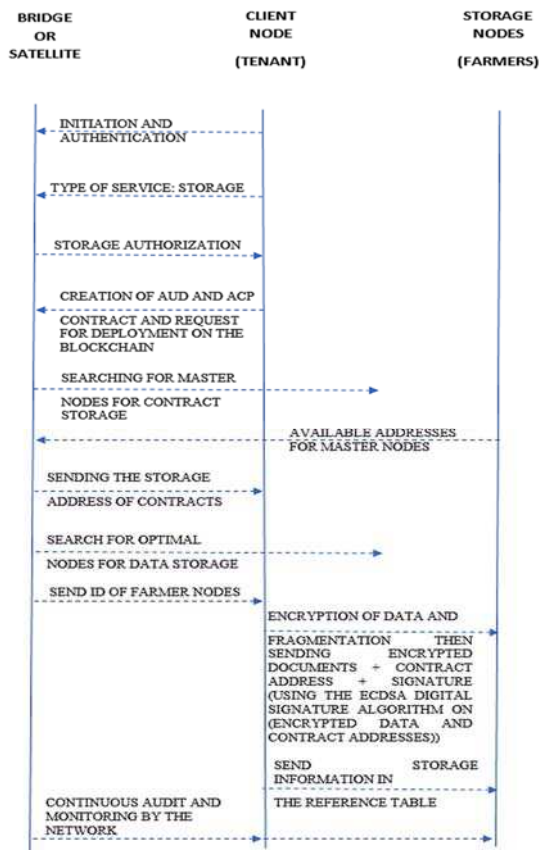


Fig. 3.5. Flowchart of the data uploading process.

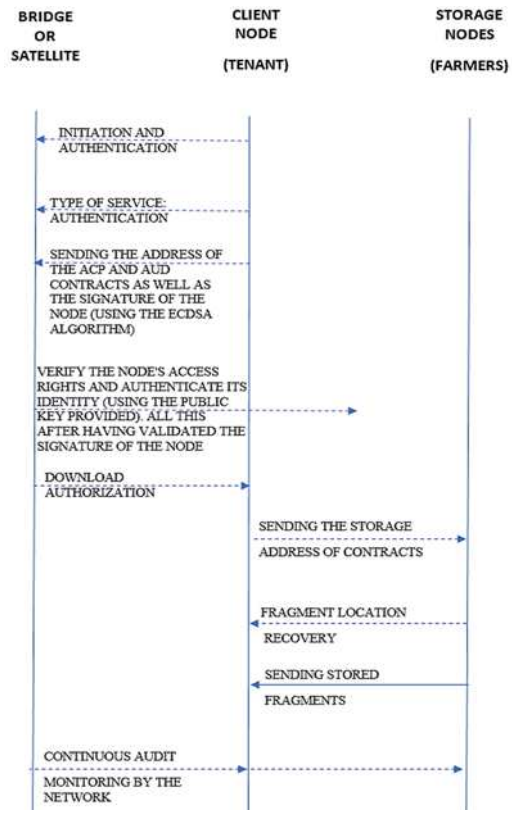


Fig. 3.6. Flowchart of the data downloading process.

3.5.3. Model Architecture

Here is the proposed architecture (Fig. 3.7).

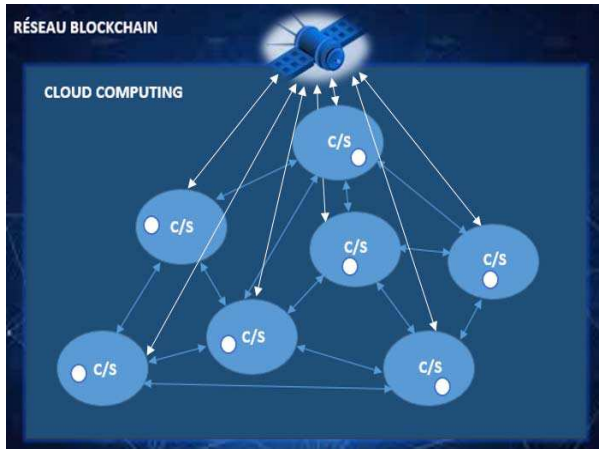


Fig. 3.7. Model Architecture.

