

Article

Blockchain and Homomorphic Encryption for Data Security and Statistical Privacy

Rahul Raj ^{1,*}, Yeşem Kurt Peker ¹ and Zeynep Delal Mutlu ²¹ TSYS School of Computer Science, Columbus State University, Columbus, GA 31907, USA; peker_yesem@columbusstate.edu² Independent Researcher, 1200 Brussels, Belgium; zeynepdelalmutlu@gmail.com

* Correspondence: raj_rahul@students.columbusstate.edu

Abstract: This study proposes a blockchain-based system that utilizes fully homomorphic encryption to provide data security and statistical privacy when data are shared with third parties for analysis or research purposes. The proposed system not only provides security of data in transit, at rest, and in use but also assures privacy and computational integrity for simple statistical computations. This is achieved by leveraging the attributes of the blockchain technology, which provides availability and data integrity, combined with homomorphic encryption, which provides confidentiality of data in use. The computations are performed on smart contracts residing on the blockchain, providing computational integrity. The proposed system is implemented on the Zama blockchain and performs statistical operations including mean, median, and variance on encrypted data. The results indicate that it is possible to perform fully homomorphic computations on the blockchain. Even though current computing limitations on the blockchain do not allow running the system for large data sets, the technology is available, and with advancements toward more efficient homomorphic operations on blockchains, the proposed system will provide an ultimate solution for providing the much-desired security properties in applications, including data and statistical privacy, confidentiality, and integrity at rest, in transit, and in use.

Keywords: data privacy; data integrity; computational integrity; homomorphic encryption; blockchain; statistical privacy; data security; statistical confidentiality



Citation: Raj, R.; Kurt Peker, Y.; Mutlu, Z.D. Blockchain and Homomorphic Encryption for Data Security and Statistical Privacy. *Electronics* **2024**, *13*, 3050. <https://doi.org/10.3390/electronics13153050>

Academic Editor: Simeone Marino

Received: 8 June 2024

Revised: 20 July 2024

Accepted: 26 July 2024

Published: 1 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Protecting the privacy of users and the confidentiality of sensitive data is vital in all systems where user data are stored, transmitted, or processed. Often, data need to be analyzed for business purposes or for the benefit of the public. For example, census data record information about the population of a country, state, city, or other well-defined geographical areas. They include sensitive information such as age, race, gender, religious beliefs, and long-term health conditions in many cases. Census data are the primary data used to understand the social, economic, and demographic conditions locally and nationally and provide information that governments need to develop policies, plan and run public services, and allocate funding. The protection of census data is essential from collection to processing. One safeguard is to allow only authorized people access to the data. Encryption is a common mechanism to implement this safeguard. Data are encrypted in storage and transmitted between the authorized users. However, there is the risk of unintentional mistakes by authorized users or an authorized user turning malicious, jeopardizing the security and privacy of the data and the integrity of the analysis. In some cases, data need to be made available to third parties (e.g., researchers) so that proper and objective analysis can be performed. In these cases, to protect user privacy, data are de-identified or anonymized before they are shared. However, these methods do not always preserve user privacy as intended. In addition to unintentional mistakes and the risk of insider

threats, when large data sets are made available, user information may be leaked due to more sophisticated attacks. especially when large data sets are made available. The 2016 study “Exposed! A Survey of Attacks on Private Data” by Dwork et al. presents a survey of common attacks against aggregate data. It describes reconstruction and tracing attacks. Reconstruction attacks involve approximating sensitive features of individuals within a data set. Tracing attacks aim to determine whether a target individual’s data are included in the data set or not [1]. Dwork’s study is followed by many studies focusing on attacks to data privacy and mechanisms to defend against them [2–6].

Two terms related to security of data in the statistical sense are statistical privacy and statistical confidentiality. Statistical privacy is “the art of designing a privacy mechanism that transforms sensitive data into data that are simultaneously useful and non-sensitive” [7]. The goal of statistical privacy methodologies is to “enable broad sharing of data across different data contexts and domains where it is desired or required that individuals’ identities or sensitive attributes are protected (e.g., census, health, genomic data, social networks)” [8].

Statistical confidentiality refers to the “protection of information of individual statistical units”, or equivalently, “the protection of data collected for statistical purposes” [9]. Statistical confidentiality is closely related to statistical privacy. Even though no clear distinction between these terms exists in the literature, based on the definitions referenced in this article, we consider statistical confidentiality to mean the protection of the statistical units, i.e., data collected on individuals or businesses, and ensuring the approved use of them. Statistical privacy is the broader security concept that includes the transformation of sensitive data into non-sensitive and useful data. Statistical confidentiality is one of the fundamental principles of official statistics endorsed by the General Assembly of the United Nations [10]. It is also a fundamental principle of European statistics, defining principles, rules, and procedures to protect confidential data while still permitting their use for statistical purposes [11].

The first publication of Innovations in Federal Statistics [12] discusses the challenges of and the need for a new paradigm in the federal statistical system. Chapter 5 of the publication titled “Protecting Privacy and Confidentiality While Providing Access to Data for Research Use” focuses on protecting the confidentiality of the information collected for statistical use. After discussing the security implications of linking multiple data sets for analysis, including the legal foundation for privacy and confidentiality, the authors draw two major conclusions: “As federal statistical agencies move forward with linking multiple data sets, they must simultaneously address quantifying and controlling the risk of privacy loss” and “Privacy-enhancing techniques and privacy-preserving statistical data analysis can potentially enable the use of private-sector data sources for federal statistics”. With these conclusions, they recommend “Federal statistical agencies should adopt modern database, cryptography, privacy-preserving, and privacy-enhancing technologies”.

