International Journal of Innovative Research and Scientific Studies, 8(1) 2025, pages: 2067-2082

IJIRSS
International Journal of Innovative Research & Scientific Studies

ISSN: 2617-6548

URL: www.ijirss.com

Examining the function of Merkle trees in enhancing security within big data technologies

Akku Kubigenova[1], Almbubi Aktayeva[2*], Altynbek Sharipbay[3], Rozamgul Niyazova[4], Aisha Sexenbayeva[5]

[1]*Department of Information and Communication Technologies, Sh. Ualikhanov Kokshetau University, 020000, Kokshetau, Kazakhstan.*
[2]*Department of Information Systems and Informatics, Abay Myrzakhmetov Kokshetau University, 020000, Kokshetau, Kazakhstan.*
[3,4]*Research Institute of Artificial Intelligence, L.N. Gumilyov Eurasian National University, 010008, Astana, Kazakhstan.*
[5]*Department of Information Security, L.N. Gumilyov Eurasian National University, 010008, Astana, Kazakhstan.*

Corresponding author: Almbubi Aktayeva (*Email: aakhtaewa@gmail.com*)

Abstract

The main purpose of this article is to study the structural characteristics of the Big Data paradigm, which presents serious data confidentiality, integrity, and security challenges. Big data technology and its associated services have become the central topic of numerous scientific studies and popular applications due to the rapid progress in this field and the development of data analysis solutions. Comprehensive solutions and innovative technologies are required to protect confidential information. However, current evaluations of Merkle tree methods primarily focus on the metrics of traditional cryptosystems. This article describes the Big Data Security ecosystem and emphasizes the importance of maintaining data integrity and authenticity during exchanges in various fields. The authors propose that Merkle tree technology can effectively solve these issues using a quantum cryptosystem algorithm, providing secure and efficient data exchange. In this work, we conducted experimental research using Python and a quantum-computing library to obtain accurate test data. The scientific novelty of the work is the development of a new method and technology for a modified Merkle signature scheme to select the optimal computational efficiency and security. Finally, experimental studies have shown that researchers can use trusted technologies to perform a complete probabilistic analysis of Merkle tree data, which allows for safe end-to-end Big Data exchanges.

Keywords: Big data analytics, Big data, Data management, Merkle trees, Quantum computing, Secure data processing.

DOI: 10.53894/ijirss.v8i1.4899
Funding: This study received no specific financial support.
History: Received: 16 January 2025/**Revised:** 11 February 2025/**Accepted:** 17 February 2025/**Published:** 24 February 2025

Competing Interests: The authors declare that they have no competing interests.
Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.
Transparency: The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.
Publisher: Innovative Research Publishing

1. Introduction

The modern interconnected digital world has generated vast volumes of data. This is due to the advent of web technologies, which have transformed user-generated content. The simplicity of content creation has led to a significant increase in generated data. The recent boom in social media has further amplified real-time content generation, resulting in the emergence of Big Data [1]. Today, more than a billion people use social media, quickly generating structured and unstructured data volumes. The main challenge behind this colossal data lies in the analysis and processing of Big Data [2].

The emerging IoT sector is a primary source of Big Data, generating data from various sensors, including medical devices, temperature sensors, and numerous additional software modules and digital devices. Big Data continues to challenge the human capacity to process continuously generated and exponentially increasing data volumes, with significant sources being modern digital technologies [3]. The primary sources of Big Data constantly generate a vast array of data (structured, unstructured, and semi-structured) that exceeds the processing capabilities of modern database systems [4].

In addition to managing and analyzing Big Data, processing models will require optimization in the coming years because the relevant data usually constitute a minimal volume of the overall data (low-density values) [5].

"Big Data" refers to the large volume, velocity, and variety of information resources that require economically efficient and innovative data processing methods rather than traditional data processing methods to enhance understanding and decision-making. Big Data is a branch of data science that explores various tools, approaches, and strategies for analyzing extensive and complex datasets, deconstructing them, and systematically extracting insights and information. The main differences between traditional and big data are the volume, variety, velocity, complexity, and potential value. Table 1 presents a comparative analysis of Classical and Big Data.

Table 1.
Comparative analysis of classical and big data.

Characteristics	Classic database	Big data
Volume of information	From gigabyte to terabyte	From petabyte to exabyte
Storage method	Centralized	Decentralized
Data structure	Structured	Semi-structured and unstructured
Data storage and processing model	Vertical model	Horizontal model
Data relationship	Strong	Weak
Processing methods	Traditional data analysis can be conducted using primary statistical methods.	Advanced analytical methods, such as machine learning and data mining, are required for Big Data analysis.
	Traditional data analysis methods are slow and gradual.	Big data analysis methods are quick and instantaneous.

Traditional data, for instance, comprises structured data that conventional methods can easily handle. By contrast, Big Data encompasses many semi-structured, structured, and unstructured data that require specialized tools. At the same time, Big Data provides valuable and profound information, which may be inconsistent or inaccurate.

Big Data is a comprehensive concept that requires a complete understanding from multiple participants, each with their own architecture and technological preferences (see Figure 1) [6].

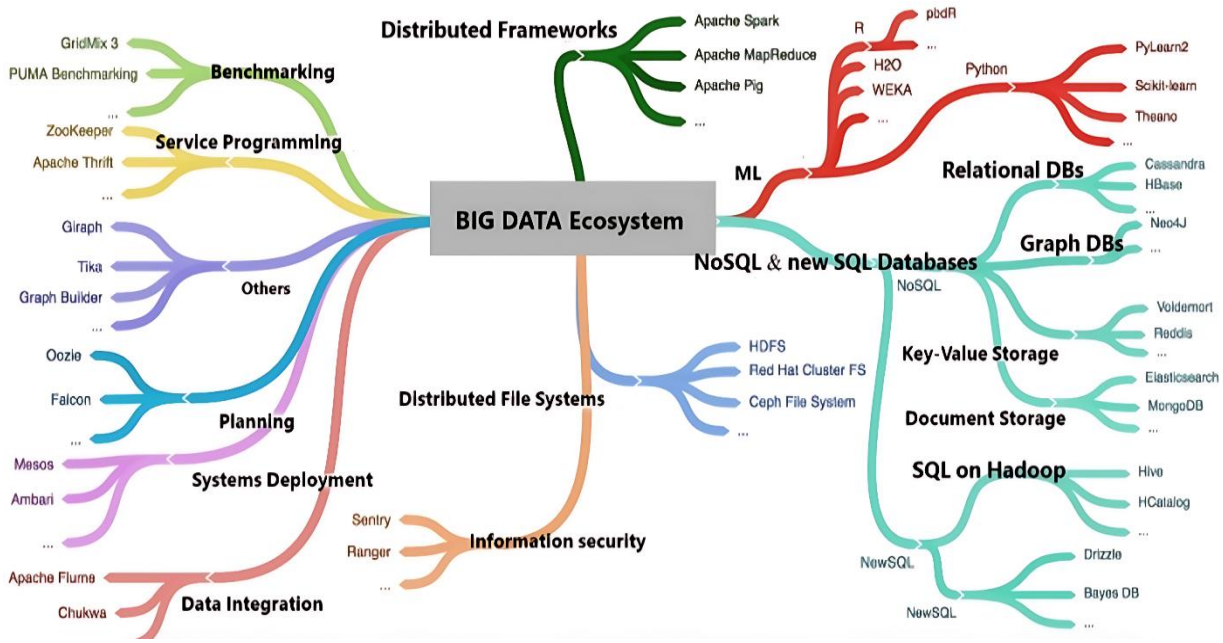

Figure 1.
Ontology of the big data concept.

Big Data Processing Technologies are defined as software utilities primarily designed for analyzing, processing, and extracting information from large datasets with highly complex structures that traditional data processing methods cannot handle [7]. Researchers have investigated what needs to be known about Big Data, analytics, and cloud computing to set up and process accurate data. They examined the coding languages needed to recognize the different types of Big Data storage technologies and their importance, longevity, and accessibility [8].

Big Data expands the capabilities of classical statistical analytical approaches by incorporating modern methods that leverage resources to run analytical algorithms. This transition is crucial as databases grow in size, variety, and complexity. Big Data can be classified according to the key components based on the primary characteristics of the eight Vs: volume, velocity, variety, veracity, value, validity, variability, and visualization. These refer to the vast amounts of data generated from various sources (see Figure 2).

Figure 2.
Main characteristics of big data — 8V.

