Decentralized Password Manager

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

Submitted by

Praksh Panab Rathour (201196)

Abhinav Thakur (201354)

Under the guidance & supervision of

Dr. Pankaj Dhiman

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **"Decentralized Password Manager"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by **Praksh Pranab Rathour (201196)** and **Abhinav Thakur(201354)** during the period from August 2023 to May 2024 under the supervision of **Dr. Pankaj Dhiman ,**(Assistant Professor(SG), Department of Computer Science and Engineering, Jaypee University of Information Technology).

Praksh Pranab Rathour

(201196)

Abhinav Thakur

(201354)

The above statement made is correct to the best of my knowledge.

Supervisor Name: Dr. Pankaj Dhiman

Designation: Assistant Professor (SG)

Department: CSE/IT

Dated:

DECLARATION

We hereby declare that the work presented in this report entitled **‘Decentralized Password Manager’**in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr Pankaj Dhiman**(Assistant Professor(SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Student Name: Praksh Pranab Rathour	Student Name: Abhinav Thakur
Roll No.: 201196	Roll No.: 201354

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Name: Dr. Pankaj Dhiman
Designation: Assistant Professor (SG)
Department: Computer Science & Engineering and Information Technology
Dated: May 15, 2024

ACKNOWLEDGMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully. I really grateful and wish my profound my indebtedness to Supervisor Dr Pankaj Dhiman, Assistant Professor(SG) Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of my supervisor in the field of "Research Area" to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

I would like to express my heartiest gratitude to Dr Pankaj Dhiman Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Praksh Pranab Rathour (201196)

Abhinav Thakur (201354)

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS, SYMBOLS, OR NOMENCLATURE

Abbreviation	Name
DPM	Decentralized Password Manager
P-2-P	Photovoltaic Effect
E2E	End to End
AES	Advanced Encryption Standard
PKI	Public Key Infrastructure
OTP	One-Time Password
MFA	Multi-Factor Authentication
UI	User Interface
API	Application Programming Interface
SDK	Software Development Kit

ABSTRACT

The design and execution of a decentralized password manager that provides improved security and privacy through the use of secured data division and sharing are described in this project report. The project's goal is to reduce the hazards associated with centralized password managers by giving users an effective and safe way to manage their passwords without depending on a central authority.

In order to guarantee that user data is stored safely and redundantly throughout the network, the system uses a distributed design that makes use of the peer-to-peer network idea. The passwords are secured using encryption techniques, which makes it harder for bad actors to steal them.Then it is used for user authentication and to guarantee that only authorized users have access to their passwords using public-private key encryption techniques. The study discusses the several security procedures used to guarantee user data availability, confidentiality, and integrity. It emphasizes how to safeguard passwords and stop unauthorized access to user data by using cryptographic techniques. Evaluations of the system's usability and scalability have also been conducted, and the findings indicate that it is a competitive substitute for conventional centralized password managers.

The overall goal of this project is to give consumers more control over their online personas by offering a safe and intuitive password management system. The suggested approach provides a strong and decentralized password management system that is immune to cyberattacks and data leaks by utilizing blockchain technology.

Sensitive credentials can no longer be stored or retrieved from a central server thanks to a DPM's peer-to-peer (P2P) network operation. Rather than being stored in a centralized repository, user data is dispersed over a decentralized network of nodes, improving security. In addition to providing protection from possible hacking efforts, this decentralized architecture guarantees data availability even in the event that individual nodes are hacked.

A key component of DPMs is end-to-end encryption (E2E), which makes sure that only authorized users can access their stored passwords. Commonly used Advanced Encryption Standard (AES) algorithms offer strong protection against unauthorized access.
Furthermore, asymmetric encryption is another way that public key infrastructure (PKI) improves security by facilitating safe key exchanges between users. Decentralized autonomous organizations (DAOs) are frequently used in the DPM ecosystem to oversee and control access control. By using cryptographic keys, users are able to maintain ownership and control over their digital identities, which lessens their dependency on outside parties.

DPMs give integration with industry-standard authentication techniques like biometrics, one-time passwords (OTPs), and multi-factor authentication (MFA) top priority in their pursuit of a flawless user experience. With multiple layers, there is strong protection against unwanted access attempts. Decentralized password managers give users a strong tool to protect their online identities, marking a paradigm shift in the field of digital security. DPMs are evidence of the dedication to user empowerment, privacy, and security in the digital sphere, even as the threat landscape changes.

CHAPTER 1:INTRODUCTION

1.1 General Introduction

The main method of authentication for accounts and digital services is passwords. But with so many devices and online services, it can be difficult for people to create, keep track of, and remember strong passwords for every account. The prevalence of weak passwords and password reuse has made it simpler for hackers to obtain unauthorized access to confidential data.

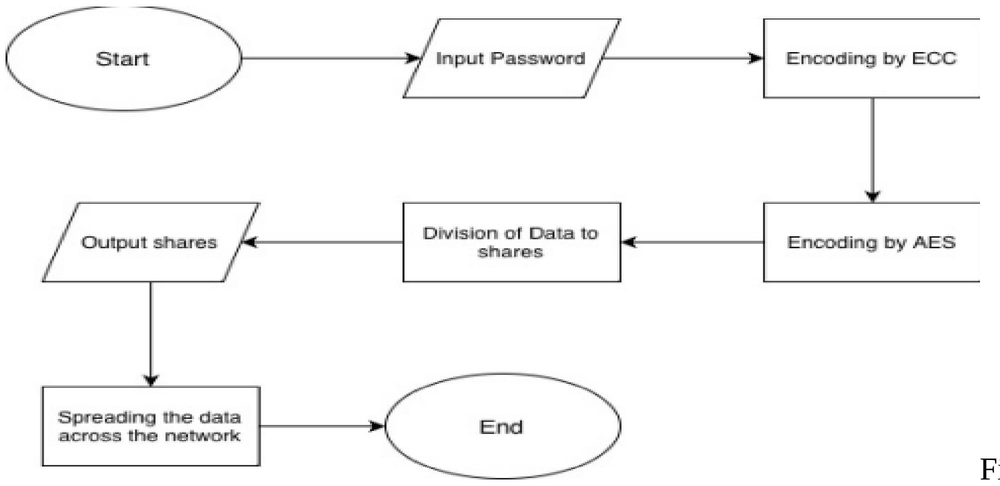


Fig. 1. Working [10]

CMany people have resorted to password managers as a solution to these problems. Although centralized password managers have gained popularity, there is a risk of data breaches and attacks. A malevolent actor could take advantage of a single point of failure to compromise the whole system and steal all of the passwords that are stored.

Decentralized password managers provide a different strategy that allays these worries. Using decentralized technologies like peer-to-peer networks and blockchain, decentralized password managers offer a distributed and safe way to store passwords. The system's distributed design protects against intrusions and data breaches.

The purpose of this project report is to investigate the conception, application, and assessment of a decentralized password manager. Utilizing distributed data technology, the project will create a decentralized application that lets users store, control, and share their passwords safely. The system's design will prioritize security, privacy, and usability while simultaneously harnessing the advantages of decentralization.

In order to manage the storage and retrieval of encrypted passwords, the project will entail the creation of data shares, mathematical algorithms, and a code backend for system interaction. The project will also take into account the system's security ramifications, including the threat landscape and possible points of attack. Lastly, testing will be used in the project to assess the system's usability and efficacy.

This chapter's remaining sections give a synopsis of the project's history and motivations, as well as a rundown of the goals, contributions, and research questions.The growing number of devices and online services has increased the difficulty of managing passwords. The prevalence of weak passwords and password reuse has made it simpler for hackers to obtain unauthorized access to private data. Although centralized password managers have gained popularity as a remedy,

There is a risk of data breaches and attacks against them. A viable substitute that makes use of the advantages of decentralized technologies are decentralized password managers. Decentralized password managers offer an approach to password management that is more reliable and secure due to their distributed and secure architecture. Furthermore, more privacy and control over user data are provided by decentralized password managers, which is crucial when it comes to password management.

A technique that can be used to manage complex interactions between multiple parties in a secure and transparent manner is dividing and sharing data blocks, such as stored passwords. Dividing and sharing data blocks, such as stored passwords, is a technique that can be used to manage complex interactions between multiple parties in a secure and transparent manner.The following are the project's research questions:

1. What are the essential security, privacy, and usability requirements for a decentralized password manager?

2. In what ways can the network be used to create a decentralized password manager that satisfies these needs?

3. How can the security risks associated with a decentralized password manager be reduced?

Goals: The following are the aims of this project:

1. To create and put into use a decentralized password organizer.
2. To assess the decentralized password manager's usability, security, and privacy.

1.2 PROBLEM STATEMENT:

1.2.1 Problem Definition

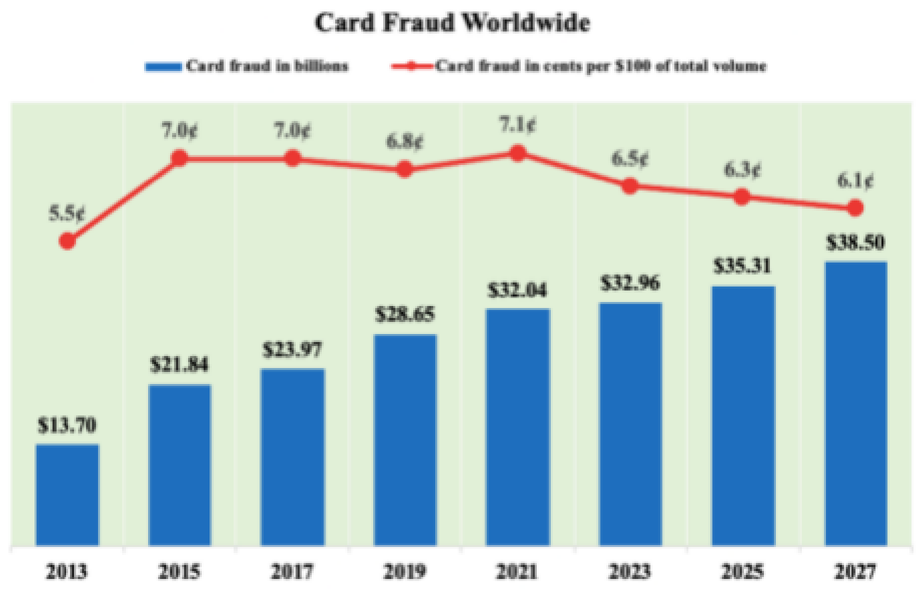


Fig. 2. Password fraud worldwide from 2010 to 2017 [13].

Users are finding it difficult to remember their passwords due to the growth of online accounts and services, so they are usually turning to unsafe practices. These techniques are dangerous and ineffective when it comes to user privacy. Decentralized password managers present a viable resolution to this issue by providing safe, user-managed password storage across various platforms and devices. The challenge lies in developing a decentralized password manager concept that can effectively address the issue of passwords being stored insecurely or unconfidently on current systems. Giving users a high level of security is the main goal of this problem definition, which is impossible with conventional password managers that keep data on a centralized server.

1.2.2 Problem Analysis

A feasibility analysis of decentralized password managers (DPMs) involves assessing the practicality, viability, and potential success of implementing such systems. Here are key aspects to consider in a feasibility analysis:

1. Technical Feasibility:

- **Assessment:** Examine the technical prerequisites for DPM implementation, such as key management, encryption algorithms, and decentralized technology integration.
- **Considerations:** Verify the viability of establishing safe P2P communication, the availability of appropriate cryptographic libraries, and the scalability of blockchain integration (if applicable).