3.6. Link between Our BAC-DC Scheme and the One Studied BACC

The point to be discussed here is the combination of the BACC model with the decentralized cloud concept used in our BAC-DC solution. The blockchain usually contains the transaction blocks and smart contracts as shown in the figure below.

In the BACC model, the blockchain is used to provide access control tasks which are defined as smart contracts. Each of these smart contracts performs a specific task. The system therefore consists of four smart contracts, namely, AUD-Contract, ACP-Contract, LKP-Contract and CLU-Contract. These contracts are created and deployed on the blockchain by the data owners and the cloud storage server. The execution of these contracts will therefore ensure the access control tasks (Example in figure 3.8).

In the case of our solution, access control will also be provided by smart contracts, but not in the same way. This is because first of all these contracts are no longer created or deployed by the cloud storage server or what are called data owners, they will be deployed by the storage nodes which can also be clients. Then the access control procedure will be roughly the same as with the BACC solution.

In addition, in BACC, the authors introduce a new node, called master node, in the blockchain network. The master node, represented by M-MN as in the figure, is the minor node that is also a security server.

The security server in BACC is the holder of the data decryption key. On the other hand, in our model, it is these nodes that will be the owners of the main elements entering into the authentication procedure of the various nodes, or more simply the access control procedure. It is emphasized here that all the sharing elements are distributed between the nodes of the network via a secure channel.

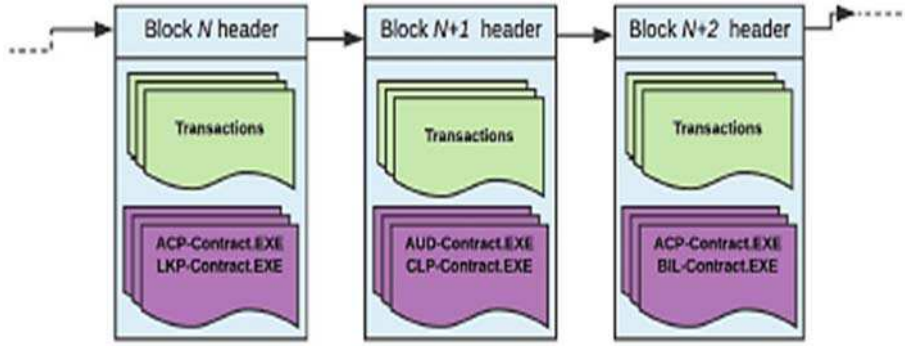


Fig. 3.8. Blockchain blocks illustration.

We have made a comparative table (Table 3.2) between the two solutions (the one studied and the one proposed) for a better understanding of the common points.

Table 3.2. Comparison between BACC and BAC-DC.

	BAC-DC	**BACC**
Decentralization	Decentralization of storage and access control	Decentralization of access control
Use of blockchain	Storage of data and access control	Access control tasks
Smart contracts	Deployed by storage nodes	Deployed by cloud server and data users
Security server role (master node)	Access control procedure	Store the decryption keys
Communication Channels between network entities	Secure Socket Layer (SSL)	Secure Socket Layer (SSL)
Storage	The blockchain's nodes used for storage	Cloud for data storage
Data Encryption	AES-128	AES-128
Cost	Cheap	Expensive
Integrity of data	ECDSA (Elliptic Curve Digital Signature Algorithm)	ECDSA (Elliptic Curve Digital Signature Algorithm)
Security objectives	Confidentiality and integrity	Confidentiality and integrity
Loss of data	No single point of failure	Single point of failure
Redundancy	Yes	No
Bandwidth	Unlimited	Limited
System deployment	Complex	Complex

3.7. Security Analysis

3.7.1. File Partitioning Before Storage

The first step in securing a file before storage is file sharding. This means that when you want to store a file, you must first divide the file into several smaller pieces. The advantage of file sharding is twofold. First, you can send and recall fragments of the file in parallel, which speeds up file transfer. Second, no single entity owns all of your file. You are the only person who knows where all the shards are.

