This research report demonstrates introductions and motivations for the selected computer science topic as well as related challenges, shows reviews and critiques for current solutions, and illustrates the mechanism of the proposed solution.

# Research on utilizing blockchains to improve data security

Project 2 Final Written Report

The University of Adelaide
Computer Science (Advanced)
Jiajun Yu
a1806320

# Project 2 Final Written Report

## Topic Introduction

The topic of this research is about utilizing blockchains to improve data security.

At the beginning of 2020, the number of bytes in the 'digital universe' was 40 times bigger than the number of stars in the observable universe (Vuleta 2021). The data of a single person might not be valuable, but combining the data generated by millions of people is a completely different story (ODSC 2019). Data security is the practice of protecting digital information from unauthorized access, corruption, or theft throughout its entire lifecycle (IBM 2021). The security of data is crucial to the public nowadays; therefore, companies have the legal and moral obligation to enhance data security to prevent users' data from being misused for unethical purposes (Harrington 2021).

A blockchain is a distributed decentralized database that maintains a continuously growing list of ordered records (Synopsys 2021). In the blockchain network, nodes collectively adhere to a protocol to communicate and validate new blocks (Wikipedia 2021).

## Topic Motivation

In June 2021, data associated with 700 million LinkedIn users were stolen and posted for sale in a dark web forum, and this data exposure accident impacted around 92% of the total LinkedIn users (Tunggal 2021). Furthermore, in early October 2013, Adobe reported that hackers had stolen almost three million encrypted customer credit card records and login data for an undetermined number of user accounts. (Hill 2021). It can be seen from the news that data breaches frequently occurred in the past decade; therefore, enhancing data security is of vital importance.

As an emerging technology, blockchain becomes increasingly popular. The decentralization feature makes it an excellent tool to ensure data security and integrity.

## Challenge Introduction

In this report, a solution will be proposed to address two challenges that exist in the data security field. The first challenge is brute force attack and another one is quantum computing. The brute force attack is a technique that uses trial and error to guess information. Attackers systematically check all potential possibilities until correctly guessing targeted information, such as the user's password (Kaspersky 2021). Quantum computing is a type of special computation technology that harnesses the collective properties of quantum states, such as superposition, interference, and entanglement, to perform extremely fast calculations in quantum computers (Paul et al. 2021).

## Challenge Motivation

During the COVID-19 pandemic, a vast majority of people worked at home. It was revealed that there was a rise in brute force attacks attempting to exploit the Windows remote desktop protocol (RDP). Brute force attack raises a great concern of people because there are over 80% of breaches involve brute force attacks or the use of lost or stolen credentials (Mission Critical 2020).

In 2019, Google claimed that they have made a quantum supremacy breakthrough because its designed quantum computer was able to perform a very complex calculation in 200 seconds while it would have taken 10,000 years or more on a traditional supercomputer. Despite claiming to have hit the quantum supremacy milestone, Google said they still have a long way to go before such computers become useful (Porter 2021).

Brute force attacks and the advent of quantum computers threaten data security to a large extent. Therefore, it is important to design a new data secure storage model (a mechanism) by utilizing blockchains.

## Literature Review (Presentation of current challenge solutions)

A wide range of literature has been reviewed for this research. Two proposed models for data secure storage and one paper for data decentralized storage will be discussed in detail.

### 1. Review of the paper, Decentralizing Privacy: Using Blockchain to Protect Personal Data

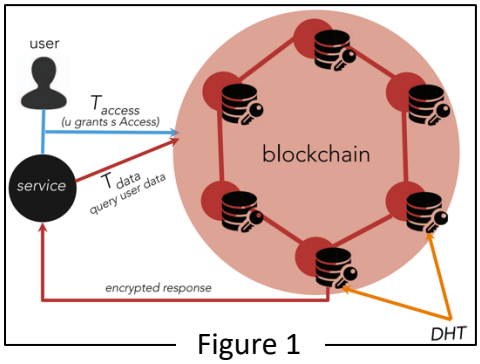

Figure 1

In the first paper, researchers designed a blockchain-based access framework (Figure 1) combined with off-blockchain storage techniques (distributed hash table, DHT) to implement data secure storage. The mechanism of this model is to split the identities of 'user' and 'service' first by their respective digital identities, while users have higher priority to access data and manage access permissions. The blockchain accepts two types of transactions: $T_{access}$, used for access control management; and $T_{data}$, for data storage and retrieval. When storing data by this model, data will be encrypted using a shared encryption key and then sent to the blockchain in a $T_{data}$ transaction, which subsequently returns a pointer (SHA-256 hash of the data) of the encrypted data and stores this pointer on the permissioned blockchain. Meanwhile, one off-blockchain storage platform will be utilized for storing the encrypted data (Zyskind et al. 2015). In this model, data security is ensured by the blockchain for verifying and accepting different identities and requests.