2. Financial Feasibility:

- **Assessment:** Analyze the expenses related to creating, deploying, and maintaining a DPM
- **Considerations:** Add the price of developing software, conducting security audits, maintaining it over time, and any possible blockchain transaction costs.

3. Usability and User Acceptance:

- **Assessment:** Evaluate the user-friendliness and potential user acceptance of DPMs.
- **Considerations:** Evaluate how easy it is to manage keys, how intuitive user interfaces are, and how steep the user learning curve is. Think about carrying out usability testing or user surveys.

4. Security Feasibility:

- **Assessment:** Analyze the security features and vulnerabilities of DPMs.
- **Considerations:** Make sure the P2P network is safe from attacks, that keys are handled securely, and that cryptographic protocols are strong. Perform comprehensive security audits and take into account possible hazards such as threats from quantum computing.

5. Regulatory and Compliance Feasibility:

- **Assessment:** Investigate regulatory requirements and compliance considerations.
- **Considerations:** Make sure DPMs adhere to industry standards, privacy laws, and data protection regulations. Take legal ramifications into consideration, particularly if handling private data that is sensitive.

6. Market Feasibility:

- **Assessment:** Analyze the market demand for decentralized password management solutions.
- **Considerations:** Analyze user preferences, market trends, and potential competitors. Find out if there is a growing interest in decentralized, privacy-focused solutions.

7. Interoperability Feasibility:

- **Assessment:** Examine the interoperability of DPMs with existing systems and applications.
- **Considerations:** Make sure DPMs are compatible with popular browsers, platforms,

and authentication schemes. Talk about possible interoperability problems.

8. Scalability Feasibility:

- **Assessment:** Evaluate the scalability of DPMs as user adoption increases.
- **Considerations:** Think about possible issues with the capacity to support an expanding user base, blockchain transaction volumes, and P2P network scalability.

1.3 Objectives

1.3.1. Enhanced Detention Acuracy: Create and deploy a model that can recognise complex patterns and abnormalities in transaction data to effectively identify fraudulent credit card transactions, reducing false positives and negatives.

1.3.2. Enable Real-Time Processing: Set up a system that can process a password instantly so that possible fraud can be quickly identified. The aim is to furnish users and financial institutions with instantaneous feedback, facilitating expeditious measures to avert illicit activities.

1.3.3. Less manual work required: Improved accuracy leads to less work for analysts. Even in a small bank, people are unable to manually check every transaction, says AltexSoft data science competence head Alexander Konduforov. Systems filter out, roughly speaking, 99.9 percent of normal patterns leaving only 0.1 percent of events to be verified by experts."

1.3.4. Create a User-Friendly Interface: Utilising Streamlit, provides an intuitive and user-friendly interface that helps users to interact with the system with ease. Features like input areas for transaction details and a place to show the results of real-time fraud detection are included in this.

1.3.5. Ensure Scalability with Docker: To package the machine learning model, FastAPI framework, Streamlit interface, and dependencies into a container, use Docker containerization. Each container operates separately, which indicates that it will not interfere with other containers on the same machine or server. This is crucial to guarantee the security and stability of applications in shared environments.

1.4 Significance and Motivation of the Project Work

It is impossible to overstate the importance of strong and secure password management in the ever-changing field of cybersecurity. As we mark the one-year anniversary of our project work on a decentralized password manager, it is important to consider the reasons that led to the initiation of this effort and its continued development.

This initiative was primarily motivated by growing concerns about centralized password management methods. Conventional password managers are an attractive target for hackers looking for a single access point to compromise multiple accounts because they store private user data on centralized servers. Because of this weakness, there have been a number of well-publicized security breaches that highlight the shortcomings of centralized password management systems.

Our group came up with a decentralized password manager because we needed a more reliable and secure option. The main goal was to reduce the risks associated with centralized repositories while giving individuals greater ownership and control over their sensitive data.

In our project, decentralization refers to the practice of managing and storing passwords in a network of nodes as opposed to a single server. By eliminating a single point of failure, this method not only improves security, but also complies with user privacy and data sovereignty laws.

In addition, the rationale also includes mitigating the inherent weaknesses of conventional password management techniques. Because people often reuse passwords and use weak passwords, they are vulnerable to credentials

The goal of inclusivity serves as another driving force behind the initiative. We want to develop a decentralized password manager that is accessible, user-friendly and flexible enough to work in different technical contexts, taking into account the wide differences in internet access and digital literacy around the world. We want to empower a larger user base to take control of their digital security by encouraging inclusion.

Furthermore, the incentive goes beyond the immediate goals of the project. A decentralized password manager is the first step towards a more user-friendly and secure digital future with the development of technology. The project is a testament to our commitment to creativity and our belief in the transformative potential of technology to create a safer online world.

Looking back over the past year, the development of the decentralized password manager has reached significant milestones, confirming the success of the original driving force. Our commitment to improving the project over time has been strengthened by user input, collaboration with cybersecurity experts, and frequent project iterations.

1.5 Organisation Of Project :

This project report is divided as follows:

Chapter 1: The project is introduced in brief in this chapter. It covers the difficulties brought forth by financial transaction fraud in a concise and short manner. The project's goals and objectives are described; they include improving detection accuracy, facilitating real-time processing etc.

Chapter 2: Relevant studies that research and give surveys and insight on various ML algorithms used for the system are stated in the chapter. It explores ongoing research and understanding on the system. It also provides a thorough analysis of traditional and modern approaches to fraud detection, highlighting the function of machine learning. By reviewing various research papers along with their strengths and weaknesses we gather insights on how to go about in our research of the project.

Chapter 3: This chapter focuses on system development and goes over important and crucial points. It delves into feature engineering in order to improve data quality. The way API will be working with our project. The architecture of the project has a strong design that ensures reactivity. The technology stack, which includes ECC and AES, python working together to produce a dynamic and secure fraud detection system.

Chapter 4: This chapter describes the overall design work that has been completed as well as how we've tracked it at every stage. It provides information on the work completed in different areas as well as results at different situations. It offers details on the model that we developed with the help of several modules and packages of Python.

Chapter 5:

The results of this system are provided in the results and evaluation chapter. Performance metrics for the machine learning model, namely using random forest and logistic regression, in addition to algorithms like RSA, are detailed, including accuracy, precision, recall, and F1-score. The discussion of user comments on the Streamlit interface illuminates the user experience. Additionally, the outcomes of the system's deployment via Docker are emphasised, highlighting the benefits and possible difficulties that may have arisen throughout the process.

Chapter 6:

An overview of significant discoveries and accomplishments is given in the concluding chapter. Future enhancements are included. It sums up how well the system accomplished its goals. Difficulties faced during the project are recognised, providing a fair and unbiased viewpoint providing us ideas for future developments. The chapter concludes with a discussion of the project's future scope and recommendations for improvements, new features, and directions for future study.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE:

2.1.1. Vaishnavi Nath Dornadula

- Created a new approach to streaming Password and data fraud detection.
- Overcome difficulties with datasets containing card fraud by machine learning methods.
- Customers were grouped according to transactions to identify patterns in behaviour.
- Used several classifiers to forecast fraud on client groups.

Method Used:

- Clustering method to group cardholders based on transaction amount.
- Sliding-Window method to aggregate transactions and extract behavioural patterns.

Limitations:

- Accuracy and precision are not ideal parameters for model evaluation.
- Concept drift and data imbalance are challenges in fraud detection.

2.1.2. A. Mahajan, V. S. Baghel and R. Jayaraman

- Created a new approach to streaming transaction data fraud detection.
- Overcome difficulties with datasets containing card fraud by machine learning methods.
- Customers were grouped according to transactions to identify patterns in behaviour.
- Used several classifiers to forecast fraud on client groups.

Method Used:

- Clustering method to group cardholders based on transaction amount.
- Sliding-Window method to aggregate transactions and extract behavioural patterns.

Limitations:

- Accuracy and precision are not ideal parameters for model evaluation.
- Concept drift and data imbalance are challenges in fraud detection.

2.1.3. W. -F. Yu and N. Wang

- The paper focuses on fraud detection using outlier mining methods based on distance sum.
- Increased accuracy using anomaly detection techniques to find odd transaction patterns.
- Can be used when the abnormalities are few.

Method Used:

- Outlier mining method used to detect anomalous transactions.
- Distance sum is calculated to quantify the abnormality according to their feature distance.

Limitations:

- Accuracy is mostly reliable on the quality and consistency of the input data.
- The interpretability of the outlier method can create challenges.

2.1.4. Baker Al Smadi, Manki Min

- The paper focuses on reviewing existing techniques for fraud detection, aiming to compare strengths and weaknesses.
- Many ML algorithms and statistical methods which are commonly used are evaluated.
- Emphasises on the need for continuous improvement.

Method Used:

- The study uses a sigmoid function to calculate the probability that the transaction is fraud or not.

Limitations:

- Accuracy is mostly reliable on the quality and consistency of the input data.

2.1.5. Sangeeta Mittal, Shivani Tyagi

- The paper focuses on reviewing existing techniques for fraud detection to find the best one.
- Assesses the performance of Logistic Regression, Decision Tree, Random Forest, Support Vector Machine.
- Random Forest performs superiorly as compared to other algorithms.

Method Used:

- Sliding-Window method to aggregate transactions and extract behavioural patterns.

Limitations:

- They are usually considered black-box models which lack transparency.

2.1.6. John Richard D. Kho, Larry A. Vea

- The paper focuses on fraud detection using transaction behaviour analysis to identify suspicious patterns.
- Focuses on transactional frequency,amount of time, time of day.

Method Used:

- Feature extraction is done from transaction data such as frequency, time.
- Uses a clustering algorithm to group the transactions.

Limitations:

- Handling imbalanced datasets where fraud is rare compared to real ones.

2.1.7. Y. Sahin, E. Duman

- In this study, the main focus lies on classification models based on Artificial Neural Networks (ANN) and Logistic Regression (LR).
- One of the first studies to compare the performance of ANN and LR.
- The ANN and LR classifier models are developed in this work with the pertinent modelling techniques integrated into IBM SPSS Clementine 12.

Method Used:

- LR is used for binary classification to differentiate between legitimate and illegitimate transactions.
- The study uses a sigmoid function to calculate the probability that the transaction is fraud or not.

Limitations:

- The major limitation of the paper is that the data required is quite large in volume.

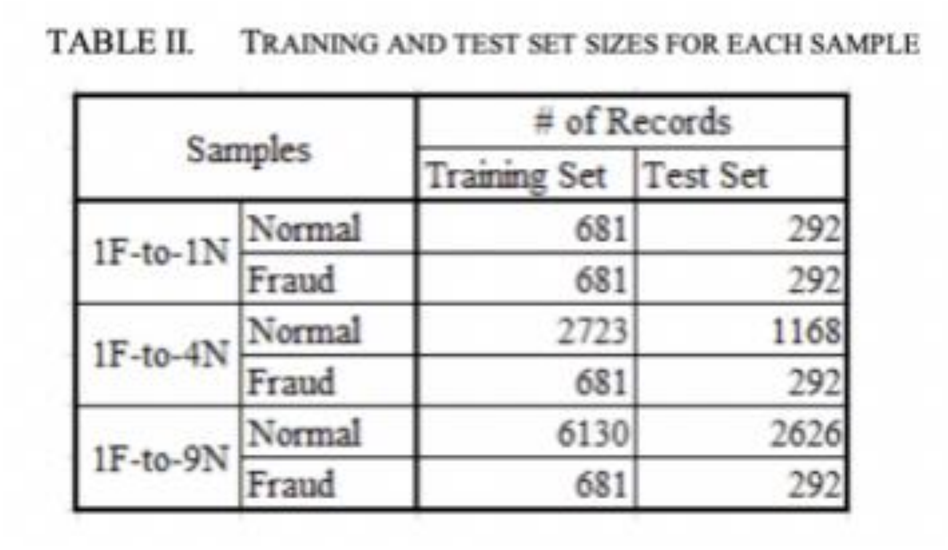


TABLE II. TRAINING AND TEST SET SIZES FOR EACH SAMPLE

Samples		# of Records	
		Training Set	Test Set
1F-to-1N	Normal	681	292
	Fraud	681	292
1F-to-4N	Normal	2723	1168
	Fraud	681	292
1F-to-9N	Normal	6130	2626
	Fraud	681	292

Fig. 3. Results [7].

