Project Report

Secure Banking Application and Password Manager Using Elliptic Curve Cryptography (ECC)

CSE1011 – Cryptography Fundamentals

Submitted by

17BCI0185 – Adrijeet Deb

17BCI0149 – Ankit Bando

17BCI0124 – Hritik Amar Shah

17BCI0079 – Amit Mathew

17BCI0184 – Ariyan Chowdhury

Under the guidance of

Prof. Madhu Viswanatham V.

Bachelor of Technology
in
Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

Introduction

Theoretical Background

Human beings are suspicious species by nature. It can be blamed on their curiosity or their intelligence. As an intelligent being, man thirsts to know. They try to pry into information if required. This kind of nosiness of man has led to some infamous scandals in history. The Watergate scandal, 2G housing scam, Wikileaks and many others are evidence of man's intrusiveness. So, to prevent this kind of intrusion human beings needed a tool and that tool is what is Cryptography.

Public Key Cryptography

Public key cryptography is a modern form of encryption that allows different parties to exchange information securely over an insecure network without first agreeing on secret keys. The primary use of public key cryptography is to provide information security in computer science, such as securely transmitting email, credit card details or other secret information between senders and receivers over the Internet.Maintaining the Integrity of the Specifications

Securely transferring information from person A to person B over an insecure network involves three steps. These are

The encryption of the original information, called plaintext, the transmission or ciphertext of the encrypted message, and the decryption of the ciphertext back to the plaintext. Since the ciphertext is transmitted over an insecure network, any spy can access the ciphertext, so the original information can be accessed if he can decrypt the message. Therefore, a successful cryptosystem must be able to encrypt the original message in such a way that only the intended recipient can decrypt the ciphertext. The purpose of public key cryptography is to make it difficult to decrypt encrypted messages in a reasonable amount of time (by brute force) unless certain key facts are known. Ideally, only the intended sender and receiver of the message should know these specific key facts.

Any information that is necessary to decrypt a message is called a key. The key specifies a specific function that converts the original message to ciphertext and vice versa. Public key encryption relies on two keys, a private and public keys. The public key is published where anyone can access it. However, each person or computer will also choose a private key that only the individual knows. The importance of the key pair encryption algorithm should be so great that it is computationally infeasible to recover the original message from the encrypted data if the key is lost.

In 1976, Diffie and Hellman proposed using some one-way function called the trapdoor function, making it almost impossible to decrypt encrypted data without a key.

Elliptic Curve Cryptography (ECC)

In 1985, Neal Koblitz and Victor Miller independently suggested the use of elliptic curves in public key cryptography. ECC requires a much smaller key than that used in traditional public key cryptosystems while maintaining the same level of security. Therefore, the use of elliptic curves allows faster encryption and decryption.

Strictly speaking, it is not a new type of cryptosystem, but an elliptic curve version of the existing cryptosystem; but because of its many unique advantages, it has been studied separately from the beginning. It is widely believed that the elliptic curve cryptosystem will become the most important public key cryptosystem in the 21st century.

Elliptic curve public key cryptography is a very difficult premise based on the elliptic curve discrete logarithm problem; in fact, for multiplicative groups over a finite field, the discrete logarithm function is much farther. As mentioned earlier, groups are often used for public key encryption as the domain in which we define encryption functions. This is because we need each element of the domain to have an inverse and vice versa. In elliptic curve cryptography, the group used is the set of points on a given elliptic curve.

This is how elliptic curve public key encryption works.

In order for Alice and Bob to communicate securely over an insecure network, they can exchange private keys on the network in the following ways:

1. A specific rational base point P is published in the public domain for use with a specific elliptic curve E (Fq) published in the public domain.

2. Alice and Bob select random integers kA and kB, respectively, which are used as private keys.

3. Alice calculates kA*P, Bob calculates kB*P and they exchange these values on an insecure network.

4. Using the information they receive from each other and their private key, both Alice and Bob calculate (kA * kB) * P = kA*(kB*P) = kB*(kA* P). This value is only the shared secret of Alice and Bob.

Since Alice and Bob share secrets that third parties can hardly find, they can use this shared secret.

Motivation

Nowadays storage of passwords has become a hassle. With the increase of social media sites and online services, the use of passwords has increased. It would be stupid to use a universal password for all purposes. So, the solution to the problem is using a password manager. It would solve the issues regarding this matter. Nowadays plenty of password managers have been released for public use but none of them is free e.g.: McAfee True Key, LastPass, Kaspersky Password Manager. All of them require you to pay a monthly or yearly subscription fee. It is very expensive for people living in third world countries. So, we were motivated to make a password manager as a freeware and release it to the public so that people wouldn't have to rely on subscription-based software to store their passwords.

Another application our project is securing online transactions. Another application our project is securing online transactions. Banks have a lot of security so that they are able to protect their clients' data and their personal data. It is very important because the data stored by any bank is very sensitive. It can be used to exploit or extort money from famous businessmen. Bank should keep an eye on their employees and have security checks in place to not allow unauthorized or unrestricted access to the database. So, a simulation of how banks encrypt data and allow access to their employees and users based on their designation will be showed in our project and we'll be using ECC specifically ECIES will be used for encryption.

Problem/Proposed Statement:

Passwords have become one of the most reliable ways to secure our data and entity authentication. With the rise of online websites and how we've become depended on online services, it is not possible to generate unique passwords for each website and then remember then accordingly. Our password manager will capture passwords from the website where the user will insert his login details and then encrypt it using a master password. One master password will allow the user to access all his passwords at once. User can manually enter his/her login details personally. An extension for Google Chrome made by us will allow the user to capture the passwords automatically when he/she logs in or signs up to a website.

Our program will encrypt data and give access to bank employees based on their class level access on their accounts. That will allow the bank to provide access to their employees according to their designations. Not all employees will have unrestricted access. Information displayed to them will be encrypted according to what they are permitted to see by the organization. This will be based on the designation of the employee at the organization. A low-level employee cannot access information that can only be seen by a higher-level employee.

Aim of the Proposed Work:

Aim of our project is to provide a free password manager to users around the world. This will allow users to have a free password manager and they won't have to subscribe to other password managers.

The aim of the secure banking application is to provide an idea to common folk about how their bank transactions are secured from breaches from hackers and other attackers.

Objectives of the Proposed work:

The objectives of password manager:

1. Provide a free password manager for common users.
2. Free people from the clutches of capitalist industries like McAfee, LastPass, etc. that are profiting from the need of a password manager by people.
3. Allow users to login securely and generate pseudo-random passwords.
4. Maintain a database of passwords for the user and encrypt it using ECIES.

The objectives of secure banking application:

1. Implements class level-based access for viewing of users' personal records.
2. Gives an idea about how transactions are secured in a bank.
3. Allows the user to encrypt his/her personal data using ECIES.
4. Maintains a record of the users' transactions and provides security for it.

These are the primary objectives of our 2 applications using ECC. We have multiple secondary objectives that won't be in the project but planned to be added to our applications later.

Literature Survey

Survey Of Existing Models

1.
   Syeda Farha Shazmeen , Shyam Prasad in their research paper **" A Practical Approach for Secure Internet Banking based on Cryptography"** International Journal of Scientific and Research Publications propose a challenge/response -based short-time password authentication methods using Elliptic curve cryptography in combination with Software Security model.

Gaps identified in the survey:

In this approach bank hides customer transaction data is secure using ECIES algorithm. Customer application decrypts data in secure manner the encryption and decryption are characterized by private key. They use SIM application toolkit where bank can stored the application and encryption keys on SIM.

Link:

Syeda Farha Shazmeen , Shyam Prasad in their research paper " A Practical Approach for Secure Internet Banking based on Cryptography" International Journal of Scientific and Research Publications.
http://www.ijsrp.org/research-paper-1212/ijsrp-p12122

2.

Ehab M. Alkhateeb, Mohammad A. Alia, Adnan A. Hnaif in their research paper **"**The Generalised Secured Mobile Payment System Based on ECIES**"** International Journal of Scientific and Research Publications propose Elliptic Curve Integrated Encryption Scheme ECIES.