These characteristics provide the foundation for understanding the challenges and opportunities presented by Big Data. Studying these characteristics of Big Data is key to its proper utilization and fundamental to realizing its potential. Additionally, the main factors are usually associated with further difficulties in storage, analysis, and application of subsequent procedures or result extraction. Table 2 provides descriptions of the main characteristics of Big Data [9].

Table 2.
Description of the main characteristics of big data.

No	Big data characteristics	Brief description
1.	Volume	Volume refers to the amount of data measured in gigabytes, zettabytes (ZB), and yottabytes (YB). According to industry trends, the volume of data is expected to rise substantially in the coming years.
2.	Velocity	Velocity refers to the speed of data processing. High velocity is crucial for the performance of any Big Data process. It consists of the rate of change, activity bursts, and the linking of incoming data sets.
3.	Value	Value refers to the benefits that an organization derives from the data. Does it match the organization's goals? Does it help the organization improve itself? It's among the most important core characteristics of big data.
4.	Variety	Variety refers to the different types of Big Data. It is one of the biggest issues the Big Data industry faces, as it affects performance. It's vital to manage and organize a variety of data correctly.

		Variety refers to the various types of data gathered from multiple sources.
5.	Veracity	Veracity refers to data accuracy. It is one of the most critical characteristics of Big Data, as low veracity can significantly undermine the accuracy of results.
6.	Validity	How valid and relevant is the data for the intended purpose?
7.	Volatility	Big data is constantly changing. Data gathered from a source a day ago might differ from today. This is called data variability, and it affects data homogenization.
8.	Visualization	Visualization refers to presenting significant data-generated insights through visual representations such as charts and graphs. It has become prevalent recently as Big Data professionals regularly share their insights with nontechnical audiences.

Big Data analysis explores vast volumes of complex data to identify hidden patterns and correlations. This provides the foundation for choosing the right technologies, implementing effective data management strategies, and obtaining meaningful insights from the extensive and dynamic world of Big Data. Nevertheless, knowing the main features is not enough to understand what Big Data is all about; one also needs to be able to use these features in complicated tasks with many variables and fuzzy logic. However, this sharp increase in data consumption has led to numerous data security issues. There is an apparent contradiction between the security and privacy of Big Data and their widespread use.

This article focuses on privacy and security issues in the realm of Big Data, distinguishes between privacy and security, and discusses the privacy requirements in Big Data. The results provide practical and reliable mechanisms for information security in complex and interconnected Big Data environments. This research contributes to the theoretical progress of threat modeling and prediction and offers practical solutions for improving the cybersecurity status of interconnected Big Data.

2. Materials and Methods

2.1. Research on Big Data Security

Research and widespread applications in big data and related services have increased dramatically due to the rapid development of data analytics technologies and solutions. However, there is an urgent need for a comprehensive system to verify the integrity of big data.

New research [10] stresses the need for a strong system to check the integrity of large amounts of data, outlines a way to do it, and emphasizes how important speed and accuracy are in the checking process. This approach aims to introduce a model for the integrity of Big Data and consider the verification process. Ensuring the integrity of big data involves both processing speed and the accuracy of the verification process. The authors of this study [10] use traditional data verification methods, machine learning, and the Cerberus Framework to make their case for a strict testing methodology to ensure the integrity of big data. They assert that this makes data more accurate and reliable in many situations. The research has utilized the Python programming language with precise test data and incorporated the latest technologies and programming practices.

The authors of this paper [11] emphasize that ensuring data security is crucial as organizations collect and process vast amounts of data; a lack of security can lead to significant financial losses and damage an organization's reputation. Additionally, the study by Abdullah-Al-Musa, et al. [11] involves developing a comprehensive security model (BDS), analyzing significant data challenges, integrating data analytics, identifying security needs, and providing actionable recommendations to enhance considerable data security and integrity in organizational settings. The authors Abdullah-Al-Musa, et al. [11] present a Big Data Security (BDS) model that aims to improve data security in organizations. The study suggests that as data generation rises, the challenges related to significant data security will also increase, making the implementation of the BDS model and other security measures vital for mitigating the risks associated with data breaches and unauthorized access.

In this article, Jiana, et al. [12] present a technical framework for the life cycle and protection of personal information in big data, summarize key technologies such as authorization, access control, security audits, traceability audits, and data desensitization, and promote the implementation of a cyber power strategy.

In addition, in-depth research on these key technologies has been conducted to provide complete solutions for protecting personal information in big data, cracking down on illegal user information trading activities, and creating a positive network security environment.

The article by Vardanyan, et al. [13] argues that digital integrity, as a new basis for digital rights and a new form of the limited idea of human dignity, could help protect people better in the digital world and fill in the gaps in protecting people's digital rights. The paper [13] highlights that protecting personal data within big data frameworks is inadequate; it emphasizes digital integrity as a foundation for enhancing data protection and individual rights.

This study by Yin [14] argues that data integrity is critical in big data environments to prevent corruption and loss, provide reliable information for decision-making, and maintain trust in shared data across platforms. To ensure the security and integrity of data, technology professionals need to establish a data authentication result detection algorithm that provides reliability and usability of data by improving limited verification results. This paper by Yin [14] uses an algorithm to find unreliable verification results and protect against forgery attacks. The results of the experiments show that the algorithm can guarantee the accuracy of data verification.

This paper by Mehmood, et al. [15] aims to provide a comprehensive overview of privacy-preserving mechanisms in big data and introduce the challenges for existing mechanisms. It illustrates the Big Data infrastructure and state-of-the-art

privacy-preserving mechanisms at each stage of the Big Data life cycle. In addition, the paper discusses the challenges and future research directions related to privacy preservation in big data.

The author of this paper [16] analyzes the technical challenges of implementing Big Data security and privacy protection and discusses some key technologies and their latest developments. The analysis shows that Big Data effectively solves information security problems while introducing security issues. It brings new opportunities for the development of information security.

The study by Dongpo [17] analyzes the security problems of Big Data, proposes protection strategies for its security and privacy, and studies the data collection and storage process. It argues that many professional privacy protection technologies are needed to protect Big Data information and that users' privacy protection awareness must be improved to ensure privacy information security.

Using reputation-based redundancy computation, this study by Gao, et al. [18] develops a way to protect the integrity of big data processing in the cloud. It shows that the solution only adds a limited cost to achieving integrity protection and is practical for real-world applications. The results of the implementation and experiment demonstrate that the solution only adds a limited cost to achieving integrity protection and is practical for real-world applications.

This paper by Lebdaoui and Hajji [19] illuminates integrity issues in big data and introduces a new model for preserving integrity in this context. The authors discuss some aspects of Big Data that affect data integrity and present a new model for managing and protecting data integrity in the face of Big Data challenges.

2.2. Quantum-based Cryptography Technique

The article by Sookhak, et al. [20] discusses creating a universal architecture for replicated metadata services in distributed file systems. The authors introduce a brand-new structure called RMS (Replicated Metadata Services) that addresses the problems with current high-availability (HA) solutions and simplifies setup and testing. The paper provides two examples of distributed file systems, PVFS and HDFS, to demonstrate the benefits of RMS. The study shows that the HDFS RMS variant performs comparably to the original implementations.

This article by Aujla, et al. [21] proposes secure storage, verification, and auditing (SecSVA) of big data in the cloud environment. The study shows that SecSVA can offer safe third-party auditing while maintaining data integrity across multiple domains in a cloud setting. In this article, the authors propose a scheme, SecSVA, that enables secure storage, verification, and auditing of big data in the cloud environment. They have developed an attribute-based encryption (ABE-based) scheme to grant users access to the encrypted data in the cloud. The Merkle hash tree (MHT)-based algorithm protects the integrity of the data, and the proposed scheme can handle different kinds of attacks on cloud-based data.

This study by Joseph, et al. [22] introduces a novel enhanced encryption technique with quantum-based neuro-cryptography. The encryption uses logical shift and modulo operation-based transformation that eliminates data extrusion. In quantum cryptography, the stream of photons generates a key, which neuro-cryptography then converts into the secret key. Therefore, the researchers have developed a novel inherited process incorporating a cyclic crossover operation. Combining ciphertext and key achieves one-point cyclic crossover and bit string mutation, thereby increasing confusion and eliminating timing attacks. The proposed system increases throughput and reduces encryption and decryption time with high diffusion and confusion properties.

This study by Shafia, et al. [23] proposes that cloud computing is an Internet-based technology that has emerged rapidly in the last few years due to the widespread demand for services required by various institutions, organizations, and individuals. The cloud server rapidly transfers structured, unstructured, and semi-structured data. The authors of this paper provide an overview of the characteristics and state of security.