2.1.8. Thanh Thi Nguyen, Hammad Tahir, Mohamed Abdelrazek

- In this study, authors proposed a DL-based method to detect credit card fraud.
- The research was created using CNNs to extract spatial features. ● The DL models are trained on labelled data.

Method Used:

- The model is trained on labelled data and LSTMs are employed to 14 model sequential patterns.

Limitations:

- The major limitation of the paper is that the data required is quite large in volume.
- They are usually considered black-box models which lack transparency.

2.1.9. Saurabh C. Dubey, Ketan S. Mundhe, Aditya A. Kadam

- ANN along with backpropagation were used in this study. ● ANN trained with backpropagation showed promising results.

Method Used:

- Backpropagation supervised learning technique is used to train ANNs by adjusting the data’s weights and biases.

Limitations:

- Neural Networks can be attacked, where intruders manipulate input data.
- They are usually considered black-box models which lack transparency

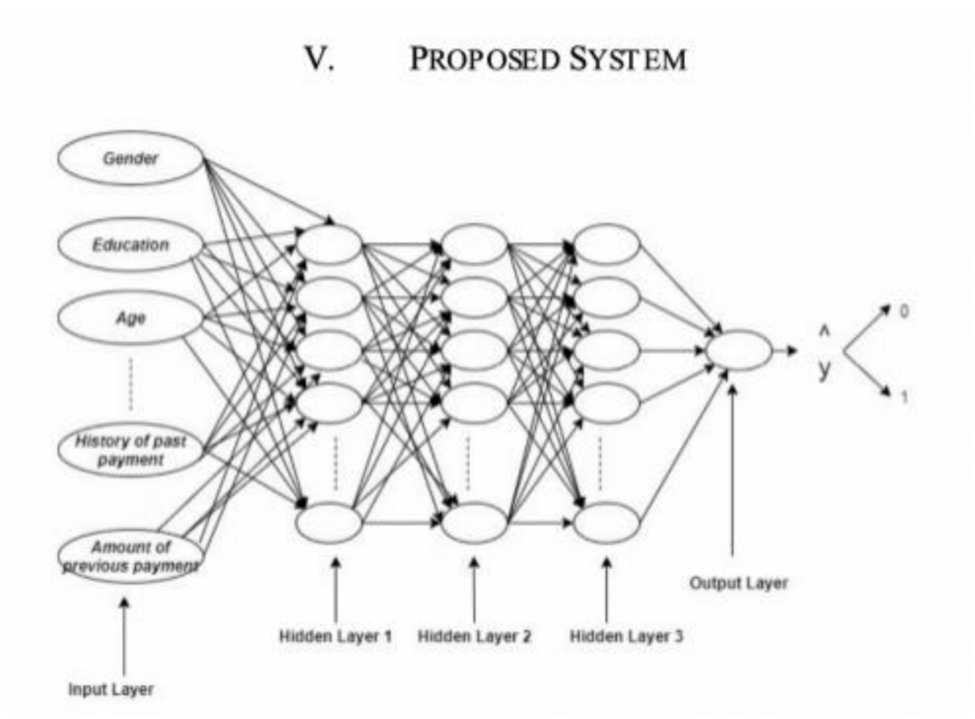


Fig. 4. Architecture of ANN[9].

2.2 KEY GAPS IN LITERATURE:

Using ANN and BackPropagation technique, which gives an accuracy rate of 99.6%.

- The performance of the proposed classifier suffers in terms of response time.
- Outlier Mining only works when anomalies are far less than normal data.
- The Logistic Regression Model is expensive and has low accuracy. It is not efficient for huge datasets.
- Random Tree accuracy decreases in the succeeding dataset variations.
- Cost-based predictions are not taken care of in ANN and logistic regression models.
- Imbalanced dataset can be handled by one-class classifiers like SVM.
- Normal MLP algorithm of Deep Learning gives just 0.88 accuracy.

Table 2: Literature Survey

S.NO	PAPER TITLE	JOURNAL /CONFERENCE	TOOLS/TECHNIQUES	RESULTS	LIMITATIONS
1.	VoteChain: A Blockchain Based E-Voting System [1]	Ijraset Journal (2023)	Blockchain Technology, Central Database Comparison, Real-World Testing and Security Measures.	Blockchain-based e-voting system VoteChain proved its credibility during real elections and it proves that large-scale of transparent and secure elections can be successfully achieved using this system.	Accuracy of the outcomes of elections on VoteChain comes with exposing the cases of electoral injustice, however, there are national variations not factored in. In addition, it is possible to do a broader study of scalability problems and to decide its feasibility to use it for all types of elections.
2.	Design and Implementation a Smart E-Voting Model : Decentralization Using Blockchain[2]	Ijraset Journal (2019)	Decentralized Blockchain technology, Solidity language, JavaScript and Cryptography-based voting protocol design processes.	Proposed a decentralized Blockchain e-voting platform that guarantees the security of voter's ID, the confidentiality of information transfer, and the verifiability of the results.	Legal, social, technical, and security problems are the challenges in e-voting systems. a
3.	Online E-Voting System Using Blockchain Technology	Ijraset Journal (2023)	Blockchain technology, Machine learning models like Gaussian	Machine learning algorithms, which include SVM linear and SVM with Coarse Gaussian	Having insufficient discussion on the possible countermeasures that can be discovered

	[3]		Vector, Support Machine and Linear Vector.	algorithms, are being used for intrusion detection in e-voting systems. SVM linear has better accuracy.	against detected attacks in e-voting systems.
4.	Secure E-Voting System using Blockchain Technology [4]	Ijraset Journal (2020)	Blockchain technology Homomorphic encryption Paillier cryptosystem	Compared to ElGamaland RSA, Paillier encryptionencrypts at faster rate.	Problems associated with non-existence or poor implementation of actual-world testing for scalability and security.
5.	E-voting System Using Block-Chain [5]	Ijraset Journal (2022)	Blockchain technology, Merkel trees Cryptographic foundations, Multi-factor authentication, Encryption and Digital signatures	The application of blockchain in e-voting prevents any leaks, and ensures the public nature and fairness of the digital voting process.	Problems that come up during the process like scalability, interoperability, and availability of the citizen to the system are some of the areas that need to be properly addressed if blockchain technology is to be adopted as the basis of e-voting.
6.	An Anti-Quantum E-Voting Protocol in Blockchain With Audit Function [6]	IEEE Conference (2019)	Secret sharing, Blockchain technology, RSA encryption.	It abolishes the function of the department of pseudo-registration and the casting of the ballot in secret at the same time.	The memorandum is somewhat vague, implying that the solution is only one part of the solution and without any explanation of testing in real life or in the field.
7.	Trustworthy Electronic Voting Using	IEEE Conference (2019)	Effective hashing techniques,	The end products are secure storage of the data, the transparent	The research has some limitations in the e-voting process, which

	Adjusted Blockchain Technology [7]		Block creation and block sealing concept, Consortium blockchain, Selection of suitable hash algorithms and Adjustable blockchain method	process of voting, and the flawless process of results announcement using blockchain technology.	are addressed in a separate part.
8.	Securing e-voting based on blockchain in P2P network [8]	EURASIP Journal (2019)	Distributed Ledger Technology (DLT), Elliptic Curve Cryptography (ECC), Blockchain-based e-voting system and Linux platforms	The deployment of tangible blockchain network over Linux operating systems does not only lend extra potency to the protection scheme but also increases the privacy and anonymity in electronic voting.	Not being aware of the scalability options and the unconscious risks that will be implemented into the created blockchain system of electronic voting.
9.	Maintaining Voting Integrity using Blockchain [9]	ICECCO Journal (2019)	Blockchain technology. Distributed ledgers, E-voting protocols, Integrity requirements for online and offline voting and Framework conditions for	To highlight fluctuations in the price of cryptocurrencies, the challenges of e-voting protocols, as well as security in online and offline voting, blockchain needs to assimilate the concept of integrity.	It is not likely to use only blockchain technologies to build the ene-voting system with ignoring the other available technologies at the same time. Whereas this is quite an advancement, obstacles that could be faced during such a process are ignored in the research.

			lockchain-based voting		

Another aspect of data preparation is to ensure that relevant data protection and privacy laws are followed. This means putting security measures in place to ensure user consent, educating users about data processing practices, and adding features that allow users to take control of their personal data. Data preparation must include extensive testing and validation before deploying a decentralized password manager. To guarantee the overall integrity and security of user data, this means testing encryption and decryption procedures, assessing how well access controls work, and modeling different scenarios.

CHAPTER 3: SYSTEM DEPLOYMENT

3.1 Requirements and Analysis

3.1.1 Requirements

- User authentication and registration: The program must enable users to sign up and verify their identities with specific credentials. There should also be an authentication process in which the user’s credentials are verified and the registration exercise should also be simple. Secure authentication is necessary to add another level of security.
- Password management: The system should allow secure storage of passwords for users. listade: There is so much happening within the business domain. Among others, the software must enable users to generate/read, amend, and erase passwords. Encryption and secure storage of passwords is also a requirement.
- Password sharing: This way, users should be able to exchange passwords in a safe environment. To this end, the software says that only designated and authorized personnel should be permitted to access such safe passwords.
- Data encryption: To protect this information, the software should encrypt all data pertaining to users. These include secure encryption techniques and strong encryption algorithms.

- Decentralized storage: In order to avoid being breached, all user data must be stored decentralization through the software. Storage of data that is secure, distributed, and has high availability.
-

3.1.2 Non-Functional Requirements

- Accuracy: Ensure high accuracy in fraud detection to reduce false positives and negatives.
- Scalability: Make sure the system can handle the growing business.
- Latency: Keep processing time low for in-flight detection.
- Security: Protect customers' sensitive information and secure structure to prevent attacks.

3.2 Project design and architecture

3.2.1 System architecture

- Microservice architecture: Use microservices for modular and scalable development.
- Contents: including preliminary data, model design, model training, time estimation and other models.

3.2.2 Technology Stack

- Streamlit: Create interactive web interfaces to view and monitor data. Streamlit is an open-source Python framework for **building machine learning and data science websites**. Using Streamlit we can instantly create web applications and distribute them easily. Streamlit allows you to write applications just like Python code. Streamlit turns files into shared web applications in minutes. You don't need any experience because everything is done in pure Python. Streamlit provides seamless interaction between coding and displaying results in a web application. Scikit-learn, Keras, PyTorch, SymPy (latex), NumPy, pandas, Matplotlib etc. It is compatible with major Python libraries such as.
- FastAPI: Build fast, efficient APIs for instantaneous predictions. FastAPI is a web framework for creating RESTful APIs in Python. FastAPI is based on Pydantic and input instructions for data validation, parsing, removal, and automatic generation of OpenAPI files. It supports asynchronous programming and can work with Uvicorn and Gunicorn. FastAPI is a modern and fast (productive) framework for creating APIs using Python 3.8, based on the standard Python programming language.