The location of the fragments is a key distinction. Our model, as a cloud storage provider, obviously prioritizes protecting user privacy. The customer or tenant should be the only person who knows where all the fragments of their file have gone. This is where blockchain and cryptography come in. We are implementing what is called a distributed hash table so that users can locate all the fragments of their original file. This hash table requires a private key to identify the fragments. Without the private key, it would be almost impossible to correctly guess the location of an exploded file.

3.7.2. End-to-end Encryption

This is where data privacy is taken to the next level. Fragmentation already adds a layer of privacy because no data host (the one that stores the data also called a farmer) can read the entire file. But even being able to read a fragment of a file is problematic. It could still contain sensitive information.

To counter this phenomenon, our model helps its users who want to store their data (called tenants) to compress and encrypt their files before fragmentation. The encrypted file has only one key, and the holder keeps that key locally on his computer.

As the sole owner of the encryption key, the tenant is the only person who can read the file. When a data host receives a fragment, it has already been encrypted as part of a larger file. The data hosted by the farmer is useless without all the other fragments and the encryption key.

To hack our model and gain access to a file, we would have to locate all the fragments in a file. This is almost impossible without the hash table private key. Then, you would have to convince the farmers who host these fragments to send them to you without the appropriate signature. Finally, one would have to guess (which is highly improbable) or steal the encryption key from the tenant.

Therefore, our model based on decentralized file storage is much more secure than traditional centralized options. In our solution, we talked about data encryption before storage. Client-side encryption is an exceptionally strong data security strategy. By eliminating the possibility that service providers (or third parties who may compel service providers to provide data access) to be able to view the data, client-side encryption ensures that data stored in the cloud can only be viewed on the client side. This prevents data loss and unauthorized disclosure of personal or private files, providing greater peace of mind for both personal and business users. Among various algorithms that we consider efficient (DES, RSA, IDEA, 3 WAY AND RCS) we have opted for the AES algorithm which remains the one that most attracts our attention.

Advanced Encryption Standard or AES, also known as Rijndael, is a symmetric encryption algorithm. It is currently the most widely used and the safest.

This algorithm takes as input a block of 128 bits (16 bytes), the key is 128, 192 or 256 bits. The 16 input bytes are swapped according to a previously defined table. These bytes are then placed in an array of 4×4 elements and its rows are rotated to the right. The increment for the rotation varies depending on the row number. A linear transformation is then applied to the matrix, it consists of the binary multiplication of each element of the matrix with polynomials resulting from an auxiliary matrix, this multiplication is subject to special rules according to GF (28) (Galois group or finite body). The linear transformation guarantees better diffusion (propagation of the bits in the structure) over several turns. Finally, an exclusive OR XOR between the matrix and another matrix makes it possible to obtain an intermediate matrix. These different operations are repeated several

times and define a "tour". For a key of 128, 192 or 256, AES requires 10, 12 or 14 turns, respectively.

3.7.3. Parity Fragments for Redundancy

Individual file fragments are sent to ordinary computers across the BAC-DC network. But what if one of these computers shuts down? Are the fragments that were stored on this computer lost?

We clearly need to implement some type of redundancy in the system. This is done with parity fragments. When you upload a file, you can choose the level of redundancy you need for your file. With enough parity fragments, you can greatly reduce the chances of losing a data fragment from your file.

However, over a longer period of time, the likelihood of losing a shard increases. Our model then performs regular audits and uses other verification methods to make sure this does not happen. However, it is best to recall and rebuild files periodically before restocking them again.

