VIT®

Vellore Institute of Technology

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

PRIVACY PRESERVATION IN TRANSACTIONS

USING FLASH K-ANONYMITY

J COMPONENT PROJECT REPORT
FALL SEM 2021-22

Submitted by

CHINTHALA HARDHEEK - 19BCI0163
SHIV THAKER - 19BCI0167
KONIJETI MAHESH SAI - 19BCI0164

BCI2001 - Data Privacy
Slot – A1+TA1

Under the Guidance of
Prof. Vijaya Kumar K

Computer Science and Engineering
DECEMBER 2021

ACKNOWLEDGEMENT

We would like to acknowledge all those without whom this project would not have been successful. Firstly, we would like to thank our Data Privacy 'Prof. Vijaya Kumar K' who guided us throughout the project and gave his immense support. He made us understand how to complete this project and without his presence, this project would not have been complete. We also got to know a lot about parallelization and its benefits. This project has been a source to learn and bring our theoretical knowledge to the real-life world. Once again, thanks to everyone for making this project successful.

Place : Vellore Institute of Technology, Vellore

Date : Dec 2, 2021

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

ABSTRACT

The age of information is also the age of digital information assets. With it comes the need for the professional programmer to protect these assets cryptographically as well as to devise ways for efficient computing. As the security in ATMs is increasing the access with privacy is decreasing. In today's world, ATM's have become a very important and useful resource for everyone.

To make sure our transactions are secure, we need some security protocols. Through data privacy and protection techniques we can make our system secure. The algorithms and the tools used for data privacy and data protection are discussed in subsequent sections along with proper implementation explanation.

Keywords - Data privacy, Data protection, Cryptography

INTRODUCTION

The core theme of the project focuses on the development of Security in ATM Transaction System in Python using Hash, RSA algorithms and using SQLite and anonymizing the dataset using Flash algorithm in Amnesia tool. We will establish a model of an ATM.

What is an ATM?

ATM stands for Automated Teller Machine. It's a specialized computer that makes it convenient to manage your money. For example, almost all ATMs allow you to withdraw money, and many allow you to make deposits. At some ATMs, you can print a statement (a record of your account activity or transactions); check your account balances (the amount of money in your accounts right now); transfer money between your accounts; and even purchase stamps. You can usually access the most services at an ATM that's operated by your own bank.

Why use ATMs?

ATMs are a safe and convenient way to manage your money. There are millions of ATMs worldwide and you can use many ATMs 24 hours a day, 7 days a week. Some allow you to select the language you want to use.

Is there a fee for using an ATM?

Check with your bank to see if they charge any ATM fees to customers. Almost all banks do charge a fee to non-customers who use their ATMs. Keep in mind that even though using ATMs may cost you money, it's much less expensive than using a check cashing service.

Road - Map

A database and two modules. One will be for the back where all the account creation and modification will be made and other will be the ATM which will provide a way for the user to

connect. Security protocols are used for storing details in encrypted form ensuring security to the user. The gathered dataset can then be anonymised and protected in a way that the data anonymizer can create different anonymized dataset. These anonymized datasets are measured in the way of both data privacy and utility based approach.

RELATED WORK / LITERATURE SURVEY

Transaction data are usually useful for data mining. While it is high-dimensional data, traditional anonymization techniques such as generalization and suppression are not suitable. The simulation experiments on real datasets and the results of association rules mining on the anonymized transaction data showed that our algorithm can safely and efficiently preserve the privacy in transaction data publication, while ensuring high utility of the released data.

K-anonymization is an important technique for the de-identification of sensitive datasets. In this paper, we briefly describe an implementation FLASH framework which has been carefully engineered to meet the needs of an important class of k-anonymity algorithms. They have implemented and evaluated two major well-known algorithms within this framework and show that it allows for highly efficient implementations. Regarding their runtime behavior, they were able to closely reproduce the results from previous publications but also found some algorithmic limitations. In contrast to the current state-of-the-art, our algorithm offers algorithmic stability, with execution time being independent of the actual representation of the input data. Experiments with different real-world datasets show that their solution clearly outperforms the previous algorithms.

Table 1 Literature Survey

Title	Year of publication	Journal name	Methodology and implementation	Citation	Limitations / Future research / Gaps identified
Flash: Efficient, Stable and Optimal K-Anonymity	2012	2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on	This work is based upon a generic framework for the efficient implementation of k-anonymity algorithms. They presented an efficient implementation of the OLA algorithm on top of it.	F. Kohlmaye r, F. Prasser, C. Eckert, A. Kemper and K. A. Kuhn, "Flash: Efficient,	To better leverage the capabilities of modern multi-core processors by parallelizing implementation framework as well as the Flash algorithm. Early experiments with simple intra-operator parallelization within our framework were promising, but the parallelization

		Privacy, Security, Risk and Trust		Stable and Optimal K-Anonymity," 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing, 2012, pp. 708-717, doi: 10.1109/SocialCom-PASSAT.2012.52.	The k-anonymity algorithm itself is challenging.
FLASH: Fast and Robust Framework for Privacy-preserving Machine Learning	2020	April 2020 Proceedings on Privacy Enhancing Technologies 2020(2):459-480	Privacy-preserving machine learning (PPML) via Secure Multi-party Computation (MPC) has gained momentum in the recent past. Assuming a minimal network of pair-wise private channels, we propose an efficient four-party PPML framework over rings Z2` , FLASH, the first of its kind in the regime of PPML framework, that achieves the strongest security notion of Guaranteed Output Delivery (all parties obtain the output irrespective of adversary's behaviour). The state of the art ML	TY - JOUR AU - Byali, Megha AU - Chaudhari, Harsh AU - Patra, Arpita AU - Suresh, Ajith PY - 2020/04/01 SP - 459 EP - 480	The latency and throughput are evaluated over both LAN and WAN settings. The communication complexity is measured independent of the network. For the aforementioned algorithms, the throughput is calculated as the number of queries that can be computed per second and min in LAN and WAN respectively.

			frameworks such as ABY3 by Mohassel et.al (ACM CCS'18) and SecureNN by Wagh et.al (PETS'19) operate in the setting of 3 parties with one malicious corruption but achieve the weaker security guarantee of abort.	T1 - FLASH: Fast and Robust Framework for Privacy-preserving Machine Learning VL - 2020 DO - 10.2478/popets-2020-0036 JO - Proceedings on Privacy Enhancing Technologies ER -	
Privacy Preservation in Transaction Databases based on Anatomy technique	2020	The 5th International Conference on Computer Science & Education Hefei, China. August 24–27, 2010	This paper considers the problem of privacy preserving transaction data publishing. Transaction data are usually useful for data mining. While it is high-dimensional data, traditional anonymization techniques such as generalization and suppression are not suitable. In this paper, we present a novel technique based on anatomy technique and propose a simple linear-time anonymous algorithm that meets the l-diversity requirement. The simulation experiments on real datasets and the results of association rules mining on the anonymous transaction data showed that our	Yingjie Wu School of Computer Science and Engineering Southeast University Nanjing, China College of Mathematic and Computer Science Fuzhou University Fuzhou, China yjwu@fzu.edu.cn	Many privacy models, such as k-anonymity [4,5,6], l diversity [7] and t-closeness [8], are used to prevent reidentification attacks on low-dimensional relational data, but directly adopt these to high-dimensional unstructured data will not produce good results. These models anonymize the public attributes (i.e., quasi-identifiers) to prevent the attacker using link attack to infer the privacy of personal information. However, for transaction data, any attribute (i.e., item) permutations and combinations are considered as the attacker's background knowledge.

			algorithm can safely and efficiently preserve the privacy in transaction data publication, while ensuring high utility of the released data.	Shangbin Liao, Xiaowen Ruan, Xiaodong Wang College of Mathematics and Computer Science Fuzhou University Fuzhou, China N080320044@fzu.edu.cn	
Preserving Transactional Data	2016	DPC Technology Watch Report 16-02 May 2016	This paper discusses requirements for preserving transactional data and the accompanying challenges facing the companies and institutions who aim to re-use these data for analysis or research. It presents a range of use cases – examples of transactional data – in order to describe the characteristics and difficulties of this 'big' data for longterm access. Based on the overarching trends discerned in these use cases, the paper will define the challenges facing the preservation of these data early in the curation lifecycle. It will point to potential solutions within current legal and ethical frameworks, but will focus on positioning the problem of re-using these data from a preservation perspective.	Bruno Ferreira, Miguel Ferreira, and Luís Faria, KEEP SOLUTIONS and José Carlos Ramalho, University of Minho ISSN: 2048-7916	As public-facing networks, they are also in a position to build trust with the larger population when it comes to re-use of data. Institutions who traditionally preserve digital content (e.g. repositories, libraries, archives) have also faced the need to demonstrate their trustworthiness to the general public.
A ROBUST TECHNIQUE FOR PRIVACY	2014	IMPACT: International Journal of	This paper provides an enhanced technique for preserving	VINEET RICHHARIYA1 &	Proposed algorithm have reduced the time complexity, space complexity as well as false