The authors of this study, Majdoubi, et al. [24], explore the landscape of quantum computing in the Big Data Era, drawing parallels with classical methodologies. Their findings underscore the significance of quantum cryptographic methods, fueled by further exploration and development in this dynamic and promising field that contributes to data security.

Most recent studies generally touch on the generic features of attribute-based encryption schemes, such as user revocation, scalability, flexibility, data confidentiality, and scope in pairing-based attribute-based encryption schemes.

This research by Jemihin, et al. [25] states that attribute-based encryption (ABE) is commonly used to address the problem of scalability in new public key infrastructure (PKI). The study also delves into recent challenges in attribute-based encryption cryptography in the post-quantum era and highlights its differences from conventional pairing-based attribute-based encryption schemes. The authors also suggest reviewing existing quantum-resistant attribute-based encryption schemes based on their algorithm design, security, and functionalities.

Therefore, the use of various existing cryptography techniques has improved information security in big data. However, the current methods lead to data extrusion due to shortened transformation and diffusion properties in the key generation and encryption processes, resulting in data loss and leakage. This study builds on these findings and incorporates scenario-based modeling to better manage threats and mitigate risks in increasingly complex and interconnected big data environments. By summarizing the strengths and addressing the shortcomings of existing methodologies, this study contributes to the development of interconnected big data cybersecurity frameworks.

3. Results

3.1 Merkle Tree Principle

Big data technology continues to evolve, and new security and privacy challenges arise with this development. Despite its widespread use, current research lacks a complete understanding of the security aspects of Big Data. Applying quantum

computing using Merkle trees to assess data quality is one possible direction that can help address security concerns and increase confidence in big data technologies.

This study proposes a hybrid approach to secure data management. This approach enhances security and privacy in data management and addresses the challenges of big data technologies. In this study, we propose using Merkle trees with quantum and post-quantum computing, which can effectively check the integrity of data, provide security, and increase trust in big data technologies:

1. A Merkle tree, also known as a hash tree, is a data structure that efficiently stores and verifies the integrity and content of large amounts of data. It works by dividing a large amount of data into smaller blocks and generating a hash for each block. The system then combines these hashes into larger ones, forming a root hash. This root hash is a complete summary of all the underlying data, which makes data integrity checking very efficient.
2. A Merkle proof establishes the existence of a data item within a specific Merkle tree. With these features of the Merkle tree, a Merkle proof can quickly verify the existence of a data element in the Merkle tree. This approach helps check the integrity and content of large amounts of data.
3. Merkle Tree Principle:

A. Hash trees are primarily used to ensure the integrity of large data structures by verifying and validating the integrity of their contents.

B. A hash tree is a generalized version of a hash list and chain, most of which are binary.

C. Using a branched Merkle tree, regardless of its branching, possesses all the characteristics of a tree structure.

D. The value of the end node is the hash of the data block, and the value of the non-leaf node is based on the values of all child nodes or the values of the leaf nodes below it.

E. Hash trees are calculated according to the hash algorithm.

3.2. The Hash of the Parent Element = Hash (Hash of Child Element 1 + ... + Hash of Child Element N), Where + Denotes String Union

Figure 3 shows the basic structure of the Merkle tree, demonstrating its hierarchical structure from the data blocks to the Merkle root. The graph illustrates how to use cryptographic hashing to turn data blocks into final nodes. It then shows how to build parent nodes in a loop, which ends with creating the Merkle root.

A hash is a function that converts arbitrary-length data into fixed-length data. The main difference between a hash tree and a hash list is that a branch of a hash tree can be loaded in a single pass, and the integrity of each branch can be checked immediately, even if the data of the whole tree is incomplete. When computing the hash of the final node, 0x00 is added to the data, and 0x01 is added to the internal node's hash value; the tree's depth is limited to the depth of the subtree corresponding to the node value.

Therefore, we only consider an extracted hash chain valid if the prefix decreases at each step and remains optimistic when it reaches the final node. The hash tree uses hash operations such as SHA-224, SHA-256, SHA-512, etc. However, one could use the checksum algorithm if the security requirements are not high, and it is necessary to ensure that the data is not accidentally corrupted. To check the integrity of the data, the easiest way is to hash all the data to obtain a fixed-length hash value and then publish it to the network so that when the user loads the data, it is hashed again. Specifically, we use a combination of the RSA algorithm with 512-bit SHA encryption along with quantum and post-quantum cryptography to ensure the data remains private and secure.

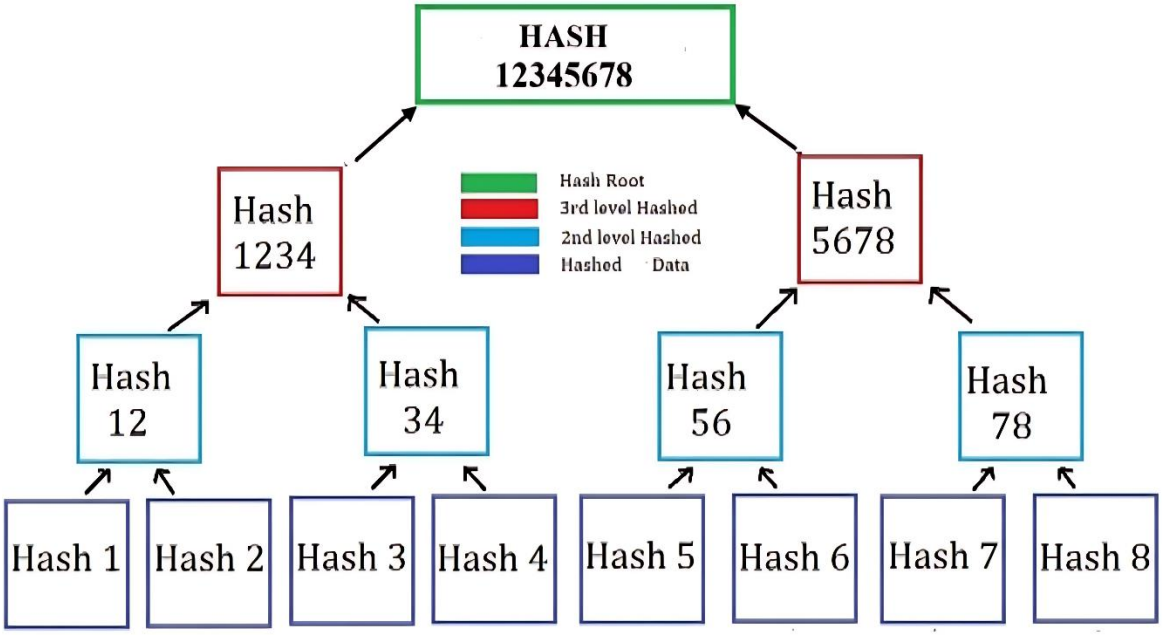

Figure 3.
Algorithm for calculating the Merkle tree hash.

A unique feature of the Merkle tree is that the existence of any node or leaf can be cryptographically proven by computing the root (See Table 3). The message signature is created using the private key from the selected key pair. Signature verification involves computing the root based on the transmitted parameters and comparing it with a reusable public key. These parameters are:

- Signature.
- Root.
- A one-time key, the private part of which is used to sign the message.
- Hashes from the tree lie on the path from the selected leaf to the root.

Table 3.
Advantages of Merkle trees in big data.

No	Advantages	Note
1	Efficient data verification processes	The Merkle tree provides an efficient means of verifying transactions without consuming much computing power.
2	Less memory footprint	Verifying transactions using the Merkle tree does not require loading all the data. Compared to other data structures, it requires less space for computation.
3	Fast transactions	Since transactions are paired together and a single hash is created, information transmission over the network becomes faster. This is one of the main reasons why cryptocurrency transfers are very fast.
4	Tamper detection	The Merkle tree allows you to detect when a transaction has been tampered with. When a transaction is hashed and stored, a change in the original information also causes a change in the hash. This can be determined by comparing the current hash with the hash stored in the block header.

This study aims to fill this gap by developing a theoretical framework for data security that considers different scenarios based on Merkle path length and hash length. Empirical experiments validate the theoretical models by investigating simulations with different hash lengths and Merkle path lengths. A Merkle tree with data blocks Y1, Y2, ..., Yn is given. This study proposes a solution encompassing three primary experimental work functions.

3.2. Description of the Experimental Methodology

Tests are conducted to determine whether theoretical ideas about how different parameters, such as hash function length and Merkle tree root path length, affect hybrid technologies that utilize quantum computing. Additionally, tests are performed to assess how various parameters influence Big Data security.