### 2. Review of the paper, A Secure Data Sharing Platform Using Blockchain and Interplanetary File System

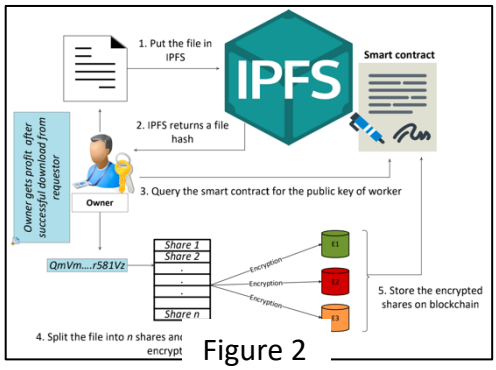

Figure 2

In the second paper, researchers designed a data secure storage model (Figure 2) combined with blockchain and IPFS (InterPlanetary File System) to implement data secure storage. The mechanism of this model is to encrypt the file hash (pointer) returned from IPFS and store that encrypted file hash on the blockchain. IPFS is a protocol and peer-to-peer network for storing and sharing data in a distributed file system and the storage method of IPFS is fragmented distributed storage (Desai et al. 2020). File hash value returned by IPFS is critical for retrieving data back from IPFS; therefore, in this paper, its SSS algorithm is used to split the IPFS file hash, and the data owner can decide the number of random keys for the file hash segment encryption. Once designated file hash segments are encrypted, they will be stored in the blockchain along with other important information. In this model, data security is ensured by the encrypted file hash value (Naz et al. 2019).

### 3. Review of the paper, A Blockchain-based Decentralized Data Storage and Access Framework for PingER

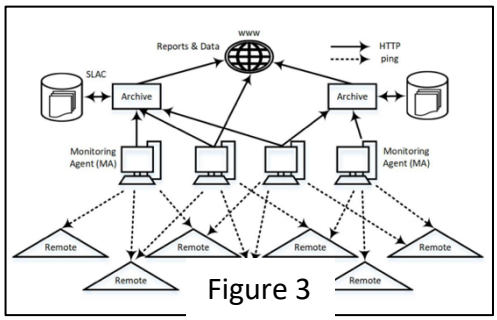

Figure 3

In the third paper, researchers proposed a model (Figure 3) for storing files of a worldwide end-to-end Internet performance measurement project to remove its total dependence on a centralized repository (Ali et al. 2018). This paper does not propose a data secure storage model but specifies some details of the implementation for a decentralized data storage model. Merkle tree, permissioned blockchain, IPFS, and other

techniques are integrated into its model, and it provides a paradigm for designing a data secure storage model.

## Critique of Literature (Critique of current challenge solutions: Paper 1 & Paper 2)

There exist some issues in the first model (Figure 1). Firstly, the security of the blockchain which stores a lot of file hash values cannot be assured. Researchers did not specify attributes of this blockchain, instead, the paper only briefly mentioned that the model would retain pointers to data on the public ledger (blockchain). If the blockchain is public, anyone can access the contents of the blockchain and get those file hash values to retrieve data back from the off-blockchain storage platform. Secondly, this model did not consider the file hash storage framework inside the blockchain. After the user's identity is verified, it is difficult to know how the hash value of the targeted file is searched and fetched, because the model does not label the owner of those file hash values. Lastly, it is difficult to change the compound identity for accessing data. A compound identity is a shared identity for two or more parties, where some parties (owners) own the identity, and the rest (guests) have restricted access to it (Zyskind et al. 2015). The compound identity is generated by a specific algorithm in the early stage; therefore, it might be hard to add or delete certain identities in the future.

The second model also has some drawbacks. When it comes to encrypting the file hash value, its SSS algorithm splits the file hash into n shares and randomly chooses keys for encryption. In terms of slicing the hash value of a file, it is unsafe because the model executes linearly slicing which means that data is cut into sequential segments before being selected for encryption. By executing a high-speed brute force attack on obtained public keys, getting the file hash is theoretically attainable by fetching cracked private keys and using different private keys to decrypt different encrypted file hash segments. In terms of choosing keys for encryption, it might be hard for the user to manage a lot of private keys used for decryption after storing the encrypted file hash value on the blockchain.

## Solution to Challenge

A new data security model was designed and proposed (Figure 4) to enhance data security. The mechanism of this model is to store segments of information on multiple blockchains and store large data through IPFS to implement a more distributed storage. By increasing the difficulty of obtaining data slices, we can ensure that data will be more resistant to brute force attacks and quantum computing attacks.

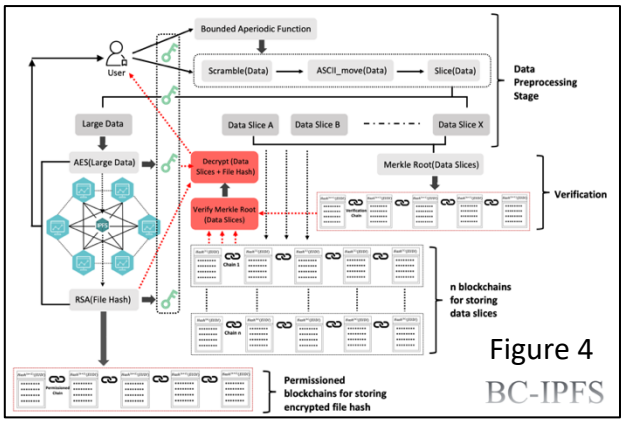

Figure 4

Firstly, raw data from the user will be pre-processed to be in the form of disordered data segments (Figure 5, 6, 7, 8 in appendix). The pre-processing stage consists of three steps, scrambling, doing ASCII arithmetic, and slicing data. The contents of pre-processed data segments are highly dependent on a randomly generated bounded aperiodic function $\{f(x)|-10 \leq f(x) \leq 10\}$.