PRESERVATION OF OUTSOURCED TRANSACTION DATABASE		Research in Engineering & Technology (IMPACT: IJRET) ISSN(E): 2321-8843; ISSN(P): 2347-4599 Vol. 2, Issue 6, Jun 2014, 51-58 © Impact Journals	privacy of association rules as well as private data of individuals in an outsourced business transaction database. As the importance of business transaction data has increased manifolds and the data has become an essential part of any business. This paper implement privacy by using a perturbation technique using jointly Gaussian Function that will not only maintain the privacy of association rules present in the dataset but also the sensitive attributes of individuals contained in it. Using this approach we are reducing time complexity, space complexity, and fake and false rules problems	PRATEEK CHOUREY2 1HOD, Department of Computer Science & Engineering, LNCT, Bhopal, Madhya Pradesh, India 2Research Scholar, Department of Computer Science & Engineering, LNCT, Bhopal, Madhya Pradesh, India	rules problems in an effective manner from the previous work.
Two Privacy-Preserving Approaches for Publishing Transactional Data Streams	2017	Digital Object Identifier 10.1109/ACCESS.2017.DOI	Many privacy-preserving approaches have been proposed for publishing static transactional data. Due to the characteristics of data streams, which must be processed quickly, static data anonymization methods cannot be directly applied to data streams. In this paper, we first analyze the privacy problem in publishing transactional data streams based on a sliding window. Then, we present two dynamic algorithms with	JINYAN WANG, CHAOJI DENG, AND XIANXIAN LI 1Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin,	The information losses are near to those caused by batch processing with the existing static anonymization methods.

			generalization and suppression to anonymize continuously a sliding window to make it satisfy ρ-uncertainty by structuring an affected sensitive rules trie, because the removal and addition of transactions may make the current sliding window fail to satisfy ρ-uncertainty. Experimental results show that our methods are more efficient than sliding window anonymization with batch processing by using existing static anonymization methods	541004, China 2School of Computer Science and Information Technology, Guangxi Normal University, Guilin, 541004, China Corresponding author: Xianxian Li (e-mail: lixx@gxnu.edu.cn)	

PROPOSED PROBLEM FORMULATION

Privacy Preservation

Flash K-anonymity algorithm: The algorithm determines the optimal k-anonymity solutions by binary search on the lattice of generalization. The algorithm searches through all paths in the generalization lattice and terminates when all paths are traversed completely.

Algorithm 1: Outer loop of the Flash algorithm

```
   Input: Lattice lattice
1  begin
2      heap ← new min-heap
3      for l = 0 → lattice.height − 1 do
4          foreach node ∈ level[l] do
5              if !node.tagged then
6                  path ← FindPath(node)
7                  CheckPath(path, heap)
8                  while !heap.isEmpty do
9                      node ← heap.extractMin
10                     foreach up ∈ node.successors do
11                         if !up.tagged then
12                             path ← FindPath(up)
13                             CheckPath(path, heap)
```

Figure 1. Flash algorithm pseudo code

It traverses the lattice in a bottom-up breadth-first manner and constantly generates paths which branch like lightning flashes. It is based upon the following observations:

1. Predictive tagging can be best exploited if the lattice is traversed vertically and in a binary fashion.
2. When traversing a lattice vertically, the execution time becomes volatile in terms of the representation of the input dataset (e.g., the order of the columns). This must be prevented by implementing a stable strategy.
3. In order to achieve the best performance, the algorithm should prefer nodes that allow the application of the previously presented optimizations.

Amnesia tool:

The Amnesia anonymization tool is software written in Java and JavaScript and should be used locally for anonymizing personal and sensitive data. The basic idea behind anonymization is that we load a file containing personal data (original data) to Amnesia, and Amnesia transforms it into an anonymous dataset, which can then be stored locally. The transformation is guided by user selections and provides an anonymization guarantee for the resulting dataset. Amnesia currently supports k-anonymity and km-anonymity guarantees.

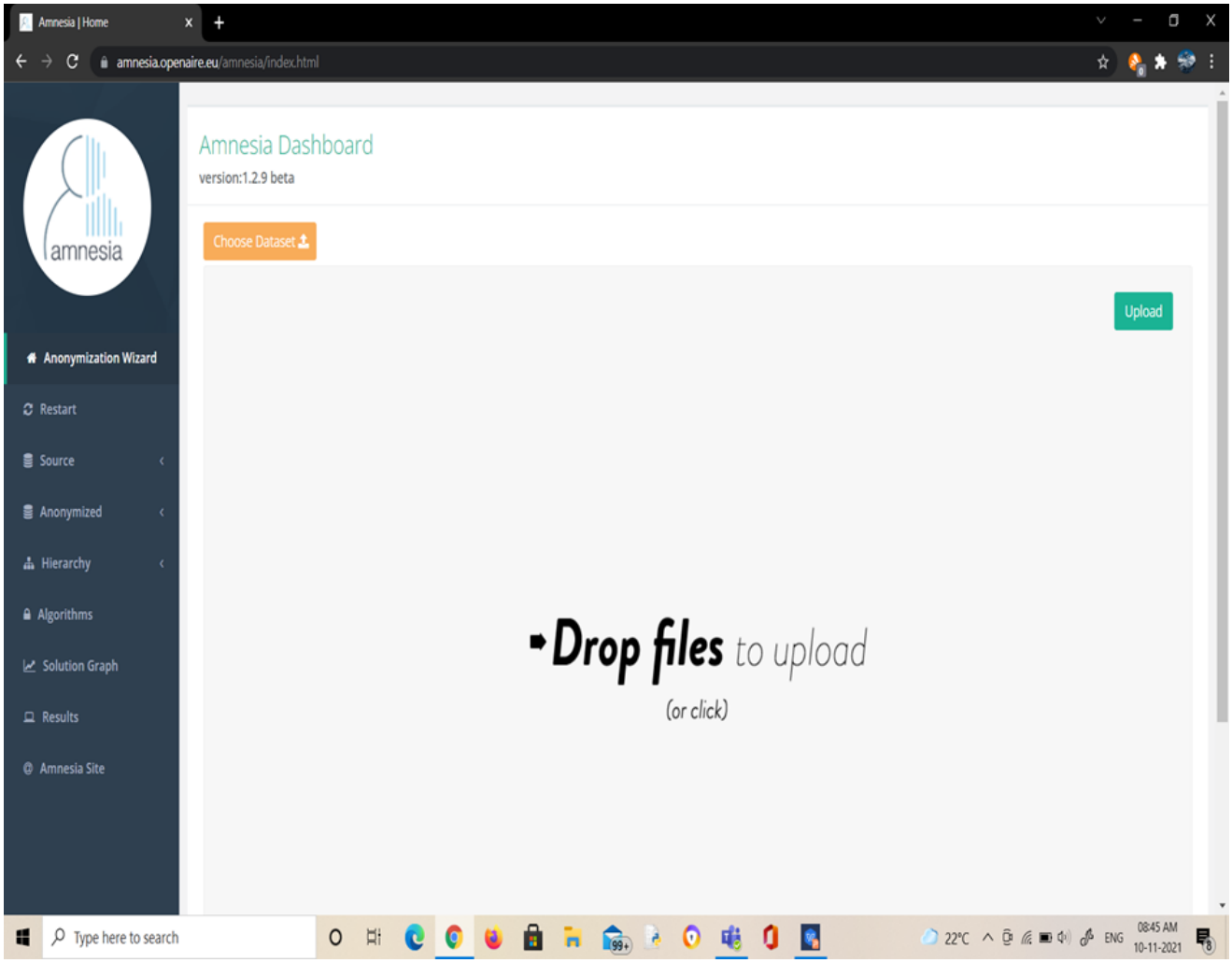


Figure 2. Amnesia anonymization tool

Hash Function Algorithm

A hash function is a function that takes input of a variable length sequence of bytes and converts it to a fixed length sequence. It is a one way function. This means if f is the hashing function, calculating f(x) is pretty fast and simple, but trying to obtain x again is not possible practically. The value returned by a hash function is often called a hash, message digest, hash value, or checksum. Hash function will produce a unique output for a given input. However, depending on the algorithm, there is a possibility to find a collision due to the mathematical theory behind these functions. Hash functions are used inside some cryptographic algorithms, in digital signatures, message authentication codes, manipulation detection, fingerprints, checksums (message integrity check), hash tables, password storage and much more. Some of the Hash function algorithm examples are MD5, SHA (SHA1, SHA224, SHA256, SHA384, SHA512).

Flask Framework

For the front end/UI design we have decided to use the flask web development framework. Flask has an inbuilt library called SQLAlchemy which can be used to provide support for the backend of our UI.

SQLite

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.