This method of conducting experiments allows us to test our ideas about how likely the Merkle tree root hash function will be computed in hybrid technologies that utilize quantum computing. It also enables us to assess the value and reliability of these ideas in various situations. During the experimental phase of our study, we concentrated on the empirical validation of three strategies:

- Developing a Merkle tree structure to organize the security of a privacy-appropriate chain of data blocks.
- Merkle trees are used in quantum cryptography as a data hashing function, and the SHA-512 algorithm (RSA 1024 or RSA 2048) is utilized for data security.

The application of Merkle trees in post-quantum cryptography, utilizing a data hashing function with the KYBER algorithm (simulation) for data security, is discussed.

3.3. Experimental Characteristics

In the experimental scheme, Merkle tree nodes are written as key-value pairs (merkle_key, merkle_value):

$$N_j^i = merkle\_key, \quad (1)$$

where:

- The top index, I, indicates the level at which the node is located in the Merkle tree.
- i = 0 indicates that the node is obtained by computing the hash of a blockchain transaction.
- i > 0 indicates that the node is obtained by combining its two child nodes.
- i = Root indicates that the node is the root node of the Merkle tree.
- A more significant value of "i" indicates that the node is closer to the root node of the Merkle tree.
- The lower index j indicates the serial number I of the node in the Merkle tree.

The lower index of each Merkle tree node increases from left to right. The merkle_value signifies the hash associated with that Merkle tree node. Non-leaf nodes of the Merkle tree have both left and right child nodes. Suppose one could find a Merkle tree node by locating its left and right child nodes if two Merkle-proof paths go through it to its left and right, respectively. This node can be obtained directly without using any auxiliary information.

For this purpose, a Python program was developed to compare the experimental data results. The experimental design has been carefully thought out to ensure that the experimental data models can be reliably validated.

Example_1: The RSA 2048 quantum computing algorithm hash function is used to design and build a Merkle tree with 16 leaf nodes and five levels. This is done to ensure that Big Data is correct and authentic. Figure 4 shows the creation of a Merkle tree model of 16 data blocks.

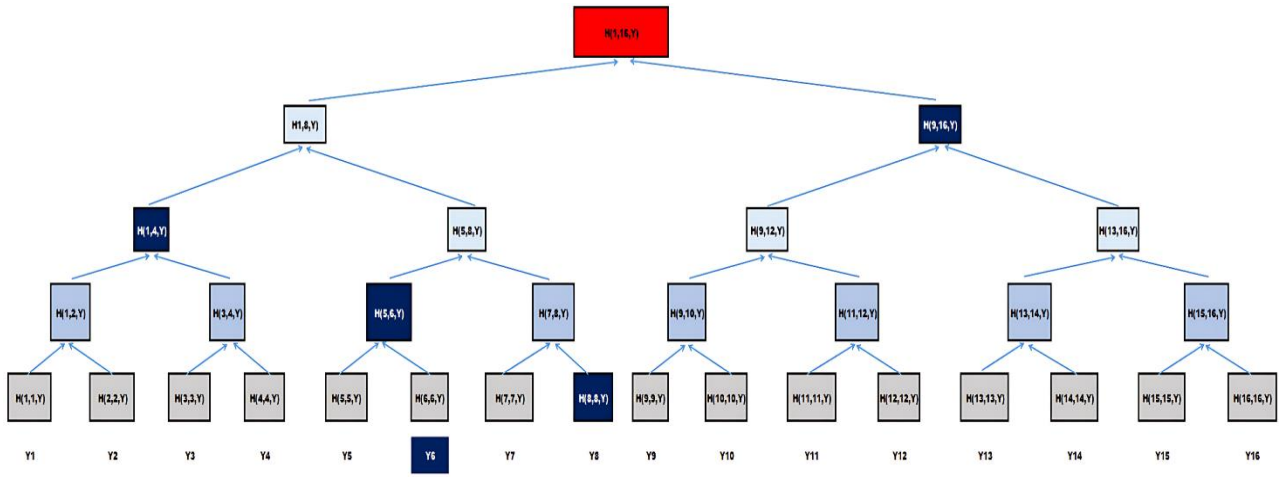

Figure 4.
Graph visualization of creating a Merkle tree model of 16 data blocks.

Appendix 1 shows the pseudocode procedure used by the program to create a Merkle tree from 16 data blocks implemented in Python. In addition, the program's pseudocode includes a modification to compute the root hash using fractional distance scaling. The pseudocode and algorithm provide an overview of the functionality of the program, and key concepts include:

- RSA key generation is used to create public and private keys.
- RSA digital signatures ensure message integrity and authenticity.
- Merkle tree construction for efficient hashing and integrity checking in big data.
- A graph visualization to show the structure of the Merkle tree.

Using Merkle trees, the program successfully combines cryptographic principles and techniques to protect the integrity and authenticity of data. It also demonstrates how these structures work visually. This pseudocode will create a Merkle tree using RSA quantum cryptography, derive the root hash and key pair, visualize the tree in classical form, and output it in a readable format.

A path in a Merkle tree is a sequence of nodes and hashes required to verify the integrity of a particular block of data. The characteristics of the hash function ensure a unique output value corresponding to any combination of two input data sets. This function could serve as a tool to confirm the characteristics of the input parameters. Figure 5 displays the experimental results.

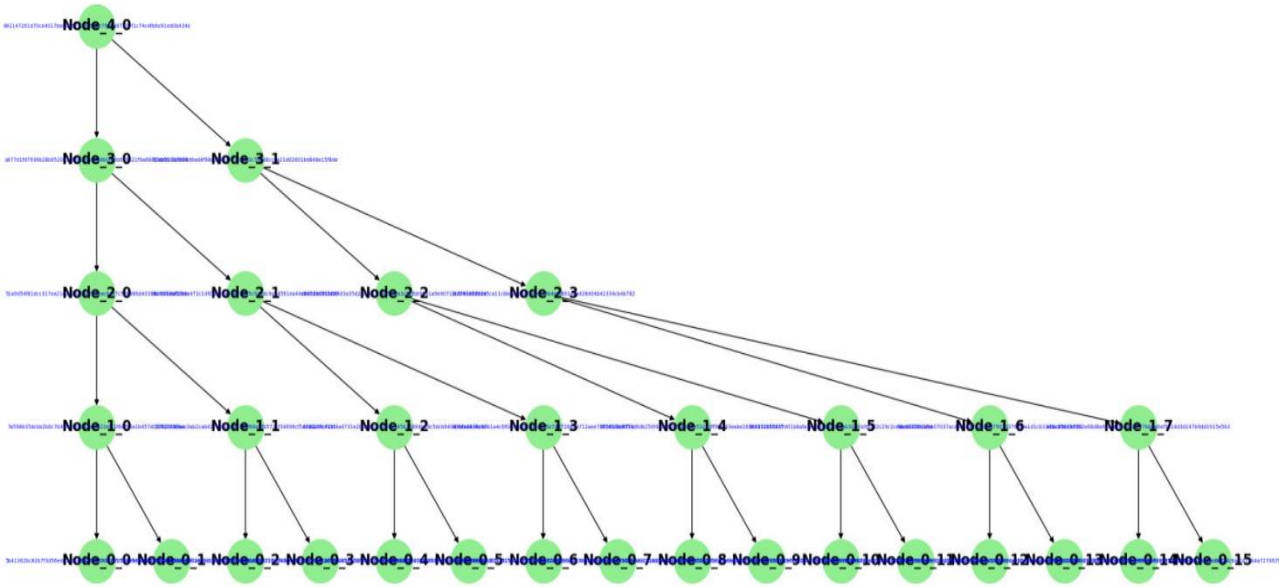

Figure 5.
Graph visualization to show the structure of the Merkle tree.

For example, as shown in the figure above, there are a total of 16 end nodes, and there can be a total of 16 hash operations.

$$\begin{aligned} H_1 &= H(H_1, Y_1) \\ H_2 &= H(H_1, Y_2) \\ H_3 &= H(H_2, Y_3) \\ &\dots \\ H_{16} &= H(H_{15}, Y_{16}) \end{aligned} \qquad (2)$$

Using the Merkle tree, we will need 32 hash operations, and by simplifying the validation dataset, we can save a significant amount of memory space. Thus, we use this Merkle tree structure to organize the security of a privacy-appropriate chain of data blocks.

Example_2: The pseudocode algorithm describes the program's basic steps, which use SHA3-512 hashing and generate public and private keys for post-quantum encryption. Figure 6 and Appendix 2 display the experimental results.