In this study, we propose a system that provides data security, statistical privacy, and confidentiality using techniques and tools purely from mathematics and computer science. We explore the use of a blockchain-based system that allows performing statistical calculations on encrypted data by utilizing fully homomorphic encryption on the blockchain using smart contracts. The proposed system provides end-to-end security of the data from the time they leave the data owner to the point when the researcher receives the results of statistical computations. The provided security includes the integrity and confidentiality of the data in transit, at rest, and in use. Not only the individual data points are never revealed in plaintext, but the results are available only to authorized users in plaintext form. Also, because the computations are carried out on the blockchain, tampering with the algorithms for the statistical computations is almost impossible. Moreover, requesting statistics other than what is approved on the smart contract is not possible. As such, the proposed system provides statistical privacy and confidentiality desired in various applications, including federal statistical systems. In Section 2, we provide brief descriptions of recent studies where blockchain and homomorphic encryption are used to provide data security. In

Section 3, we describe our proposed system and the implementation of a proof of concept for the system. In Section 4, we present the results of running the proof of concept, and finally, in Section 5, we provide a discussion of the results.

2. Related Work

There have been numerous studies that have attempted to introduce data security and integrity not only when they are at rest but also when they are in use through various means, including homomorphic encryption and blockchain technology. One such study proposes a system for a collaborative data training paradigm for medical image data sharing where the blockchain ledger provides the decentralization of the federated learning models without relying on a central server, and the homomorphic encryption alleviates the concerns related to raw data sharing [13]. Their system encrypts the gradients homomorphically and shares them through the blockchain.

Liang et al. [14] propose the integration of blockchain and homomorphic encryption to address the challenges in circuit copyright protection. Their study proposes a homomorphic encryption-based mathematical model within the blockchain that secures the transactions while also ensuring integrity and confidentiality of data by utilizing smart contracts.

Yaji et al. [15] proposed a system that utilizes Goldwasser–Micali and Paillier encryption schemes for the comparative evaluation study with a focus on data privacy techniques using blockchain technology for AI applications. The study found that attacks on the blockchain such as collision, preimage, and attacks on the wallet can be avoided through encrypting blocks using the proposed Goldwasser–Micali and Paillier encryption schemes.

Mutlu et al. [16] proposed a system that uses blockchain technology and homomorphic encryption and enables third parties (researchers) to perform linear regression on encrypted data. They use the Paillier algorithm to calculate the sum required for linear regression through smart contracts. The data owner encrypts the data using the public key of the researcher and sends them to the smart contract where the calculation is performed. The encrypted result can then be accessed by the researcher, who can decrypt it on their system using their private key.

Vanin et al. [17] propose a model to secure Personal Health Record (PHR) that uses an interplanetary protocol file system based on distributed hash tables (DHTs) along with the blockchain. PHR metadata are stored on the blockchain and shared across the network, while PHR data are stored off-chain through the IPFS network. They use two elements: Data Steward (DS), which is responsible for storing PHR on behalf of the individual, and Shared Data Vault (SDV), which is a temporary IPFS storage area where health institutions can access PHR with the consent of the individual. Encrypted data are available to the public through statistical portals, where they can perform operations on the data using homomorphic encryption to obtain meaningful results. For this purpose, they use the Microsoft Simple Encrypted Arithmetic Library (SEAL), which implements the BFV algorithm in JavaScript.

Umar et al. [18] proposed a model for e-voting using the Paillier algorithm. Once the voter casts their vote, it is encrypted homomorphically. A new block of the transaction is then created, which contains the encrypted ballots, the pseudonymous address of the voter and the admin, the timestamp of the block creation, the hash of the previous block of the transaction, as well as the hash of the current block. Then the new block of the transaction is mined using the consensus mechanism. After mining, the new block is committed to the ledger of the blockchain. This process continues until the end of the election, after which the admin tallies the encrypted votes using the Paillier algorithm, which yields the final sum, which can then be decrypted to determine the results.

Shrestha et al. [19] conducted a study that analyzes the security concerns with the Internet of Things (IoT). One of the major concerns that has been highlighted is the privacy of data. This study also explores the possibility of integrating the IoT with the blockchain and with homomorphic encryption. The benefits include data immutability, unforgettability, removing single points of failure, and confidentiality of data.

Caldarola et al. [20] proposed the Neural Fairness Protocol, a consensus mechanism that integrates the Elliptic Curve Digital Signature Algorithm (ECDSA) and cryptographic techniques. This protocol enhances anonymity and accountability in blockchain transactions, particularly through the development of a robust threshold ECDSA algorithm. Beyond its initial application, this innovation shows potential for improving security across various cryptocurrencies, such as Litecoin and Mastercoin.

The proposed system in this study can be compared to the work proposed by Liang [14], Yaji [15], and Mutlu [16] in the sense that it utilizes blockchain technology along with homomorphic encryption to provide privacy of data while they are being used. Furthermore, the system proposed by Mutlu [14] utilizes partial homomorphic encryption over the smart contract, which is similar to the architecture of our proposed system. However, the above-mentioned studies utilize the Pallier algorithm, which is partially homomorphic, whereas this study uses a fully homomorphic system that not only allows addition but also multiplication on the encrypted data, allowing a broader range of computations to be performed securely.