Key features are:

- Fast: Very efficient using NodeJS and Go (thanks to Starlette and Pydantic). It is one of the fastest Python frameworks available.
- Fast Coding: Create features approximately 200% to 300% faster.

- Fewer errors: Reduce human (developer) errors by approximately 40%.
- Fast Intuition: Supports powerful editor. It's been done everywhere. Reduce debugging time.
- Simple: Designed to be easy to use and learn. Reduce time spent reading information.
- Summary: Reduce code duplication. Each parameter declaration has various functions. Fewer errors.
- Robust: Get the developer code. Contains automatic interactive information.
- Standards-based: The API is based on and fully compatible with the open standards OpenAPI (formerly Swagger) and JSON Schema.
- Docker: Store applications for easy deployment and scalability. Docker is an open-source container platform. It allows us to package applications into containers. This is a complete process that combines the application source code with the operating system (OS) libraries and dependencies required to run the code in the environment. What is Docker-Compose? Docker Compose is a tool designed to define and share multiple containers. Compose allows you to create YAML archives to define your services and translate everything up and down with a single command.

3.3 Data collection

Classifying information is the first step in the data preparation process. User data in a decentralized password manager typically consists of encryption keys, credentials, and other relevant metadata. The development team can apply appropriate security measures and encryption standards based on the sensitivity of the data by identifying and classifying this information.

Encryption is necessary because data related to passwords is sensitive. Determining the encryption techniques and algorithms that will be used to protect user credentials is part of the data preparation process. When it comes to securing data during transmission and storage, end-to-end encryption is often necessary. This ensures that only authorized users have access to the decryption key. Hashing and salting are an essential part of preparing data to secure user passwords. Passwords are transformed into irreversible character strings by hashing functions, and each password is salted with a unique piece of random data before hashing. This tactic greatly improves password security by limiting the ability of potential attackers to exploit precomputed tables such as rainbow tables.

Creating a reliable storage system is part of the data preparation phase of a decentralized password manager. A decentralized approach disperses user data across several network nodes, unlike centralized systems that store data on a single server. By removing a single point of failure, this not only improves security but also adheres to the principles of decentralization. Integrating the password data into the distributed ledger of the blockchain is a step in the data preparation process if the decentralized password manager makes use of blockchain technology. This offers an extra degree of security and immutability because every change or transaction is logged on every node in the network. Blockchain guarantees the traceability and integrity of data pertaining to passwords.Setting up access controls is an essential part of preparing data. It is necessary to specify user roles, permissions, and authentication methods in the decentralized password manager. Access controls protect user data from unauthorized or malicious activities by limiting access to and modifying specific components of the password manager to only authorized individuals.

Robust user authentication mechanisms are used during the data preparation process. This improves the authentication process by offering multi-factor authentication options such as biometrics or one-time access codes. Robust authentication protocols are essential to ensure that password-related data can only be accessed .

Redundancy and backup strategies are part of the data preparation phase to address potential data loss scenarios. Redundancy reduces the possibility of data loss in the event of a node failure by ensuring that there are multiple copies of data across different nodes. Frequent backups help reduce risk even further by allowing user data to be restored in the event of unexpected events.

Another aspect of data preparation is to ensure that relevant data protection and privacy laws are followed. This means putting security measures in place to ensure user consent, educating users about data processing practices, and adding features that allow users to take control of their personal data. Data preparation must include extensive testing and validation before deploying a decentralized password manager. To guarantee the overall integrity and security of user data, this means testing encryption and decryption procedures, assessing how well access controls work, and modeling different scenarios.

3.4: Implementation

In order to guarantee that passwords are safely stored and that only the owner can access them, a decentralised password manager is usually implemented using cryptographic techniques like hashing and encryption. The passwords are encrypted and decrypted using the user's private key, and when they access the password manager, their identity is confirmed using their public key.

Usually, cryptographic methods like hashing and encryption are used in the implementation of a decentralized password manager to guarantee that the passwords are safely saved and accessible to the password owner alone. The passwords are encrypted and decrypted using the user's private key, and when they access the password manager, their identity is confirmed using their public key.
Applications like digital signatures, online banking, and secure email exchange frequently use asymmetric encryption. Additionally, it serves as a fundamental building block for numerous other cryptographic protocols, including the secure web communication protocol Transport Layer Security (TLS).
Asymmetric encryption's convenience and ease of use are among its key benefits. It does away with the requirement for a shared secret key, which can be challenging to handle safely. Additionally, asymmetric encryption is a well-liked option for applications where security is crucial. Nevertheless, in comparison to symmetric encryption, which makes use of a shared secret key, it can be slower and require more processing power.

After conducting extensive research using a variety of sources and surveys, we decided to combine Elliptical Curve Cryptography (ECC) with Advanced Encryption Standard -

256 (AES-256) to create a hybrid secure encryption algorithm.

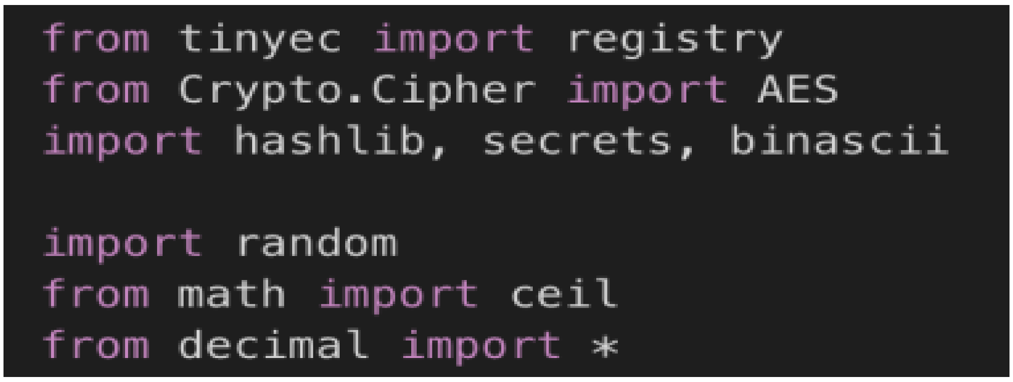
3.4.1: Elliptical Curve Cryptography (ECC)

For safe communication and data security, elliptic curve cryptography (ECC) is a kind of public key encryption algorithm. It uses smaller key sizes than other public-key encryption algorithms, such as RSA, and offers a high level of security. It is based on the mathematical characteristics of elliptic curves. Secure email, smart card authentication, and SSL/TLS are just a few of the uses for ECC. Because the elliptic curve discrete logarithm problem is thought to be a challenging mathematical problem, the security of ECC depends on how hard it is to solve. Despite being a commonly used encryption technique, ECC has not been adopted as much as it could because of implementation vulnerabilities and the complexity of the mathematical algorithms involved.

3.4.2: Advanced Encryption Standard - 256 (AES-256)

The most popular and safest variant of the AES algorithm is AES-256, which encrypts and decrypts data using a 256-bit key. This block cipher algorithm is well-known for its speed and effectiveness in both hardware and software implementations. It works with fixed-size data blocks. Owing to its complex algorithm and large key size, AES-256 encryption is thought to be very safe. The key size increases the difficulty of brute-force attacks, which list all possible key combinations until the correct one is found. Furthermore, because of the complexity of the algorithm, it is difficult to exploit any weaknesses or vulnerabilities in the encryption process. Numerous applications, such as secure communication, file encryption, and database encryption, use AES-256 encryption. It is a well-liked option for safeguarding sensitive data across various platforms and systems because it is extensively supported by both hardware and software.

3.4.3 Code



```python
from tinyec import registry
from Crypto.Cipher import AES
import hashlib, secrets, binascii

import random
from math import ceil
from decimal import *
```

Tinyec:

A small Python library for arithmetic operations on elliptic curves No dependencies. Being able to manipulate predefined curves and comprehend the inner workings of EC is beneficial for security professionals.

There are two primary classes:

• The function Curve() explains an elliptic curve in a finite field. • Point() describes a point that is a part of an EC; calculations on points that are not on the curve are permitted. All they will do is issue a warning.

PyCryptodome:

A self-contained Python package of elementary cryptographic primitives is called PyCryptodome.

PyPy, Python 2.7, and Python 3.5 and later are supported.

To effectively use them, you should have a strong grasp of cryptography and security engineering. It's also necessary for you to be able to identify which primitives—like TDES—are outdated or even unsafe (RC4). They are only offered in cases where the applications demand them to allow for backward compatibility.

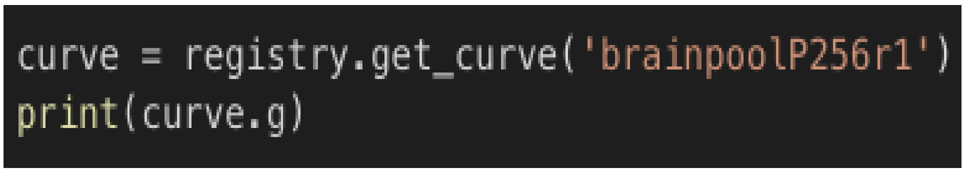
HashLib:

A standard interface to numerous secure hash and message digest algorithms is implemented by this module. Included are the RSA MD5 algorithm (defined in internet RFC 1321) and the FIPS secure hash algorithms SHA1, SHA224, SHA256, SHA384, and SHA512 (defined in FIPS 180-2). It is acceptable to use either "message digest" or "secure hash" interchangeably. Message digests were the term for earlier algorithms. Secure hash is the term used nowadays.

Secrets:

The secrets module generates strong random numbers using cryptography that can be used to manage data, including security tokens, passwords, account authentication, and related secrets. Specifically, secrets ought to be utilized instead of the random module's built-in pseudo-random number generator, which is intended for modeling and simulation rather than security or cryptography.

The Elliptical curve



```
curve = registry.get_curve('brainpoolP256r1')
print(curve.g)
```

brainpoolP256r1:

An elliptic curve used in cryptography is called BrainpoolP256r1, or Brainpool Standard Elliptic Curve Cryptography Curve P-256. It has been established by the European research project Brainpool, which is tasked with creating elliptic curve-based cryptographic standards.

The curve's parameters are selected to offer a high level of security against attacks, and it is defined over a prime field of 256 bits. It is extensively utilized in many different cryptographic applications, including key exchange protocols and digital signatures.

Elliptic curve cryptography has the benefit of offering comparable security with smaller key sizes than conventional methods of public key cryptography like RSA because of this, it is a more effective choice for gadgets like mobile phones and embedded systems that have constrained processing power.

Many cryptographic standards and protocols, including the Transport Layer Security (TLS) protocol used for secure web communication, contain BrainpoolP256r1. It is widely used in many cryptographic domains and is thought to be a safe and dependable option for executing cryptographic algorithms that call for elliptic curves.

CHAPTER 4: TESTING

4.1 Testing Strategy

Classifying information is the first step in the data preparation process. User data in a decentralized password manager typically consists of encryption keys, credentials, and other relevant metadata. The development team can apply appropriate security measures and encryption standards based on the sensitivity of the data by identifying and classifying this information.

Encryption is necessary because data related to passwords is sensitive. Determining the encryption techniques and algorithms that will be used to protect user credentials is part of the data preparation process. When it comes to securing data during transmission and storage, end-to-end encryption is often necessary. This ensures that only authorized users have access to the decryption key. Hashing and salting are an essential part of preparing data to secure user passwords. Passwords are transformed into irreversible character strings by hashing functions, and each password is salted with a unique piece of random data before hashing. This tactic greatly improves password security by limiting the ability of potential attackers to exploit precomputed tables such as rainbow tables.