To be more specific, in the data scrambling stage, characters will be put on the x-axis and their corresponding move values will be calculated based on the absolute round values of $f(x)$. First part characters with index 'i' will be swapped with the character with index 'i + move value', and second part characters with index 'i' will be swapped with the character with index 'i - move value' (Figure 9, 10, 11 in appendix). In the changing character ASCII stage, characters will be placed on the x-axis again and their corresponding change values will be calculated based on the round values of $f(x)$. The changed character ASCII will be its original ASCII plus its corresponding change value (Figure 12 in appendix). In the data slicing stage, the data will be divided into segments of 10 characters. For each segment, the absolute round values of $f(x)$ will determine

which character in the segment will be stored in which data slice (Figure 13 in appendix). In addition, the rest characters will be stored by IPFS (characters selected will be replaced by spaces).

Before data slices are stored in different blockchains, a Merkle tree will be generated (Figure 14). In the proposed model, data slices will be stored in random blockchains, so when it comes to decrypting data, the correct order of data slices must be verified. The root hash ($H_{ABCDEFGH}$) will be obtained through the hash calculation of every two sequential adjacent data slices. Meanwhile, the root hash will be stored in the 'Verification Chain'. By permutating obtained data slices in different orders and comparing the calculated root hash with the root hash stored on the blockchain, the validity of data slices, as well as their correct order, can be verified.

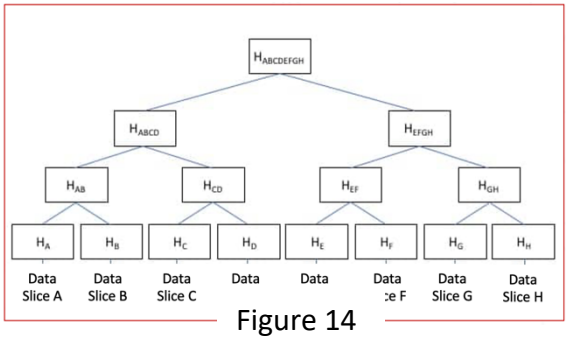

Figure 14

In the storage stage, Large data will be stored by IPFS combined with two effective encryption algorithms. Before data is stored by IPFS, a symmetric encryption algorithm, AES, will encrypt the data in a short time, and the decryption key will be returned to the user. After IPFS stores the data, an asymmetric encryption algorithm, RSA, will be utilized to encrypt the hash value of the file (which will be used to retrieve the file from IPFS in the future), and the encrypted file hash will be stored in a permissioned blockchain which can be managed by leading companies with an excellent reputation. Meanwhile, data slices will be stored in random public blockchains with the recursive hash value of the user's digital identity labeling the owner of them ($Hash^{(n)}(UDI) = Hash\left[Hash^{(n-1)}(UDI) + UDI\right]$, UDI = User's Digital Identity, n = labeled blockchain number) to avoid attackers effortlessly obtaining data slices of the same user in different blockchains.

## Discussions

Discussions for a lot of contents have been covered in previous sections. The designed model above can be discussed in more detail. One advantage of this model is utilizing multiple blockchains not only to store the file hash which returned from the off-blockchain platform (IPFS) as well as the information for the verification purpose but also to store fragments of information in different blockchains to avoid attackers retrieving everything easily from IPFS by attacking and obtaining the file hash. Secondly, the recursive hash value of the user's digital identity is utilized to label the owner of data slices on different blockchains to avoid attackers effortlessly fetching data slices of the same user at the same time. Thirdly, adopting a bounded aperiodic function to scramble and do character ASCII arithmetic increases the difficulty to restore the original data. Lastly, data owners only need to keep three keys secure (AES decryption key, RSA private key, bounded aperiodic function $f(x)$) to ensure the data security and integrity to a large extent.

## Conclusions

In conclusion, the data security model plays a significant role in dealing with data breaches. Brute force attack and quantum computing are two main challenges in this field. The proposed solution (model) which integrates blockchains is designed to enhance data security. The main idea of this designed model is to slice data and store fragments of information at different places combined with effective encryption algorithms and recursive hash values labelling. By increasing the difficulty of fetching data slices, we can make important data largely resistant to brute force attacks and quantum computing attacks. Further research can be devoted to the design of quantum-resistant algorithms, the research of combinations of existing encryption algorithms, and the improvement of data transmission security.

## References


Ali et al. 2018, ‘A blockchain-based decentralized data storage and access framework for PingER’, *IEEE*, doi:10.1109/TrustCom/BigDataSE.2018.00179, viewed 25 September 2021, < https://ieeexplore.ieee.org/document/8456048>.

Desai et al. 2020, ‘Blockchain based secure data storage and access control system using IPFS’, *Journal of critical reviews*, vol. 7, no. 19, viewed 21 September 2021, < http://www.jcreview.com/fulltext/197-1594989773.pdf>.

Harrington, D 2021, *Data security: importance, types, and solutions*, Varonis, viewed 27 October 2021, < https://www.varonis.com/blog/data-security/>.