Gaps identified in the survey:

In this paper Elliptic curve cryptography is used to mobile payment system. Elliptic Curve Integrated Encryption Scheme ECIES cryptographic protocols have been applied to enhance the security of the proposed mobile payment system. However, the proposed system is secure, easy and straightforward payment process. Elliptic curve cryptography is actually adopted to its efficiency and requires small key size comparing to RSA cryptosystem with the same security level. As well as, USSD technology is used in this system for PIN authentication process with high security performance.

Link:

Ehab M. Alkhateeb, Mohammad A. Alia, Adnan A. Hnaif in their research paper "The Generalised Secured Mobile Payment System Based on ECIES" International Journal of Scientific and Research Publications.

https://www.researchgate.net/publication/300715543_The_Generalised_Secured_Mobile_Payment_System_Based_on_ECIES_and_ECDSA

3.

Ruchika Markan, Gurvinder Kaur in their research paper **"Literature Survey on Elliptic Curve Encryption Techniques"** International Journal of Scientific and Research Publications propose Elliptic Curve Architectures which is based on the arithmetic of elliptic curves and discrete logarithmic problems.

Gaps identified in the survey:

In this paper they are using ECC schemes which are public-key based mechanisms that provide encryption, digital signatures and key exchange algorithms. The best known encryption scheme is the Elliptic Curve Integrated Encryption Scheme (ECIES) which is included in IEEE and also in SECG SEC 1 standards. ECC provides greater security and more efficient performance than the first generation public key techniques. Even though ECIES provides some valuable advantages over other cryptosystems as RSA, the number of slightly different versions of ECIES included in the standards may obstruct the adoption of ECIES.

Link :

Ruchika Markan, Gurvinder Kaur in their research paper "Literature Survey on Elliptic Curve Encryption Techniques" International Journal of Scientific and Research Publications
http://ijarcsse.com/Before_August_2017/docs/papers/Volume_3/9_September2013/V3I9-0358

4.
Ehab M. Alkhateeb, Mohammad A. Alia, and Adnan A. Hnaif Kaur in their research paper **"An Improved Approach for Secured Mobile Payment System based on ECIES Cryptography"** International Journal of Scientific and Research Publications propose a new approach to mobile payment system based on Elliptic Curve cryptography ECC.

Gaps identified in the survey:

The proposed mobile payment system includes three main processes: Authentication process, Member recognition process, and Payment process. Furthermore, Elliptic Curve Integrated Scheme ECIES cryptographic protocol has been applied to enhance the security of the proposed mobile payment system.

Link :

Ehab M. Alkhateeb, Mohammad A. Alia, and Adnan A. Hnaif Kaur in their research paper "An Improved Approach for Secured Mobile Payment System based on ECIES Cryptography" International Journal of Scientific and Research Publications.

http://www.academia.edu/12491086/An_Improved_Approach_for_Secured_Mobile_Payment_System_based_on_Ecies_Cryptography

5. Setiadi, I., Kistijantoro, A. I., & Miyaji, A. (2015) in their research paper "Elliptic curve cryptography: Algorithms and implementation analysis over coordinate systems" *2015 2nd International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*. doi:10.1109/icaicta.2015.7335349,proposes a step-by-step tutorial to transform ECC over prime field GF(p) from mathematical concept to the software implementation.

Gaps identified in the survey: This paper contains a step-by-step guide for implementing ECC over Primefield
GF(p) from mathematical concept to the software implementation. This paper also provides several alternatives and trade-off between different coordinate systems in computational process.

Link: https://freedomofkeima.com/files/Elliptic%20Curve%20Cryptography%20-%20Algorithms%20and%20%20Implementation%20Analysis%20over%20Coordinate%20Systems.pdf

6. Jiang, L., Liu, N., Wang, Y., & Guo, D.(2013) in their research paper ” Remote mutual password authentication for online payment with the elliptic curve cryptography based key agreement” *2013 International Conference on Information and Network Security (ICINS 2013)*. doi:10.1049/cp.2013.2463,proposes a remote mutual password authentication scheme specially designed for third-party payment platform via elliptic curve cryptography.

Gaps identified in the survey:
This paper proposes a mutual password authentication scheme specifically designed for third party platforms using elliptic curve cryptography, and a third-party payment model that uses the excellent capabilities of the proposed scheme to ensure the security of online payment. Meanwhile, rigorous cryptanalysis is done in order to ensure the good characteristics of the proposed scheme in resisting various attacks.

Link: https://ieeexplore.ieee.org/document/6826012

7. Laiphrakpam Dolendro Singh and Khumanthem Manglem Singh (2015) in their research paper ”Implementation of Text Encryption using Elliptic Curve Cryptography” proposes a new technique where the classic technique of mapping the characters to affine points in the elliptic curve has been removed.
Gaps identified in the survey: Elliptic Curve Cryptography has been a recent area of research in the field of
cryptography. It provides more security with smaller key size compared to other cryptographic methods. This paper proposes a new technique in which the classic method of mapping characters to affine points in an elliptic curve has been removed. Relevant ASCII Values of the plain text is paired up. The paired values serve as input for elliptic curve cryptography. This new method avoids expensive mapping work and the need to share a common lookup table between sender and recipient. The algorithm designed in such a way that it can be used to encrypt or decrypt any type of script with certain ASCII values.

Link: https://www.sciencedirect.com/science/article/pii/S1877050915013332?via%3

8. V. Gayoso Martínez, L. Hernández Encinas, and C. Sánchez Ávila
(2010) in their research paper ” , JOURNAL OF COMPUTER SCIENCE AND ENGINEERING, VOLUME 2, ISSUE 2,A Survey of the Elliptic Curve Integrated Encryption Scheme” offer a comprehensive introduction to ECIES, detailing the encryption and decryption procedures and the list of functions and special characteristics included in aforementioned standards.
Gaps identified in the survey: Elliptic curve cryptographic schemes are encryption, digital signature and key
exchange mechanisms. The most famous encryption scheme is based on at ECC – Elliptic Curve Integrated Encryption Scheme(ECIES) included in ANSI X9.63, ISO /

IEC 18033-2, IEEE 1363a and SECG SEC 1.In this paper a comprehensive Introduction for ECIES is given, which describes in detail encryption and decryption procedures,as well as a list of functions and special functions included in the aforementioned standards.
Link:https://www.sciencedirect.com/science/article/pii/S1877050915013332?via%3 Dihub

9. **S. Nithya, Dr. E. George Dharma Prakash Raj in their research paper "ANALYSIS OF ELLIPTIC KEY ALGORITHMS"** International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) propose a new approach to mobile payment system based on Elliptic Curve cryptography ECC.

Gaps identified in the survey:
In this paper, the elliptic curve integrated encryption system (ECIES) is the standard elliptic curve based on encryption algorithm. It is a hybrid scheme that uses a public key system to transport a session key for the use of symmetric cipher. It generates a random number for public key generation and enhances security for public key generation by converting the public key into a cipher text.
Link :
S. Nithya, Dr. E. George Dharma Prakash Raj in their research paper "ANALYSIS OF ELLIPTIC KEY ALGORITHMS" International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3 Issue 8, August 2014
http://ijarcet.org/wp-content/uploads/IJARCET-VOL-3-ISSUE-8-2690-2694

Overview of the Proposed System

Introduction and Related Concepts

Key generation: The parameters that must be agreed upon are the elliptic curve parameters: p (prime no.) as the finite field size, a and b as the curve parameters, G as the generator point, n as the order of G, and h as the cofactor.