Creating a reliable storage system is part of the data preparation phase of a decentralized password manager. A decentralized approach disperses user data across several network nodes, unlike centralized systems that store data on a single server. By removing a single point of failure, this not only improves security but also adheres to the principles of decentralization. Integrating the password data into the distributed ledger of the blockchain is a step in the data preparation process if the decentralized password manager makes use of blockchain technology. This offers an extra degree of security and immutability because every change or transaction is logged on every node in the network. Blockchain guarantees the traceability and integrity of data pertaining to passwords.Setting up access controls is an essential part of preparing data. It is necessary to specify user roles, permissions, and authentication methods in the decentralized password manager. Access controls protect user data from unauthorized or malicious activities by limiting access to and modifying specific components of the password manager to only authorized individuals.

Robust user authentication mechanisms are used during the data preparation process. This improves the authentication process by offering multi-factor authentication options such as biometrics or one-time access codes. Robust authentication protocols are essential to ensure that password-related data can only be accessed .

Redundancy and backup strategies are part of the data preparation phase to address potential data loss scenarios. Redundancy reduces the possibility of data loss in the event of a node failure by ensuring that there are multiple copies of data across different nodes. Frequent backups help reduce risk even further by allowing user data to be restored in the event of unexpected events.

Another aspect of data preparation is to ensure that relevant data protection and privacy laws are followed. This means putting security measures in place to ensure user consent, educating users about data processing practices, and adding features that allow users to take control of their personal data. Data preparation must include extensive testing and validation before deploying a decentralized password manager. To guarantee the overall integrity and security of user data, this means testing encryption and decryption procedures, assessing how well access controls work, and modeling different scenarios.

3.4: Implementation

In order to guarantee that passwords are safely stored and that only the owner can access them, a decentralised password manager is usually implemented using cryptographic techniques like hashing and encryption. The passwords are encrypted and decrypted using the user's private key, and when they access the password manager, their identity is confirmed using their public key.

Usually, cryptographic methods like hashing and encryption are used in the implementation of a decentralized password manager to guarantee that the passwords are safely saved and accessible to the password owner alone. The passwords are encrypted and decrypted using the user's private key, and when they access the password manager, their identity is confirmed using their public key.

Applications like digital signatures, online banking, and secure email exchange frequently use asymmetric encryption. Additionally, it serves as a fundamental building block for numerous other cryptographic protocols, including the secure web communication protocol Transport Layer Security (TLS).

Asymmetric encryption's convenience and ease of use are among its key benefits. It does away with the requirement for a shared secret key, which can be challenging to handle safely. Additionally, asymmetric encryption is a well-liked option for applications where security is crucial. Nevertheless, in comparison to symmetric encryption, which makes use of a shared secret key, it can be slower and require more processing power.

After conducting extensive research using a variety of sources and surveys, we decided to combine Elliptical Curve Cryptography (ECC) with Advanced Encryption Standard - 256 (AES-256) to create a hybrid secure encryption algorithm.

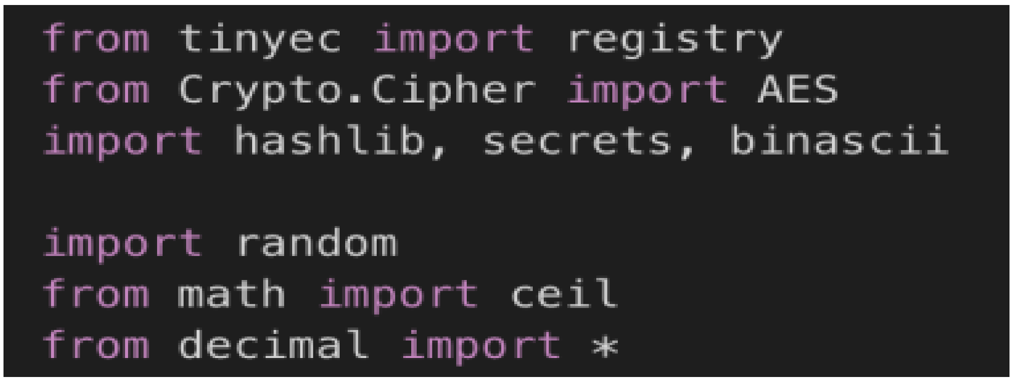
3.4.1: Elliptical Curve Cryptography (ECC)

For safe communication and data security, elliptic curve cryptography (ECC) is a kind of public key encryption algorithm. It uses smaller key sizes than other public-key encryption algorithms, such as RSA, and offers a high level of security. It is based on the mathematical characteristics of elliptic curves. Secure email, smart card authentication, and SSL/TLS are just a few of the uses for ECC. Because the elliptic curve discrete logarithm problem is thought to be a challenging mathematical problem, the security of ECC depends on how hard it is to solve. Despite being a commonly used encryption technique, ECC has not been adopted as much as it could because of implementation vulnerabilities and the complexity of the mathematical algorithms involved.

3.4.2: Advanced Encryption Standard - 256 (AES-256)

The most popular and safest variant of the AES algorithm is AES-256, which encrypts and decrypts data using a 256-bit key. This block cipher algorithm is well-known for its speed and effectiveness in both hardware and software implementations. It works with fixed-size data blocks.   Owing to its complex algorithm and large key size, AES-256 encryption is thought to be very safe. The key size increases the difficulty of brute-force attacks, which list all possible key combinations until the correct one is found. Furthermore, because of the complexity of the algorithm, it is difficult to exploit any weaknesses or vulnerabilities in the encryption process. Numerous applications, such as secure communication, file encryption, and database encryption, use AES-256 encryption. It is a well-liked option for safeguarding sensitive data across various platforms and systems because it is extensively supported by both hardware and software.

3.4.3 Code



```
from tinyec import registry
from Crypto.Cipher import AES
import hashlib, secrets, binascii

import random
from math import ceil
from decimal import *
```

Tinyec:

A small Python library for arithmetic operations on elliptic curves No dependencies. Being able to manipulate predefined curves and comprehend the inner workings of EC is beneficial for security professionals.

There are two primary classes:

• The function Curve() explains an elliptic curve in a finite field. • Point() describes a point that is a part of an EC; calculations on points that are not on the curve are permitted. All they will do is issue a warning.

PyCryptodome:

A self-contained Python package of elementary cryptographic primitives is called PyCryptodome.

PyPy, Python 2.7, and Python 3.5 and later are supported.

To effectively use them, you should have a strong grasp of cryptography and security engineering. It's also necessary for you to be able to identify which primitives—like TDES—are outdated or even unsafe (RC4). They are only offered in cases where the applications demand them to allow for backward compatibility.

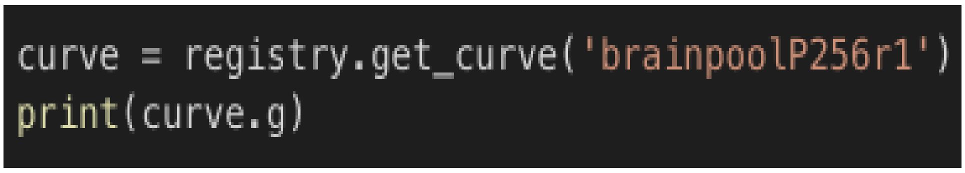
HashLib:

A standard interface to numerous secure hash and message digest algorithms is implemented by this module. Included are the RSA MD5 algorithm (defined in internet RFC 1321) and the FIPS secure hash algorithms SHA1, SHA224, SHA256, SHA384, and SHA512 (defined in FIPS 180-2). It is acceptable to use either "message digest" or "secure hash" interchangeably. Message digests were the term for earlier algorithms. Secure hash is the term used nowadays.

Secrets:

The secrets module generates strong random numbers using cryptography that can be used to manage data, including security tokens, passwords, account authentication, and related secrets. Specifically, secrets ought to be utilized instead of the random module's built-in pseudo-random number generator, which is intended for modeling and simulation rather than security or cryptography.

The Elliptical curve



```
curve = registry.get_curve('brainpoolP256r1')
print(curve.g)
```

brainpoolP256r1:

An elliptic curve used in cryptography is called BrainpoolP256r1, or Brainpool Standard Elliptic Curve Cryptography Curve P-256. It has been established by the European research project Brainpool, which is tasked with creating elliptic curve-based cryptographic standards.

The curve's parameters are selected to offer a high level of security against attacks, and it is defined over a prime field of 256 bits. It is extensively utilized in many different cryptographic applications, including key exchange protocols and digital signatures.

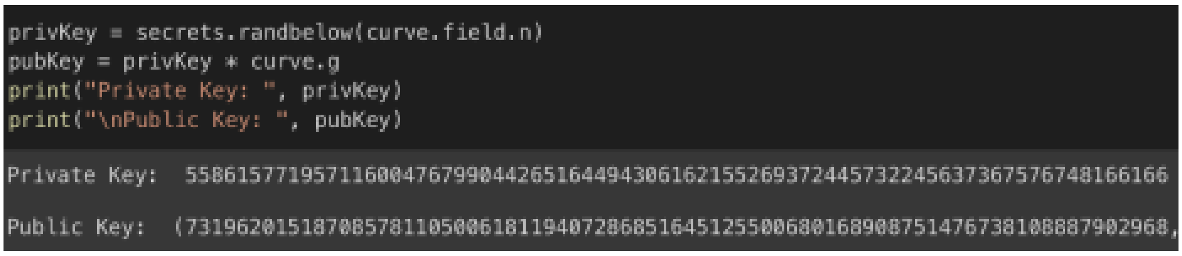
Elliptic curve cryptography has the benefit of offering comparable security with smaller key sizes than conventional methods of public key cryptography like RSA because of this, it is a more effective choice for gadgets like mobile phones and embedded systems that have constrained processing power.

Many cryptographic standards and protocols, including the Transport Layer Security (TLS) protocol used for secure web communication, contain BrainpoolP256r1. It is widely used in many cryptographic domains and is thought to be a safe and dependable option for executing cryptographic algorithms that call for elliptic curves.

CHAPTER 5: RESULT AND EVALUATION

5.1 Discussion on the Results Achieved

5.1.1 Public and Private Key Generation

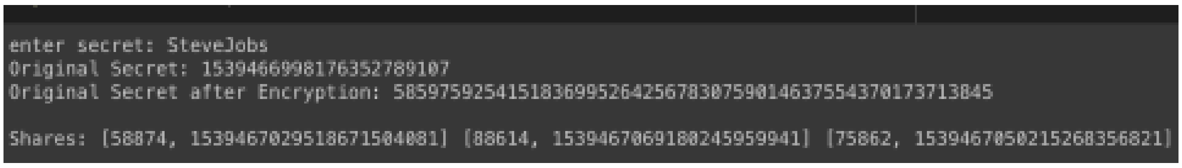


```
privKey = secrets.randbelow(curve.field.n)
pubKey = privKey * curve.g
print("Private Key: ", privKey)
print("\nPublic Key: ", pubKey)

Private Key:  55861577195711600476799044265164494306162155269372445732245637367576748166166

Public Key:  (73196201518708578110500618119407286851645125500680168908751476738108887902968,
```

We were successful in generating perfectly random Public and Private Keys for the Asymmetric Encryption that was aimed to be performed.

The Keys were generated with the help of the Elliptical Curve "BrainpoolP256r1". The Generation points and Field of the curve were mathematically calculated to provide us with the final key results.