3. Materials and Methods

In this section, we describe a system that allows secure data sharing and analysis between data owners and third parties. The data owners are entities or individuals that have access to the plain data. They could also be seen as sensors with the capability to process data and transmit them. The owners are authorized or give consent to have the data analyzed with the condition that privacy of the data is preserved. The third parties in the system are researchers. They could be individuals or an entity authorized to do certain types of analysis on the data. They may not have the authorization to view the data in plaintext.

3.1. Description of the Proposed System

The system encrypts the data using a homomorphic encryption scheme before they leave the data owner. The encrypted data are stored on a smart contract on the blockchain. The analyses that can be performed on the data are also stored on the smart contract. When a researcher needs to do analysis on the data, assuming that the type of analysis is available on the smart contract, the analysis is performed on the encrypted data using homomorphic computations. The results of the homomorphic computations are also encrypted. To receive the result on their end, the researcher requests the result to be re-encrypted with the researcher's key so that it can be transmitted securely to the researcher and decrypted only by the researcher. It should be noted here that the plaintext data are never revealed during the entire process, even during re-encryption. When the researcher sends their public key to the smart contract for re-encryption, the smart contract initiates a distributed decryption protocol through which parts of the result are decrypted by different validators, re-encrypted using the researcher's public key, and then combined to obtain the final re-encrypted result [21]. Figure 1 shows an overview of the system with a data owner and researcher as main actors.

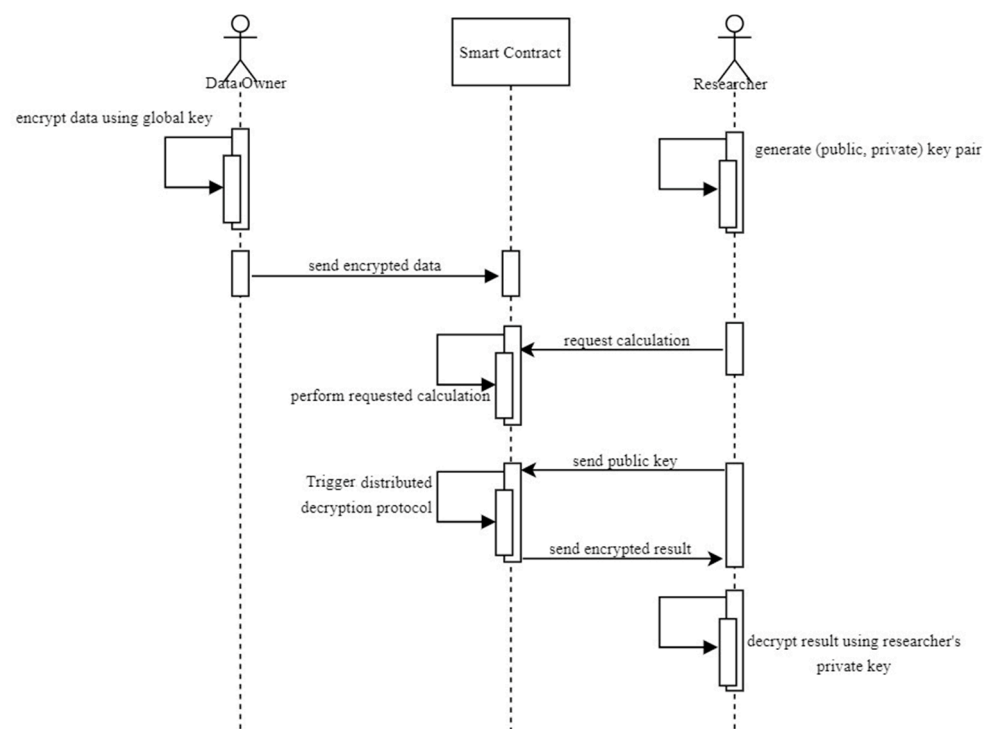


Figure 1. An overview of the proposed system.

3.2. Implementation

The proposed system is implemented on the Zama Blockchain, an Ethereum Virtual Machine (EVM)-based blockchain that supports computation on encrypted values [18]. Like Ethereum, which has a currency known as Ether, Zama has its own currency, called ZAMA. The basic idea behind Zama is to provide confidential smart contracts. This means the data sent to or received from the smart contract are encrypted and cannot be read if the data transfer is intercepted. Zama uses asymmetric encryption and hence uses two keys: public and private. The public key, also known as the global key, is stored publicly on-chain and is used by every user to encrypt their data and perform calculations on those encrypted data. The global key is generated during a setup phase by the initial validators and securely re-shared when the validator set is changed [21]. This allows mixing of encrypted data from multiple users and across multiple smart contracts. The private key is used to decrypt the data and is not owned by any single user. Instead, Zama uses a threshold protocol. In a threshold protocol, pieces of the private key are distributed among validator nodes in the network, and a certain number of validators need to cooperate to decrypt the data. In the case of Zama, the participation of at least one-third of the validators is necessary to perform decryption [18]. This method enhances security through decentralization, preventing rogue use of the key. Zama uses a probabilistic encryption scheme, which means that the encrypted value for a single plaintext will not always be the same. For example, consider an array having 4 elements [2, 4, 1, 4]. Once the elements of this array are encrypted, hypothetically, it would become [d5048, e2314, x3215, o9849]. Notice that the number 4 appears twice in the plaintext array at index 1 and index 3, but its encrypted version does not have any repetitions, and the values at index 1 and index 3 are different. This scheme further enhances security by preventing attackers from deducing information about the plaintext based on patterns in the ciphertext.