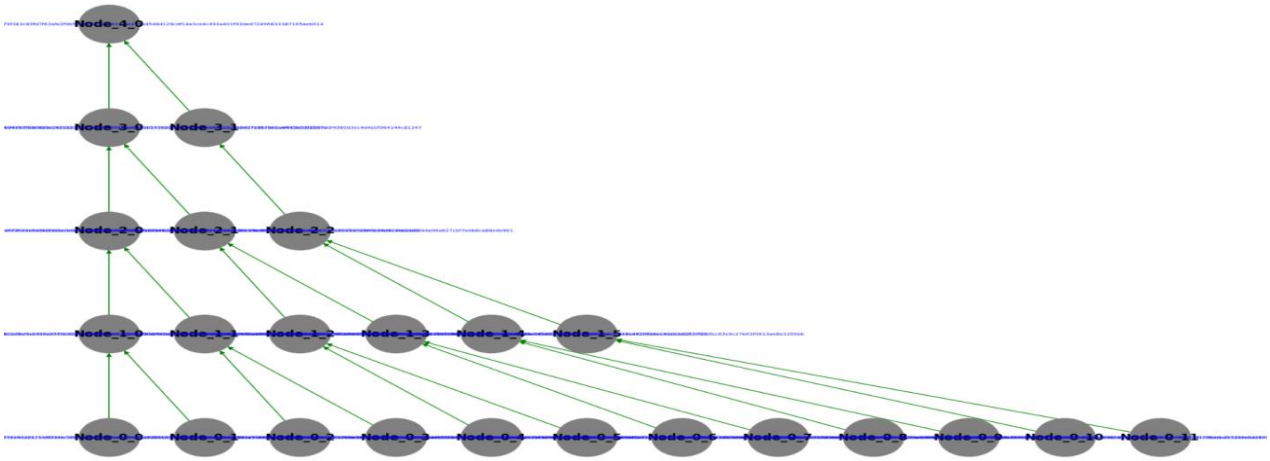

Figure 6.
Graph visualization to show the structure of the Merkle tree in Example 2.

Example_3: Simulation of KYBER post-quantum cryptography using the Merkle Trees hash function for 12 big data technology data blocks. The Kyber algorithm employs quantum-resilient methods for key exchange and data encryption. It is based on a lattice and provides security even against attacks by quantum computers. Figure 7 and Appendix 3 display the experimental results.

The program demonstrates how to work with data hashing, construct and visualize Merkle trees, and other essential aspects of big data technology and cryptography.

The key components of the program with data hashing are:

- *Hashing:* The hash_data() function applies SHA-512 to all data and intermediate levels of the tree.
- *Merkle tree:* At each level, hashing pairs of elements builds the tree from the leaves (data) to the root.
- *Visualisation:* NetworkX displays Merkle's tree as a directed graph, with nodes holding the hash values for each level.
- *Post-quantum keys:* Random hexadecimal strings simulate key generation for Kyber-type cryptography.

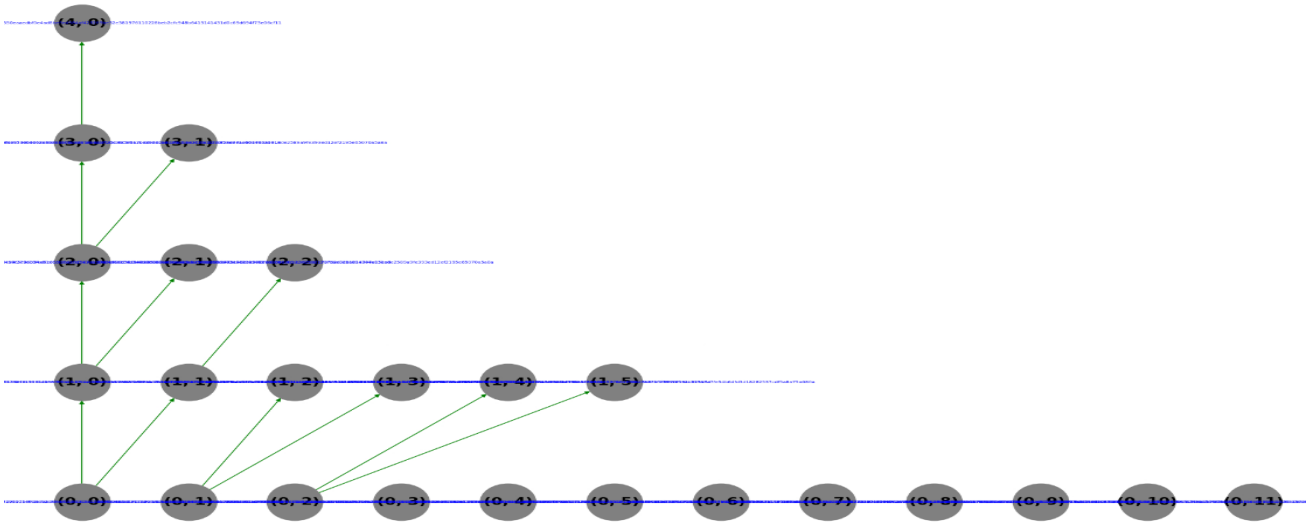

Figure 7.
Graph visualization to show the structure of the Merkle tree in Example 3.

This program performs the following tasks:

- *Key generation for post-quantum cryptography:* The program generates random key pairs (public and private) to simulate post-quantum encryption using the Kyber algorithm.
- *Data hashing:* The program uses the SHA-512 algorithm to compute the data hash.
- *Merkle tree construction:* The program constructs a Merkle tree from a list of hashed data. At each level, the data is hashed until only one root hash remains.
- *Merkle tree visualisation:* The program uses the NetworkX library and Matplotlib to visualize the Merkle tree. It displays the tree as a directed graph, with each node representing a hash of data and edges signifying the relationships between its levels.

Post-quantum cryptography, such as the Kyber algorithm, and Merkle trees for hashing and retrieving information make this algorithm secure and effective for big data technologies.

Despite their widespread use, current research lacks a complete understanding of the security aspects of Big Data. Our research aims to fill this gap by developing a theoretical framework for data security that considers different scenarios based on Merkle path length and hash length. Empirical experiments validate the theoretical models by investigating simulations with different hash lengths and Merkle path lengths.

As part of the study's experimental phase, we conducted several experiments. The results show the importance of finding the optimal hash length and Merkle path length to simultaneously make Big Data safe and efficient.

4. Discussion

This study is particularly pertinent as BIG DATA technologies expand, gathering and processing progressively intricate data structures. First, this suggests that we must meticulously evaluate the depth of Merkle trees in BIG DATA technologies to maintain an optimal level of security. Second, testing the theoretical model in the real world confirms the links between hash length and Merkle path length in hybrid security technologies, such as quantum computing. It offers a pragmatic instrument for assessing the security of Merkle tree implementations. The alignment of theoretical computations with experimental evidence lends confidence to our conclusions. It provides a robust basis for future security assessments in Big Data technologies to uphold an optimal security standard.

Prospective avenues for research. The intricate relationship between hash length and Merkle path length in hybrid security systems, including quantum computing, presents avenues for additional research. Examining the ideal combinations of these factors for various application scenarios might yield more precise recommendations for designing systems utilizing big data and hybrid security methods.

5. Conclusions

This study not only identifies existing gaps in current research but also offers directions for future research. It emphasizes the potential of combining big data with new technologies, such as quantum computing, which paves the way for new developments and applications. In exploring these interdisciplinary connections, the authors highlight unexplored areas that could lead to significant technological advances. This forward-looking approach ensures that the research remains relevant and in line with rapid technological advances.

Finally, a thorough study that combines theoretical models with experimental checks not only provides a complete picture of the connection between hash length and Merkle path length but also enables the verification of data integrity in Merkle tree implementations in hybrid security technologies like quantum computing. The findings from this study play an essential role in advancing Big Data security structures, making significant contributions to data integrity and privacy.

References

[1] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management*, vol. 35, no. 2, pp. 137-144, 2015. https://doi.org/10.1016/j.ijinfomgt.2014.10.007

[2] M. Mahmoudian, S. M. Zanjani, H. Shahinzadeh, Y. Kabalci, E. Kabalci, and F. Ebrahimi, "An overview of big data concepts, methods, and analytics: Challenges, issues, and opportunities," presented at the IEEE Xplore: 5th Global Power, Energy and Communication Conference. https://doi.org/10.1109/GPECOM58364.2023.10175760, 2023.