Let P be the private key of the user. Then, the public key Q can be determined by the expression:

$$\mathbf{Q=[P].G}$$

Encryption Scheme: Algorithm used- ECIES (Elliptic Curve Integrated Encryption Scheme):

INPUT: Q where Q is the public key of other party
m where m is the intended message

OUTPUT: e where e is the encrypted message

C (x,y) where C is the chosen point

```
ALGORITHM:
1. Choose random value k [1,n-1]
2. C = [k].G
3. R = [k].Q
4. e = ($R_x$*m) modulo p
5. Return (e,C)
Decryption Scheme:
INPUT: e where e is the encrypted message
       C where C is the chosen point
       $P_k$ where $P_k$ is the private key of receiver
OUTPUT: d where d is the decoded message
ALGORITHM:
1. R = [$P_k$].C
2. d = (e*$(R_x)^{-1}$) modulo p
3. Return(d)
```

Framework, Architecture for the Proposed System

a) Password Manager- In this application, the user has to input a master password which becomes his private key and all the passwords which are stored by the user are encrypted using the public key which is calculated from the user's private key. Whenever the user wants to retrieve any of his stored passwords he enters the private key (i.e. the master password) which decrypts the password details in the stored database and thus the user can access the password details. Any other attacker trying to access the password database cannot do so because it can be decrypted only through the user's private key.

b) Secure Bank Application- In this application, there is a transaction log where we can see the details of the transactions of various users like sender's, receiver's id, net balances, the amount sent and the date and the time of the transaction. Here some of the private details of the users like the amount of transaction, net balance will be in encrypted form to protect the privacy of the financial details of the users. While, the users can access their details and perform secure transactions using their private keys which they enter as their bank PINs.

Results and Analysis:

Password manager source code

main.py

```
from PyQt4 import QtGui

from PyQt4 import QtCore

from Crypto.Cipher import DES3

from Crypto import Random

from transform import convertpass, convertToMUL8, convertToOriginal

import sys, sqlite3, hashlib


class StartWindow(QtGui.QMainWindow):

    def __init__(self, parent = None):

        super(StartWindow, self).__init__(parent)

        self.setGeometry(750, 450, 400, 300)

        self.setWindowTitle("Password Manager")

        self.LoginPage()
```

```
def LoginPage(self):
    self.TextPlacer1 = QtGui.QLabel(self)
    self.TextPlacer1.setText("User Name")
    self.TextPlacer1.resize(self.TextPlacer1.sizeHint())
    self.TextPlacer1.move(100, 100)
    self.TextPlacer2 = QtGui.QLabel(self)
    self.TextPlacer2.setText("Password")
    self.TextPlacer2.resize(self.TextPlacer2.sizeHint())
    self.TextPlacer2.move(100, 140)
    self.UserField = QtGui.QLineEdit(self)
    self.UserField.resize(self.UserField.sizeHint())
    self.UserField.move(180, 100)
    self.PassField = QtGui.QLineEdit(self)
    self.PassField.resize(self.PassField.sizeHint())
    self.PassField.move(180, 140)
    self.PassField.setEchoMode(QtGui.QLineEdit.Password)
    self.loginbtn = QtGui.QPushButton("Login", self)
    self.loginbtn.resize(self.loginbtn.sizeHint())
    self.loginbtn.move(180, 170)
    self.CreateNewAcc = QtGui.QPushButton("Create New Account", self)
    self.CreateNewAcc.resize(self.CreateNewAcc.sizeHint())
    self.CreateNewAcc.move(180, 200)
    self.ExitBtn = QtGui.QPushButton("Exit", self)
    self.ExitBtn.resize(self.ExitBtn.sizeHint())
```

```
        self.ExitBtn.move(180, 230)

        self.ExitBtn.clicked.connect(self.close_application)

        self.CreateNewAcc.clicked.connect(self.on_pushbutton_clicked)

        self.QDialog = NewApplicationWindow(self)

        self.loginbtn.clicked.connect(self.on_pushbutton_login)

        self.show()


    def close_application(self):

        sys.exit()


    def on_pushbutton_clicked(self):

        self.QDialog.show()


    def on_pushbutton_login(self):

        self.m = hashlib.sha512()

        self.m.update(self.PassField.text().encode())

        self.conn = sqlite3.connect('Data.db')

        self.c = self.conn.cursor()

        data = self.c.execute("SELECT * FROM USERS")

        for row in data:

            if row[1] == self.UserField.text() and row[2] == self.m.digest():

                self.QDialog = Login(self, row[0], self.PassField.text())

                self.QDialog.show()
```

```
class NewApplicationWindow(QtGui.QDialog):


    def __init__(self, parent = None):
        super(NewApplicationWindow, self).__init__(parent)
        self.createFormGroupBox()
        buttonBox = QtGui.QDialogButtonBox(QtGui.QDialogButtonBox.Ok |
QtGui.QDialogButtonBox.Cancel)
        buttonBox.accepted.connect(self.accepted)
        buttonBox.rejected.connect(self.reject)


        mainLayout = QtGui.QVBoxLayout()
        mainLayout.addWidget(self.formGroupBox)
        mainLayout.addWidget(buttonBox)


        self.setLayout(mainLayout)
        self.setWindowTitle("Form Layot")


    def createFormGroupBox(self):
        self.formGroupBox = QtGui.QGroupBox("Enter Details")
        layout = QtGui.QFormLayout()
        self.Username = QtGui.QLineEdit()
        self.Password = QtGui.QLineEdit()
        self.Password.setEchoMode(QtGui.QLineEdit.Password)
        layout.addRow(QtGui.QLabel("UserName:"), self.Username)
        layout.addRow(QtGui.QLabel("Password:"), self.Password)
        self.formGroupBox.setLayout(layout)


    def accepted(self):

        self.m = hashlib.sha512()

        self.m.update(self.Password.text().encode())

        self.conn = sqlite3.connect("Data.db")

        self.c = self.conn.cursor()

        self.c.execute("INSERT INTO USERS(USERNAME, PASSWORD) VALUES(?, ?)",
[(self.Username.text()), (self.m.digest())])

        self.conn.commit()

        print("Successfully Added the account.")

        self.accept()


class Login(QtGui.QMainWindow):

    def __init__(self, parent = None, id = 0, passwordString = None):

        super(Login, self).__init__(parent)

        self.passwordStringLocal = passwordString

        self.setGeometry(650, 450, 800, 600)

        self.id = id

        self.setWindowTitle("Password Manager")

        self.conn = sqlite3.connect('Data.db')

        self.c = self.conn.cursor()

        self.profile()


    def profile(self):
        self.Add = QtGui.QPushButton("New", self)
        self.Add.resize(self.Add.sizeHint())
        self.Add.move(100, 200)
        self.Logout = QtGui.QPushButton("Logout", self)
        self.Logout.resize(self.Logout.sizeHint())
        self.Logout.move(100, 300)
        self.Logout.clicked.connect(self.pushLogout)
        self.Add.clicked.connect(self.AddIndex)
        self.view = QtGui.QPushButton("Account List", self)
        self.view.resize(self.view.sizeHint())
        self.view.move(100, 250)
        self.view.clicked.connect(self.AccountList)
        self.show()

    def AccountList(self):
        self.QDialog = passwordList(self, self.id, self.passwordStringLocal)
        # self.QDialog.show()

    def pushLogout(self):
        sys.exit()

    def AddIndex(self):
        self.QDialog = AddPassword(self, self.id, self.passwordStringLocal)
        self.QDialog.show()
```

```
class AddPassword(QtGui.QDialog):
    def __init__(self, parent = None, id = 0, passStringLocalAdd = None):
        super(AddPassword, self).__init__(parent)
        self.id = id
        self.passStringAdd = passStringLocalAdd
        self.createFormGroupBox()
        buttonBox = QtGui.QDialogButtonBox(QtGui.QDialogButtonBox.Ok |
QtGui.QDialogButtonBox.Cancel)
        buttonBox.accepted.connect(self.accepted)
        buttonBox.rejected.connect(self.reject)

        mainLayout = QtGui.QVBoxLayout()
        mainLayout.addWidget(self.formGroupBox)
        mainLayout.addWidget(buttonBox)

        self.setLayout(mainLayout)
        self.setWindowTitle("Add Password")

    def createFormGroupBox(self):
        self.formGroupBox = QtGui.QGroupBox("New Account Details")
        layout = QtGui.QFormLayout()
        self.Username = QtGui.QLineEdit()
        self.Password = QtGui.QLineEdit()
        self.Password.setEchoMode(QtGui.QLineEdit.Password)
        self.Company = QtGui.QLineEdit()

        layout.addRow(QtGui.QLabel("UserName/Email"), self.Username)

        layout.addRow(QtGui.QLabel("Password:"), self.Password)

        layout.addRow(QtGui.QLabel("Company:"), self.Company)

        self.formGroupBox.setLayout(layout)


    def accepted(self):

        self.conn = sqlite3.connect("Data.db")

        self.c = self.conn.cursor()

        key = convertpass(self.passStringAdd)

        iv = Random.new().read(DES3.block_size)

        cipher_encrypt = DES3.new(key, DES3.MODE_OFB, iv)

        passwordpadded = convertToMUL8(self.Password.text())

        self.c.execute("INSERT INTO PASSES VALUES(?, ?, ?, ?, ?)", [(self.id),
(self.Company.text()), (self.Username.text()),
(cipher_encrypt.encrypt(convertToMUL8(self.Password.text()))), (iv)])

        print("A New Password FIELD ADDED.")

        self.conn.commit()

        self.accept()




class passwordList(QtGui.QDialog):

    def __init__(self, parent = None, id = 0, passString = None):

        super(passwordList, self).__init__(parent)

        self.passStringLocal = passString

        self.table = QtGui.QTableWidget()
        self.tableItem = QtGui.QTableWidgetItem()
        self.table.setWindowTitle("List of Accounts")
        self.table.resize(800, 600)
        self.table.setColumnWidth(100,100)
        self.conn = sqlite3.connect("Data.db")
        self.c = self.conn.cursor()
        data = self.c.execute("SELECT * FROM PASSES WHERE ID = ?;",[(id)])
        self.counter = 0
        extData = []
        for i in data:
            self.counter += 1
            extData.append(i)
        self.table.setRowCount(self.counter)
        self.table.setColumnCount(3)
        self.table.setColumnWidth(0, 200)
        self.table.setColumnWidth(1, 200)
        self.table.setColumnWidth(2, 200)
        self.table.setHorizontalHeaderLabels(["Company", "Email/Username", "Password"])
        key = convertpass(self.passStringLocal)
        for i in range(self.counter):
            iv = extData[i][4]
            cipher_decrypt = DES3.new(key, DES3.MODE_OFB, iv)
            self.table.setItem(i, 0, QtGui.QTableWidgetItem(extData[i][1]))
            self.table.setItem(i, 1, QtGui.QTableWidgetItem(extData[i][2]))
```

```
            self.table.setItem(i, 2,
QtGui.QTableWidgetItem(convertToOriginal(cipher_decrypt.decrypt(extData[i][3]).decode())))
        self.table.show()




def run():
    app = QtGui.QApplication(sys.argv)
    GUI = StartWindow()
    sys.exit(app.exec_())

run()
```