Hill, M & Swinhoe, D 2021, *The 15 biggest data breaches of the 21st century,* CSO, viewed 28 September 2021, <https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21-st-century.html>.

IBM 2021, *What is data security*, IBM, viewed 28 October 2021, < https://www.ibm.com/topics/data-security>.

Kaspersky 2021, *Brute force attack: definition and examples*, Kaspersky, viewed 22 October 2021, < https://www.kaspersky.com/resource-center/definitions/brute-force-attack>.

Mission Critical 2020, *Brute-force attacks still a major threat*, Mission Critical, viewed 24 October 2021, < https://www.missioncriticalmagazine.com/articles/brute-force-attacks-still-a-major-threat>.

Naz et al. 2019, ‘A secure data sharing platform using blockchain and Interplanetary File System’, *MDPI*, doi:10.3390, viewed 19 September 2021, < https://www.mdpi.com/2071-1050/11/24/7054>.

ODSC 2019, *Data valuation — what is your data worth and how do you value it*, Open Data Science, viewed 24 October 2021, <https://medium.com/@ODSC/data-valuation-what-is-your-data-worth-and-how-do-you-value-it-b0a15c64e516>.

Paul et.al 2021, *Quantum computing*, Wikipedia, viewed 24 October 2021, < https://en.wikipedia.org/wiki/Quantum_computing>.

Porter, J 2021, *Google wants to build a useful quantum computer by 2029*, The Verge, viewed 24 October 2021, < https://www.theverge.com/2021/5/19/22443453/google-quantum-computer-2029-decade-commercial-useful-qubits-quantum-transistor>.

Synopsys 2021, What is blockchain, SYNOPSYS, viewed 27 October 2021, < https://www.synopsys.com/glossary/what-is-blockchain.html>.

Tunggal, A 2021, *The 59 biggest data breaches (updated for October 2021)*, UpGuard, viewed 23 October 2021, < https://www.upguard.com/blog/biggest-data-breaches>.

Vuleta, B 2021, *How much data is created every day*, blog, viewed 24 September 2021, <

https://seedscientific.com/how-much-data-is-created-every-day/ >.

Wikipedia 2021, Blockchain, Encyclopaedia, Wikipedia, viewed 24 September 2021, < https://en.wikipedia.org/wiki/Blockchain >.

Zyskind et al. 2015, 'Decentralizing privacy: using blockchain to protect personal data', *IEEE*, doi:10.1109/SPW.2015.27, viewed 20 September 2021, < https://ieeexplore.ieee.org/document/7163223>.

## Appendix

```
value=10*math.sin(15*math.cos(10*i)-3*math.log(abs(math.cos((190)*i)*150)))
```

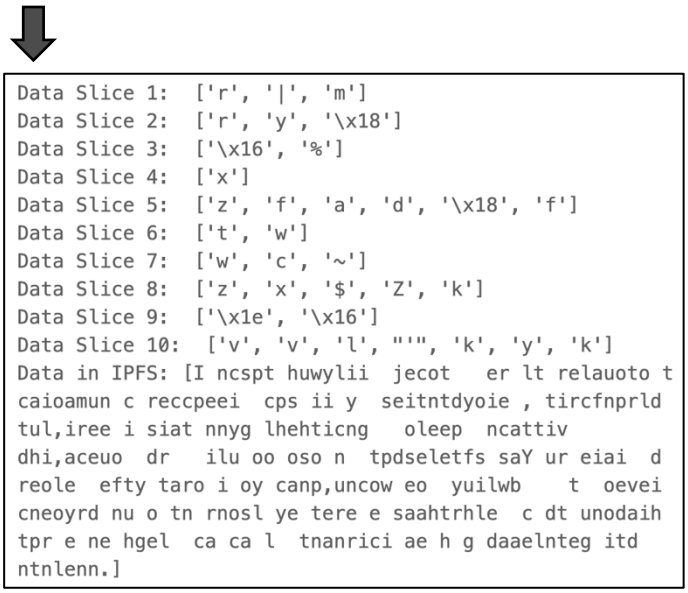


Figure 5 (Data pre-processing example 1)

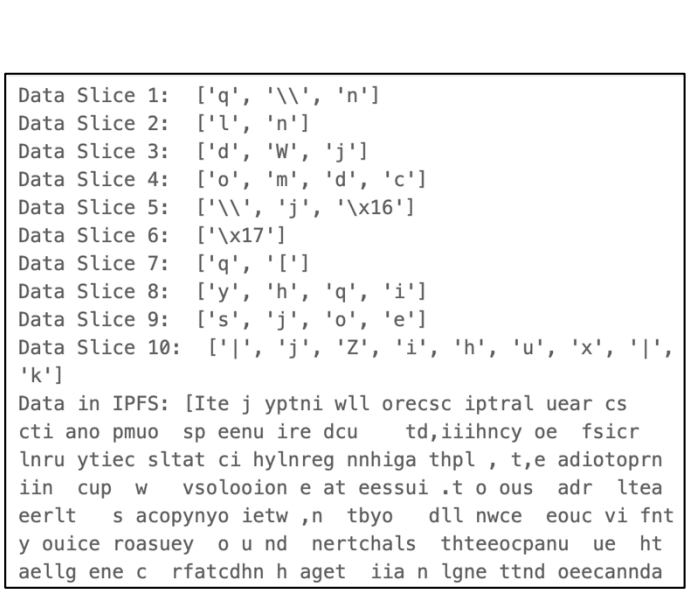


Figure 6 (Data pre-processing example 2)

```
value=10*math.sin(15*math.cos(100*i)-3*math.log(abs(math.cos((190)*i)*150)))
```

```
value=10*math.sin(16*math.cos(100*i)-3*math.log(abs(math.cos((190)*i)*150)))
```