3.7.4. File Verification

One question remains: how can we be sure our files are still there? Couldn't a farmer just wipe out the shards he owns or shut down his computer?

To address this concern, the BAC-DC model performs a records check. To be paid, farmers must prove that they have the shards sent to them. Therefore, the system sends a request to the farmers, and if the farmer has modified or deleted the encrypted fragment, he will not be able to respond to the request. If the farmer currently owns the case, they can properly respond to the request. He receives a micro payment for the storage and maintenance of the file. Thus, the farmers are encouraged to store the files and to remain active on the network.

3.7.5. Data Loss

In the established system, to avoid the loss of data, following a failure or the compromise of a node etc., only a number of fragments (for example 1/3 of the fragments constituting a file) will be necessary for its reconstruction.

3.7.6. Access Control

Among the various existing access control solutions, that of S. Fugkeaw et al. [17] drew our attention. Indeed, the authors propose a role-based encryption scheme (LW-C-CP-ARBE) of lightweight collaborative ciphertext policy attribute to support lightweight and thin access control for the cloud environment.

They apply the CP-ABE approach as basic cryptographic access control and introduce a new proxy re-encryption (PRE) protocol to reduce data re-encryption and decryption costs for mobile users.

To this end, the overhead of performing the cryptographic operation at the end user device is low. In addition, they are developing a secure access policy and re-encryption sharing protocol to allow users with write privileges to update data and request the proxy to perform data re-encryption.

We opted for attribute-based encryption because it is useful in a peer-to-peer context where there is no central server to manage access policies.

In order to demonstrate the strengths of our scheme, we have compared it with some solutions studied in our previous work [1] in Tables 3.3 and 3.4.

Table 3.3. Comparison of BAC-DC with some solutions ensuring data confidentiality in the cloud.

	In addition to the cloud	**File partitioning before storage**	**End-to-end encryption**
BAC-DC	Blockchain	✓	✓
Solution based on Intel SGX	Software Guard Extensions Technology	×	✓
HPI-Secure	5 mains components acting with the cloud	✓	✓
Confidentiality preserving through encryption and obfuscation	Mathematical function	×	✓
TR-MABE	Multi-Authority attribute-based encryption	×	✓
Blockchain for cloud-based data privacy	Blockchain	×	✓
CAFE	Hypervisor level technique	✓	×
ARIANNA	Arianna app	×	×
Multiple Storage solution	Multiple Store	✓	×
Matrix-based solution	Matrices	×	×

Table 3.4. Comparison of BAC-DC with some solutions ensuring data confidentiality in the cloud.

	Parity fragments for redundancy	File verification	Data loss	Access control
BAC-DC	✓	✓	✓	✓
Solution based on Intel SGX	×	×	×	×
HPI-Secure	×		×	×
Confidentiality preserving through encryption and obfuscation	×		×	×
TR-MABE	×	×	×	✓
Blockchain for cloud-based data privacy	×	×	×	✓
CAFE	×	×	×	✓
ARIANNA	×	×	×	×
Multiple Storage solution	×	×	×	×
Matrix-based solution	×	×	×	×

3.8. Security against Attacks

There are many types of traditional attacks to which our schema is exposed. We have listed some of them by developing for each the solutions to be adopted to be protected.

3.8.1. The Google Attack

A Google attack is so called by Bitcoin Core developer Peter Todd, to describe the coordinated attack on the network by a large company or entity controlling large amounts of computing power / storage space. Our research has shown that Google currently stores around 8,000 PB of data [18]. We introduce the concept of user space, or collective unused free space on the computers of average consumers. This user space could expand if users began to use decentralized cloud storage platforms as their primary platform. We estimate this figure to be around 250,000 BP [18]. So even if Google suspended all its services to attack the network, it would not be possible to exceed the resources provided by user space. We assert that the collective user space is greater and always will be greater than that of the centralized cloud computing industry as a whole.