The proposed system uses the libraries for homomorphic calculations provided by Zama. Once the contract is compiled, it produces the Application Binary Interface (ABI). The ABI specifies the functions available in the smart contract along with their parameters and return types, allowing users to interact with the smart contract. The data owner and researcher both have their own JavaScript applications that utilize the ABI and methods

provided by the Ethers library to communicate with the smart contract. The data owner sends the data to the smart contract, and the researcher requests analysis from the smart contract. Figure 2 depicts the interaction of the actors with the smart contract.

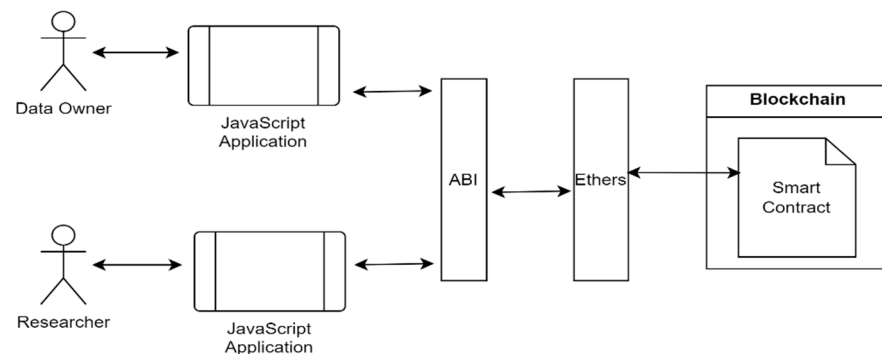


Figure 2. Interaction with the smart contract.

Following is a list of tools and technologies that were used to implement the system:

- Solidity: A programming language that is used to write smart contracts on various platforms, including Ethereum.
- Node.js: It is a JavaScript runtime environment that allows execution of JavaScript code outside the web browser. Version 18.18.0 was used for this system.
- Remix: It is a browser-based Integrated Development Environment (IDE) used for the development and deployment of smart contracts.
- MetaMask: It is a cryptocurrency wallet that is used to connect to and interact with the blockchain.
- Visual Studio Code: It is an IDE that supports development in various programming languages, including JavaScript.
- Ethers.js: It is a JavaScript library that allows interaction with smart contracts. Version 6.10.0 was used for this system.
- FhEVM: It is a library provided by Zama that allows the creation of confidential smart contracts on EVM using Solidity. Version 0.4.0 was used for this system.
- FhEVMjs: It is a JavaScript library provided by Zama that allows interaction with smart contracts, including encrypting the data and generating public/private keys. Version 0.4.0 was used for this system.

A proof of concept for the proposed system is implemented as part of this study. Due to limitations in the libraries for homomorphic computations on Zama and working on a public blockchain, as will be elaborated in Section 3.2.1, the implementation included calculations of simple descriptive statistics such as mean, median, and variance. The proof of concept was tested with a small number of data points, with the range of numbers between 1 and 50.

3.2.1. Challenges

While implementing the system, the following limitations and challenges were identified in Solidity and Zama:

- Data Types: fhEVM only supports unsigned integers that are either 8-bit, 16-bit, 32-bit, or 64-bit. Furthermore, the methods provided by fhEVM for calculations on encrypted data such as addition, subtraction, multiplication, and division return an encrypted unsigned integer. In the case of division, there is an added limitation where an encrypted integer can only be divided with a plaintext integer. This means it is not currently possible to divide two encrypted integers. Even if there was a possibility for such division, the result would always be an encrypted integer because of its return type; hence, it is not possible to work with decimal numbers. Moreover, the integers

are unsigned; therefore, it is not possible to handle negative integers either. These limitations restrict the number of analyses that can be performed on the smart contract.

- **Gas Limit:** Computations on the blockchain run on gas, a unit of cost associated with performing a transaction or computation on the network. Gas is used to pay validators for the resources they use to conduct transactions. Gas limit refers to the maximum amount of gas that can be spent on a transaction or computation. Setting a limit on gas consumption is important because it ensures that transactions have a predefined upper limit on resource consumption to safeguard against scenarios where code execution might enter an infinite loop. Currently, there is a gas limit of 10,000,000 on Zama devnet [22]. However, homomorphic calculations are quite expensive in terms of resources, and since these calculations are performed on the smart contract, they consume a lot of gas. Gas consumption depends on the operations that are performed on the encrypted data and increases with the complexity of the algorithm and bits of data type (32-bit operations would consume more gas compared to 8-bit operations). The current gas limit restricts the functionality in terms of the number of data points that can be processed.
- **Solidity:** Unlike traditional programming languages such as Python and Java, Solidity has various limitations. One of them is that the data structure used to store key-pair values (mappings) is quite limited in terms of functionality, and it is not possible to iterate over keys or values. This makes it challenging to implement various algorithms, such as the calculation of the mode (the most frequent value in an array).

3.2.2. Calculation of Mean

For a set of values $\{x_1, x_2, \dots, x_n\}$, the arithmetic mean, denoted by $\bar{x}$, is the sum of the values divided by the number of values, as shown in Equation (1).