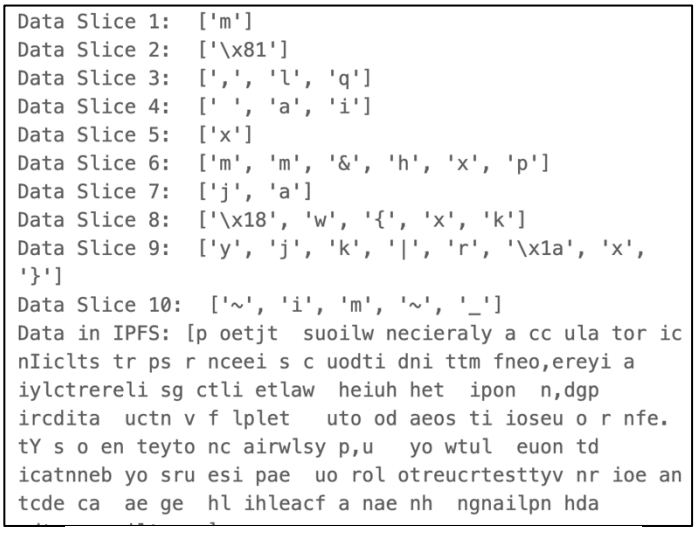


Figure 7 (Data pre-processing example 3)

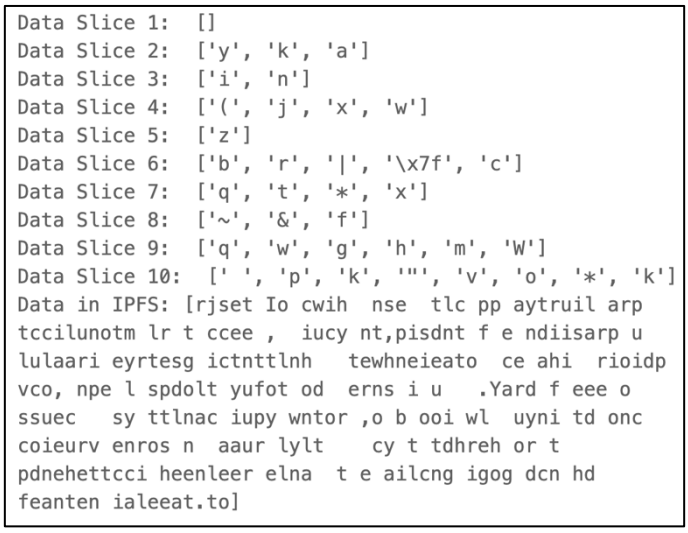


Figure 8 (Data pre-processing example 4)

```
value=10*math.sin(17*math.cos(100*i)-3*math.log(abs(math.cos((190)*i)*150)))
```

| Function | $f(x) = 10\sin(16\cos 100x - 3ln\lvert\cos(190x) * 150\rvert)$ | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data | 'H' | 'i' | ',' | space | 'e' | 'v' | 'e' | 'r' | 'y' | 'o' | 'n' | 'e' | ',' | space | 'l' | space | 'a' | 'm' | space | 'J' | 'a' | 's' | 'o' | 'n' |
| $f(x)$ | $f(0)$ | $f(1)$ | $f(2)$ | $f(3)$ | $f(4)$ | $f(5)$ | $f(6)$ | $f(7)$ | $f(8)$ | $f(9)$ | $f(10)$ | $f(11)$ | $f(12)$ | $f(13)$ | $f(14)$ | $f(15)$ | $f(16)$ | $f(17)$ | $f(18)$ | $f(19)$ | $f(20)$ | $f(21)$ | $f(22)$ | $f(23)$ |
| Value | 8.2 | 5.8 | -7.9 | 8.9 | 9.7 | -6.2 | 6.0 | -7.9 | -2.3 | 3.2 | 8.2 | 5.9 | 9.2 | -9.2 | -6.3 | 5.3 | 4.4 | 9.3 | -3.8 | -7.0 | 8.1 | 0.5 | 9.9 | -0.2 |
| Move Value | 8 | 6 | 8 | 9 | 10 | 6 | 6 | 8 | 2 | 3 | 8 | 6 | 9 | 9 | 6 | 5 | 4 | 9 | 4 | 7 | 8 | 0 | 10 | 0 |
| Swap Index | 0+8 | 1+6 | 2+8 | 3+9 | 4+10 | 5+6 | 6+6 | 7+8 | 8+2 | 9+3 | 10+8 | 11+6 | 12-9 | 13-9 | 14-6 | 15-5 | 16-4 | 17-9 | 18-4 | 19-7 | 20-8 | 21-0 | 22-10 | 23-0 |
| Scrambled Data | 'y' | 'r' | 'n' | 'o' | spac e | 'e' | spac e | spac e | 'v' | 'e' | 'i' | 'm' | 'o' | 'l' | 'H' | spac e | ',' | 'e' | ',' | 'a' | 'J' | 's' | 'a' | 'n' |