3.8.2. The Sybil Redundancy Attack

An attacker who wants to exploit the network by claiming that he has several redundant copies of a fragment, when he only has one, cannot do so. Indeed, each partition is uniquely encrypted. Therefore, an attacker can only pass audits if he has a copy of the unique redundant copies. The client node must take steps to share its redundant copies to geographically distributed and independent nodes. If it is distributed randomly, the probability that the same redundant element is hosted on the same monitored node is statistically quite low. By using random distribution and the standard single encryption method for redundant copies, an attacker cannot perform a Sybil redundancy attack.

3.8.3. The Incorrect Distribution

Here we are looking at a malicious farmer who would like to attack the network. This attacker does not care about financial incentives. The attacker wants to create distrust of the network by simultaneously removing the data and its redundant copies from the network, making the file unrecoverable. In collusion with one or more malicious farmers, the farmer aims to receive several redundant copies of the fragment. Our first line of defense is not to use a single node when uploading a sensitive file over the network, but rather to relay the fragments through multiple nodes. This way, even if one of the nodes is malicious, it cannot carry out this attack successfully. The nodes should collude, and if the client selects nodes at random, the probability of selecting colluding nodes is low. In the worst-case scenario where the farmer colludes with more than one of the relay nodes, the second line of defense consists of two additional mechanisms. First, the use of erasure and redundancy coding systems with different parameters ensures that the farmer cannot know how many fragments to destroy. In addition, we can design a method where all the verification is done on a GVN (Group of Verification Nodes). With this, the farmer has no way to determine the associated fragments if he has multiple redundant copies.

3.8.4. Cheating Client Node

Imagine a scenario where a malicious actor would like to host a file on the network, but avoid paying for the file, or intentionally rejects correct audits in order to avoid payment. In this case, there would be a disagreement between the two nodes and the transaction would cease. The Merkle roots integrated into the blockchain, serve as sufficient and irrefutable proof of the validity of the customer's disputes.

3.8.5. The Hostage Byte Attack

The hostage byte attack was first described by Vitalik Buterin, in which a malicious farmer transfers data to a customer but holds the last piece of data hostage for higher payment [19]. We address this problem with our erasure-encoding scheme, so that the fragments needed to reassemble the file are not linear. As long as the client keeps the limits of its erasure by encoding a secret, the malicious farmer cannot know which is the last byte.

3.8.6. Geppetto's Honest Attack

In this attack vector, a farmer is honestly offering a large amount of storage space for an extended period. At one point, he pulls the strings in an attempt to destroy much of the network. We can make this prohibitive in two ways. First, through the use of GVNs (Group of Verification Nodes), we can delay payment to farmers. Farmers would have demonstrated their reliability for a certain amount before starting to receive payment, and if they suddenly became malicious or unable to provide adequate service, the pending funds would be lost. Second, a linked smart contract could be created between the customer and the farmer. They would both agree to a service level agreement (SLA) to follow, and if the farmer deviated from the agreement, the customer could be paid. Otherwise, the funds could be destroyed.

3.9. Advantages and Disadvantages of the BAC-DC Model

3.9.1. Advantages

The advantages of the proposed model are numerous among which we can mention:

- Decentralization: The principle of decentralization of the cloud used in our model makes it possible to avoid computer attacks, guarantees data confidentiality and ensures that data access is available at all times (no maintenance period);
- Profit: People can earn money by renting their unused hard drive space;
- Cheaper solution for the market: The storage cost is lower than the solutions offered by the traditional cloud industry;
- Cheaper solution for storage providers: The costs of traditional cloud infrastructure were colossal compared to the proposed model: electricity, bandwidth, computer hardware, maintenance etc.;
- No more single point of failure: our solution is based on a decentralized storage model which fragments files and distributes them to several network nodes. Since the fragments are scattered over several nodes, there is no longer a single point of failure;
- Farmer-side privacy: It is impossible for farmers to access stored data because they only have part of it;
- Redundancy: The files are copied to several computers at the same time in order to be able to cope with the technical problems that some farmers may encounter;
- Unlimited bandwidth: this is precisely the parameter that drives up companies' bills for data storage;
- Storage durability: we require a reduced number of nodes necessary to rebuild a file.