[3] S. H. Islam, N. Mishra, S. Biswas, B. Keswani, and S. Zeadally, "An efficient and forward-secure lattice-based searchable encryption scheme for the Big-data era," *Computers & Electrical Engineering*, vol. 96, p. 107533, 2021. https://doi.org/10.1016/j.compeleceng.2021.107533

[4] U. Demirbaga, G. S. Aujla, A. Jindal, and O. Kalyon, *Big data analytics," In theory, techniques, platforms, and applications*. Cham, Switzerland: Springer, 2024.

[5] Z. Chenbin, L. Xu, and L. Jiguo, "Fuzzy identity-based dynamic auditing of big data on cloud storage," *IEEE Access*, vol. 7, pp. 160459-160471, 2019. https://doi.org/10.1109/ACCESS.2019.2950938

[6] A. A. Kharlamov and M. Pilgun, "Data analytics for predicting situational developments in smart cities: Assessing user perceptions," *Sensors*, vol. 24, no. 15, p. 4810, 2024. https://doi.org/10.3390/s24154810

[7] N. Gavric, A. Shalaginov, A. Andrushevich, A. Rumsch, and A. Paice, "Enhancing security in international data spaces: A STRIDE framework approach," *Technologies*, vol. 13, no. 1, p. 8, 2024. https://doi.org/10.3390/technologies13010008

[8] N. Stefanovic, M. Radenkovic, Z. Bogdanovic, J. Plasic, and A. Gaborovic, "Adaptive cloud-based big data analytics model for sustainable supply chain management," *Sustainability*, vol. 17, no. 1, p. 354, 2025. https://doi.org/10.3390/su17010354

[9] S. Ren, P. S. Fong, and Y. Zhang, "Enriching value of big data cooperative assets from a time-horizon perspective," *Sustainability*, vol. 16, no. 24, p. 10961, 2024. https://doi.org/10.3390/su162410961

[10] F. Alyami and S. Almutairi, "Implementing integrity assurance system for big data," *Wireless Personal Communications*, vol. 122, no. 3, pp. 2585-2601, 2022. https://doi.org/10.1007/s11277-021-09013-x

[11] A. S. Abdullah-Al-Musa, N. Z. Khidzir, and T. G. Tan, "Towards the big data and digital evidences integrity," *Jurnal Intelek*, vol. 14, no. 1, pp. 1-8, 2019.
[12] B. Jiana, Y. Guo, N. He, and S. Wang, "Research on key technologies of personal information security protection in big data," *Academic Journal of Engineering and Technology Science*, vol. 6, no. 4, pp. 42-47, 2023. https://doi.org/10.25236/AJETS.2023.060407
[13] L. Vardanyan, V. Stehlík, and H. Kocharyan, "Digital integrity: A foundation for digital rights and the new manifestation of human dignity," *TalTech Journal of European Studies*, vol. 12, no. 1, pp. 159-185, 2022. https://doi.org/10.2478/bjes-2022-0008
[14] S. Yin, "A study on data integrity verification result detection algorithm in big data warehouse," *Journal of Physics: Conference Series*, vol. 1574, 2020. https://doi.org/10.1088/1742-6596/1574/1/012008
[15] A. Mehmood, I. Natgunanathan, Y. Xiang, G. Hua, and S. Guo, "Protection of big data privacy," *IEEE Access*, vol. 4, pp. 1821-1834, 2016. https://doi.org/10.1109/ACCESS.2016.2558446
[16] Z. Wentao, "Big data security and privacy protection," in *Proceedings of the 2nd International Conference on Computer Science and Advanced Materials (CSAM 2019). https://doi.org/10.25236/csam.2019.012*, 2019, pp. 52–55.
[17] Z. Dongpo, "Big data security and privacy protection," in *Proceedings of the 8th International Conference on Control and Information Science (ICMCS 2018). https://doi.org/10.2991/icmcs-18.2018.56*, 2018, pp. 275–278.
[18] Z. Gao, N. Desalvo, and K. Pham, "Weidong integrity protection for big data processing with dynamic redundancy computation," *Proceedings of the IEEE International Conference on Autonomic Computing. https://doi.org/10.1109/ICAC.2015.34*, pp. 159 – 160, 2015.
[19] I. Lebdaoui and S. Hajji, "Orhanou Ghizlane Managing big data integrity," in *Proceedings of the IEEE International Conference on Engineering & MIS (ICEMIS). https://doi.org/10.1109/ICEMIS.2016.7745332*, 2016, pp. 1–6.
[20] M. Sookhak, F. R. Yu, and A. Y. Zomaya, "Auditing big data storage in cloud computing using divide and conquer tables," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 5, pp. 999-1012, 2017. https://doi.org/10.1109/TPDS.2017.2784423
[21] G. S. Aujla, R. Chaudhary, N. Kumar, A. K. Das, and J. J. Rodrigues, "SecSVA: Secure storage, verification, and auditing of big data in the cloud environment," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 78-85, 2018. https://doi.org/10.1109/MCOM.2018.1700379
[22] A. Joseph, P. Mathew, and M. Cherian, "Enhanced cryptography techniques using inherited process with cyclic crossover operation in big data," *International Journal of Modeling, Simulation & Scientific Computing*, vol. 15, no. 5, 2024. https://doi.org/10.1142/S1793962324500375
[23] R. Shafia, A. H. Khan, and M. Haroon, "Big data security and privacy: Current challenges and future research perspective in cloud environment," in *Proceedings of the International Conference on Information Management and Technology (ICIMTech)*, 2020, pp. 977–982, doi: https://doi.org/10.1109/ICIMTech50083.2020.9211239
[24] C. Majdoubi, S. El Mendili, and G. Youssef, "Quantum cryptology in the big data security era," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 15, no. 7, pp. 1-10, 2024. http://dx.doi.org/10.14569/IJACSA.2024.0150761
[25] Z. B. Jemihin, S. F. Tan, and G.-C. Chung, "Attribute-based encryption in securing big data from post-quantum perspective: A survey," *Cryptography*, vol. 6, no. 3, p. 40, 2022. https://doi.org/10.3390/cryptography6030040

Appendix 1.
Pseudocode Example_1.

No	Steps	Activity	Description of Key Functions
1.	Function generate_rsa_keys ():	– Generate RSA key pair (2048 bit). – Return public and private keys.	1. This function generates a pair of RSA keys (private and public) with a key size of 2048 bits using the RSA class from the Crypto.PublicKey library. 2. The public and private keys are exported and returned in byte format.
2.	Function sign_message (*private_key*, *message*):	– Calculate the message hash using SHA-256. – Sign the hash using the RSA private key using the PKCS1_v1_5 scheme. – Return the signature.	1. The function calculates the message's hash using SHA-256 and signs the hash with the private RSA key using the PKCS1_v1_5 signature scheme, a widely used signature scheme in cryptography. 2. The function returns the generated signature.
3.	Function verify_signature (*public_key*, *message*, *signature*):	– Calculate the message hash using SHA-256; – Verify the signature using the RSA public key and the PKCS1_v1_5 scheme; – If the signature is valid, return True; – - If not, return False.	1. The function checks whether the provided signature matches the message using the public RSA key. 2. Verify the signature using the public RSA key and the PKCS1_v1_5 scheme. 3. The function returns True if the signature is valid and False if it is invalid.
4.	Function hash_data(data):	– Apply SHA-256 to compute the hash of the data.	This function calculates the SHA-256 hash of the input data and returns the hash as a hexadecimal string.