Figure 9 (Data Scrambling)

| Function | $f(x) = 10\sin(16\cos 100x - 3ln\lvert\cos(190x) * 150\rvert)$ | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data | 'H' | 'i' | ',' | space | 'e' | 'v' | 'e' | 'r' | 'y' | 'o' | 'n' | 'e' | ',' | space | 'l' | space | 'a' | 'm' | space | 'J' | 'a' | 's' | 'o' | 'n' |
| $f(x)$ | $f(0)$ | $f(1)$ | $f(2)$ | $f(3)$ | $f(4)$ | $f(5)$ | $f(6)$ | $f(7)$ | $f(8)$ | $f(9)$ | $f(10)$ | $f(11)$ | $f(12)$ | $f(13)$ | $f(14)$ | $f(15)$ | $f(16)$ | $f(17)$ | $f(18)$ | $f(19)$ | $f(20)$ | $f(21)$ | $f(22)$ | $f(23)$ |
| Value | 8.2 | 5.8 | -7.9 | 8.9 | 9.7 | -6.2 | 6.0 | -7.9 | -2.3 | 3.2 | 8.2 | 5.9 | 9.2 | -9.2 | -6.3 | 5.3 | 4.4 | 9.3 | -3.8 | -7.0 | 8.1 | 0.5 | 9.9 | -0.2 |
| Move Value | 8 | 6 | 8 | 9 | 10 | 6 | 6 | 8 | 2 | 3 | 8 | 6 | 9 | 9 | 6 | 5 | 4 | 9 | 4 | 7 | 8 | 0 | 10 | 0 |
| Swap Index | 0+8 | 1+6 | 2+8 | 3+9 | 4+10 | 5+6 | 6+6 | 7+8 | 8+2 | 9+3 | 10+8 | 11+6 | 12-9 | 13-9 | 14-6 | 15-5 | 16-4 | 17-9 | 18-4 | 19-7 | 20-8 | 21-0 | 22-10 | 23-0 |
| Scrambled Data | 'y' | 'r' | 'n' | 'o' | spac e | 'e' | spac e | spac e | 'v' | 'e' | 'i' | 'm' | 'o' | 'l' | 'H' | spac e | ',' | 'e' | ',' | 'a' | 'J' | 's' | 'a' | 'n' |

Figure 10 (Data scrambling example 1)

| Function | $f(x) = 10\sin(16\cos 100x - 3ln\lvert\cos(190x) * 150\rvert)$ | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data | 'H' | 'i' | ',' | space | 'e' | 'v' | 'e' | 'r' | 'y' | 'o' | 'n' | 'e' | ',' | space | 'l' | space | 'a' | 'm' | space | 'J' | 'a' | 's' | 'o' | 'n' |
| $f(x)$ | $f(0)$ | $f(1)$ | $f(2)$ | $f(3)$ | $f(4)$ | $f(5)$ | $f(6)$ | $f(7)$ | $f(8)$ | $f(9)$ | $f(10)$ | $f(11)$ | $f(12)$ | $f(13)$ | $f(14)$ | $f(15)$ | $f(16)$ | $f(17)$ | $f(18)$ | $f(19)$ | $f(20)$ | $f(21)$ | $f(22)$ | $f(23)$ |
| Value | 8.2 | 5.8 | -7.9 | 8.9 | 9.7 | -6.2 | 6.0 | -7.9 | -2.3 | 3.2 | 8.2 | 5.9 | 9.2 | -9.2 | -6.3 | 5.3 | 4.4 | 9.3 | -3.8 | -7.0 | 8.1 | 0.5 | 9.9 | -0.2 |
| Move Value | 8 | 6 | 8 | 9 | 10 | 6 | 6 | 8 | 2 | 3 | 8 | 6 | 9 | 9 | 6 | 5 | 4 | 9 | 4 | 7 | 8 | 0 | 10 | 0 |
| Swap Index | 0+8 | 1+6 | 2+8 | 3+9 | 4+10 | 5+6 | 6+6 | 7+8 | 8+2 | 9+3 | 10+8 | 11+6 | 12-9 | 13-9 | 14-6 | 15-5 | 16-4 | 17-9 | 18-4 | 19-7 | 20-8 | 21-0 | 22-10 | 23-0 |
| Scrambled Data | 'y' | 'r' | 'n' | 'o' | spac e | 'e' | spac e | spac e | 'v' | 'e' | 'i' | 'm' | 'o' | 'l' | 'H' | spac e | ',' | 'e' | ',' | 'a' | 'J' | 's' | 'a' | 'n' |