Transform.py

```
def convertpass(plaintext):
    key = [plaintext[0]]
    counter = 1
    i = 0
    while counter != 16:
        key.append(plaintext[(i+i//2)%len(plaintext)])
        i += 9
        counter += 1
    return "".join(key)


def convertToMUL8(m):
    length = len(m)
    spaces = length % 8
    spaces = 8 - spaces
    return m + " "*spaces


def convertToOriginal(ciphertext):
    return ciphertext.rstrip()
```

CreateDB.py

```
import sqlite3



if __name__ == "__main__":
    conn = sqlite3.connect("Data.db")
    c = conn.cursor()
    c.execute("CREATE TABLE USERS(ID INTEGER PRIMARY KEY AUTOINCREMENT,
USERNAME TEXT, PASSWORD TEXT);")
    c.execute("CREATE TABLE PASSES(ID INTEGER, APPLICATION TEXT, USERNAME
TEXT, PASSWORD TEXT, IV TEXT);")
    print("DataBase Created.")
```

Banking simulation source code

app.py

```
import sys

from PyQt4 import QtGui

import sqlite3

import hashlib

import seccure

import datetime




class Window(QtGui.QMainWindow):

    def __init__(self):

        super(Window, self).__init__()

        self.setWindowTitle("Bank Simulation")

        self.resize(400, 300)

        self.move(700, 400)

        self.Controls()

        self.show()


    def Controls(self):

        self.UserName = QtGui.QLineEdit(self)

        self.Password = QtGui.QLineEdit(self)
        self.Password.setEchoMode(QtGui.QLineEdit.Password)
        self.UserName.move(180, 100)
        self.Password.move(180, 150)
        self.TextUser = QtGui.QLabel("UserName: ", self)
        self.TextPass = QtGui.QLabel("Password: ", self)
        self.TextUser.move(100, 100)
        self.TextPass.move(100, 150)
        self.ExitButton = QtGui.QPushButton("Exit", self)
        self.ExitButton.move(150, 250)
        self.ExitButton.clicked.connect(self.on_click_exit)
        self.LoginButton = QtGui.QPushButton("Login", self)
        self.LoginButton.move(200, 200)
        self.LoginButton.clicked.connect(self.on_click_login)
        self.SignUpButton = QtGui.QPushButton("Sign Up", self)
        self.SignUpButton.move(100, 200)
        self.SignUpButton.clicked.connect(self.on_click_signup)

    def on_click_signup(self):
        self.SignUp = SignUpWindow()


    def on_click_exit(self):
        sys.exit()


    def on_click_login(self):
        self.conn = sqlite3.connect("BankDB.db")
        self.c = self.conn.cursor()
        self.h = hashlib.sha512()
        self.h.update(self.Password.text().encode())
        self.hash = self.h.digest()
        for i in self.c.execute("SELECT * FROM USERS;"):
            if i[1] == self.UserName.text() and i[2] == self.hash:
                self.Login = LoginedScreen(i[3])
                self.close()
class LoginedScreen(QtGui.QMainWindow):
    def __init__(self, accID):
        super(LoginedScreen, self).__init__()
        self.accID = accID
        self.resize(400, 400)
        self.move(600, 250)
        self.setWindowTitle("Welcome {}".format('USER'))
        self.ButtonTL = QtGui.QPushButton("Transaction Log", self)
        self.ButtonTL.clicked.connect(self.on_click_TL)
        self.ButtonTL.resize(self.ButtonTL.sizeHint())
        self.ButtonTF = QtGui.QPushButton("Transfer Funds", self)
        self.ButtonTF.move(100, 300)
        self.ButtonTL.move(100, 200)
        self.ButtonTF.resize(self.ButtonTF.sizeHint())
        self.ButtonTF.clicked.connect(self.on_click_TF)
        self.ButtonExit = QtGui.QPushButton("Login Out", self)
        self.ButtonExit.resize(self.ButtonExit.sizeHint())
        self.ButtonExit.move(100, 350)
        self.ButtonExit.clicked.connect(self.on_click_exit)
        self.TotalBalance = QtGui.QPushButton("Net Balance", self)
        self.TotalBalance.resize(self.TotalBalance.sizeHint())
        self.TotalBalance.move(100, 150)
        self.TotalBalance.clicked.connect(self.on_click_net_balance)
        self.show()

    def on_click_net_balance(self):
        self.NetBalanceScreen = NetBalanceScreen(self.accID)

    def on_click_exit(self):
        sys.exit()

    def on_click_TF(self):
        self.TF = TransferFundsWindow(self.accID)
    def on_click_TL(self):
        self.TL = Transaction_log(self.accID)

def hashTransaction(time, accID):
    hash = hashlib.sha512()
    time = str(time).split(':')
    time = ''.join(time) + str(accID)
    hash.update(time.encode())
    return hash.digest()
```

```
class TransferFundsWindow(QtGui.QMainWindow):
    def __init__(self, accID):
        super(TransferFundsWindow, self).__init__()
        self.resize(400, 400)
        self.accID = accID
        self.move(700, 400)
        self.LabelSendTo = QtGui.QLabel("Send To:", self)
        self.LabelAmount = QtGui.QLabel("Amount:", self)
        self.LabelSendTo.move(145, 100)
        self.LabelAmount.move(145, 200)
        self.ReceiversID = QtGui.QLineEdit(self)
        self.Amount = QtGui.QLineEdit(self)
        self.ReceiversID.move(200, 100)
        self.Amount.move(200, 200)
        self.ButtonSend = QtGui.QPushButton("Send", self)
        self.BUttonCancel = QtGui.QPushButton("Cancel", self)
        self.ButtonSend.move(100, 300)
        self.BUttonCancel.move(200, 300)
        self.BUttonCancel.clicked.connect(self.on_click_cancel)
        self.ButtonSend.clicked.connect(self.on_click_send)
        self.show()


    def on_click_send(self):
        self.conn = sqlite3.connect('BankDB.db')
        self.c = self.conn.cursor()

        self.date = datetime.datetime.now()

        self.c.execute('INSERT INTO SAMPE VALUES(?, ?, ?, ?, ?, ?, ?);', [(self.accID),
(self.accID), (self.ReceiversID.text()), (seccure.encrypt(self.Amount.text().encode(),
b'8W;>i^H0qi|J&$coR5MFpR*Vn')) , (str(self.date.time())), (str(self.date.date())),
(hashTransaction(self.date.time(), self.accID))])

        print("Successfully Transfered")

        # self.fetchdata = self.c.execute('SELECT NETBALANCE FROM USERS where
ACCOUNT_ID = ?', [(self.accID)])

        # for i in self.fetchdata:

        #     self.intbalnow = i[0]

        # self.c.execute('UPDATE USERS SET NETBALANCE = ? WHERE ACCOUNT_ID =
?',[(self.intbalnow - int(self.Amount.text())), (self.accID)])

        self.conn.commit()

        self.close()