		– - Return the SHA-256 hash of the input data as a string.	
5.	Function build_merkle_tree (leaves):	– Calculate the hash and store each element in the leaves (data) at level 0. – If the number of elements at a level is odd, duplicate the last element. – -While there is more than one element at a level: – -Hash each adjacent element, creating a new level. – -Return the root of the Merkle tree and all hashes at the levels.	1. The function builds a Merkle tree from the leaves (data), repeatedly hashing pairs of nodes at each level to create a new tree level. 2. The process continues until only one node remains (the root of the Merkle tree). 3. The function also keeps track of the hashes at each tree level. 4. If the number of elements at a level is odd, the last element is duplicated to ensure parity. 5. Once the computation is complete, the root of the tree and the hashes at each level are returned.
6.	Function visualize_merkle_tree (levels):	– Create a directed graph using NetworkX. – Add nodes and edges to represent the Merkle tree structure. – Assign positions to nodes for visualization (leaves at level 0). – Link nodes to parents and neighbors. – Plot the tree using ***matplotlib***. – Label nodes with hashes and plot the graph. – Plot the graph using ***matplotlib***.	1. The function visualizes the Merkle tree using the **NetworkX** library to create the graph and **matplotlib** to display it. 2. Create a directed graph using the **NetworkX** library to visualize the Merkle tree. 3. Nodes are added to the graph for each level of the Merkle tree. 4. The positions of the nodes are calculated based on the tree level to visualize the hierarchical structure. 5. The relationships between parent and child nodes are displayed. 6. The graph is output with signed hashes on the nodes.
7.	Main Function:	1. Create a list of data (e.g., 16 elements). 2. Generate a pair of RSA keys. 3. Output public and private keys. 4. Sign the message and verify the signature. 5. Build a Merkle tree and output the root of the tree. 6. Visualize the Merkle tree. 7. Verify and output a valid RSA signature.	
8.	Result	RSA Public Key: -----BEGIN PUBLIC KEY----- MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA+JoEsfpFiAEXDrHbBcFQdcHMSbxe8ooeWyfYjF3p0AfX6h8gPL51FcFk9WU8wze2eTx72il7+25iHt4cRIbrBjnVB7+XPmlPa89SfSzoT2hFWp6tc5YuPHxfW+NV2JoDbUmqrxZSXwBqm3fHDMJMIUGj0vn0IZCNBNGoPrLuj7NHAVEnTLKqD8HbctBh2n1S+H0Vl1F9wt/dysS7UzWW4GLw3JBJG+v/1dxhxkgd+CeVGq6u73htvheY6Pi2vg9KzAVvsKNkhWm39DNvxLvBMm+qieZDN4A4VijPj4uqbkHOS1Nogm5UuPpWQyPrxh5V14t6HWzJx4fEkCJizBIN9wIDAQAB -----END PUBLIC KEY----- RSA Private Key: -----BEGIN RSA PRIVATE KEY----- MIIEogIBAAKCAQEA+JoEsfpiAEXDrHbBcFQdcHMSbxe8ooeWyfYjF3p0AfX6h8gPL51FcFk9WU8wze2eTx72il7+25iHt4cRIbrBjnVB7+XPmlPa89SfSzoT2hFWp6tc5YuPHxfW+NV2JoDbUmqrxZSXwBqm3fHDMJMIUGj0vn0IZCNBNGoPrLuj7NHAVEnTLKqD8HbctBh2n1S+H0Vl1F9wt/dysS7UzWW4GLw3JBJG+v/1dxhxkgd+CeVGq6u73htvheY6Pi2vg9KzAVvsKNkhWm39DNvxLvBMm+qieZDN4A4VijPj4uqbkHOS1Nogm5UuPpWQyPrxh5V14t6HWzJx4fEkCJizBIN9wIDAQABAoIBAHmz0NOebXao7fRELrCRFEYpw0xfuu2qnzTJ+3GwMhwlEtcrOLl0Fda6MCYQUyBQwHR2nz1MrMUIo2FwI8+KazlHi5o4+Z5UqMxjGQNtvibNqZI09z4MSgNTcc2GrAqXT1ErzLLdTyLyDb6hcyE0Z1eqjjxlIFZkMR0oVGCpcS+IQhYZY4MJ/H01f8IOfxh/j88CBVUndloGcKy7PiuSNgE8IcG66FuEXXFsMj5tMJfgnfyRHKGYMCmqg2M0KYoy7nPYe/7TweM/OWhuMIWF29j+1Bf/FYtPJkXxBLKdpsXEWr0q2a72AFar8uWkxNuw+m+F4PkHdxmoQvqW0PHXAtUCgYEA+NY

		Bx2qEK8l4Yoi5Z6/yjiiAIqr/Ef5r1Gr90i5ciBUftlPdEeLUfKVie27rb4dv7E5nodCHbhlEjtlG0Jka qzAzHyW8ajH/K2SJmliHXYStsHgnxfpRxBaDlVaJ3BSVmdWsfjeeVFdLihKaehDrXWcDhloX SP2QixK5QCBy6nMCgYEA/8JIyGlOFzCitjO5PJGm2Vy9/pVTnB0EDiLmh2ymRAYGMSXH 5qpzRPQVqwDL5ByZZ0V/F2iK9G8rxHD07BLxWtNw38hGrat6PzKkMFTuKmfFW/quz09H2 uuDvd/D31/gHxRF/UOlTnblq318nRdGyCEvABZs+QnFt3djVpoFGW0CgYBHKjVSCeWI/FK go4ea3tgYzf5va8YNKF1AX+T9kmLGTHxIUc28ysmIaGg08vEEcZCR3Y6H2bKRy0ovLHazO 8ZDxUIpm0o6eNgIWfV8nesA1OwztvlxBnkSAnpZYGmG9qAeYdy2MEPozRp+f3mMhYA6sz Zv2x4jstj+3Tj73QvgfwKBgHLv0N2f1WQ+iyDupbQ+kIjmUMv0N5rUv1nK/WamwSGYw3eaz vcuapb2SjaLKRXedGIgrTjL1oZ632RsjuT4rlZu5y7qRjkN8uYrz/WYdV/BOsRXJ/AqZNRROuc 9hD0CTSoqZjmnCMb2T8h9oF7IcxrFLuYPLfHNHnmQwafNl7URAoGAbKNRwEt4nUzjzUZ 3i01A607HfXdQpIcAix5gQO1qPHJoXJfWV8a9jclbEGPkQsPqqfP0JXmakKJv53B+igz+Nbxn S88TNdUpGAyBgFWc2tKTvAR7IlACC5R+kqU+ZY1Gim4hvPN4mwb34hggtDFarEZiNoTlos PcXTHKFgVhTuM= -----END RSA PRIVATE KEY----- Building Merkle Tree... Level 0 (leaves): ['5b41362bc82b7f3d56edc5a306db22105707d01ff4819e26faef9724a2d406c9', 'd98cf53e0c8b77c14a96358d5b69584225b4bb9026423cbc2f7b0161894c402c', 'f60f2d65da046fcaaf8a10bd96b5630104b629e111aff46ce89792e1caa11b18', '02c6edc2ad3e1f2f9a9c8fea18c0702c4d2d753440315037bc7f84ea4bba2542', 'e195da4c40f26b85eb2b622e1c0d1ce73d4d8bf4183cd808d39a57e855093446', '9c67b4b76a18503009f542ef7c93dc7ac94aebbc6141515bea4e63e3068373a6', '28b815d8952fd6517f2584a3f5f300a9b1289faeadfac4fc6d6bd5a781e75df5', 'b5cc74ab5bb5a5f1acc7407be3e4cbce8611c5ed07354ab9e510b74ee0b273cb', 'bbe0aa41024faeac81813a0194a95637d54cc65c025e0efd857ce0afcd51573f', '1d7cf04971d73d83d66e430a406a1fc0c9cce8d90cb108c7c9d253e96b94cb85', '29e8c5e2631fcef93a0912f6a6526b77e499eacf80cf4150a858ae2e8cd1e5b5', 'f4c7ef276b49b82aabf55d2ca438be349d1e5ee16544f426467fa05aefa132e2', '0e69703c8a1492efdf1fe666a3b4b0ad2133a4aca7bb4789eb4c8d54844c0f04', '15033b3da8a3f4ee46a9fea3eb5b2341c08df8981c2b616ce0b4925cae6981e6', '87fef323635f22fc88999957a585c3cf3f8383029c8501e52928a14028c0af2c', '86f4eb3ab6578e5d16e3ed2e04457c7cea474c8af2471b4a7274fcf9a1a2b2e1'] Level 1: ['7a598b35dcbb2b6c7b45ffc1e4152a1f822ef41f68fff3a1b457d057629d89ec', '23431736aac0ab2cab427b40cae8253bf66e3fb5721f34696cf54730aefce451', 'ceab270c71ef4a4731e2bbedec99789e945b15864b99c5dcb64087bfa6b38cbf', 'ebd4aa436e4db1a4c6f0d78c98cef9c465b74ff728804712aee78716bbbdf77a', '995613b3d53d6db250f42b9b205daca9e532a3ff9fe153eabe2650a112c658c5', 'de7d7c850737d01b8afa3ec99489791dab30afcb96432c29c2cedeaa303f4a9e', '54e9529bc1feb07037acd631a823a88d47901d3768fda1d1cb33efbc49c1b380', 'e1ac858e56762e68d8a92efa4f0c3e2f737bb6a6ef5814d3d147b9dd1915e5b3'] Level 2: ['51a0d54f81dcc317ea21d2125c65d796eac64e7c52b886d40388cf1f1abf93eb', '4bceb66a020ea472c1d904bc8a5082e8d5c972dc9a24561ea44e8a63808f3db9', 'cb658ad910083d3a35d2ac2b110345ef0a3c52fbb9531a9e40712b5f40453874', '0c4761b88caa5ca11cbbd21d5af9e0bcf8b4e9f861eda428404b42334cb4b782'] Level 3: ['a977d1fd7936b28b9520d3266b2ebc9f68b0f66b676621f9a688c9666c0ef496', '52aa51303b88d6ad4f98ed59b37732f2b0c7b448cc6a21d02601bd848e15f8de'] Level 4: ['841147261d70ce4017be0035772905d227976487277f1c74c4fb6e91ed0b434e'] Merkle tree root: 841147261d70ce4017be0035772905d227976487277f1c74c4fb6e91ed0b434e

Appendix 2.
Pseudocode for Example_2

No	Steps	Explanation of Pseudocode
1.	Function to hash data using the SHA3-512 algorithm	– Function hash_data(data): – Convert the string 'data' to bytes using UTF-8 encoding. – Apply the SHA3-512 hash function to these bytes. – Return the hash as a hexadecimal string.
2.	Function to generate keys for post-quantum cryptography	– Function generate_post_quantum_keys(): – Public_key = Generate a string of 128 random characters from "0123456789abcdef". – Private_key = Generate a string of 128 random characters from "0123456789abcdef". – Return public_key and private_key.