3.9.2. Disadvantages

Most computer solutions have drawbacks, since the world of technology is only virtual and infinite. Thus, despite the advantages mentioned above, our solution encounters some limits to be reviewed and corrected. These disadvantages are as follows:

- Node trust: Despite the phenomenon of trusting nodes in the network, can we still trust these storage entities? especially when the files to be stored are of the order of "highly confidential";
- Fragment loss: can we predict, despite the trust placed in a node, that it can be functional all the time? otherwise we risk the loss of the stored fragments;
- Network stagnation: we were talking above about a phenomenon of redundancy to ensure the reconstruction of a file. This redundancy is ensured by the duplication of the fragments. However, can this method not cause a jam in the network?
- Complexity: the deployment of such a network and the processing remains very complex;
- Adaptation: the decentralized cloud is not always suitable for all applications because it can sometimes be difficult to set up.

We have also proposed a comparative table (table 3.5) of our solution with those studied in our previous article [1]. We took into account the few fundamental aspects noted in this part when discussing the advantages and disadvantages of our solution.

Table 3.5. Comparison of the advantages and disadvantages of our solution BAC-DC with some solutions ensuring data confidentiality in the cloud.

	Decentralization	Cheap solution for the market	No Single point of failure	Redundancy	Confident Cloud provider	Complexity
BAC-DC	✓	✓	✓	✓	✓	✓
Solution based on Intel SGX	×	✓	×	×	✓	×
HPI-Secure	×	✓	×	×	×	×
Confidentiality preserving through encryption and obfuscation	×	✓	×	×	×	×
TR-MABE	×	×	×	✓	✓	✓
Blockchain for cloud-based data privacy	✓	×	✓	✓	✓	✓
CAFE	×	✓	×	×	×	×
ARIANNA	×	✓	×	×	×	×
Multiple Storage solution	✓	✓	✓	✓	✓	×
Matrix-based solution	×	✓	×	×	×	✓

3.10. Conclusion

Cloud computing is a popular option for people and businesses for a number of reasons like cost savings, increased productivity, speed, efficiency, performance, and security. Cloud providers enable users to store files and applications on remote servers and then access all the data via the Internet. They allow users to access the applications and computing power of their servers while also retaining complete control over those resources. At the same time, the connectivity of the cloud makes it possible to design decentralized networks that incorporate edge computing strategies.

Decentralized file storage systems have the potential to eliminate high costs and inefficiencies from the marketplace and provide a much higher level of privacy, reliability and quality of service than today. Decentralized storage is a use case that could become very compelling for peer-to-peer networks and distributed ledger technology in a few years.

The solution proposed in this paper brings many advantages in terms of security. We integrate blockchain technology into the cloud while emphasizing access control managed by specific nodes called master nodes and not by a central entity. Indeed, it responds to security vulnerabilities identified in existing decentralized cloud computing solutions. Among the many advantages, it is very profitable, emphasizes data confidentiality, offers good redundancy and unlimited bandwidth.

The few weaknesses noted in our solution will be taken into account and we will propose in future work an improved version of the scheme to reduce its complexity and expand its range of use. We will also try to measure the contribution of improvements by comparing the two versions via implementations.

References

[1]. Wayzani S., Diallo C., Confidentiality Concerns in Cloud Environments, in Advances in Networks, Security and Communications (S. Yurish, Ed.), Vol. 3, Chapter 2, *IFSA Publishing,* 2022, pp. 29-70.

[2]. D. Li, R. Du, Y. Fu, H. M. Au, Meta-Key: A secure data-sharing protocol under blockchain-based decentralized storage architecture, *IEEE Networking Letters*, Vol. 1, Issue 1, 2019, pp. 30-33.