Figure 3. SQLite

The SQLite file format is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way through the year 2050. SQLite database files are commonly used as containers to transfer rich content between systems and as a long-term archival format for data. There are over 1 trillion (1e12) SQLite databases in active use.
SQLite source code is in the public-domain and is free to everyone to use for any purpose.

As to why we use hashing,
Password hashing is not for the security of passwords while being transmitted over the wire, but for *storage* on the server. We hash passwords because it sometimes happens (more often than we would like) that bad people get a peek at the server's files and/or database.
Password hashing is used to protect users in the case of a database being compromised.
Transmitting the password in plaintext allows an attacker to sniff the password. Computing the hash on the client side essentially makes the hash the password, so an attacker can just sniff the hash and use that instead.

How all the modules, algorithms, tools and techniques come together will be explained in the next section.

IMPLEMENTATION / MODELING

We use the Faker tool integrated along with the flask framework to create our own dataset. The dataset is then anonymized using the Amnesia tool. The results show the UI that we made to interact with the database. Then we implement the Flash K-anonymity algorithm on the same database. Results for the same are shown below.

RESULTS AND DISCUSSION

Generalized hierarchies:

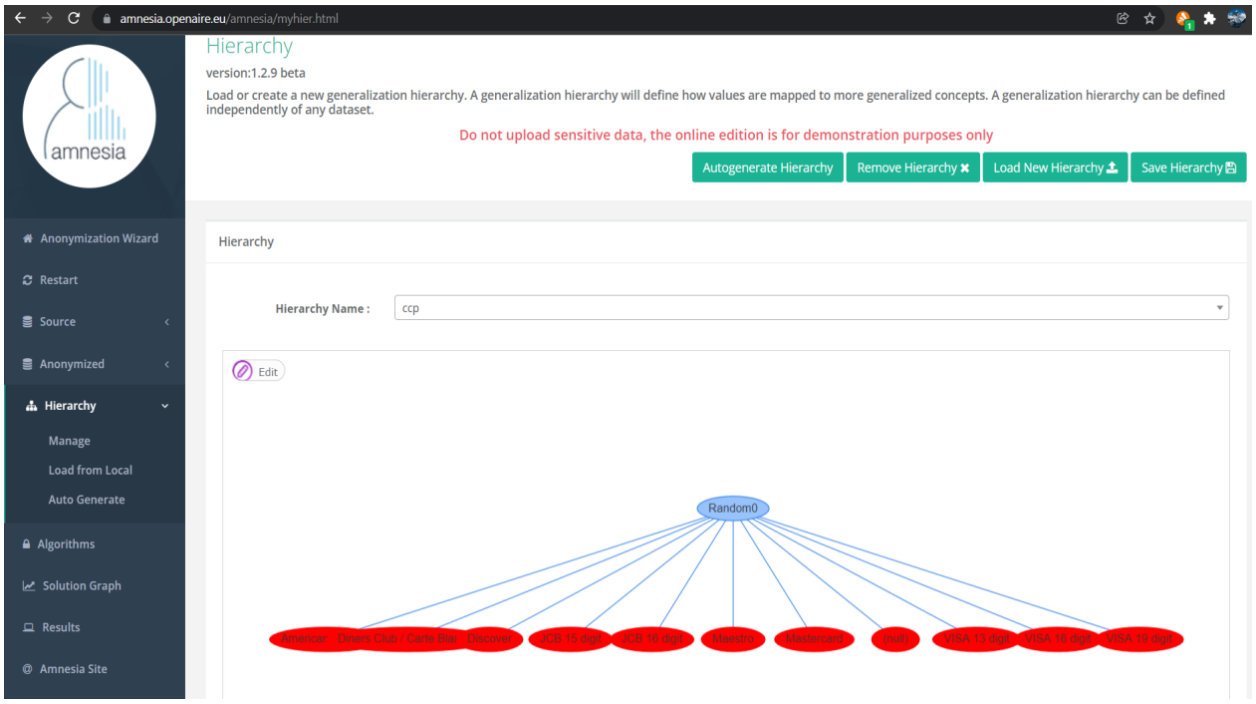


Figure 4. CCPROVIDER

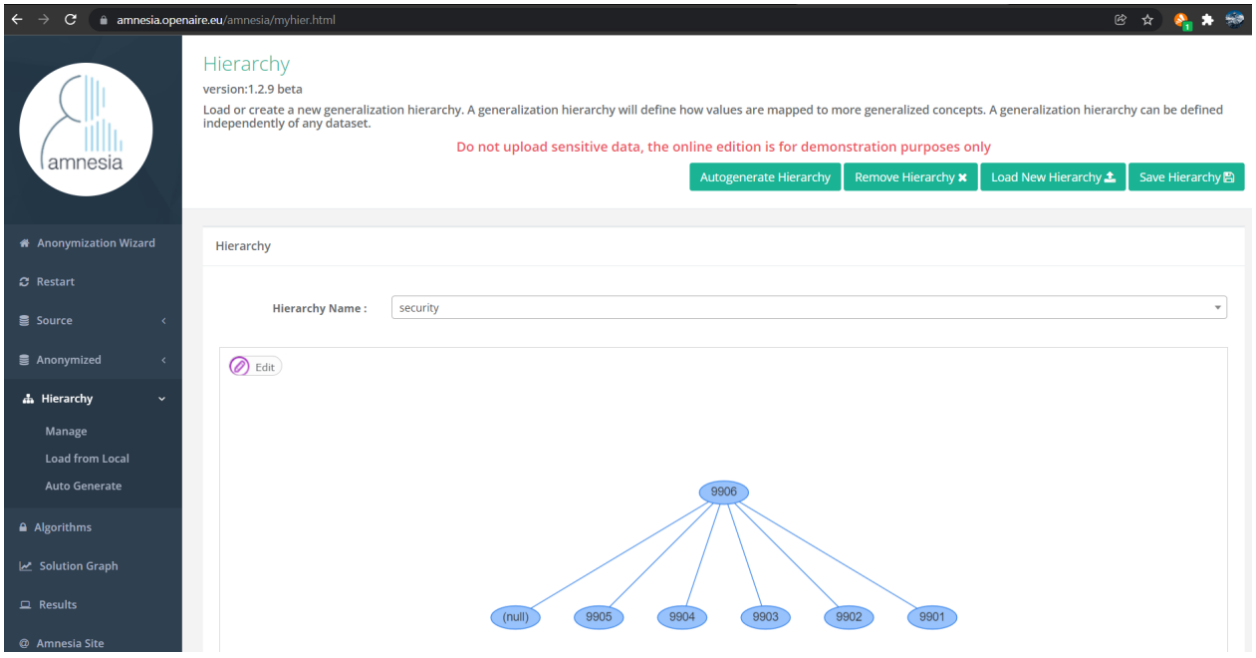


Figure 5. CCSECURITY CODE

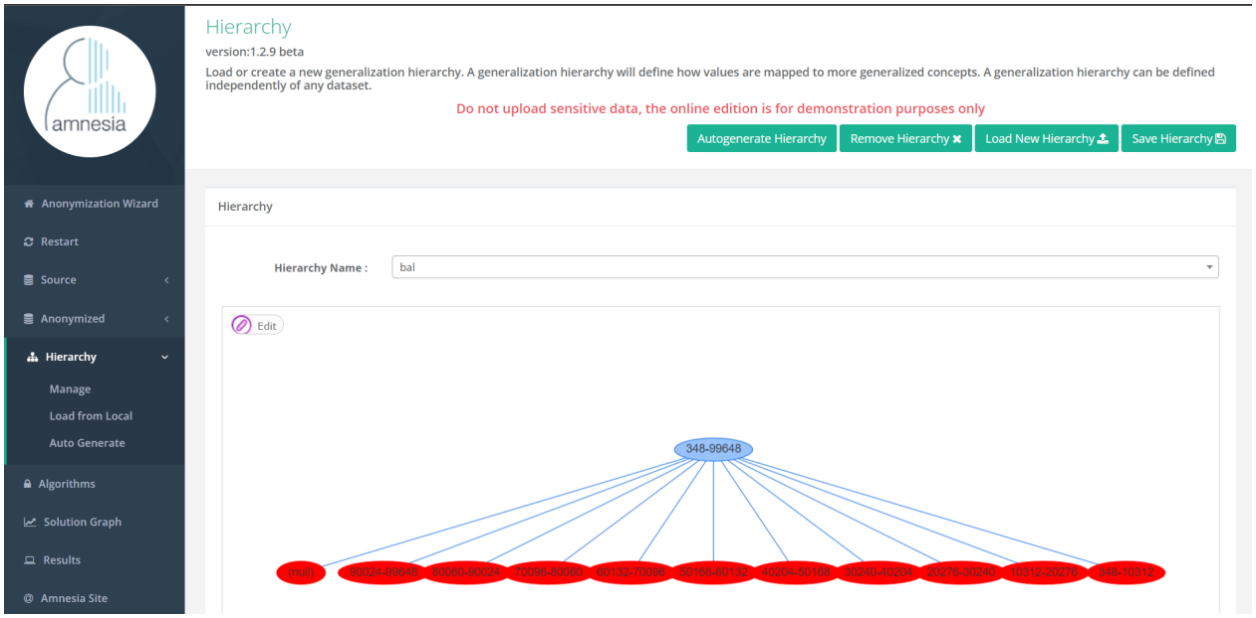


Figure 6. BALANCE

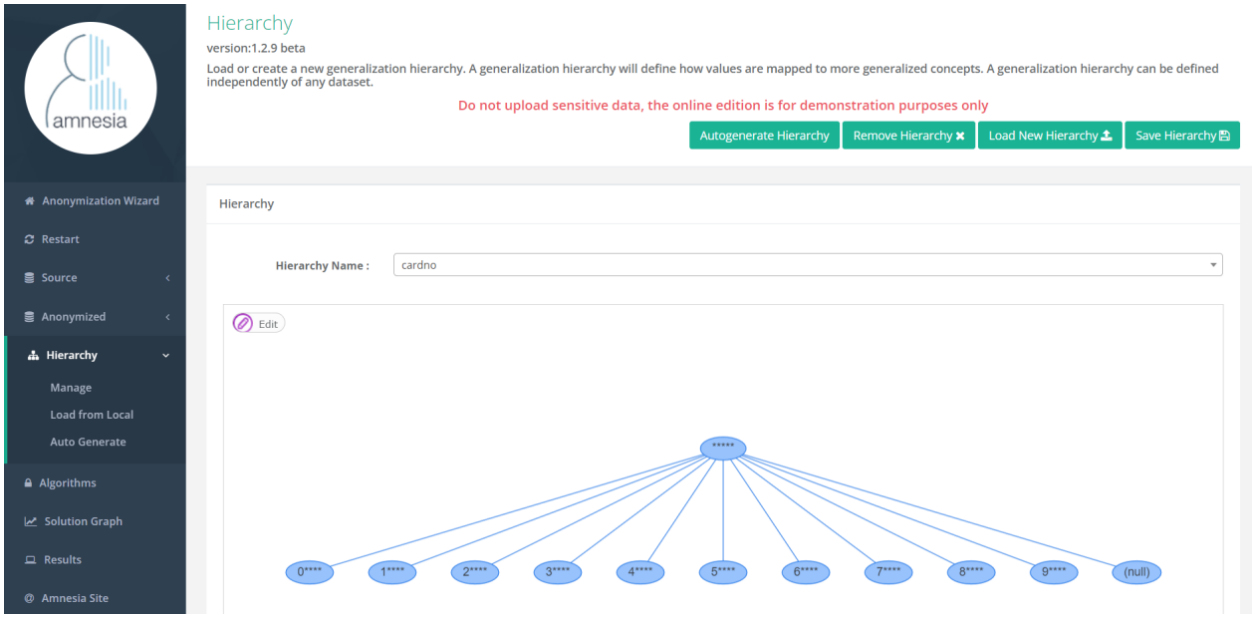
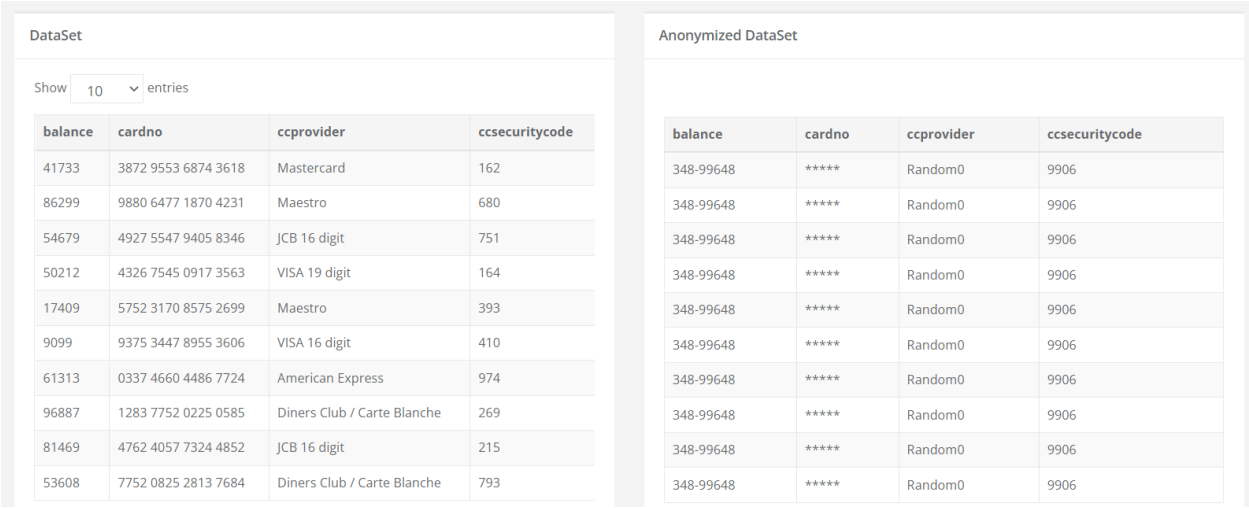


Figure 7. CARD NUMBER

By seeing the below datasets before anonymization and after anonymization we can say that the datasets are secured to use for the ATM transactions and preserve both privacy and utility.

Sample demo:



DataSet

Show 10 entries

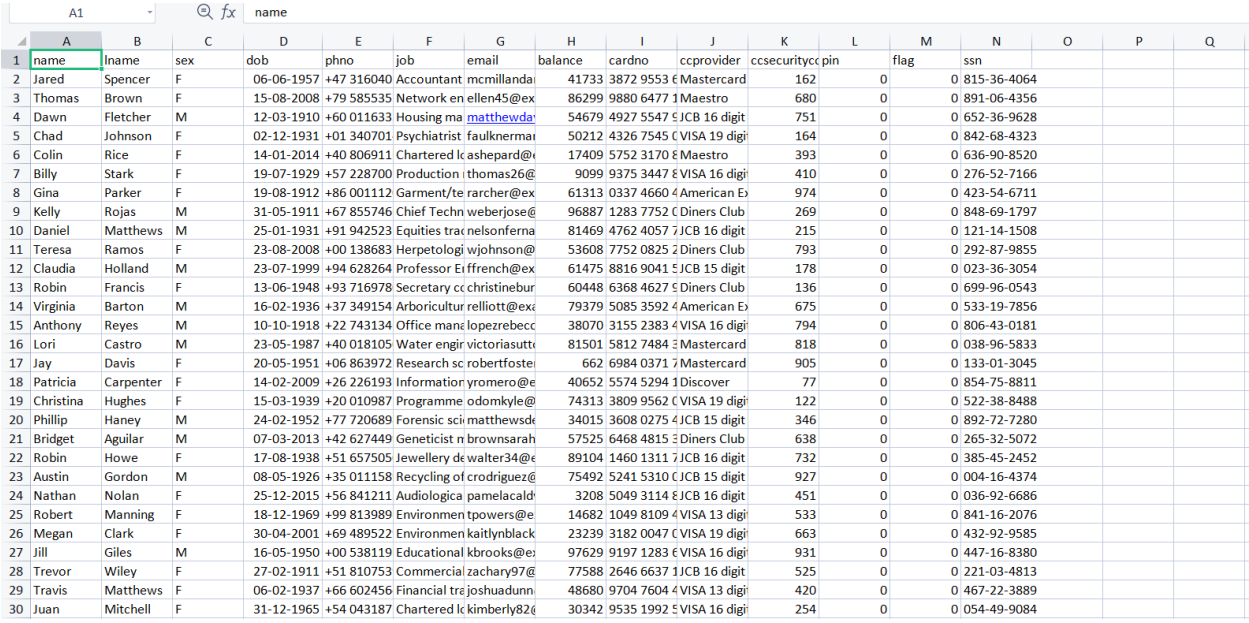
balance	cardno	ccprovider	ccsecuritycode
41733	3872 9553 6874 3618	Mastercard	162
86299	9880 6477 1870 4231	Maestro	680
54679	4927 5547 9405 8346	JCB 16 digit	751
50212	4326 7545 0917 3563	VISA 19 digit	164
17409	5752 3170 8575 2699	Maestro	393
9099	9375 3447 8955 3606	VISA 16 digit	410
61313	0337 4660 4486 7724	American Express	974
96887	1283 7752 0225 0585	Diners Club / Carte Blanche	269
81469	4762 4057 7324 4852	JCB 16 digit	215
53608	7752 0825 2813 7684	Diners Club / Carte Blanche	793