    def on_click_cancel(self):

        self.close()






class Transaction_log(QtGui.QTableWidget):

    def __init__(self, accID):

        super(Transaction_log, self).__init__()

        self.accID = accID

        self.setWindowTitle("Welcome User")
        self.resize(1200, 600)

        self.tableItem = QtGui.QTableWidgetItem()

        self.setRowCount(10)

        self.setColumnCount(6)

        self.setHorizontalHeaderLabels(('Sender', 'Receiver', 'Transaction Amount','Time of
Transaction', 'Date of Transaction', 'Transaction ID'))

        self.setColumnWidth(0, 200)

        self.setColumnWidth(1, 200)

        self.setColumnWidth(2, 200)

        self.setColumnWidth(3, 200)

        self.setColumnWidth(4, 200)

        self.setColumnWidth(5, 200)

        self.fetch()

        self.show()


    def fetch(self):

        self.conn = sqlite3.connect('BankDB.db')

        self.c = self.conn.cursor()

        self.data = self.c.execute('SELECT SENDER_ID, RECEIVER_ID,
TRANSACTION_AMOUNT, DOT, TOT, TRANSACTION_ID from SAMPE where
ACCOUNT_ID = ? or RECEIVER_ID = ?;', [(self.accID), (self.accID)])

        j = 0

        for i in self.data:

            self.setItem(j, 0, QtGui.QTableWidgetItem(str(i[0])))

            self.setItem(j, 1, QtGui.QTableWidgetItem(str(i[1])))

            self.setItem(j, 2, QtGui.QTableWidgetItem(str(i[2])))
            self.setItem(j, 3, QtGui.QTableWidgetItem(str(i[3])))
            self.setItem(j, 4, QtGui.QTableWidgetItem(str(i[4])))
            self.setItem(j, 5, QtGui.QTableWidgetItem(str(i[5])))
            j += 1


class SignUpWindow(QtGui.QMainWindow):
    def __init__(self):
        super(SignUpWindow, self).__init__()
        self.setWindowTitle("Sign Up Window")
        self.resize(500, 500)
        self.Control()
        self.show()

    def Control(self):
        self.TextName = QtGui.QLabel("Name:", self)
        self.TextName.move(200, 50)
        self.TextUserName = QtGui.QLabel("User Name:", self)
        self.TextUserName.move(170, 100)
        self.TextPassword = QtGui.QLabel("Password:", self)
        self.TextPassword.move(180, 150)
        self.TextAccountNumber = QtGui.QLabel("Account Number:", self)
        self.TextAccountNumber.move(130, 200)
        self.TextAccountNumber.resize(self.TextAccountNumber.sizeHint())
        self.TextNetBalance = QtGui.QLabel("Net Balance:", self)
        self.TextNetBalance.move(160, 250)

        self.NameField = QtGui.QLineEdit(self)

        self.NameField.move(250, 50)

        self.User_NameField = QtGui.QLineEdit(self)

        self.User_NameField.move(250, 100)

        self.PasswordField = QtGui.QLineEdit(self)

        self.PasswordField.setEchoMode(QtGui.QLineEdit.Password)

        self.PasswordField.move(250, 150)

        self.Account_NumberField = QtGui.QLineEdit(self)

        self.Account_NumberField.move(250, 200)

        self.NetBalance = QtGui.QLineEdit(self)

        self.NetBalance.move(250, 250)

        self.SubmitButton = QtGui.QPushButton("Submit", self)

        self.SubmitButton.move(280, 400)

        self.CancelButton = QtGui.QPushButton("Cancel", self)

        self.CancelButton.move(180, 400)

        self.SubmitButton.clicked.connect(self.on_click_submit)

        self.CancelButton.clicked.connect(self.on_click_cancel)


    def on_click_submit(self):

        self.conn = sqlite3.connect("BankDB.db")

        self.c = self.conn.cursor()

        self.h = hashlib.sha512()

        self.h.update(self.PasswordField.text().encode())

        self.hash = self.h.digest()
        self.publicKey = str(seccure.passphrase_to_pubkey(b'my private key'))

        self.date = datetime.datetime.now()

        self.c.execute("INSERT INTO USERS(NAME, USERNAME, PASSWORD, ACCOUNT_ID, PubKey) VALUES(?, ?, ?, ?, ?);", [(self.NameField.text()), (self.User_NameField.text()), (self.hash), (self.Account_NumberField.text()), (self.publicKey)])

        self.c.execute('INSERT INTO SAMPE VALUES(00001, 00001, ?, ?, ?, ?, ?)', [(self.Account_NumberField.text()), (seccure.encrypt(self.NetBalance.text().encode(), b'8W;>i^H0qi|J&$coR5MFpR*Vn')), (str(self.date.time())), (str(self.date.date())), (hashTransaction(self.date.time(), self.Account_NumberField.text()))])

        self.conn.commit()

        self.conn.close()

        self.close()


    def on_click_cancel(self):

        self.result = QtGui.QMessageBox.question(self, 'Alert', "Do you want to leave this page ?", QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)

        if self.result == QtGui.QMessageBox.Yes:

            self.close()


class NetBalanceScreen(QtGui.QMainWindow):

    def __init__(self, accID):

        super(NetBalanceScreen, self).__init__()

        self.resize(200, 200)

        self.accID = accID

        self.CalBAL()

        self.Label = QtGui.QLabel("Net Balance", self)

        self.Balance = QtGui.QLineEdit(self)
        self.Balance.setText(str(self.NETBAL))
        self.Balance.setReadOnly(True)
        self.Label.move(50, 50)
        self.Balance.move(50, 100)
        self.show()

    def CalBAL(self):
        self.conn = sqlite3.connect('BankDB.db')
        self.c = self.conn.cursor()
        self.data = self.c.execute('SELECT TRANSACTION_AMOUNT FROM SAMPE WHERE
RECEIVER_ID = ?;', [(self.accID)])
        self.pos = 0
        for i in self.data:
            # self.pos += int(i[0])
            self.pos += float(seccure.decrypt(i[0], b'my private key').decode())
            # print(seccure.decrypt(str(i[0]).encode(), b'my private key'))
        self.data = self.c.execute('SELECT TRANSACTION_AMOUNT FROM SAMPE WHERE
SENDER_ID = ?;', [(self.accID)])
        self.neg = 0
        for i in self.data:
            pass
            self.neg += float(seccure.decrypt(i[0], b'my private key').decode())
            # print(seccure.decrypt(str(i[0]).encode(), b'my private key'))
        # self.bald = self.c.execute('SELECT NETBALANCE FROM USERS WHERE
ACCOUNT_ID = ?;',[(self.accID)])
        print(self.accID)
        self.NETBAL = self.pos - self.neg


if __name__ == "__main__":
    App = QtGui.QApplication(sys.argv)
    GUI = Window()
    sys.exit(App.exec_())
```