Figure 11 (Data scrambling example 2)

| Function | $f(x) = 10\sin(16\cos 100x - 3ln\lvert\cos(190x) * 150\rvert)$ | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Scrambled Data | 'y' | 'r' | 'n' | 'o' | space | 'e' | space | space | 'v' | 'e' | 'i' | 'm' | 'o' | 'l' | 'H' | space | ',' | 'e' | ',' | 'a' | 'J' | 's' | 'a' | 'n' |
| $f(x)$ | $f(0)$ | $f(1)$ | $f(2)$ | $f(3)$ | $f(4)$ | $f(5)$ | $f(6)$ | $f(7)$ | $f(8)$ | $f(9)$ | $f(10)$ | $f(11)$ | $f(12)$ | $f(13)$ | $f(14)$ | $f(15)$ | $f(16)$ | $f(17)$ | $f(18)$ | $f(19)$ | $f(20)$ | $f(21)$ | $f(22)$ | $f(23)$ |
| Value | 8.2 | 5.8 | -7.9 | 8.9 | 9.7 | -6.2 | 6.0 | -7.9 | -2.3 | 3.2 | 8.2 | 5.9 | 9.2 | -9.2 | -6.3 | 5.3 | 4.4 | 9.3 | -3.8 | -7.0 | 8.1 | 0.5 | 9.9 | -0.2 |
| Change Value | 8 | 6 | -8 | 9 | 10 | -6 | 6 | -8 | -2 | 3 | 8 | 6 | 9 | -9 | -6 | 5 | 4 | 9 | -4 | -7 | 8 | 0 | 10 | 0 |
| Original ASCII | 121 | 114 | 110 | 111 | 32 | 101 | 32 | 32 | 118 | 101 | 105 | 109 | 111 | 73 | 72 | 32 | 44 | 101 | 44 | 97 | 74 | 115 | 97 | 110 |
| Changed ASCII | 129 | 120 | 102 | 120 | 42 | 95 | 38 | 24 | 116 | 104 | 113 | 115 | 120 | 64 | 66 | 37 | 48 | 110 | 40 | 90 | 82 | 115 | 107 | 110 |
| ASCII Changed Data | '-' | 'x' | 'f' | 'x' | '*' | '_' | '&' | '\x18' | 't' | 'h' | 'q' | 's' | 'x' | '@' | 'B' | '%' | '0' | 'n' | '(' | 'Z' | 'R' | 's' | 'k' | 'n' |

Figure 12 (Data ASCII changing)

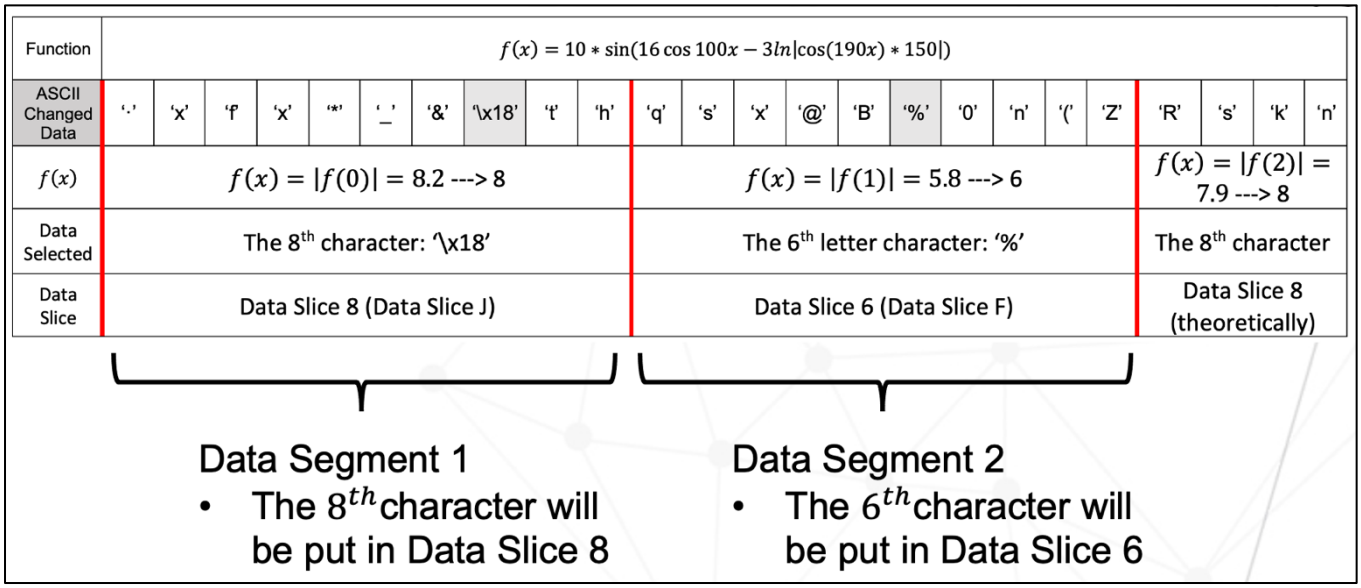


Figure 13 (Data slicing)

## Code Implementation (Pre-processing stage)

```
import random
import math
import matplotlib.pyplot as plt
✓ 0.8s

# Give the row data
text_list=[]
# text=input("Please enter data: ")
text="Hi, everyone, I am Jason"
# text="In this project you will select a particular topic i
text.split()
for i in text:
    text_list.append(i)

print("The data that needs to be preprocessed is: \n")
for i in range(len(text_list)):
    print(text_list[i],end="")

text_len=len(text)
print("\n\nThe length of the data is: ",text_len)
✓ 0.4s
The data that needs to be preprocessed is:

Hi, everyone, I am Jason

The length of the data is:  24
```