Anonymized DataSet

balance	cardno	ccprovider	ccsecuritycode
348-99648	*****	Random0	9906
348-99648	*****	Random0	9906
348-99648	*****	Random0	9906
348-99648	*****	Random0	9906
348-99648	*****	Random0	9906
348-99648	*****	Random0	9906
348-99648	*****	Random0	9906
348-99648	*****	Random0	9906
348-99648	*****	Random0	9906
348-99648	*****	Random0	9906

Figure 8.COMPARISION

Our dataset:



A1 fx name

	name	lname	sex	dob	phno	job	email	balance	cardno	ccprovider	ccsecurityco	pin	flag	ssn
2	Jared	Spencer	F	06-06-1957	+47 316040	Accountant	mcmillanda	41733	3872 9553 6	Mastercard	162	0	0	815-36-4064
3	Thomas	Brown	F	15-08-2008	+79 585535	Network en	ellen45@ex	86299	9880 6477 1	Maestro	680	0	0	891-06-4356
4	Dawn	Fletcher	M	12-03-1910	+60 011633	Housing ma	matthewda	54679	4927 5547 9	JCB 16 digit	751	0	0	652-36-9628
5	Chad	Johnson	F	02-12-1931	+01 340701	Psychiatrist	faulknermar	50212	4326 7545 0	VISA 19 digi	164	0	0	842-68-4323
6	Colin	Rice	F	14-01-2014	+40 806911	Chartered lo	ashepard@e	17409	5752 3170 8	Maestro	393	0	0	636-90-8520
7	Billy	Stark	F	19-07-1929	+57 228700	Production i	thomas26@	9099	9375 3447 8	VISA 16 digi	410	0	0	276-52-7166
8	Gina	Parker	F	19-08-1912	+86 001112	Garment/te	rarcher@ex	61313	0337 4660 4	American Ex	974	0	0	423-54-6711
9	Kelly	Rojas	M	31-05-1911	+67 855746	Chief Techn	weberjose@	96887	1283 7752 0	Diners Club	269	0	0	848-69-1797
10	Daniel	Matthews	M	25-01-1931	+91 942523	Equities trac	nelsonferna	81469	4762 4057 7	JCB 16 digit	215	0	0	121-14-1508
11	Teresa	Ramos	F	23-08-2008	+00 138683	Herpetologi	wjohnson@	53608	7752 0825 2	Diners Club	793	0	0	292-87-9855
12	Claudia	Holland	M	23-07-1999	+94 628264	Professor Ei	ffrench@ex	61475	8816 9041 5	JCB 15 digit	178	0	0	023-36-3054
13	Robin	Francis	F	13-06-1948	+93 716978	Secretary co	christinebur	60448	6368 4627 9	Diners Club	136	0	0	699-96-0543
14	Virginia	Barton	M	16-02-1936	+37 349154	Arboriculture	relliott@exa	79379	5085 3592 4	American Ex	675	0	0	533-19-7856
15	Anthony	Reyes	M	10-10-1918	+22 743134	Office mana	lopezrebecc	38070	3155 2383 4	VISA 16 digi	794	0	0	806-43-0181
16	Lori	Castro	M	23-05-1987	+40 018105	Water engir	victoriasutt	81501	5812 7484 3	Mastercard	818	0	0	038-96-5833
17	Jay	Davis	F	20-05-1951	+06 863972	Research sc	robertfoste	662	6984 0371 7	Mastercard	905	0	0	133-01-3045
18	Patricia	Carpenter	F	14-02-2009	+26 226193	Information	yromero@e	40652	5574 5294 1	Discover	77	0	0	854-75-8811
19	Christina	Hughes	F	15-03-1939	+20 010987	Programme	odomkyle@	74313	3809 9562 0	VISA 19 digi	122	0	0	522-38-8488
20	Phillip	Haney	M	24-02-1952	+77 720689	Forensic sci	matthewsde	34015	3608 0275 4	JCB 15 digit	346	0	0	892-72-7280
21	Bridget	Aguilar	M	07-03-2013	+42 627449	Geneticist n	brownsarah	57525	6468 4815 3	Diners Club	638	0	0	265-32-5072
22	Robin	Howe	F	17-08-1938	+51 657505	Jewellery de	walter34@e	89104	1460 1311 7	JCB 16 digit	732	0	0	385-45-2452
23	Austin	Gordon	M	08-05-1926	+35 011158	Recycling of	crodriguez@	75492	5241 5310 0	JCB 15 digit	927	0	0	004-16-4374
24	Nathan	Nolan	F	25-12-2015	+56 841211	Audiologica	pamelacald	3208	5049 3114 8	JCB 16 digit	451	0	0	036-92-6686
25	Robert	Manning	F	18-12-1969	+99 813989	Environmen	tpowers@e	14682	1049 8109 4	VISA 13 digi	533	0	0	841-16-2076
26	Megan	Clark	F	30-04-2001	+69 489522	Environmen	kaitlynblack	23239	3182 0047 0	VISA 19 digi	663	0	0	432-92-9585
27	Jill	Giles	M	16-05-1950	+00 538119	Educational	kbrooks@ex	97629	9197 1283 6	VISA 16 digi	931	0	0	447-16-8380
28	Trevor	Wiley	F	27-02-1911	+51 810753	Commercial	zachary97@	77588	2646 6637 1	JCB 16 digit	525	0	0	221-03-4813
29	Travis	Matthews	F	06-02-1937	+66 602456	Financial tra	joshuadunn	48680	9704 7604 4	VISA 13 digi	420	0	0	467-22-3889
30	Juan	Mitchell	F	31-12-1965	+54 043187	Chartered lo	kimberly82@	30342	9535 1992 5	VISA 16 digi	254	0	0	054-49-9084

Figure 9. Dataset

Overall generalized hierarchy:

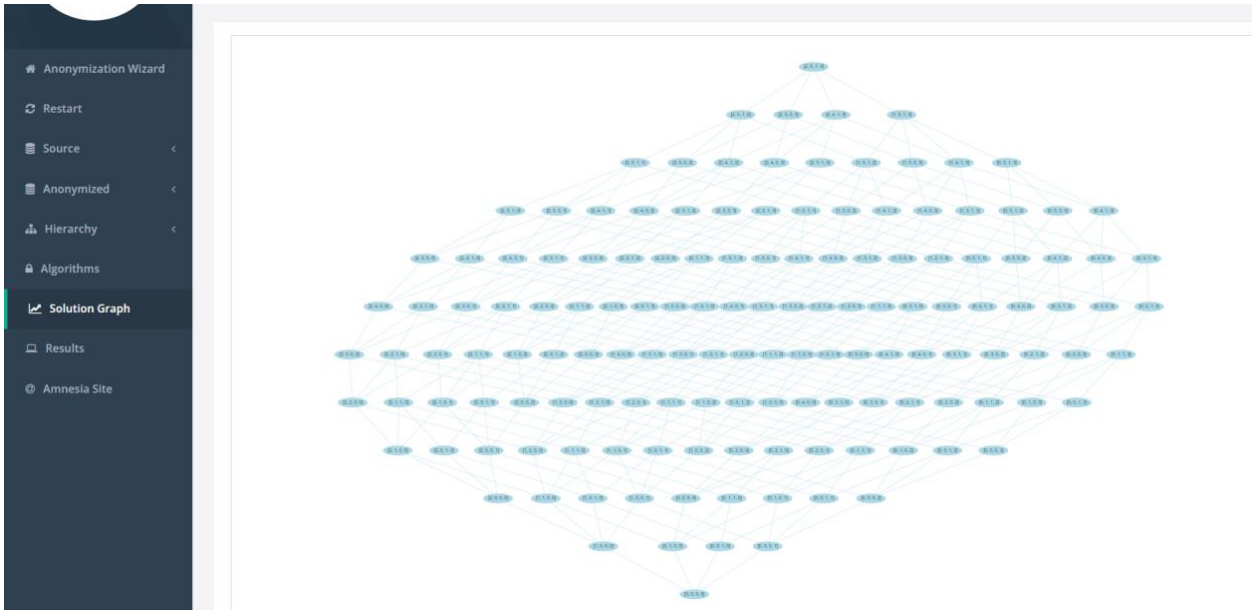


Figure 10. Generalized hierarchy

SQLiteStudio (3.2.1) - [acc (acc)]

Database Structure View Tools Help

Structure | Data | Constraints | Indexes | Triggers | DDL

Table name: acc WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	
1	name	STRING							NULL
2	lname	STRING							NULL
3	sex	CHAR							NULL
4	address	STRING							NULL
5	dob	DATE							NULL
6	phno	STRING							NULL
7	job	STRING							NULL
8	email	STRING							NULL
9	balance	INTEGER							NULL
10	cardno	STRING							NULL
11	ccprovider	STRING							NULL
12	ccsecuritycode	INTEGER							NULL
13	pin	INTEGER							0
14	flag	INTEGER							0
15	ssn	STRING							NULL