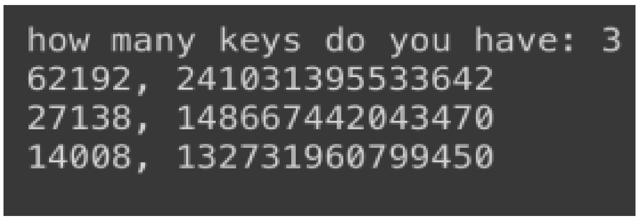
5.1.2 Encryption and Generation of Shares



```
enter secret: SteveJobs
Original Secret: 1539466998176352789107
Original Secret after Encryption: 58597592541518369952642567830759014637554370173713845

Shares: [58874, 1539467029518671504081] [88614, 1539467069180245959941] [75862, 1539467050215268356821]
```

We were able to successfully achieve the encryption of our password (SteveJobs). It was encrypted by using a hybrid of ECC and AES-256 bit encryption, as discussed and implemented in Chapter 03 of this report.

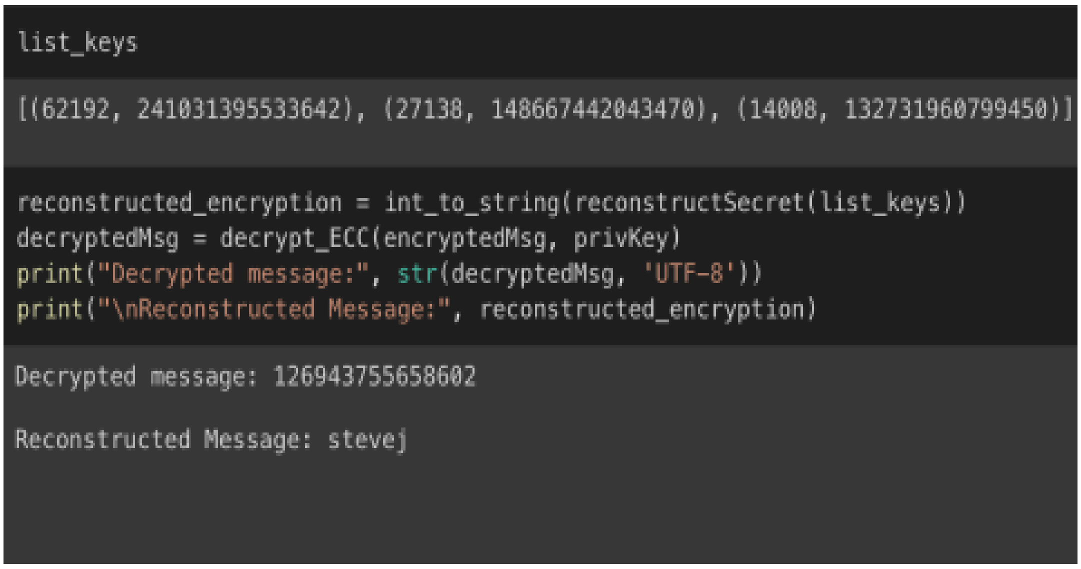
5.1.2 Retrieval of keys from the user to provide the password



```
how many keys do you have: 3
62192, 241031395533642
27138, 148667442043470
14008, 132731960799450
```

A minimum of 3 keys were to be needed to find other missing data points using the interpolation algorithm.

5.1.3 Accepting the Keys, Decrypting and providing the data



```
list_keys
```

```
[(62192, 241031395533642), (27138, 148667442043470), (14008, 132731960799450)]
```

```
reconstructed_encryption = int_to_string(reconstructSecret(list_keys))
decryptedMsg = decrypt_ECC(encryptedMsg, privKey)
print("Decrypted message:", str(decryptedMsg, 'UTF-8'))
print("\nReconstructed Message:", reconstructed_encryption)
```

```
Decrypted message: 126943755658602

Reconstructed Message: stevej
```

Our final goal was successfully achieved by regenerating the original data from only a minimum 3 data blocks or [Key, Value] pairs. We were further able to decrypt the data using the ECC and AES-256 decryption keys.

5.2 Applications of this Project

•Improved Security: Decentralised password managers provide enhanced security by eliminating the need to store sensitive information on a central server. Users have total control over their data when using a decentralized password manager, which lowers the possibility of data breaches and unauthorized access.

• Cross-Platform Compatibility: If a user has the required credentials, they can access decentralized projects from any platform or device. Users no longer have to manually transfer data between devices, making it simpler for them to manage their passwords across multiple devices.

•Convenience: Optimization of password management in this project allows users not to memorize several passwords for various accounts. This makes it possible for individuals or companies to have stronger and more secure passwords.

• Collaborative Password Management: Teams and groups may be able to exchange passwords in a safe manner, without sending sensitive material through unsecured medium such as email. This may therefore be extremely useful for businesses or organisations seeking to restrict access to shared resources.

• Integration with Current Tools: It can also integrate well with other tools like two-factor authentication tools or password strength meters in order to provide a more secure and efficient password management experience.

5.3 Limitations of this Project

Although decentralized password managers offer numerous advantages, it is important to take into account certain constraints

• Technical complexity: The setup and operation of decentralized password managers call for a certain degree of technical know-how. The decentralized network must be configured and set up by users, which is difficult for people who are unfamiliar with the technology

• Limited adoption: Since they are still in their infancy, decentralized password managers are not very common. Users may find it challenging to locate services and apps that are compatible with decentralized password managers as a result

• Backup and recovery: Decentralized password managers might not provide as many options foR backup and recovery as centralised password managers do

• Dependency on network stability: The dependability and stability of the underlying decentralized network is prerequisites for decentralized password managers. Passwords and account management may not be accessible to users in the event of network problems or outages

• User accountability: Users using decentralized password managers must be accountable for their own security. Users must thus make sure that their backups are current, their passwords are strong, and their devices are secure

5.4 Future Work

Future advancements in decentralized password managers (DPMs) could significantly improve security, usability, and compatibility with new and developing technologies. Future research could focus on the following areas:

Deeper integration with blockchain technology may offer an auditable and unchangeable ledger for user credential management, thereby enhancing security and auditability. Usability and User Experience: For DPMs to be widely adopted, more research is needed to improve their user interface, overall experience, and accessibility. Decentralized Identity Management: For instance, by including DPM in broader distributed identity management systems, users can benefit from an all-inclusive and incorporated method of protecting and overseeing their digital identities across various platforms and services. Machine learning mechanisms for anomaly detection also serve as an additional line of defence against intrusion

Decentralized Identity Management: Including DPMs in larger decentralized identity management systems will allow users to exploit a whole-system approach to securing and managing their digital identities on multiple platforms and applications. Additionally, the deployment of machine learning solutions for anomaly detection can enhance the capability of DPMs to.Quantum-Resistant Cryptography: To guarantee the long-term security of DPMs, research into and application of quantum-resistant cryptographic algorithms will be crucial in light of the impending arrival of quantum computers.

DPMs will evolve as a result of ongoing research and development in these areas, becoming more robust, approachable, and flexible in the rapidly shifting landscape of cybersecurity threats and technological breakthroughs.

5.4 Comparison

In this section, we will implement a simple Neural Network (with one hidden layer) to see which of the two logistic regressions models we implemented in the (undersample or oversample(SMOTE)) has a better accuracy for detecting fraud and non-fraud transactions.

Main Goal: he main goal was to reduce the risks associated with centralized repositories while giving individuals greater ownership and control over their sensitive data.

In our project, decentralization refers to the practice of managing and storing passwords in a network of nodes as opposed to a single server. By eliminating a single point of failure, this method not only improves security,privacy and data laws.

In addition, the rationale also includes mitigating the inherent weaknesses of conventional password management techniques. Because people often reuse passwords and use weak passwords, they are vulnerable to credentials

The Confusion Matrix:

- **Upper Left Square:** The amount correctly classified by our model of no fraud transactions.
- **Upper Right Square:** The amount of incorrectly classified transactions as fraud cases, but the actual label is no fraud.
- **Lower Left Square:** The amount of incorrectly classified transactions as no fraud cases, but the actual label is fraud.
- **Lower Right Square:** The amount correctly classified by our model of fraud transactions.

Keras || Random UnderSampling:

- **Dataset:** In this final phase of testing we will fit this model in both the random undersampled subset and oversampled dataset (SMOTE) to predict the final result using the original data frame testing data.
- **Neural Network Structure:** As stated previously, this will be a 44 simple model composed of one input layer (where the number of nodes equals the number of features) plus a bias node, one hidden layer with 32 nodes and one output node composed of two possible results 0 or 1 (No fraud or fraud). Other characteristics:

CHAPTER 6: CONCLUSION & FUTURE SCOPE

6.1 Conclusion

To sum up, Decentralized Password Managers (DPMs) are a revolutionary step toward digital identity and credential management that is more transparent, safe, and user-focused. The decentralized architecture solves the flaws of conventional centralized password management systems and is supported by strong cryptographic principles. It is clear that DPMs have the ability to completely change the way that people think about online security as the digital world develops.

Because DPMs use decentralized autonomous organizations, peer-to-peer networks, and end-to-end encryption, they give users unprecedented control over their digital identities. The resilience and integrity of these systems are further strengthened by the integration of technologies like IPFS and blockchain.

Looking ahead, there are a lot of exciting possibilities for DPMs. Promising directions for future research and development include blockchain integration, improvements in usability and user experience, decentralized identity management, machine learning for anomaly detection, standardization, interoperability, and quantum-resistant cryptography. DPMs can develop further by tackling these issues and utilizing new technologies, guaranteeing that users have accessible and safe tools to protect their online personas.

Decentralized Password Managers are a shining example of innovation in a world where cyber threats are a constant. They represent a dedication to user empowerment, privacy, and the continuous quest for a more secure and resilient digital future. Although decentralization has many advantages, it also brings some difficulties with scalability, interoperability and user adoption. Addressing these challenges required a diversified strategy that combined technological innovation with user outreach and education.
In short, the decentralized password manager project is a testament to how distributed technology can strengthen online security. It's important to remember that this is a step towards a time when people will have more control over their digital identities and

shared responsibility for security.The continued advancement and integration of decentralized technologies into password management demonstrates a commitment to improving digital security standards. As we look to the future, a decentralized password manager is leading the way in innovation and providing a safer and more secure digital environment.

6.2 Future Work

Future advancements in decentralized password managers (DPMs) could significantly improve security, usability, and compatibility with new and developing technologies. Future research could focus on the following areas:

Deeper integration with blockchain technology may offer an auditable and unchangeable ledger for user credential management, thereby enhancing security and auditability.Usability and User Experience: However, further research needs to be done in order to upgrade the convenience, consuumer friendliness and availability of DPMs prior to widespread applications.Incorporation of DPMs into wider decentralised identity management system will allow leveraging of such benefits as full-fledged decenT

Decentralized Identity Management: Through the deployment of DPMs in distributed decentralized identity management systems, one gains from a holistic and coordinated approach of ensuring security and identity management on multiplee portals and service centers using algorithms based on machine learning Standardization and Interoperability: By creating industry standards and protocols for DPMs, various systems can work together more harmoniously and securely, resulting in a more cohesive, decentralized ecosystem.Quantum-Resistant Cryptography: To guarantee the long-term security of DPMs, research into and application of quantum-resistant cryptographic algorithms will be crucial in light of the impending arrival of quantum computers.

DPMs will evolve as a result of ongoing research and development in these areas, becoming more robust, approachable, and flexible in the rapidly shifting landscape of cybersecurity threats and technological breakthroughs.

The process of developing a decentralized password manager is both gratifying and difficult in digital security. The main goal of this project was to provide users with a reliable and secure way to handle their sensitive credentials in a future where everything is connected.

The decentralized architecture of this password manager supports a more resilient and adaptable system in addition to increasing security. Giving them control over where and how their data is stored empowers them and promotes a sense of trust and ownership in the digital realm. In addition, the overall trustworthiness of the system is enhanced by the use of blockchain or other decentralized technologies that guarantee the transparency, immutability and integrity of stored credentials.