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad (1)$$

In the designed system, only the numerator, the sum of the encrypted values, is calculated on the smart contract. The division operation is performed at the researcher's end. This is due to the limitation mentioned in Section 3.2.1 that a division will always return an integer and hence lose precision in the result. To summarize, the smart contract sends the encrypted sum as well as the encrypted number of elements to the researcher. These two values are then decrypted by the researcher, and a final division is performed to obtain the result.

3.2.3. Calculation of Median

Calculating the median of sorted data is a simple task. Assuming the data consist of n values, the median is the middle element if n is odd and is the average of the two middle elements if n is even. The formula for the median of a sorted data set with n data points is given in Equation (2).

$$Med = \begin{cases} \left(\frac{n+1}{2}\right)^{th} \text{ element} & \text{if } n \text{ is odd} \\ \frac{\left(\frac{n}{2}\right)^{th} \text{ element} + \left(\frac{n}{2}+1\right)^{th} \text{ element}}{2} & \text{if } n \text{ is even} \end{cases} \quad (2)$$

In the system, the differentiation in the calculation of the median is simply handled by using an if statement. If n is odd, the median is calculated on the smart contract and returned to the researcher in encrypted form. If n is even, the sum of the middle two elements is calculated on the smart contract and sent to the researcher. The result is then decrypted by the researcher and divided by 2 to obtain the median. Thus, the researcher makes the decision of performing the division based on the number of elements.

The median calculation above assumes that the data are sorted. The biggest challenge in calculating the median of unsorted data, however, is the implementation of a sorting algorithm on the smart contract since it is not a trivial task to sort encrypted data. The

proposed system uses the bubble sort algorithm to sort the data. This algorithm was chosen because of its simplicity. It works by comparing two adjacent values in an array and swapping the numbers based on the result of that comparison, which makes it easy to implement. Furthermore, implementing bubble sort does not require any additional data structure. This makes it suitable to write in Solidity, considering the limitations of the programming language as well as the fhEVM library.

3.2.4. Calculation of Variance

The formula for calculating variance for a set of values $\{x_1, x_2, \dots, x_n\}$ is given in Equation (3), where $\bar{x}$ represents the mean of the data.

$$V = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1} \quad (3)$$

As was discussed in Section 3.2.2, it is not feasible to calculate the exact mean of any sequence on the smart contract because the division function in the fhEVM library always returns an encrypted integer. To eliminate the calculation of mean in the variance formula, we derived an equivalent formula that did not include mean or any division operator other than a major division operation at the last step. The step-by-step derivation of the formula starting with Equation (3) is shown below.

$$\begin{aligned} V &= \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1} \\ &= \frac{\sum_{i=1}^n (x_i - \frac{\sum_{j=1}^n x_j}{n})^2}{n - 1} \\ &= \frac{\sum_{i=1}^n (x_i^2 - 2x_i \frac{\sum_{j=1}^n x_j}{n} + (\frac{\sum_{j=1}^n x_j}{n})^2)}{n - 1} \\ &= \frac{n^2}{n^2} \frac{\sum_{i=1}^n (x_i^2 - 2x_i \frac{\sum_{j=1}^n x_j}{n} + \frac{(\sum_{j=1}^n x_j)^2}{n^2})}{n - 1} \\ &= \frac{\sum_{i=1}^n (n^2 x_i^2 - 2n^2 x_i \frac{\sum_{j=1}^n x_j}{n} + n^2 \frac{(\sum_{j=1}^n x_j)^2}{n^2})}{n^2(n - 1)} \\ &= \frac{n^2 \sum_{i=1}^n x_i^2 - 2n \sum_{i=1}^n (x_i \sum_{j=1}^n x_j) + \sum_{i=1}^n (\sum_{j=1}^n x_j)^2}{n^2(n - 1)} \\ &= \frac{n^2 \sum_{i=1}^n x_i^2 - 2n \sum_{i=1}^n (x_i \sum_{j=1}^n x_j) + n (\sum_{j=1}^n x_j^2)}{n^2(n - 1)} \end{aligned} \quad (4)$$

This approach removed the calculation of mean from the algorithm and, in turn, removed division as well. However, as shown in Equation (4), it was not possible to eliminate division completely. Keeping these limitations in mind, only the calculation of the numerator is implemented on the smart contract. The encrypted result of the numerator is sent to the researcher along with the encrypted value of n , where the final division is performed. Upon closer inspection of Equation (4), it can be observed that there are three major components in the numerator itself. Furthermore, the numerator involves subtraction as well. As discussed in Section 3.2.1, Zama only provides unsigned integers, which means any result below zero would undermine the accuracy of the result. To mitigate the issue, these three components are calculated separately, and their results are stored in separate variables. Then, the first component is added to the third, and the second component is then subtracted from the result of the addition. This ensures that the result does not fall below zero.

3.3. Security Assurances of the Proposed System

The proposed system provides the security requirements for all tenets of information security. It provides security assurances when data are in transit, at rest, and in use. Next, we describe how confidentiality, integrity, and availability are achieved in the proposed system.

