

SECURE VAULT

**Submitted in partial fulfillment of the requirements for the award
of Bachelor of Engineering degree in Computer Science and
Engineering**

by

Adarsh Kumar, Anshaj P Nair (Reg no: 38110038, 38110075)



**DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING**

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

Accredited with Grade "A" by NAAC

APRIL - 2020

DECLARATION

I **Anshaj P Nair, Bellamkonda Sai Varshith** hereby declare that the Project Report entitled “**SECURE VAULT**” done by me under the guidance of **Dr.S. Prayla Shyry, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science.

.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF Computer Science and Engineering

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the confide work of **Adarsh Kumar, Anshaj P Nair (Reg no: 38110038, 38110075))** who carried out the project entitled “**SECURE VAULT** “**under** my supervision from **November 2019** to **April 2020**.

Internal Guide

Dr.S. Prayla Shyry, M.E., Ph.D

Head of the Department

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examine

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D.**, Dean, School of Computing, **Dr. S. VIGNESHWARI, ME., Ph.D.**, and **Dr. L. Lakshmanan, ME., Ph.D.**, Head of the Department, Dept. of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.S. Prayla Shyry, M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of COMPUTER SCIENCE AND ENGINEERING, who were helpful in many ways for the completion of the project.

ABSTRACT

In today's Modern Age, where everything we do is tracked by someone or the other by legal or illegal means. Privacy and data security is a major concern for many. Leaked Passwords, Images, or messages cause a lot of trouble for many people. These leaks could happen via System hack, People taking screenshots, unauthorized person reading text within your system (Shared PC). The website WallPart (intentionally not linked to) claims to be "the world's largest online shop of posters...with over 10 billion images." What they do not tell you is that their database is filled with stolen and copyrighted images from photographers around the world. Leaked Passwords lead to many compromised accounts all over the world. Even One-One messages are being seen by some 3rd party over the internet.

Almost 4 out of 5 online users have same password for almost all the webIDs. This leads to a hectic security issue which the service provider face, be it google or Facebook. They are big tech giants and can still give provident security to the users, but not every webId is secure from cyber-attacks. We must not take this lightly that someone can breech our webId and take our personal data. Hackers need not to attack the most secure webId like Gmail or yahoo, but any other id i.e. that has the same password because it's convenient to have a common password. hackers make use of this common mistake every web user do. That is to have a common password everywhere. They can retrieve password from less secure webId and then use them to access strongly secure sites. We can avoid it, web vulnerability, by having many and multiple password on all WebId.

These are some of the problems that we may face one day. In order to overcome such problems faced by the technology users the SecureVault (Data security software) Window PC based application is not only Intuitive in use but also plays a pivotal role in privacy security. This Secure Vault software serves the purpose of storing multiple password just by remembering one password to login in the software, By using this software, it is easy we can have multiple Passwords on various webID, just having one unique VaultID and a simple secure password for the software, can provide a person a great edge of security in the world of internet.

This Project aims to provide a safe and easy way to store passwords, Messages and Images, Thus Protecting the privacy of User in windows Environment. Existing softwares either are expensive or does not contain all the features in one container which is not convenient for any user. The online tools pose a risk of data being stolen and many users mistrust Cloud Storage. This Project proposes to solve this by providing an offline software with all the features of text and image encryption. It also provides an encrypted way of sharing the information. This Software achieve this by using various cryptographic algorithm such as RSA, AES and substitution ciphers. This software can successfully store the information on the device and is capable of retrieving it to 100% accuracy. The Software takes at max 100MB of RAM space even while doing heavy computations. This Software can handle text of 100MB and Images of JPG and PNG type, which are the most common format for images. It performs good even on old end laptop that supports x64 windows operating system.

LIST OF CONTENTS

CHAPTER NO	CONTENT	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	vi
1	INTRODUCTION	1
	1.1 DOMAIN INTRODUCTION	1
	1.2 PROJECT INTRODUCTION	1
	1.3 OUTLINE OF PROJECT	2
2	LITERATURE SURVEY	3
	2.1 OVER USAGE	13
	2.2 SECURITY THREAT	15
	2.3 CONCLUSION	17
3	METHODOLOGY	20
	3.1 OBJECTIVE OF THE PROJECT:	20
	3.2 EXISTING SYSTEM	20
	3.3 PROPOSED SYSTEM	20
	3.4 MODIFICATION SYSTEM	20
	3.5 ADVANTAGES	21
	3.6 PLANS	21
	3.7 REQUIREMENTS	21
	3.8 DESIGN	22
	3.9 DEVELOPMENT TOOLS	23
	3.10 TESTING	23
	3.11 VISUAL C#	24
	3.12 SQLITE3	24
	3.13 AES ALGORITHM	26
	3.14 RSA ALGORITHM	28
	3.15 XAML	33