Decentralization turned out to be a crucial component in raising the password manager's overall security. We reduced the hazards connected with centralized databases and single points of failure by dispersing the password storage and retrieval across a network of nodes.

Users have control over their data when using a decentralized strategy. Since each user still retains ownership and access rights, the scales have tipped in favor of increased user empowerment and autonomy.

User privacy is given top priority in the architecture of our decentralized password manager. We've made great progress in protecting user data from unwanted access by reducing the visibility of personal information and implementing encryption techniques.

The collaborative nature of decentralized networks creates a strong security environment. The overall resilience of the system is supported by the combined strength of its participants, making it more resistant to many types of cyber attacks

REFERENCES

[1] Vaishnavi Nath Dornadula, S Geetha, “Credit Card Fraud Detection using Machine Learning Algorithms”,Procedia Computer Science,Volume 165,2019.

[2]A. Mahajan, V. S. Baghel and R. Jayaraman, "Credit Card Fraud Detection using Logistic Regression with Imbalanced Dataset," 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2023.

[3] W. -F. Yu and N. Wang, "Research on Credit Card Fraud Detection Model Based on Distance Sum," 2009 International Joint Conference on Artificial Intelligence, Hainan, China, 2009.

[4] B. Al Smadi and M. Min, "A Critical review of Credit Card Fraud Detection Techniques," 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2020.

[5] S. Mittal and S. Tyagi, "Performance Evaluation of Machine Learning Algorithms for Credit Card Fraud Detection," 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2019.

[6] J. R. D. Kho and L. A. Vea, "Credit card fraud detection based on transaction behaviour," TENCON 2017 - 2017 IEEE Region 10 Conference, Penang, Malaysia, 2017.

[7] Y. Sahin and E. Duman, "Detecting credit card fraud by ANN and logistic regression," 2011 International Symposium on Innovations in Intelligent Systems and Applications, Istanbul, Turkey, 2011.

[8] A. A. El Naby, E. El-Din Hemdan and A. El-Sayed, "Deep Learning Approach for Credit Card Fraud Detection," 2021 International Conference on Electronic Engineering (ICEEM), Menouf, Egypt, 2021.

[9] Dubey, Saurabh C., Ketan S. Mundhe and Aditya Kadam. “Credit Card Fraud Detection using Artificial Neural Network and BackPropagation.” 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS) (2020).

[10] G. E. Melo-Acosta, F. Duitama-Muñoz and J. D. Arias-Londoño, "Fraud detection in big data using supervised and semi-supervised learning techniques," 2017 IEEE Colombian Conference on Communications and Computing (COLCOM), Cartagena, Colombia, 2017.

[11] Melo-Acosta, German E., et al. “Fraud Detection in Big Data Using Supervised and Semi-Supervised Learning Techniques.” 2017 IEEE Colombian Conference on Communications and Computing (COLCOM), 2017.

[12] Mohammed, Emad, and Behrouz Far. “Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study.” IEEE Annals of the History of Computing, IEEE, 1 July 2018,

[13] Nilsonreport.Online:

https://nilsonreport.com/publication_newsletter_archive_issue.php?issue=1187

[14] Xuan, Shiyang, et al. “Random Forest for Credit Card Fraud Detection.” 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), 2018, doi:10.1109/icnsc.2018.8361343.

[15] Johnson Adeleke Adeyiga, Philip Gbounmi Toriola, Temitope Elizabeth Abioye(Ogunbiyi) et al. Fake News Detection Using a Logistic Regression Model and Natural Language Processing Techniques, 14 July 2023.

[16] Pumsirirat, A. and Yan, L. (2018). Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann Machine. International Journal of Advanced Computer Science and Applications, 9(1).

[17] Y. Li, X. Zhang, and D. Chen, “Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes,”

[18] Randhawa, Kuldeep, et al. “Credit Card Fraud Detection Using AdaBoost and Majority Voting.” IEEE Access, vol. 6, 2018.

[19]Johnson Adeleke Adeyiga, Philip Gbounmi Toriola, Temitope Elizabeth Abioye(Ogunbiyi) et al. Fake News Detection Using a Logistic Regression Model and Natural Language Processing Techniques, 14 July 2023.

[20] Nabila Farnaaz, M.A. Jabbar,"Random Forest Modeling for Network Intrusion Detection System", Procedia Computer Science, Volume 89, 2016.

[21] M. Shahin, M. Ali Babar and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," in IEEE Access, vol. 5.

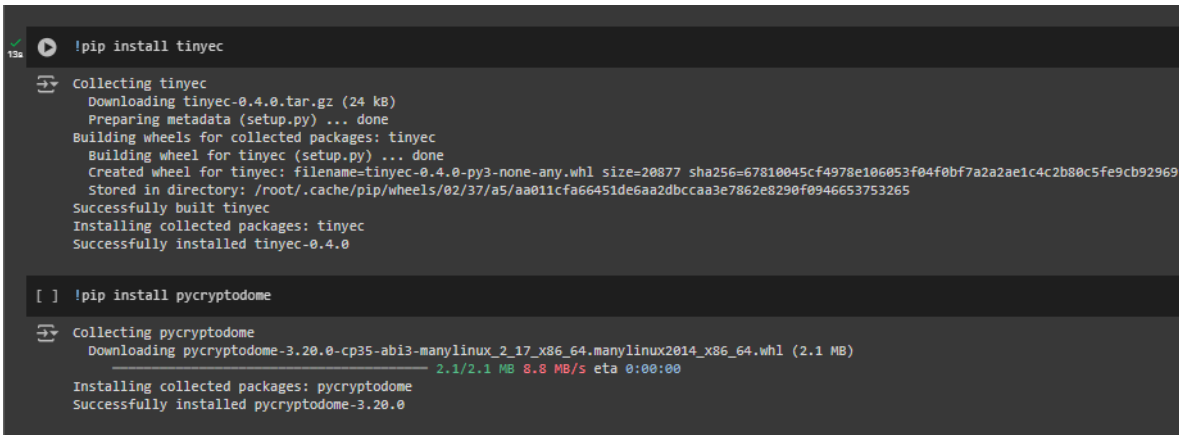
[22] Scikit-learn : machine learning in Python [Online]. https:// scikit- learn. org/ stable/

[23] Streamlit: A faster way to build and share data apps [Online].
https://streamlit.io/

[24] Deploying Machine Learning models with Streamlit, FastAPI by Rihab[Online].
https://rihab-feki.medium.com/deploying-machine-learning-models-with-
streamlit-f a stapi-and-docker

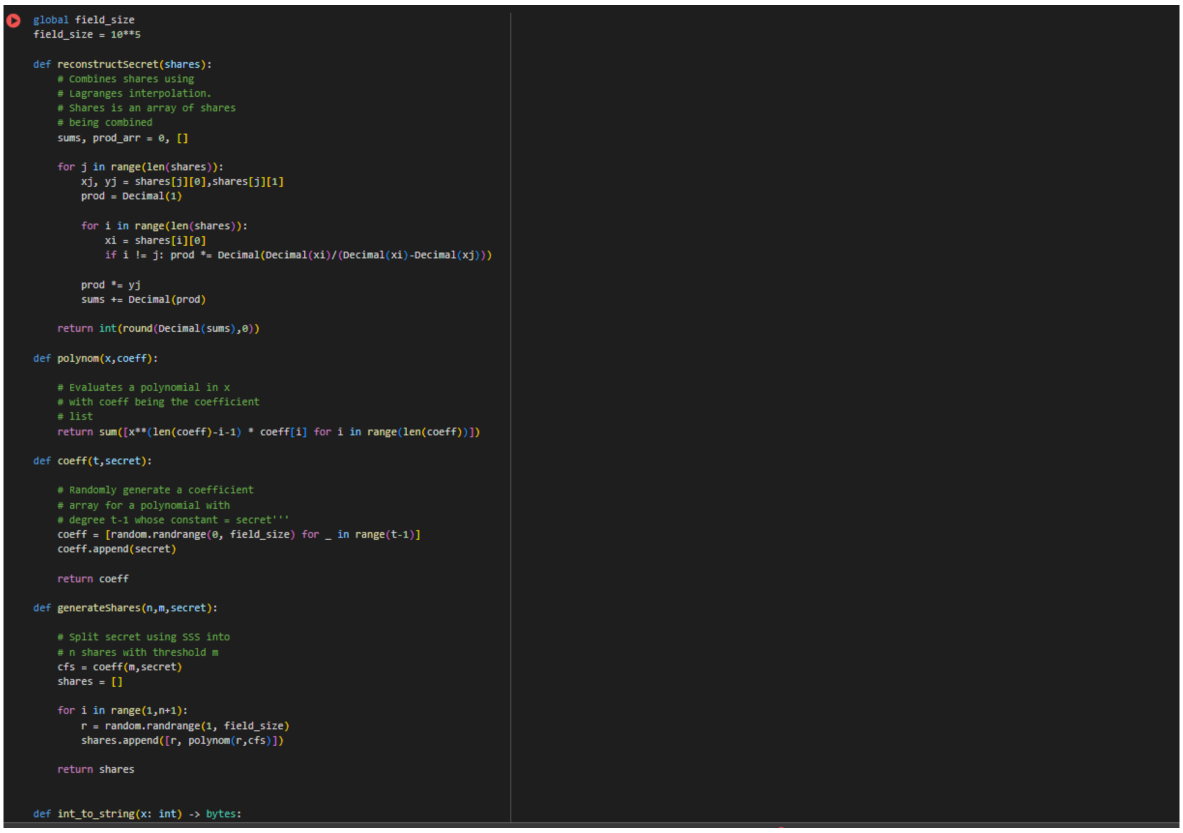
[25] Bankrate: Credit Card Fraud [Online].
www.bankrate.com/finance/credit-cards/credit-card-fraud-statics