- **Confidentiality:** Data are encrypted before they are shared and remain encrypted from then on. Data are encrypted in transit, at rest, and in use, providing statistical privacy as well as statistical confidentiality. The encrypted data are transmitted to the smart contract, where they are stored, allowing computations to be performed on them. The result of these computations is encrypted as well. Before the result is shared with the researcher, it is re-encrypted after it is internally decrypted on the blockchain using the distributed decryption protocol. This operation is not visible on the blocks of the chain and, hence, is not visible to the users. Cheating the system to obtain the decryption key would require one-third of the nodes on the blockchain to collaborate, which is an unlikely scenario for blockchains.
- **Integrity of data:** Each block in a blockchain contains the hash of the previous block, creating a chain. Tampering with one block changes the hash of that block and all the blocks that come after it. Since establishing the chain with the updated blocks is an almost impossible task, the data on the blockchain are tamper-resistant. This provides the integrity of the data.
- **Integrity of computations:** The algorithms for computations allowed on the data are implemented on the smart contract. Since smart contracts are part of the blockchain, altering them in an unauthorized way is not feasible. This provides integrity in computations and hence trust in the algorithms implemented for the computations.
- **Authorization:** Authorization in the proposed system can be considered from two perspectives. One is the control of what kind of analysis can be performed on the data. Because the functions for analysis are available on the smart contract in the proposed system and only authorized users/entities can change the functionality of the smart contract, a researcher is limited to using the functions available on the smart contract. Assuming only approved functions are implemented on the smart contract, a researcher cannot perform unapproved analysis on the data. The other authorization consideration in the system is about who can send data to the smart contract and who can request analysis of the data. Blockchain applications can be programmed to control access to the functions provided in the smart contract. Each user is uniquely identified by their public key, and this key can be used to determine their access to the functions on the smart contract.
- **Availability:** Availability is a feature that is provided by the blockchain itself because of its decentralized nature. Every node in the network has their own copy of the ledger, so even if one node becomes unavailable, the whole system would still be accessible. This removes a single point of failure and introduces fault tolerance in the system.

4. Results

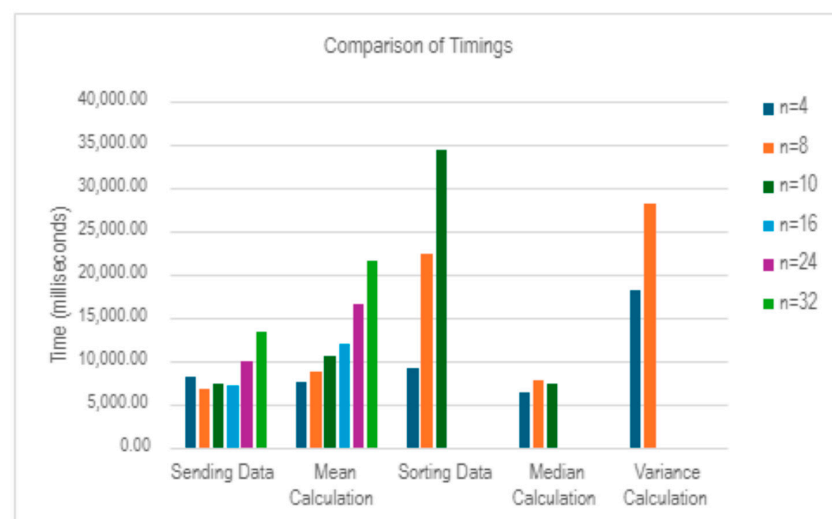
We ran the proof-of-concept implementation of the system, where the smart contract has the functionality to calculate the mean, median, and variance of the data set. We recorded the time required to perform various operations for data points of 4, 8, 10, 16, 24, and 32. These operations include setting the data (i.e., sending the data to the smart contract), mean calculation, sorting the data (which is necessary for finding the median), median calculation, and variance calculation. All experiments were carried out with a 16-bit data type. Each data set was randomly generated with points in the range of 0 to 50. Each produced data set was sent to the smart contract at once. Each data set size was tested eight times to ensure the reliability of the results. The tests were performed at different times and at different geographical locations with different connectivity. Table 1 shows the average timing of eight trials for each data set size.

Table 1. Timing of operations from request to result in milliseconds.

Number of Data Points (n)	Sending Data	Mean Calculation	Sorting Data	Median Calculation	Variance Calculation
4	8088.66679	7512.111693	9063.634214	6312.926094	18,109.45269
8	6642.155596	8792.629253	22,330.79229	7679.880587	28,161.12535
10	7243.960528	10,486.1769	34,412.11508	7358.729393	
16	7111.459872	11,948.44701			
24	9990.13181	16,596.62563			
32	13,358.13821	21,479.25259			

As exhibited in Table 1, the sizes of some data sets were not suitable for certain operations due to the gas limit imposed by Zama. For example, while the sorting process was performed on data sets containing 4, 8, and 10 data points, the variance process was applied only to data sets containing 4 and 8 data points. Sending the data sets to the blockchain and calculating the mean value could be performed on all data sets.

Figure 3 compares the timings for each operation based on the number of data points. It also compares the timings for different operations. Overall, as the number of data points increases, the time required for an operation also increases. A few exceptions to this are sending data for a smaller number of data points and the median calculation. We attribute this to the fact that operations that require a short time to complete are more susceptible to being impacted by the volume of transactions as well as the quality of the network connectivity. As depicted in the graph, variance and sorting take more time than setting data, mean calculation, and median calculation. Median calculation takes the least time since it involves a few simple operations compared to the other operations regardless of the data size.