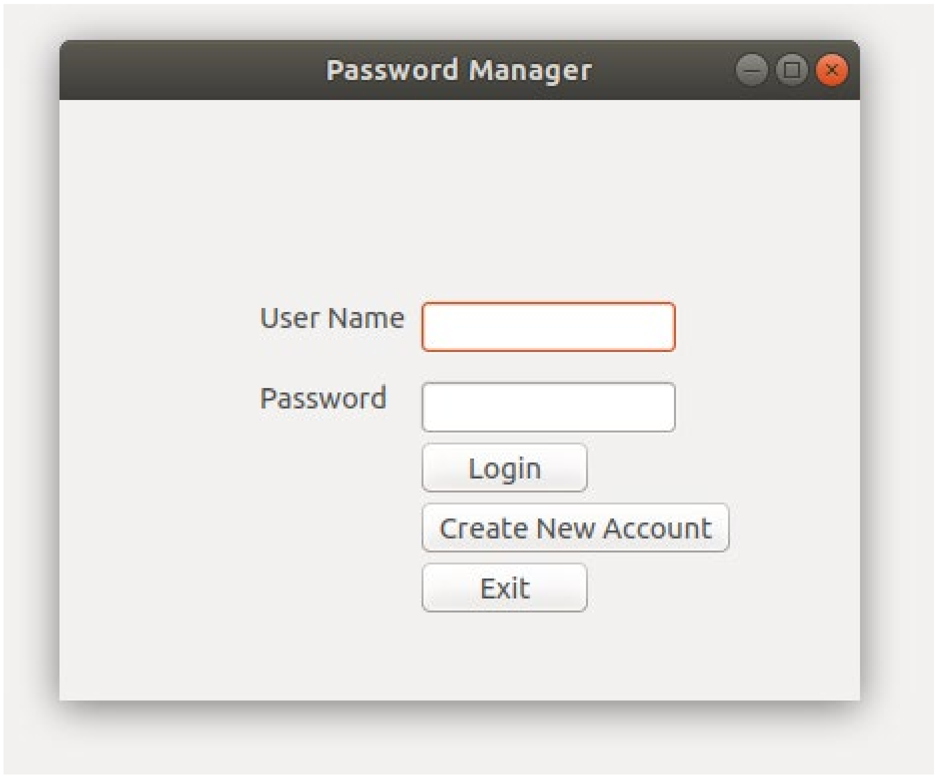
CreateBankDB.py

```
import sqlite3


conn = sqlite3.connect('BankDB.db')


c = conn.cursor()


c.execute("CREATE TABLE USERS(NAME TEXT, USERNAME TEXT, PASSWORD
TEXT, ACCOUNT_ID INTEGER, PubKey TEXT);")
conn.commit()
c.execute("CREATE TABLE SAMPE(ACCOUNT_ID INTEGER, SENDER_ID INTEGER,
RECEIVER_ID INTEGER, TRANSACTION_AMOUNT REAL, DOT TEXT, TOT TEXT,
TRANSACTION_ID TEXT);")
conn.commit()
conn.close()
```

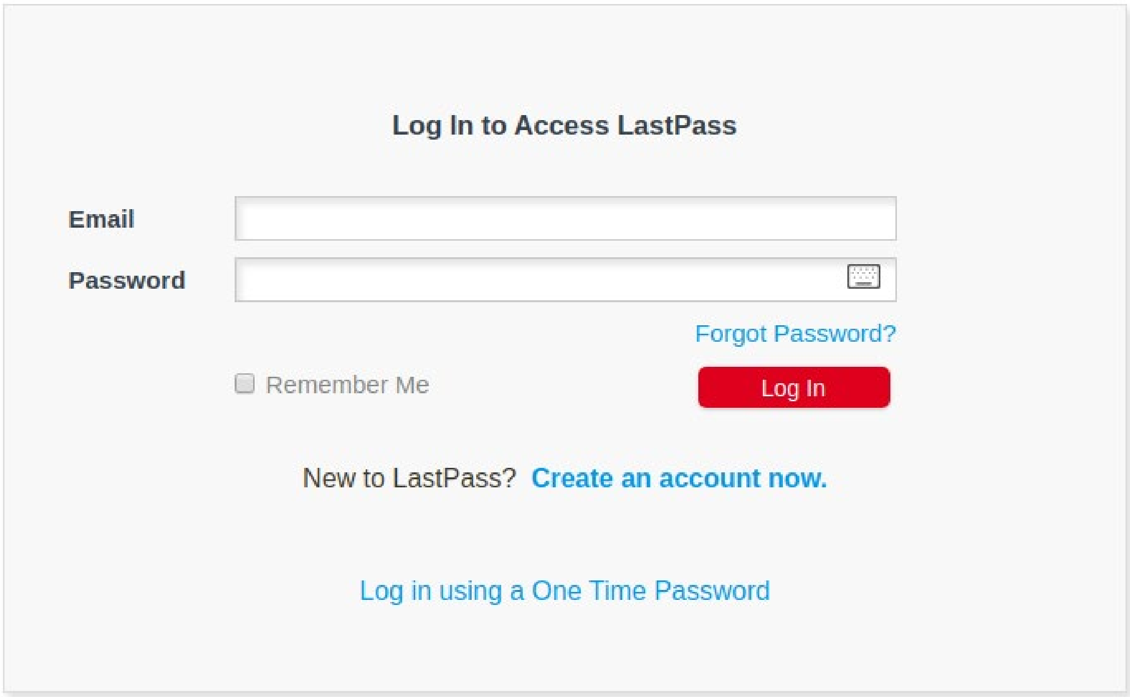
transactionview.py

```
import sqlite3


conn = sqlite3.connect('BankDB.db')

c = conn.cursor()


def trans():

    for i in c.execute("SELECT * FROM sampe;"):

        print(i)


def users():

    for i in c.execute("SELECT * FROM USERS;"):

        print(i)



while True:

    ptr = int(input("1. Users\n2. Transactions\n3. Exit\n"))

    if ptr == 1:

        users()

    elif ptr == 2:

        trans()

    elif ptr == 3:

        exit()

    else:

        print("Wrong input\n")
```

Performance Metrics:



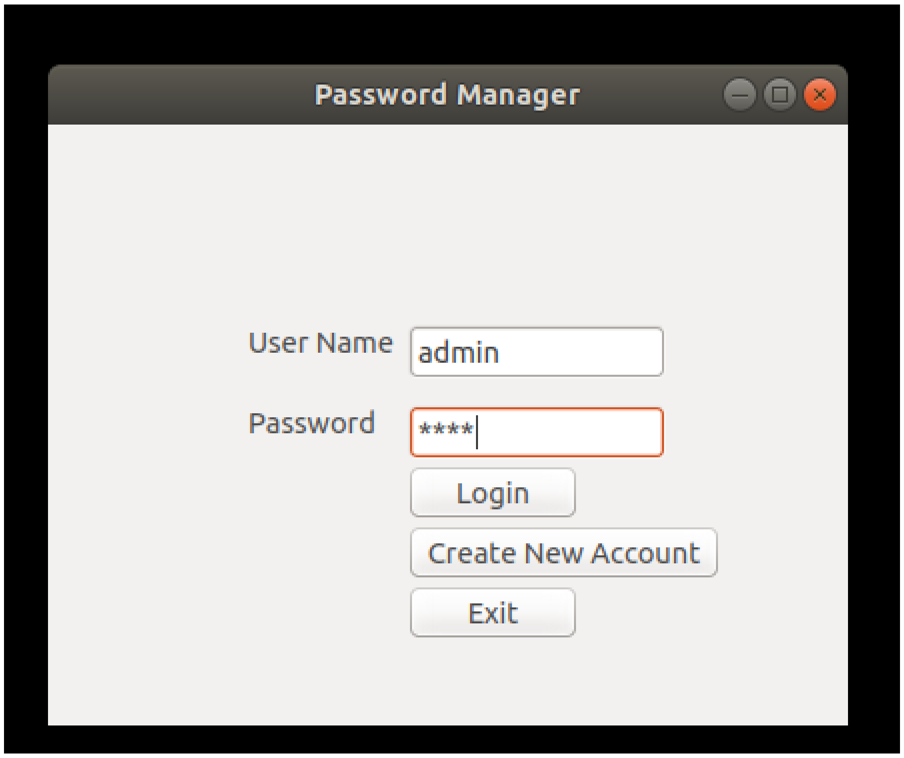
Similar products are already present in the market such as LastPass