3.	Function to build a Merkle tree.	– Function build_merkle_tree(leaves): Tree = Empty list. – For each leaf in 'leaves': Add the hash of 'leaf' to 'tree'
4.	Build the tree: hash pairs of elements at each level. – – – Return to the root of the tree.	– While the length of the 'tree' is greater than 1: – If the length of the 'tree' is odd: – Add the last element to 'tree' (duplicate it) – New_tree = Empty list – For each i from 0 to len(tree)-1 with a step of 2: Parent_hash = hash_data(tree[i] + tree[i+1]). – Add the 'parent_hash' to 'new_tree'. – Update 'tree' with the new hashes from 'new_tree' – Return tree [0].
5.	Function to visualize the Merkle tree.	– Function visualize_merkle_tree(leaves): – G = New directed graph (DiGraph)
6.	Hash all leaves and add them to the graph	– Tree = Empty list. – For each leaf in 'leaves': – Leaf_hash = hash_data(leaf) – Add 'leaf_hash' to 'tree.' – Add a node to the graph with the label 'leaf_hash.'
7.	Build the tree	– level = 0 – While the length of the 'tree' is greater than 1: Parent_level_nodes = Empty list – For each i from 0 to len(tree)-1 with a step of 2: – If i+1 == len(tree): Parent_hash = hash_data(tree[i]) – Else: – Parent_hash = hash_data(tree[i] + tree[i+1]) – Add the parent node with the label 'parent_hash' to the graph – Add edges between the nodes and the parent node – Add the parent node to 'parent_level_nodes' – Update 'tree' with the new hashes from 'parent_level_nodes.'
8.	Set up the positions for visualization.	– Pos = Empty dictionary. – For each node in the graph: – Extract the level and index from the node's identifier – Set the position of the node on the X-axis (index) and Y-axis (level)
9.	Visualize the graph. –	– Set visualisation parameters: node size, edge colour, labels, etc. – Display the graph with edges and labels.
MAIN PROGRAM:		
10.	– Main (): – Data for the Merkle tree.	1. data = ["data1", "data2", "data3", "data4", "data5", "data6", "data7", "data8", "data9", "data10", "data11", "data12"]
11.	– Generate public and private keys for post-quantum cryptography.	– Public_key, Private_key = generate_post_quantum_keys() – Print the public_key – Print the private_key
12.	– Build the Merkle tree and print the root.	– Root_hash = build_merkle_tree(data) – Print the root_hash
13.	– Visualize the Merkle tree.	– Visualize_merkle_tree(data).
14.	– RESULT:	– PUBLIC KEY: 4ea4fcc3bf493475aec12686afcbe2e955e604843700347a7d2f4a871629d845e20ce9fd2aa3e628d7c14aa5216401e043fab966b07984ba7646d3062a1626ec – PRIVAT KEY: a904509f4ee531f8cc525bbd72871d4fdb759e401f440a3e4e52bda24e4b352199678ec88765d12c7fdeb469e8ae1201e227e78f3a30e488c08977bbf461614e – Merkle tree root: 6def4162caa73d21fffc2a371429ebb2b3bf8b8dd08b866e6d90a15e3657f57325abdd881b5815b930f048855585b2dc5a2812138bfcbfe6a3f7ddad31ecae2f

Appendix 3.
Pseudocode for Example_3.

No	Steps	Description of Key Functions	Explanation of Pseudocode
1.	Function hash_data(data):	– Take input `data` (string). – Apply SHA-512 hash function to `data`. – - Return the resulting hash as a string.	This function takes a string (data) as input, applies the SHA-512 hashing algorithm, and returns the resulting hash.
2.	Function generate_post_quantum_keys ():	– Initialise an empty string `public_key`. – Generate 128 random hexadecimal characters and append them to `public_key.` – Initialise an empty string `private_key`. – Generate 128 random hexadecimal characters and append them to `private_key.` – -Return the `public_key` and `private_key`..	This function generates random hexadecimal strings for both the public and private keys. It simulates the process of generating keys for post-quantum cryptography, specifically for Kyber (though this is a simplified simulation).
3.	Function build_merkle_tree (leaves):	– Set the initial tree to be the `leaves`. – While the length of the tree is greater than 1: – If the number of elements is odd, duplicate the last element to ensure pairs. – Hash each pair of consecutive elements and create the next level. – Set the tree to the newly created level. – Return the single remaining element, which is the root hash of the Merkle tree.	This function constructs a Merkle tree by iteratively hashing pairs of elements from the leaves until a single root hash is obtained. If the number of elements at any level is odd, the last element is duplicated to ensure pairs can be hashed.
4.	Function visualize_merkle_tree (leaves):	– Create an empty directed graph `G` using `networkx`. – Create an empty list of `levels` to store the tree at each level. – For the first level (level 0), hash each leaf and add it to the `levels` list. – Set the `current_level` to be the leaves – While the `current_level` has more than one element: – Create a new list `next_level` to store the hashes of pairs. – If the number of elements is odd, directly add the last element to `next_level`. – Otherwise, hash each consecutive pair of elements and append the result to `next_level.` – Add the `next_level` to the `levels` list – Set `current_level` to be the `next_level` – Add nodes and edges to the graph for each level: – For each level, for each node, add the node to the graph and assign the corresponding hash as a label. – For each node at levels greater than 0, connect it to its parent (the corresponding node in the previous level). – Set the positions of the nodes: – Assign vertical positions such that the root node is at the top and leaf nodes are at the bottom – Display the graph using `matplotlib`: – Draw the graph with nodes and edges. – Label the nodes with their respective hash values. – Set up a title for the plot and display it.	This function builds the Merkle tree, constructs a directed graph using the NetworkX library, and visualizes the tree using Matplotlib. The graph's nodes represent the hash values at each level, and the edges represent the parent-child relationships between nodes at consecutive levels. The graph is displayed in a rotated fashion (180°).
5.	Main Program:	– Generate 12 data blocks (for example, `block_data_1`, `block_data_2`, ..., `block_data_12`).	The main program generates a set of 12 data blocks and processes them to create a Merkle tree. It also generates a pair of keys for post-quantum cryptography (Kyber, in

		– Call the generate_post_quantum_keys()` function to generate a public and private key. – Print the generated public and private keys. – Call the `build_merkle_tree()` function to generate the Merkle tree root from the data blocks. – Print the root hash of the Merkle tree. – Call the `visualize_merkle_tree()` function to visualise the Merkle tree.	this case) and prints both the public and private keys. It then builds the Merkle tree from the data, prints the root hash, and visualizes the tree.
6.	RESULT:	GENERATING POST-QUANTUM CRYPTOGRAPHY KYBER KEYS: Public Key: 4a6f4ecb773b7f2366982d866c0c607fa76537a67dec3a96ba006407c270d20ce9251785dc4c10e0187287104880d0d2adb96d22218f6e6ee09e7e9baef8633a Private Key: df5af0c22d9591834fec9d52f259008319628cb364775186aeec43760ae36cd5ae6d70356d8eafd3a24da064b7a08de964b987902945e485b13aee373bdf8cd6 MERKLE TREE ROOT: Root of the tree: bc7a280c9c308396333acf5f3f7a3683ebcace11c21e26f5b202cb0ec7471576788af8c3b055c3c2cc5e3b920dccdae5a2cabc90ef2cc1ba1a24f7576bd2317f	