APPENDIX

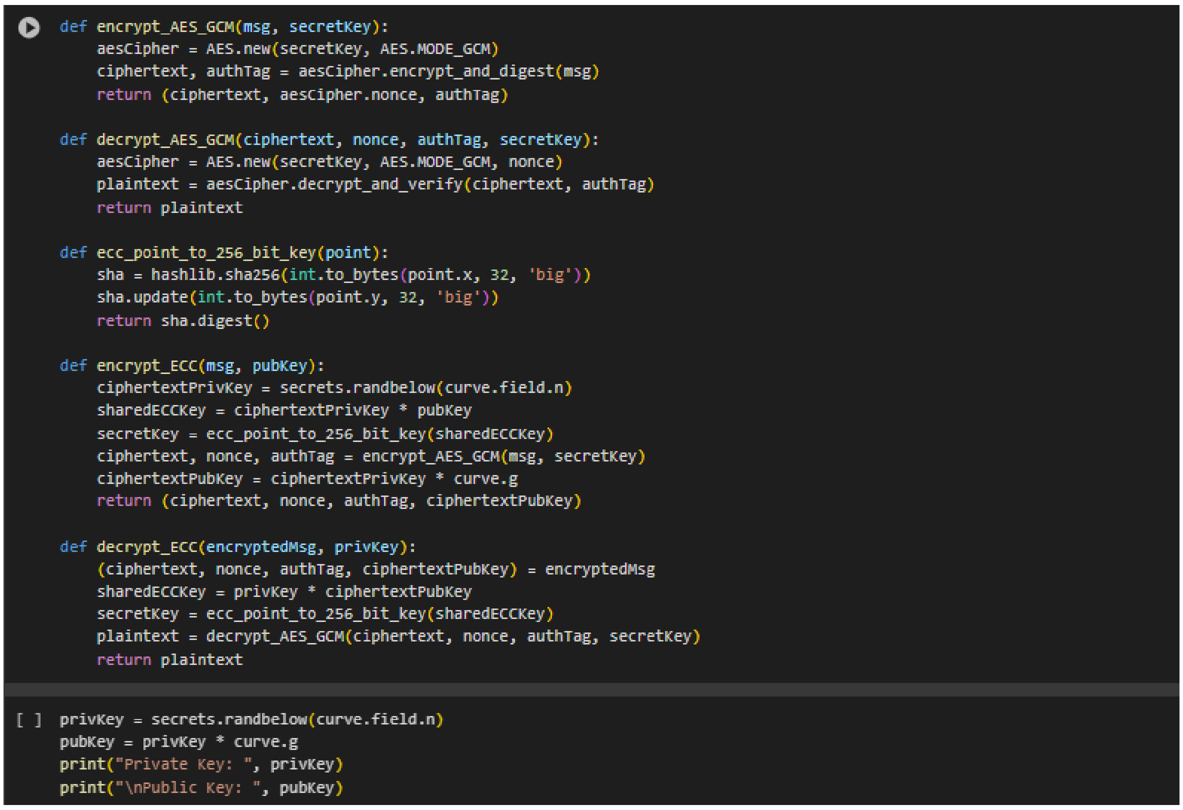


```
!pip install tinyec

Collecting tinyec
  Downloading tinyec-0.4.0.tar.gz (24 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: tinyec
  Building wheel for tinyec (setup.py) ... done
  Created wheel for tinyec: filename=tinyec-0.4.0-py3-none-any.whl size=20877 sha256=67810045cf4978e106053f04f0bf7a2a2ae1c4c2b80c5fe9cb92969
  Stored in directory: /root/.cache/pip/wheels/02/37/a5/aa011cfa66451de6aa2dbccaa3e7862e8290f0946653753265
Successfully built tinyec
Installing collected packages: tinyec
Successfully installed tinyec-0.4.0
```

```
[ ] !pip install pycryptodome

Collecting pycryptodome
  Downloading pycryptodome-3.20.0-cp35-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 8.8 MB/s eta 0:00:00
Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.20.0
```



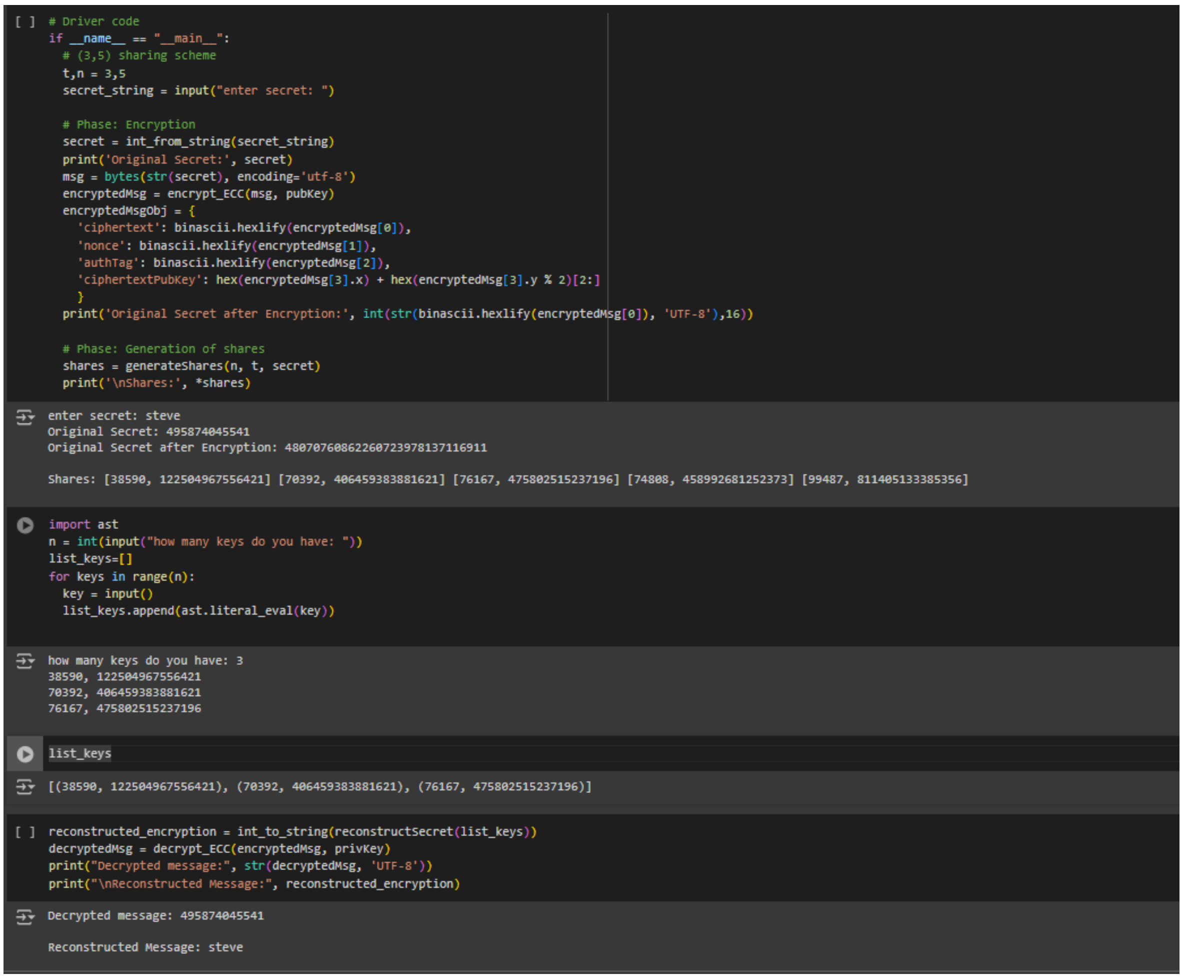
```python
global field_size
field_size = 10**5

def reconstructSecret(shares):
    # Combines shares using
    # Lagranges interpolation.
    # Shares is an array of shares
    # being combined
    sums, prod_arr = 0, []

    for j in range(len(shares)):
        xj, yj = shares[j][0],shares[j][1]
        prod = Decimal(1)

        for i in range(len(shares)):
            xi = shares[i][0]
            if i != j: prod *= Decimal(Decimal(xi)/(Decimal(xi)-Decimal(xj)))

        prod *= yj
        sums += Decimal(prod)

    return int(round(Decimal(sums),0))

def polynom(x,coeff):

    # Evaluates a polynomial in x
    # with coeff being the coefficient
    # list
    return sum([x**(len(coeff)-i-1) * coeff[i] for i in range(len(coeff))])

def coeff(t,secret):

    # Randomly generate a coefficient
    # array for a polynomial with
    # degree t-1 whose constant = secret'''
    coeff = [random.randrange(0, field_size) for _ in range(t-1)]
    coeff.append(secret)

    return coeff

def generateShares(n,m,secret):

    # Split secret using SSS into
    # n shares with threshold m
    cfs = coeff(m,secret)
    shares = []

    for i in range(1,n+1):
        r = random.randrange(1, field_size)
        shares.append([r, polynom(r,cfs)])

    return shares


def int_to_string(x: int) -> bytes:
```



```python
def encrypt_AES_GCM(msg, secretKey):
    aesCipher = AES.new(secretKey, AES.MODE_GCM)
    ciphertext, authTag = aesCipher.encrypt_and_digest(msg)
    return (ciphertext, aesCipher.nonce, authTag)

def decrypt_AES_GCM(ciphertext, nonce, authTag, secretKey):
    aesCipher = AES.new(secretKey, AES.MODE_GCM, nonce)
    plaintext = aesCipher.decrypt_and_verify(ciphertext, authTag)
    return plaintext

def ecc_point_to_256_bit_key(point):
    sha = hashlib.sha256(int.to_bytes(point.x, 32, 'big'))
    sha.update(int.to_bytes(point.y, 32, 'big'))
    return sha.digest()

def encrypt_ECC(msg, pubKey):
    ciphertextPrivKey = secrets.randbelow(curve.field.n)
    sharedECCKey = ciphertextPrivKey * pubKey
    secretKey = ecc_point_to_256_bit_key(sharedECCKey)
    ciphertext, nonce, authTag = encrypt_AES_GCM(msg, secretKey)
    ciphertextPubKey = ciphertextPrivKey * curve.g
    return (ciphertext, nonce, authTag, ciphertextPubKey)

def decrypt_ECC(encryptedMsg, privKey):
    (ciphertext, nonce, authTag, ciphertextPubKey) = encryptedMsg
    sharedECCKey = privKey * ciphertextPubKey
    secretKey = ecc_point_to_256_bit_key(sharedECCKey)
    plaintext = decrypt_AES_GCM(ciphertext, nonce, authTag, secretKey)
    return plaintext
```

```
[ ]
```

```python
privKey = secrets.randbelow(curve.field.n)
pubKey = privKey * curve.g
print("Private Key: ", privKey)
print("\nPublic Key: ", pubKey)
```



```
[ ]
```

```python
# Driver code
if __name__ == "__main__":
  # (3,5) sharing scheme
  t,n = 3,5
  secret_string = input("enter secret: ")

  # Phase: Encryption
  secret = int_from_string(secret_string)
  print('Original Secret:', secret)
  msg = bytes(str(secret), encoding='utf-8')
  encryptedMsg = encrypt_ECC(msg, pubKey)
  encryptedMsgObj = {
    'ciphertext': binascii.hexlify(encryptedMsg[0]),
    'nonce': binascii.hexlify(encryptedMsg[1]),
    'authTag': binascii.hexlify(encryptedMsg[2]),
    'ciphertextPubKey': hex(encryptedMsg[3].x) + hex(encryptedMsg[3].y % 2)[2:]
    }
  print('Original Secret after Encryption:', int(str(binascii.hexlify(encryptedMsg[0]), 'UTF-8'),16))

  # Phase: Generation of shares
  shares = generateShares(n, t, secret)
  print('\nShares:', *shares)
```

```
enter secret: steve
Original Secret: 495874045541
Original Secret after Encryption: 48070760862260723978137116911

Shares: [38590, 122504967556421] [70392, 406459383881621] [76167, 475802515237196] [74808, 458992681252373] [99487, 811405133385356]
```

```python
import ast
n = int(input("how many keys do you have: "))
list_keys=[]
for keys in range(n):
  key = input()
  list_keys.append(ast.literal_eval(key))
```

```
how many keys do you have: 3
38590, 122504967556421
70392, 406459383881621
76167, 475802515237196
```

```python
list_keys
```

```
[(38590, 122504967556421), (70392, 406459383881621), (76167, 475802515237196)]
```

```
[ ]
```

```python
reconstructed_encryption = int_to_string(reconstructSecret(list_keys))
decryptedMsg = decrypt_ECC(encryptedMsg, privKey)
print("Decrypted message:", str(decryptedMsg, 'UTF-8'))
print("\nReconstructed Message:", reconstructed_encryption)
```

```
Decrypted message: 495874045541

Reconstructed Message: steve
```

Praksh_Report

ORIGINALITY REPORT

13%	12%	4%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ir.juit.ac.in:8080 Internet Source	3%
2	www.ir.juit.ac.in:8080 Internet Source	2%
3	www.ijmtst.com Internet Source	1%
4	medium.com Internet Source	1%
5	Hema Shekhawat, Daya Sagar Gupta. "Quantum-resistance blockchain-assisted certificateless data authentication and key exchange scheme for the smart grid metering infrastructure", Pervasive and Mobile Computing, 2024 Publication	1%
6	ijritcc.org Internet Source	<1%
7	www.kaggle.com Internet Source	<1%