Log In to Access LastPass
Email
Password
Forgot Password?
Remember Me
Log In
New to LastPass? Create an account now.
Log in using a One Time Password

Comparison between our password manager and LastPass password manager

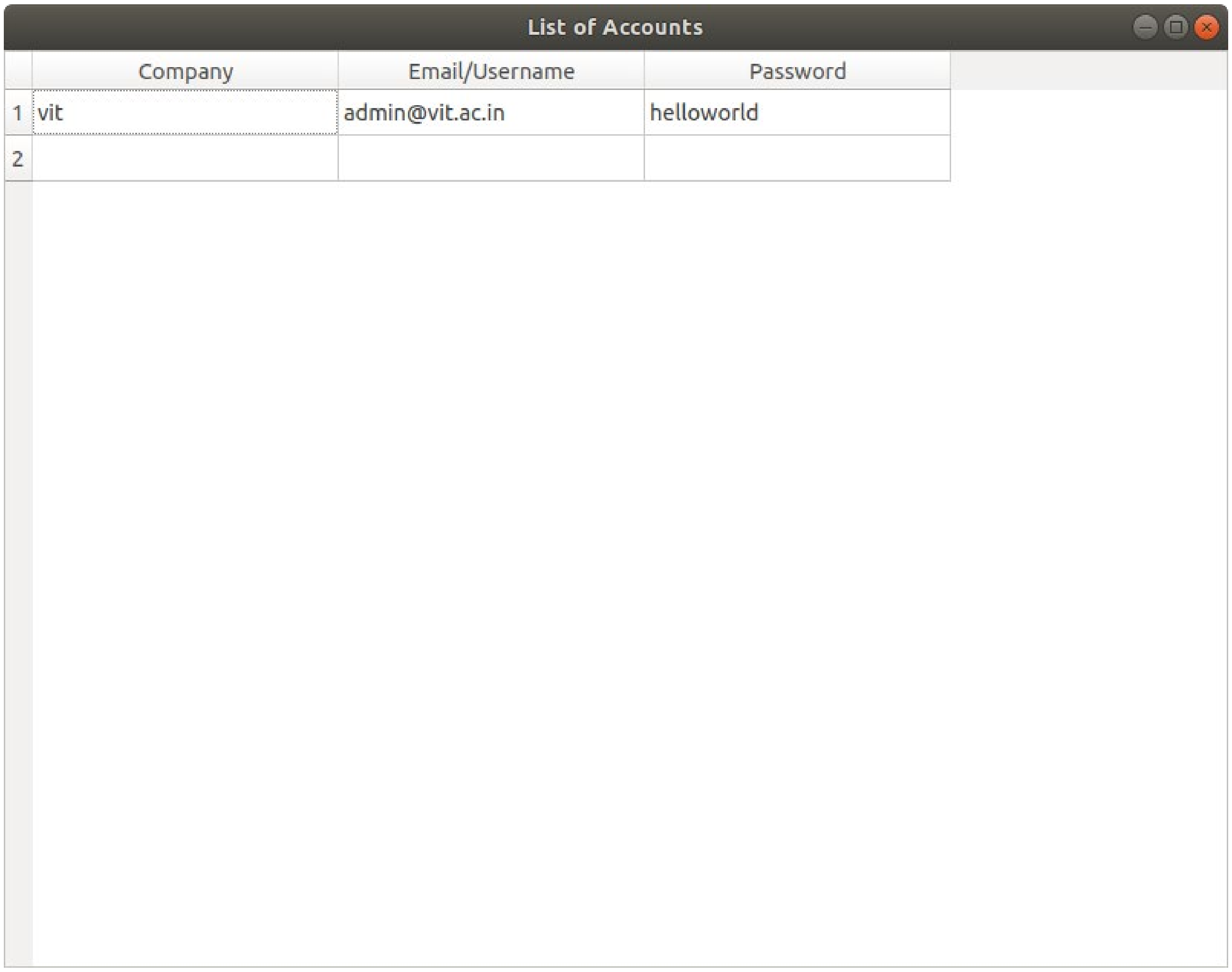
LastPass password manager	Own password manager
AES-256 with PBKDF2 SHA – 256 and salted hash	ECIES and SHA – 512 hash
Local – level encryption and decryption	Local – level encryption and decryption
Two – factor authentication	Single authentication level
Data in cloud	Data in local machine

Results Obtained:



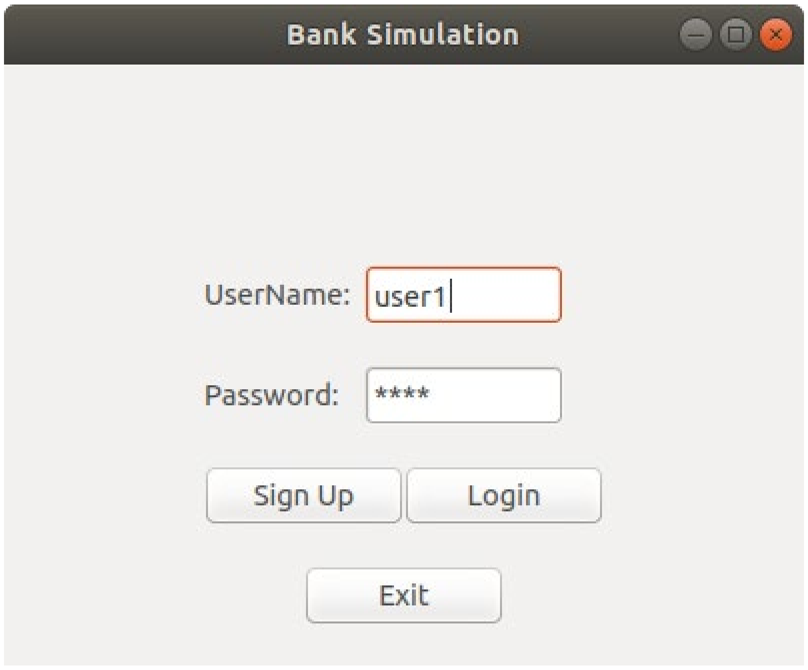
User Name: admin

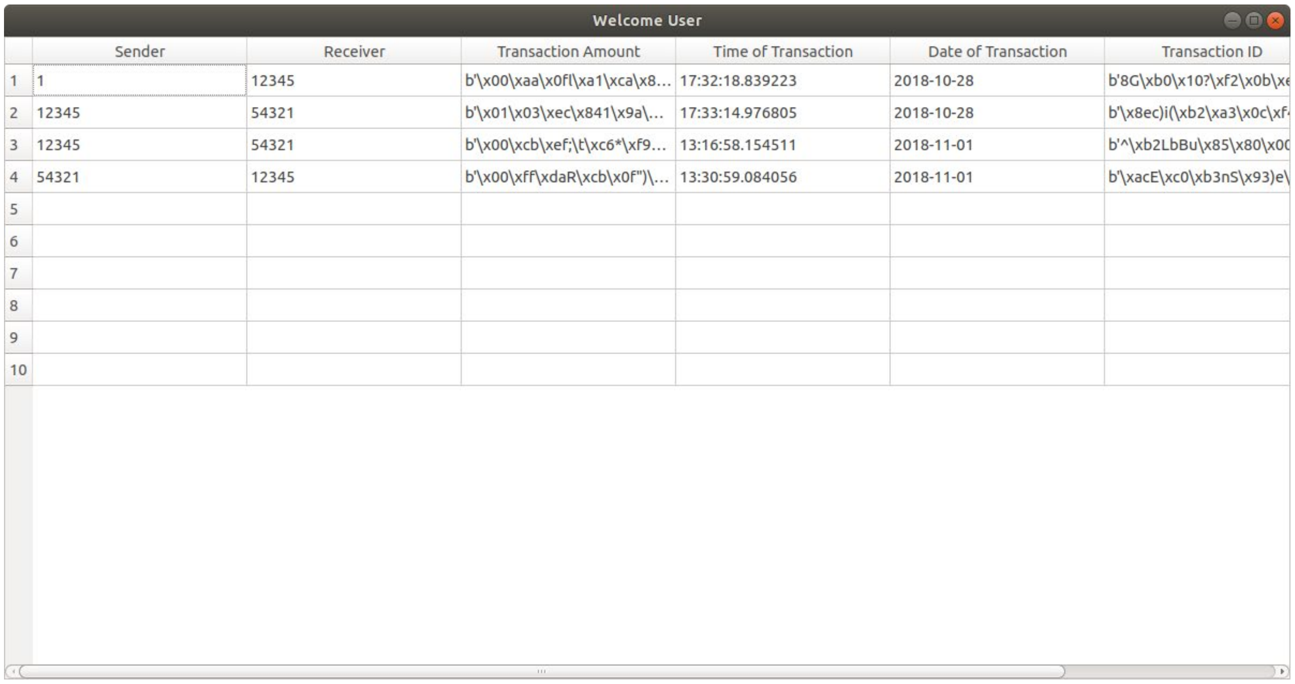
Password: ****


List of Accounts

	Company	Email/Username	Password
1	vit	admin@vit.ac.in	helloworld
2			

Banking simulation output:


Bank Simulation
UserName:
user1
Password:
****
Sign Up
Login
Exit


Welcome User

	Sender	Receiver	Transaction Amount	Time of Transaction	Date of Transaction	Transaction ID
1	1	12345	b'\x00\xaa\x0fl\xa1\xca\x8...	17:32:18.839223	2018-10-28	b'8G\xb0\x10?\xf2\x0b\xe
2	12345	54321	b'\x01\x03\xec\x841\x9a\...	17:33:14.976805	2018-10-28	b'\x8ec)i(\xb2\xa3\x0c\xf
3	12345	54321	b'\x00\xcb\xef;\t\xc6*\xf9...	13:16:58.154511	2018-11-01	b'^\xb2LbBu\x85\x80\x0
4	54321	12345	b'\x00\xff\xdaR\xcb\x0f")\...	13:30:59.084056	2018-11-01	b'\xacE\xc0\xb3nS\x93)e\
5						
6						
7						
8						
9						
10						

Conclusion

Our project was to provide a free password manager for people who cannot afford subscription-based password managers but want to get security nonetheless. We also had an idea to pride a simulation of secure online banking transaction manager so people can understand how transactions are secured in banks. We believe we have successfully fulfilled our aims and objectives. There are improvements to be made here and there which we will address shortly after releasing the application to the public over the internet.

References:

[1] Victor S. Miller, Use of Elliptic Curves in Cryptography, Advances in Cryptology-CRYPTO’85 Proceedings, Springer, vol. 218, pp. 417–426, December (2000).

[2] Neal Koblitz, Elliptic Curve Cryptosystems, Mathematics of Computation, vol. 48, issue 177, pp. 203–209, January (1987).

[3] Neal Koblitz, Alfred Menezes and Scott Vanstone, The State of Elliptic Curve Cryptography, Designs, Codes and Cryptography, vol. 19, issue 2–3, pp. 173–193, (2000).

[4] Darrel Hankerson, Alfred Menezes and Scott Vanstone, Guide to Elliptic Curve Cryptography, Springer (2004).

[5] Lawrence C. Washington, Elliptic Curves Number Theory and Cryptography, Taylor & Francis Group, Second Edition (2008).

[6] Jorko Teeriaho, Cyclic Group Cryptography with Elliptic Curves, Brasov, May (2011).

[7] S. Maria Celestin Vigila and K. Muneeswaran, Implementation of Text based Cryptosystem using Elliptic Curve Cryptography, International Conference on Advanced Computing, IEEE, pp. 82–85, December (2009).

[8] D. Sravana Kumar, Ch. Suneetha and A. Chandrasekhar, Encryption of Data Using Elliptic Curve Over Finite Fields, International Journal of Distributed and Parallel Systems (IJDPS), vol. 3, no. 1, January (2012).

[9] K. Jarvinen, Helsinki and J. Skytta, On Parallelization of High-Speed Processors for Elliptic Curve Cryptography, VLSI Systems, IEEE Transaction, vol. 16, issue 9, pp. 1162–1175, August (2008).

[10] M. Amara and A. Siad, Elliptic Curve Cryptography and its Applications, 7th International Workshop on Systems, Signal Processing and their Applications, pp. 247–250, May (2011).

[11] Gopinath Ganapathy and K. Mani, Maximization of Speed in Elliptic Curve Cryptography Using Fuzzy Modular Arithmetic over a Micro-controller based Environment, Proceedings of the World Congress on Engineering and Computer Science, vol. 1, (2009).

[12] Scott A. Vansfone, Elliptic Curve Cryptography-The Answer to Strong, Fast Public-Key Cryptography for Securing Constrained Environments, Information Security Technical Report, vol. 2, no. 2, pp. 78–87, (1997).

[13] O. Srinivasa Rao and S. Pallam Setty, Efficient Mapping Methods for Elliptic Curve Cryptography, International Jounal of Engineering Science and Technology, vol. 2(8), pp. 3651–3656, (2010).

[14] Williams Stallings, Cryptography and Network Security, Prentice Hall, 4th Edition, (2000).

[15] Lo'ai Tawalbeh, Moad Mowafi and Walid Aljoby, Use of Elliptic Curve Cryptography for Multimedia Encryption, IET Information Security, vol. 7, issue 2, pp. 67–74, (2012).

[16] www.csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf, (1999).

[17] R. Balamurugan, V. Kamalakannan, D. Rahul Ganth and S. Tamilselvan, Enhancing Security in Text Messages Using Matrix based Mapping and ElGamal Method in Elliptic Curve Cryptography, International Conference on Contemporary Computing and Informatics, IEEE, pp. 103–106, November (2014).

[18] [18] Megha Kolhekar and Anita Jadhav Implementation of Elliptic Curve Cryptography on Text and Image, International Journal of Enterprise Computing and Business Systems, vol. 1, issue 2, July (2011).

[19] Abhishek Parakh, "Oblivious Transfer using Elliptic Curves"Department of Electrical and Computer Engineering",2006,Proceedings of the 15th International Conferences on Computing.

[20] Eugen Petac "About a method for Distribution keys of a computer netwrok using elliptic curves"Department of mathematics and Computer Science,1997

[21] Guicheng shen,Xuefeng Zheng, "Research on Implementation of Elliptic Curve Cryptosystem in E-Commerce", International Symposium on Electronic Commerce and Security,2008

[22] G.V.S.Raju and Rehan Akbani,2003, "Elliptic Curve Cryptosystem and its Applications",2003,The University of Texas at san Atonio.

[23] Hans Eberle,Nils Gura,Scheduling Chang-Shantz "A Cryptographic Processor for Arbitary Elliptic Curves Over GF(2m)",2003 proceedings of the Aplication-Secific System,Architectures,and Processors(ASAP'03).

[24] Jia Xiangya, Wang Chao. "The Application of Elliptic Curve Cryptosystem in Wireless Communication", 2005 IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communication,2005

[25] Kristin Lauter,Microsoft Corporation,2004, “The Advantage of Elliptic Curve Cryptography For Wireless Security”,IEEE Wireless Communications.

[26] A.Nithin V.S,Deepthi P.P, Dhanaraj K.J,Sathidevi P.S “Stream ciphers Based on the Elliptic Curves” national Institute of Technolgy,Calicut,Internatioanl Conference on Computational Intelligence and Multimedia Applications 2007

[27] Sarwono Sutikno,Andy Surya,Ronny Effendi, “An Implementation of ElGamal Elliptic Curves Cryptosystems”,1998.Integrated System Laboratory,Bandung Institute of Technology.

[28] Qizhi Qiu and Qianxing Xiong,2003, “Research On Elliptic Curve cryptography” Wuhan University of Technology

[29] Yacine Rebahi,Jprdi Jaen Pallares,Gergely Kovacs,Dorgham Sisalem “Performance Analysis of Identity management in the session Initiation Protocol”IEEE Journal. Lay, G. and Zimmer, H. (2018).

[30] Constructing elliptic curves with given group order over large finite fields.] Available at: https://www.researchgate.net/publication/49966421_Encoding_And_Decoding_of_a_Message_in_the_Implementation_of_Elliptic_Curve_Cryptography_using_Koblitz's_Method.