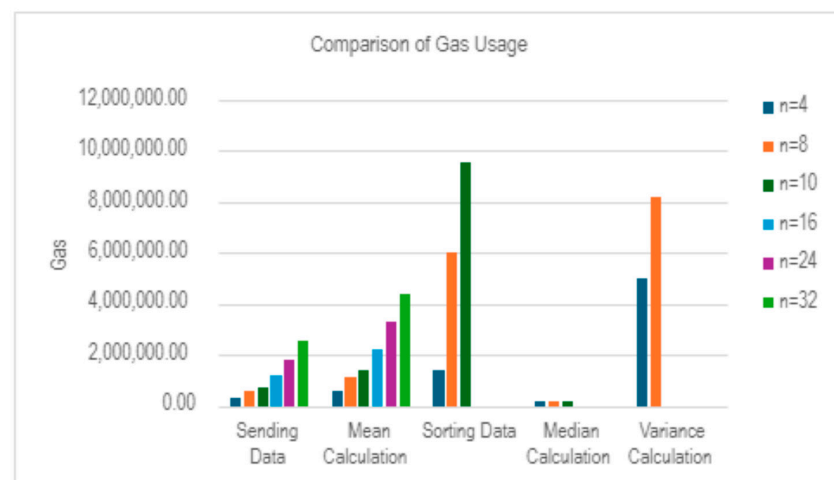
**Figure 3.** Comparison of timings of operations with respect to number of data points.

We also recorded the gas usage of each operation for each test and verified that the data sort and variance calculation reached the gas limit imposed by Zama at 10 data points and 8 data points, respectively. Table 2 shows the averages of the eight trials for each data set size.

Table 2. Gas usage of operations.

Number of Data Points (n)	Sending Data	Mean Calculation	Sorting Data	Median Calculation	Variance Calculation
4	290,491.5	577,618	1,367,854	169,517	5,022,035
8	565,458.875	1,124,706	6,007,343	169,517	8,213,723
10	709,171.5	1,398,250	9,565,005	169,517	
16	1,166,246.625	2,218,883			
24	1,835,907.625	3,313,060			
32	2,574,607.375	4,407,237			

It is no wonder that the gas usage was consistent for the given number of data points for the same operation. Figure 4 shows a comparison of the gas usage relative to the number of data points for each operation as well as a comparison of the gas usage with respect to the different operations.

**Figure 4.** Comparison of gas usage of operations with respect to number of data points.

As Figure 4 depicts, gas usage increased with the number of data points except for the median. The median calculation involved a simple operation of finding the index of the middle element, requiring only one addition and one division regardless of the number of data points. Hence, gas usage was the same for all three data set sizes. Calculating the median requires data to be sorted. However, because sorting data requires too much gas, it cannot be calculated for more than 10 data points. Similarly, variance calculation required around 10 million gas for eight data points. Adding an additional data point would require more than 10 million gas, which would be over the gas limit imposed by Zama. As is evident from the graph and the description of the implementation of the operations in Section 3, the operations that require more computations require more gas.

5. Discussion

This study proposed a system that provides data security, statistical privacy, and confidentiality using techniques and tools purely from mathematics and computer science. The system utilizes fully homomorphic encryption over a blockchain and ensures end-to-end security of the data from the moment they leave the data owner until the researcher receives the computation results. The data are secured in transit, at rest, as well as in use. Furthermore, the proposed system also provides computational integrity. With all the protections it provides, the system is an ultimate solution for meeting data security requirements.

A proof-of-concept implementation for the proposed system is carried out on the public blockchain Zama. In the implementation, a researcher can request secure calculations of mean, median, and variance. The timing results for carrying out the required operations on various numbers of data points are presented, along with the limitations of working with homomorphic encryption libraries and a public blockchain. While this study successfully demonstrates the possibility of securing data and computations on a blockchain using fully homomorphic encryption, there are certain challenges and areas that need improvement.

Fully homomorphic operations are quite expensive in terms of resources, and there are limitations on the number of calculations that can be performed on the blockchain before exceeding the gas and cost limits. This is especially critical when running the application on the main network, where transactions incur fees. This limitation can be partially mitigated on a private or permissioned blockchain network. Such a blockchain provides similar guarantees as the public blockchain and gives more control over who can use the application.

Future studies in this area can incorporate more complex algorithms across various disciplines, ranging from statistics to machine learning, and include a wider data range as more resources become available and technology advances. Furthermore, traditional algorithms that have been designed for plaintext data, such as searching and sorting, need to be carefully revised to support homomorphic computations, subject to the capabilities of the libraries available. Another possible direction for future studies could be to develop a framework that supports fully homomorphic computations over that smart contract and provides a wider range of functionality while overcoming the limitations of the current system, such as allowing division of two encrypted numbers and incorporating decimal or floating point numbers.

Author Contributions: Conceptualization, Y.K.P.; Data curation, R.R., Y.K.P. and Z.D.M.; Formal analysis, Y.K.P. and Z.D.M.; Investigation, R.R., Y.K.P. and Z.D.M.; Methodology, R.R. and Y.K.P.; Project administration, Y.K.P.; Resources, R.R., Y.K.P. and Z.D.M.; Software, R.R. and Z.D.M.; Supervision, Y.K.P.; Validation, R.R., Y.K.P. and Z.D.M.; Visualization, Y.K.P. and Z.D.M.; Writing—original draft, R.R. and Y.K.P.; Writing—review and editing, R.R., Y.K.P. and Z.D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article/ Further inquiries can be directed to the corresponding author(s).