[3]. S. Wilkinson, J. Lowry, Metadisk: Blockchain-Based Decentralized File Storage Application, http://metadisk.org/metadisk.pdf

[4]. N. Sohrabi, X. Yi, Z. Tari, I. Khalil, BACC: Blockchain-based access control for cloud data, in *Proceedings of the AustralaSIAn Computer Science Week Multiconference (ACSW'20)*, February 4-6, 2020, Melbourne, VIC, Australia, 10.

[5]. S. de Figueiredo, A. Madhusudan, V. Reniers, S. Nikova, B. Preneel, Exploring the storj network: A security analysis, in *Proceedings of the 36th Annual ACM Symposium on Applied Computing. Association for Computing Machinery*, New York, NY, USA, 2021, pp. 257-264.

[6]. C. Esposito, A. De Santis, G. Tortora, H. Chang, K.-K. R. Choo, Blockchain: A panacea for healthcare cloud-based data security and privacy ?, *IEEE Cloud Computing*, Vol. 5, Issue 1, 2018, pp. 31-37.

[7]. J. Benet, IPFS – Content Addressed, Versioned, P2P File System, https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa 3LX/ipfs.draft3.pdf
[8]. D. Vorick, L. Champine, SIA: Simple Decentralized Storage, https://SIA.tech/SIA.pdf
[9]. T. Choudhury, V. Vashisht, H. Srivastava, A secure decentralized cloud computing environment over peer to peer?, *International Journal of Computer Science and Mobile Computing*, Vol. 2, Issue 4, 2013-04-15, pp. 433-446.
[10]. How to Build a Decentralized Cloud Storage Solution Like Storj.io, https://www.devteam.space/blog/how-to-build-a-decentralized-cloud-storage-solution-like-storj-io/
[11]. P. Mockapetris, Domain Names – Implementation and Specification, STD 13, RFC Editor, November 1987, http://www.rfc-editor.org/rfc/rfc1035.txt
[12]. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications, in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'01)*, 2001, pp. 149-160.
[13]. A. I. T. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'01)*, 2001, pp. 329-350.
[14]. P. Maymounkov, D. Mazières, Kademlia: A peer-to-peer information system based on the XOR metric, in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'01)*, 2001, pp. 53-65.
[15]. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels, Dynamo: Amazon's highly available key-value store, in *Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP'07)*, 2007, pp. 205-220.
[16]. J. Paiva, L. Rodrigues, Policies for efficient data replication in P2P systems, in *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS'13)*, December 2013, pp. 404-411.
[17]. S. Fugkeaw, A fine-grained and lightweight data access control model for mobile cloud computing, *IEEE Access*, Vol. 9, 2021, pp. 836-848.
[18]. How Big is the Cloud?, http://blog.storj.io/post/95376799893/how-big-is-the-cloud/
[19]. V. Buterin, Secret Sharing and Erasure Coding: A Guide for the Aspiring Dropbox Decentralizer, https://blog.ethereum.org/2014/08/16/secret-sharing-erasurecoding-guide-aspiring-dropbox-decentralizer/

Advances in Networks, Security and Communications, Volume 3

Sergey Y. Yurish, Editor

Thy 3rd volume of *'Advances in Networks, Security and Communications'* open access Book Series contains 5 chapters written by 7 authors from 3 countries: Austria, India and Senegal. Each of the chapters contains the extended state-of-the art followed by new, obtained by the authors results, unpublished before.

Like the first two volumes of this Book Series, the 3rd volume also has been organized by topics of high interest. Every chapter in the volume is independent and self-contained and has the same structure: first an introduction to specific topic under study; second particular field description including applications, ending by well selected list of references with books, journals, conference proceedings and web sites.

With this unique combination of information in each volume, the in *'Advances in Networks, Security and Communications'* Book Series will be useful for post-graduate students, researchers, engineers and scientist working in the appropriate areas.

ISBN 978-84-09-47239-0