Figure 11. SQLite Database

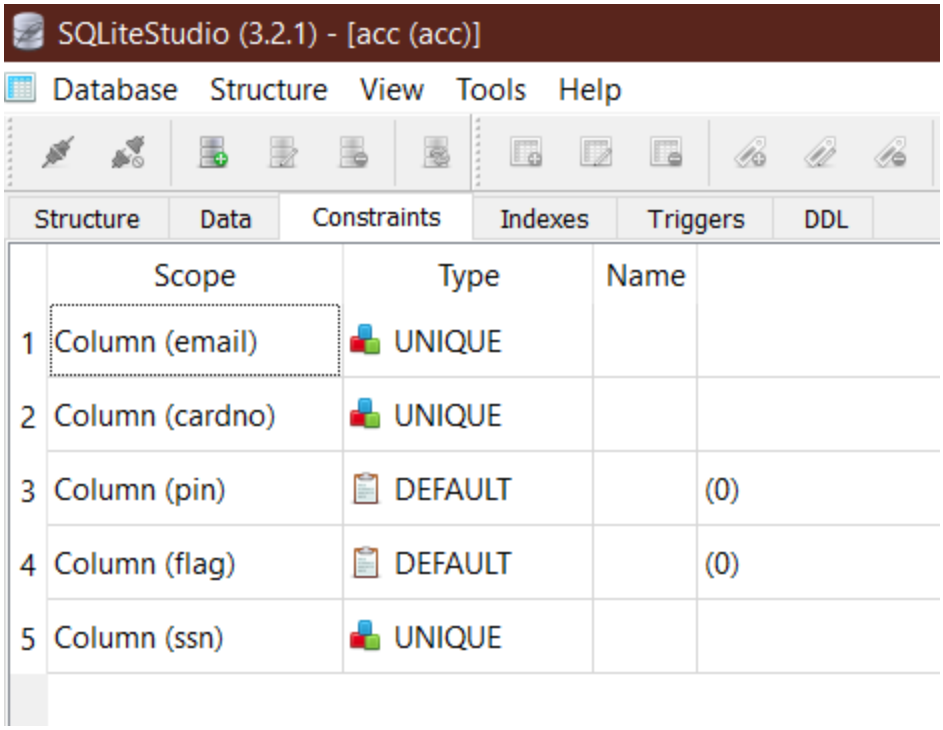


Figure 12. Constraints

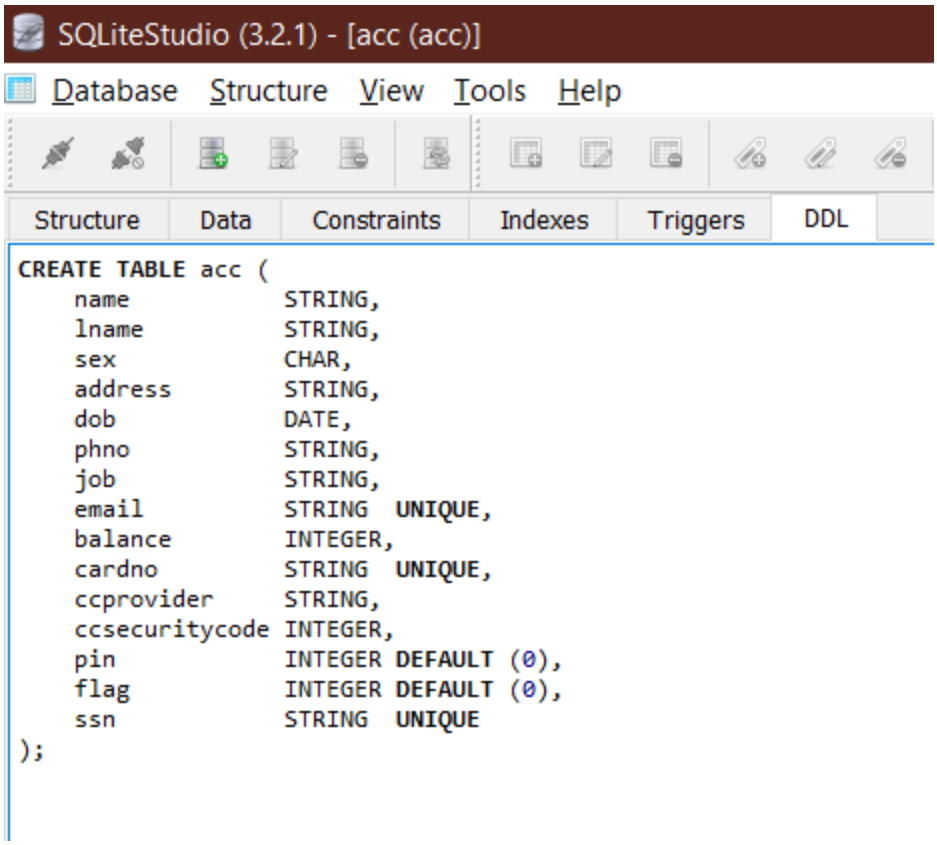


Figure 13. Table structure

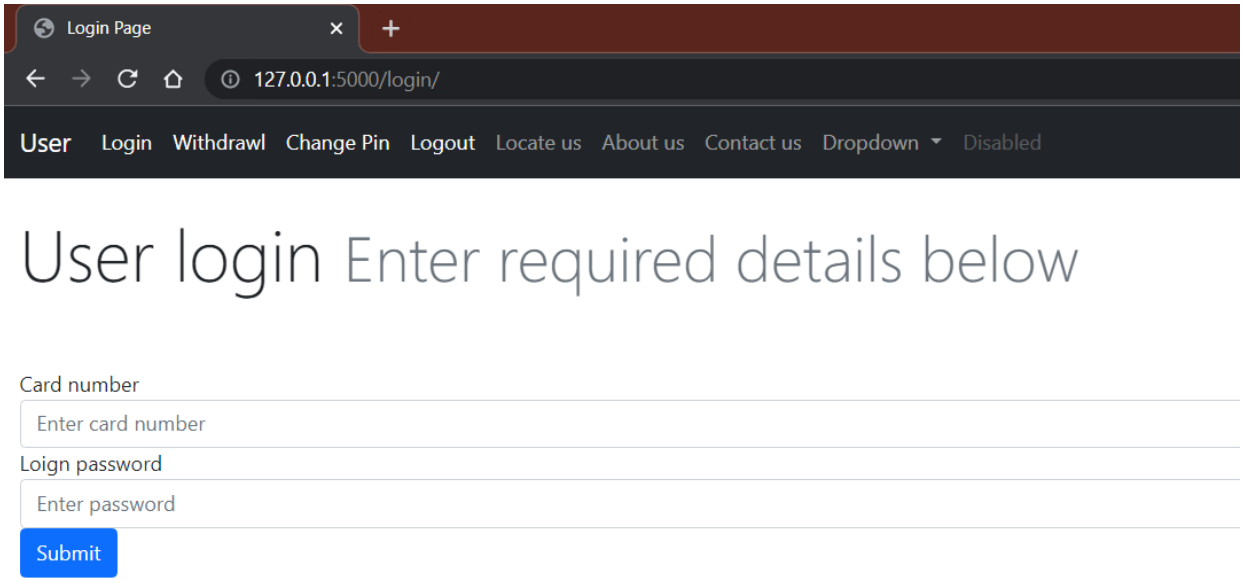


Figure 14: Page for user login

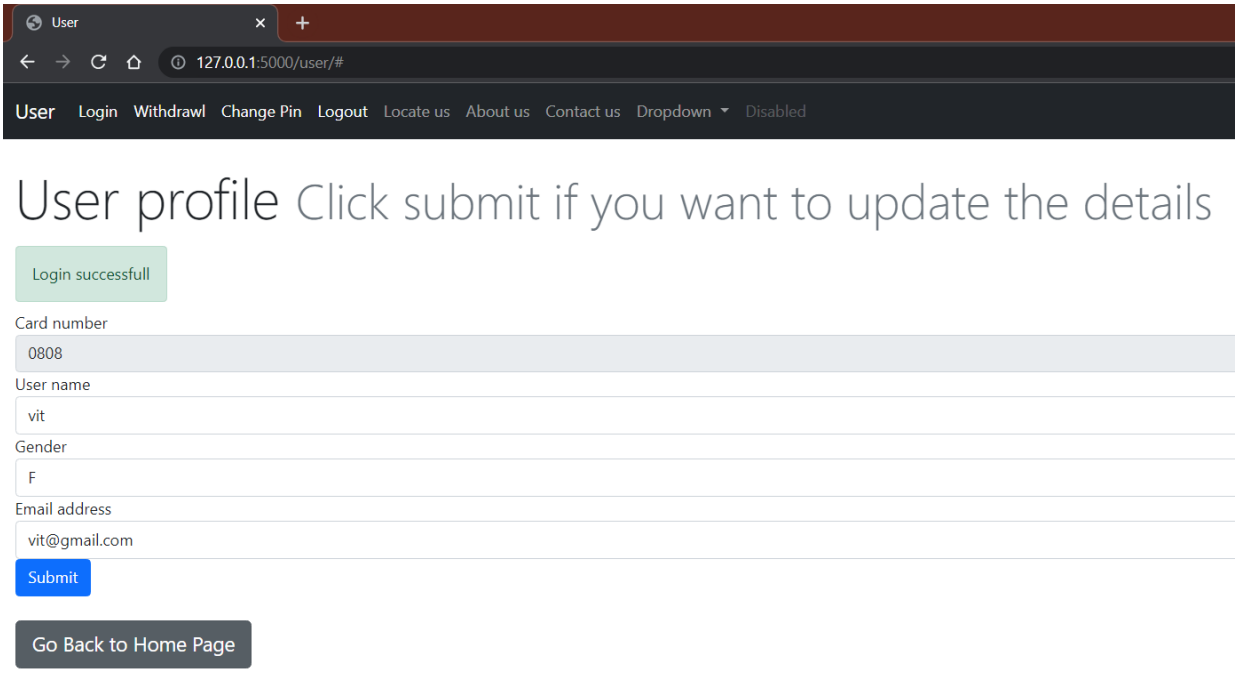


Figure 15: Page where we can see the user details

Above screenshot contains a sample user account which we used to login.. The available balances and other functionalities like change of pin/make new pin for new user are shown below.

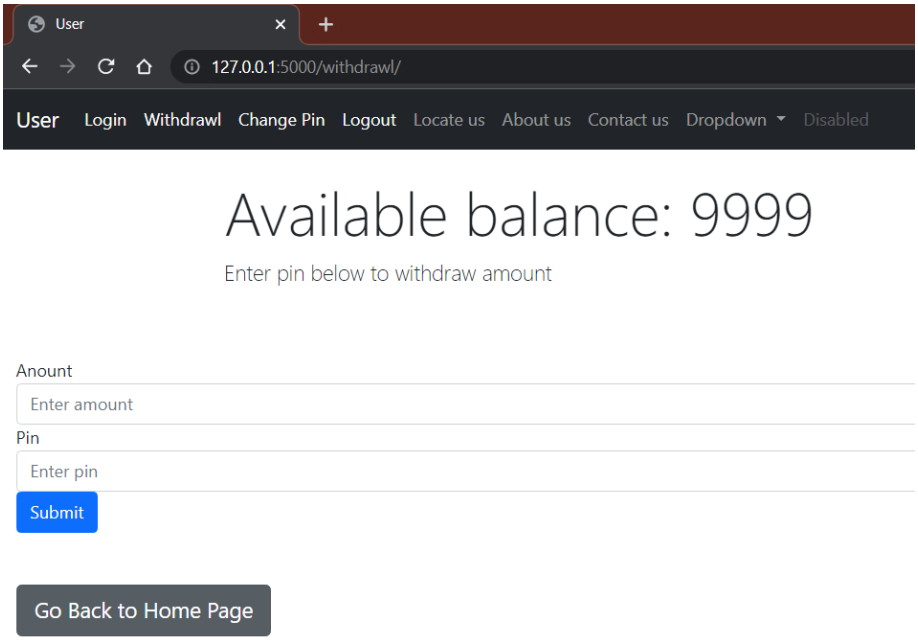


Figure 16: Page to make a withdraw

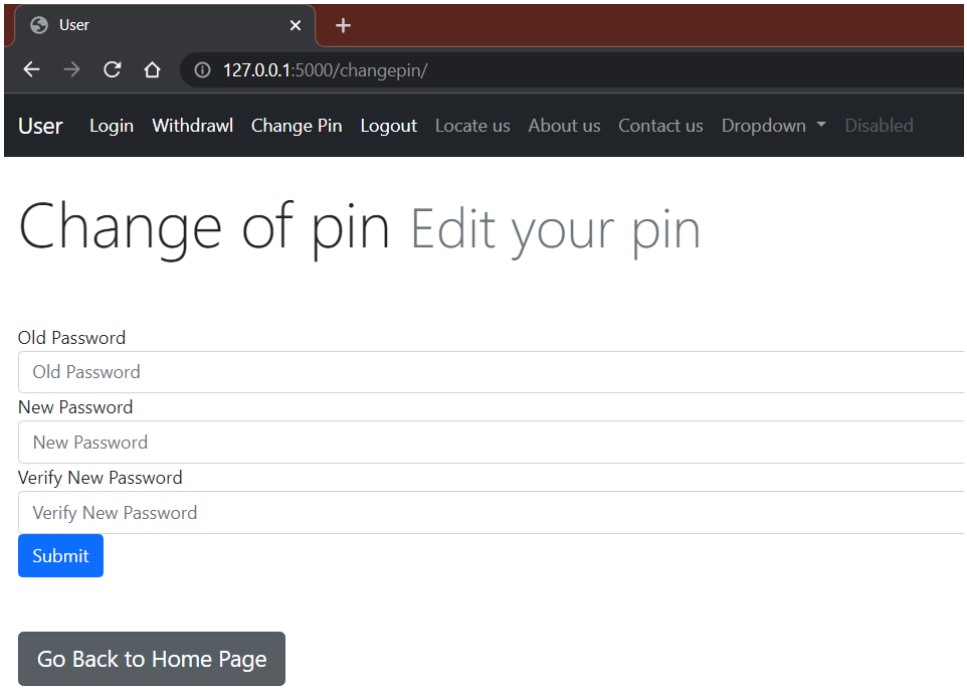


Figure 17: Page to facilitate change of pin

CONCLUSION

We have made an ATM Transaction System (Secured Data Encryption System) using Python language and it is storing details in encrypted form of user ensuring security to user. The Hash function prevents the Man in Middle attack.

We created the dataset from the frontend that we implemented, and that dataset we used is completely anonymized and secured in encrypted form using Flash algorithm with Amnesia tool. It can be implemented in the real world for personal uses.

CITATION