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

References

1. Dwork, C.; Smith, A.; Steinke, T.; Ullman, J. Exposed! A Survey of Attacks on Private Data. *Annu. Rev. Stat. Appl.* **2017**, *4*, 61–84. [CrossRef]
2. Kitamura, K.; Irvan, M.; Yamaguchi, R.S. Anonymity test attacks and vulnerability indicators for the ‘Patient characteristics’ disclosure in medical articles. In Proceedings of the 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, 6–10 June 2022; pp. 186–193. [CrossRef]
3. Nissim, K. Privacy: From Database Reconstruction to Legal Theorems. In Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, in PODS’21, Virtual Event, China, 20–25 June 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 33–41. [CrossRef]
4. Wang, Y.-R.; Tsai, Y.-C. The Protection of Data Sharing for Privacy in Financial Vision. *Appl. Sci.* **2022**, *12*, 7408. [CrossRef]
5. Agarwal, S. Sray and Mishra Data and Model Privacy. In *Responsible AI: Implementing Ethical and Unbiased Algorithms*; Springer International Publishing: Cham, Switzerland, 2021; pp. 153–170. [CrossRef]
6. Sokolovska, A.; Kocarev, L. Integrating Technical and Legal Concepts of Privacy. *IEEE Access* **2018**, *6*, 26543–26557. [CrossRef]
7. Kifer, D.; Lin, B.-R. An Axiomatic View of Statistical Privacy and Utility. *J. Priv. Confidentiality* **2012**, *4*. [CrossRef]
8. Slavković, A.; Seeman, J. Statistical Data Privacy: A Song of Privacy and Utility. *Annu. Rev. Stat. Appl.* **2023**, *10*, 189–218. [CrossRef]
9. C.3. Statistical Confidentiality—MSITS 2010 Compilers Guide—UN Statistics Wiki. Available online: <https://unstats.un.org/wiki/display/M2CG/C.3.++Statistical+confidentiality> (accessed on 4 June 2024).

10. UNSD—Fundamental Principles of National Official Statistics. Available online: <https://unstats.un.org/fpos/> (accessed on 5 June 2024).
11. Statistical Confidentiality and Personal Data Protection—Eurostat. Available online: <https://ec.europa.eu/eurostat/web/microdata/statistical-confidentiality-and-personal-data-protection> (accessed on 18 June 2024).
12. National Academies of Sciences, Engineering, and Medicine; Division of Behavioral and Social Sciences and Education; Committee on National Statistics; Panel on Improving Federal Statistics for Policy and Social Science Research Using Multiple Data Sources and State-of-the-Art Estimation Methods; Groves, R.M.; Harris-Kojetin, B.A. (Eds.) *Innovations In Federal Statistics: Combining Data Sources While Protecting Privacy*; The National Academies Press: Washington, DC, USA, 2017. [CrossRef]
13. Kumar, R.; Kumar, J.; Khan, A.A.; Ali, H.; Bernard, C.M.; Khan, R.U.; Zeng, S. Blockchain and homomorphic encryption based privacy-preserving model aggregation for medical images. *Comput. Med. Imaging Graph.* **2022**, *102*, 102139. [CrossRef] [PubMed]
14. Liang, W.; Zhang, D.; Lei, X.; Tang, M.; Li, K.-C.; Zomaya, A. Circuit Copyright Blockchain: Blockchain-Based Homomorphic Encryption for IP Circuit Protection. *IEEE Trans. Emerg. Top. Comput.* **2020**, *9*, 1410–1420. [CrossRef]
15. Yaji, S.; Bangera, K.; Neelima, B. Privacy Preserving in Blockchain Based on Partial Homomorphic Encryption System for Ai Applications. In Proceedings of the 2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW), Bengaluru, India, 17–20 December 2018; pp. 81–85. [CrossRef]
16. Mutlu, Z.D.; Peker, Y.K.; Aydın, A. Selçuk Blockchain-based Privacy Preserving Linear Regression. *J. Millimeterwave Commun. Optim. Model.* **2023**, *3*, 45–49. Available online: <https://www.jomcom.org/index.php/1/article/view/81> (accessed on 5 June 2024).
17. Vanin, F.N.; Policarpo, L.M.; Righi, R.D.; Heck, S.M.; da Silva, V.F.; Goldim, J.; da Costa, C.A. A Blockchain-Based End-to-End Data Protection Model for Personal Health Records Sharing: A Fully Homomorphic Encryption Approach. *Sensors* **2023**, *23*, 14. [CrossRef] [PubMed]
18. Umar, B.; Olaniyi, O.; Olajide, D.; Dogo, E. Paillier Cryptosystem Based ChainNode for Secure Electronic Voting. *Front. Blockchain* **2022**, *5*, 927013. [CrossRef]
19. Shrestha, R.; Kim, S. Integration of IoT with blockchain and homomorphic encryption: Challenging issues and opportunities. *Adv. Comput.* **2019**, *115*, 293–331. [CrossRef]
20. Caldarola, F.; d’Atri, G.; Zanardo, E. Neural Fairness Blockchain Protocol Using an Elliptic Curves Lottery. *Mathematics* **2022**, *10*, 3040. [CrossRef]
21. fhevm/fhevm-whitepaper.pdf at main·zama-ai/fhevm·GitHub. Available online: <https://github.com/zama-ai/fhevm/blob/main/fhevm-whitepaper.pdf> (accessed on 4 June 2024).
22. Estimate Gas | 0.3 | fhEVM. Available online: <https://docs.zama.ai/fhevm/v/0.3-2/how-to/gas> (accessed on 4 June 2024).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.