Figure 15 (Example input)

```
# Generate scramble_value, move_value, select_value
x_value=[]
y_value=[]
scramble_value=[]
change_value=[]
select_value=[]

for i in range(24):
    value=10*math.sin(16*math.cos(100*i)-3*math.log(abs(mat
    x_value.append(i)
    y_value.append(value)
    scramble_value.append(abs(round(value)))
    change_value.append(round(value))
    select_value.append(abs(round(value)))

# print(x_value);
# print(y_value)
# print(scramble_value)
print(change_value)

# Plot
plt.figure(figsize=(30,10))
plt.scatter(x_value,change_value)
plt.style.use("dark_background")
✓ 0.2s
[8, 6, -8, 9, 10, -6, 6, -8, -2, 3, 8, 6, 9, -9, -6, 5, 4, 9, -
```

Figure 16 (Generating values for pre-processing stage)

```
# Scramble data
if text_len%2==0:
    for i in range(int(text_len/2)):
        temp=text_list[i]
        text_list[i]=text_list[i+scramble_value[i]];
        text_list[i+scramble_value[i]]=temp
    for i in range(int(text_len/2),text_len):
        temp=text_list[i]
        text_list[i]=text_list[i-scramble_value[i]];
        text_list[i-scramble_value[i]]=temp
else:
    for i in range(int(text_len/2)):
        temp=text_list[i]
        text_list[i]=text_list[i+scramble_value[i]];
        text_list[i+scramble_value[i]]=temp
    for i in range(int(text_len/2)+1,text_len):
        temp=text_list[i]
        text_list[i]=text_list[i-scramble_value[i]];
        text_list[i-scramble_value[i]]=temp

print("The data that scrambled is: \n")
for i in range(len(text_list)):
    print(text_list[i],end="")
```

✓ 0.1s

```
The data that scrambled is:

yrno e  veimoIH ,e,aJsan
```

Figure 17 (Scrambling data)

```
# Move data (by Ascii value)
moved_data=[]
for i in range(text_len):
        moved_data.append(chr(ord(text_list[i])+change_value[i]))

print("The ASCII changed data is: \n")
for i in range(text_len):
    print(moved_data[i],end="")
```

✓ 0.1s

```
The ASCII changed data is:

·xfx*_& thqsx@B%0n(ZRskn
```

Figure 18 (Changing character ASCII)

```
# Slice data
data_slice1=[]
data_slice2=[]
data_slice3=[]
data_slice4=[]
data_slice5=[]
data_slice6=[]
data_slice7=[]
data_slice8=[]
data_slice9=[]
data_slice10=[]
data_IPFS=[]

group_number=text_len//10;
for i in range(group_number):
    index=select_value[i]
    if index==1:
        data_slice1.append(moved_data[10*(i)+index-1])
    elif index==2:
        data_slice2.append(moved_data[10*(i)+index-1])
    elif index==3:
        data_slice3.append(moved_data[10*(i)+index-1])
    elif index==4:
        data_slice4.append(moved_data[10*(i)+index-1])
    elif index==5:
        data_slice5.append(moved_data[10*(i)+index-1])
    elif index==6:
        data_slice6.append(moved_data[10*(i)+index-1])
    elif index==7:
        data_slice7.append(moved_data[10*(i)+index-1])
    elif index==8:
        data_slice8.append(moved_data[10*(i)+index-1])
    elif index==9:
        data_slice9.append(moved_data[10*(i)+index-1])
    elif index==10:
        data_slice10.append(moved_data[10*(i)+index-1])
```

Figure 19 (Slicing data 1)

```
hide_value=[]
for j in range(group_number):
    index=select_value[j]
    value=j*10+index-1
    hide_value.append(value)
for i in range(text_len):
        if(i not in hide_value):
            data_IPFS.append(moved_data[i])
        else:
            data_IPFS.append(" ")

print("Data Slice 1: ",data_slice1)
print("Data Slice 2: ",data_slice2)
print("Data Slice 3: ",data_slice3)
print("Data Slice 4: ",data_slice4)
print("Data Slice 5: ",data_slice5)
print("Data Slice 6: ",data_slice6)
print("Data Slice 7: ",data_slice7)
print("Data Slice 8: ",data_slice8)
print("Data Slice 9: ",data_slice9)
print("Data Slice 10: ",data_slice10)
print("Data in IPFS: [",end="")
for i in range(len(data_IPFS)):
    print(data_IPFS[i],end="")
print("]")
```

✓ 0.2s

Figure 20 (Slicing data 2)

```
Data Slice 1:  []
Data Slice 2:  []
Data Slice 3:  []
Data Slice 4:  []
Data Slice 5:  []
Data Slice 6:  ['%']
Data Slice 7:  []
Data Slice 8:  ['\x18']
Data Slice 9:  []
Data Slice 10:  []
Data in IPFS: [·xfx*_& thqsx@B 0n(ZRskn]
```

Figure 21 (Example output)