Privacy Preservation in Transactions using Flash K-anonymity, Chinthala Hardheek(19BCI0163), Shiv Thaker(19BCI0167), Konijeti Mahesh sai(19BCI0164), Vellore Institute of Technology-Vellore, BCI2001-Data privacy, Prof. Vijaya Kumar K, December 2021.

REFERENCES

[1] F. Kohlmayer, F. Prasser, C. Eckert, A. Kemper and K. A. Kuhn, "Flash: Efficient, Stable and Optimal K-Anonymity," 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing, 2012, pp. 708-717, doi: 10.1109/SocialCom-PASSAT.2012.52.

[2] JOUR, Byali, Megha,Chaudhari, Harsh,Patra, Arpita,Suresh, Ajith,2020/04/01,459,480, FLASH: Fast and Robust Framework for Privacy-preserving Machine Learning,2020, 10.2478/popets-2020-0036, Proceedings on Privacy Enhancing Technologies.

[3] Yingjie Wu ,School of Computer Science and Engineering, Southeast University, Nanjing, China, College of Mathematic and Computer Science, Fuzhou University, Fuzhou, China, yjwu@fzu.edu.cn,

Shangbin Liao, Xiaowen Ruan, Xiaodong Wang, College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, N080320044@fzu.edu.cn

[4] Bruno Ferreira, Miguel Ferreira, and Luís Faria, KEEP SOLUTIONS and José Carlos Ramalho, University of Minho ISSN: 2048-7916.

[5] VINEET RICHHARIYA1 & PRATEEK CHOUREY2

1.HOD, Department of Computer Science & Engineering, LNCT, Bhopal, Madhya Pradesh, India
2.Research Scholar, Department of Computer Science & Engineering, LNCT, Bhopal, Madhya Pradesh, India.

[6] JINYAN WANG, CHAOJI DENG, AND XIANXIAN LI

1Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin, 541004, China
2School of Computer Science and Information Technology, Guangxi Normal University, Guilin, 541004, China
Corresponding author: Xianxian Li (e-mail: lixx@gxnu.edu.cn)

APPENDIX

Appendix - A: Below code provided is the backbone of UI implementation, defining how the functions interact with each other, the connection to the database and many more.

```
from flask import Flask, redirect, url_for, render_template, request, session, flash
from datetime import timedelta
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import InputRequired, Length, ValidationError
import hashlib

app = Flask(__name__)
app.secret_key = 'hello'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.sqlite3'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.permanent_session_lifetime = timedelta(minutes=5) # days=5

db = SQLAlchemy(app)

# create a model to store information in
# inherits db.Model, class will be a database model

class User(db.Model):
    _id = db.Column('id', db.Integer, primary_key=True)
    name = db.Column('name', db.String(10))
    email = db.Column('email', db.String(10))
    gender = db.Column('gender', db.String(5))
    card_no = db.Column('card_no', db.String(20))
    balance = db.Column('balance', db.Integer)
    login_password = db.Column('login_password', db.String(100))
    password = db.Column('password', db.String(100))
    existing_user = db.Column('existing_user', db.Boolean)
    # nullable=False, unique=True

    def __init__(self, name, gender, email, card_no, balance, existing_user, password,
login_password):
        self.name = name
        self.email = email
        self.gender = gender
        self.card_no = card_no
        self.balance = balance
```

```
        self.existing_user = existing_user
        self.password = password
        self.login_password = login_password


# class RegistrationFrom(FlaskForm):
#     user = StringField(validators=[InputRequired(), Length(min=1, max=10)],
render_kw={'placeholder': 'User name'})
#     password = PasswordField(validators=[InputRequired(), Length(min=1, max=10)],
render_kw={'placeholder': 'Password'})
#     submit = SubmitField('Submit')
#
#     def validate_username(self, user):
#         existing_user_name = User.query.filter_by(name=user.data).first()
#         if existing_user_name:
#             raise ValidationError('That username already exists. Please choose a different
one.')
#
#
# class LoginForm(FlaskForm):
#     user = StringField(validators=[InputRequired(), Length(min=1, max=10)],
render_kw={'placeholder': 'User name'})
#     password = PasswordField(validators=[InputRequired(), Length(min=1, max=10)],
render_kw={'placeholder': 'Password'})
#     submit = SubmitField('Login')


@app.route('/')
def home():
    return render_template('index.html')

@app.route('/login/', methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
        session.permanent = True
        # actually get the information that was in the form and use that and send it to for
example the user page so we can display the user's name
        card_no = request.form["card_no"]
        password = hashlib.sha256(request.form['password'].encode()).hexdigest()
        # set up some session data based on whatever information they typed in
        session['card_no'] = card_no

        found_user_frm_db = User.query.filter_by(card_no=card_no).first()
        if found_user_frm_db:
            if found_user_frm_db.login_password == password:
                session['nm'] = found_user_frm_db.name
```

```
                session['email'] = found_user_frm_db.email
                session['card_no'] = found_user_frm_db.card_no
                if found_user_frm_db.gender == '_':
                    session['gender'] = 'Rather not declare'
                else:
                    session['gender'] = found_user_frm_db.gender
                flash('Login successfull', 'info')
                return redirect(url_for('user'))
            else:
                flash('Wrong password - login denied')
                return render_template('login.html')
        else:
            flash('No account found with the given card number')
            return render_template('login.html')
            # usr = User('', '', card_no, '')
            # db.session.add(usr) # add the usr model to the database
            # db.session.commit()
    else:
        # if we hit the GET request, we can just render out the login template
        # meaning we did not click on the submit button, we just clicked on the /login page
        if 'card_no' in session:
            flash('Already logged in', 'info')
            return redirect(url_for('user'))
        return render_template('login.html')

@app.route("/logout/")
def logout():
    if 'card_no' in session:
        # user = session['user']
        flash('Logout successful', 'info') # ('msg', 'built in category')
    session.pop('nm', None)
    session.pop('email', None)
    session.pop('card_no', None)
    session.pop('existing_user', None)
    session.pop('balance', None)
    session.pop('gender', None)
    # render_template('login.html', alert='alert alert-success')
    return redirect(url_for('login'))

@app.route('/user/', methods=['POST', 'GET'])
def user():
    if 'card_no' in session:
        if request.method == 'POST':
            name = request.form['nm']
            card_no = request.form['card_no']
            email = request.form['email']
```

```
            gender = request.form['gender']
            session['nm'] = name
            session['card_no'] = card_no
            session['email'] = email
            session['gender'] = gender
            if gender == 'Rather not declare':
                gender = '_'
                session['gender'] = 'Rather not declare'

            found_user_frm_db = User.query.filter_by(card_no=card_no).first()
            found_user_frm_db.name = name
            found_user_frm_db.card_no = card_no
            found_user_frm_db.email = email
            found_user_frm_db.gender = gender
            db.session.commit()
            flash('New Details saved')
            return render_template('user.html', nm=session['nm'], email=session['email'],
card_no=session['card_no'], gender=session['gender'], alert='alert alert-success')
        else:
            return render_template('user.html', nm=session['nm'], email=session['email'],
card_no=session['card_no'], gender=session['gender'], alert='alert alert-success')
    else:
        flash('You are not logged in', 'info')
        return redirect(url_for('login'))
        # , alert = 'alert alert-danger'

@app.route('/changepin/', methods=['POST', 'GET'])
def changepin():
    # card_no = None
    nm = None
    email = None
    password = None
    if 'card_no' in session:
        card_no = session['card_no']
        found_user_frm_db = User.query.filter_by(card_no=card_no).first()
        session['existing_user'] = found_user_frm_db.existing_user
        existing_user = session['existing_user']

        if request.method == 'POST' and existing_user:
            oldpass = hashlib.sha256(request.form['oldpass'].encode()).hexdigest()
            newpass = hashlib.sha256(request.form['newpass'].encode()).hexdigest()
            verifynewpass = hashlib.sha256(request.form['verifynewpass'].encode()).hexdigest()

            if found_user_frm_db.password == oldpass:
                if newpass == verifynewpass:
                    found_user_frm_db.password = newpass
```

```
                    db.session.commit()
                    flash('New Password saved')
                    return render_template('changepin.html', alert='alert alert-success',
control=True)
                else:
                    flash('New Passwords do not match')
                    return render_template('changepin.html', alert='alert alert-danger',
control=True)
            else:
                flash('Old password not correct for change of passowrd')
                return render_template('changepin.html', alert='alert alert-danger',
control=True)

        elif request.method == 'POST' and not existing_user:
            newpass = hashlib.sha256(request.form['newpass'].encode()).hexdigest()
            verifynewpass = hashlib.sha256(request.form['verifynewpass'].encode()).hexdigest()

            if newpass == verifynewpass:
                found_user_frm_db.password = newpass
                found_user_frm_db.existing_user = True
                db.session.commit()
                flash('New Password saved')
                return render_template('changepin.html', alert='alert alert-success',
control=True)
            else:
                flash('New Passwords do not match')
                return render_template('changepin.html', alert='alert alert-danger',
control=False)

        elif not existing_user:
            return render_template('changepin.html', control=False)

        elif existing_user:
            return render_template('changepin.html', control=True)

        else:
            pass
        return render_template('changepin.html')
    else:
        flash('You are not logged in', 'info')
        return redirect(url_for('login'))

@app.route('/withdrawl/', methods=['POST', 'GET'])
def withdrawl():
    if 'card_no' in session:
        card_no = session['card_no']
```

```
        found_user_frm_db = User.query.filter_by(card_no=card_no).first()
        session['existing_user'] = found_user_frm_db.existing_user
        session['balance'] = found_user_frm_db.balance
        existing_user = session['existing_user']
        balance = session['balance']

        if request.method == 'POST' and existing_user:
            withdraw_amount = int(request.form['amount'])
            pin = hashlib.sha256(request.form['pin'].encode()).hexdigest()
            # pin = request.form['pin']
            if found_user_frm_db.password == pin:
                if found_user_frm_db.balance >= withdraw_amount:
                    found_user_frm_db.balance = found_user_frm_db.balance - withdraw_amount
                    db.session.commit()
                    flash('Amount withdrawn')
                    return render_template('withdrawl.html', alert='alert alert-success',
control=True, balance=found_user_frm_db.balance)
                else:
                    flash('Withdraw amount more than balance - withdrawl blocked')
                    return render_template('withdrawl.html', alert='alert alert-danger',
control=True, balance=found_user_frm_db.balance)
            else:
                flash('Wrong pin - withdrawl blocked')
                return render_template('withdrawl.html', alert='alert alert-danger',
control=True, balance=found_user_frm_db.balance)

        elif request.method == 'POST' and not existing_user:
            return render_template('withdrawl.html', alert='alert alert-danger',
control=False)

        elif not existing_user:
            return render_template('withdrawl.html', control=False, balance=balance)

        elif existing_user:
            return render_template('withdrawl.html', control=True, balance=balance)

        else:
            pass
        return render_template('withdrawl.html')
    else:
        flash('You are not logged in', 'info')
        return redirect(url_for('login'))

@app.route('/view/')
def view():
    # get all the users and pass them as objects into our render template, to display info
```

```
    return render_template('view.html', values=User.query.all())

if __name__ == '__main__':
    # this will create the database if it doesn't already exist, very important
    db.create_all()
    app.run(debug=True)

# session, use while the user is browsing on the website
# as soon as they leave, it will dissapear, temporary, stored on the server, not on the client
side
# desinged for quick access of information, a way to pass information around the server
# all the session data is encrypted on the server, we need to define a secret key to be able
to encrypt and decrypt data



# .delete() on one specific object

'''
credentials
login pin
op  8989
io  abc
ui  9999
kl 6767
ooo 4545
uuu
'''
```