3.16 INKSCAPE	34
3.17 GUI LIBRARY	36
3.18 UPDATE LIBRARY	37
4 RESULTS AND DISCUSSION	38
4.1 WINDOWS	38
4.2 PERFORMANCE	42
5 CONCLUSION & FUTURE WORKS	44
5.1 SUMMARY	44
5.2 CONCLUSION	44
5.3 FUTURE WORKS	44
REFERENCE	45
APPENDIX	47
SAMPLE CODE	47
SCREENSHOT	52

LIST OF FIGURES

Figure No.	Title	Page No.
3.1	Architecture of Secure Vault	23
3.2	AES flow chart	28
3.3	RSA Work Flow	32
4.1	Main Window	38
4.2	SignIn Window	39
4.3	Register Window	39
4.4	Vault Window	40
4.5	Settings Window	41
4.6	Update Window	41
4.7	Debug Performance	42
4.8	Production Performance	43

CHAPTER 1

INTRODUCTION

1.1 DOMAIN INTRODUCTION

In this world of growing technologies everything is digitalized. With large number of works opportunities, the data Human mind should keep track of has also increased. Manual handling of the employee records poses huge difficulties and challenges. The use of paper work for handling the process of storing the information of the new and existing password could lead to human error, papers may end up in the wrong hands and moreover its time consuming, Thus, there is a need of software that can handle the WebId and password of large number of WebId. Similarly, we might need to keep some textual information in our system that we would not want to share, and some textual information that we want to share among a few. This could be a daunting task and not so easy for a normal user. This software can provide message security for the user and they will be able to encrypt the message with the same password.

1.2 PROJECT INTRODUCTION

The main objective of this project is to provide a user friendly yet less time-consuming process of storing Password information of multiple WebIds, Encryption of private messages using a single password of user's own, and a different password for sharing a private message, The file generated will be readable only by this software, This software will allow multiple Users on a same system. It will also provide the feature of Storing of images in the system itself but in an encrypted format that is unreadable by any software present in the system, user can share that file as well and thus protect or choose to share password to only a limited no. of users that the user feel worthy of seeing, it's will not be possible for a third person to look into the image without knowing the password that was set by the first user. This also increase the security for the cases where the hardware itself gets stolen or lost. It assures the user that only user can see the data and no one else.

Some of the project's scope are :

- Access to information and resources
- Ability to accommodate large user base
- Secure Password protection
- Secure Messages and provide storing and secure sharing
- Secure Images and provide storing and secure sharing
- Easy to use Interface
- Scalable software

The goal of the project is to deliver an Application software that can keep the password, message and image information of the user safe and retrievable, only by authorized people.

The objectives of the system are as follows:

- Design a system-based application Window to enter requirements like employee's web ID details and Password details.
- Well-designed database to store User information.
- A user friendly front-end environment for the user to interact.

1.3 OUTLINE OF PROJECT

The project describes a small, in size, software which would be useful to any person who uses internet on daily basis. The user can have a look who uses the software

and can also have multiple accounts, along with many WebIDs and Password. Just only one VaultID and password. User could also Store or share their private messages as well as images.

- Cryptography or cryptology (from Ancient Greek: κρυπτός, romanized: kryptós "hidden, secret"; and γράφειν graphein, "to write", or -λογία -logia, "study", respectively) is the practice and study of techniques for secure communication in the presence of third parties called adversaries. More generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from reading private messages; various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation are central to modern cryptography. Modern cryptography exists at the intersection of the disciplines of mathematics, computer science, electrical engineering, communication science, and physics. Applications of cryptography include electronic commerce, chip-based payment cards, digital currencies, computer passwords, and military communications.
- Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message shares the decoding technique only with intended recipients to preclude access from adversaries. The cryptography literature often uses the names Alice ("A") for the sender, Bob ("B") for the intended recipient, and Eve ("eavesdropper") for the adversary. Since the development of rotor cipher machines in World War I and the advent of computers in World War II, the methods used to carry out cryptology have become increasingly complex and its application more widespread.
- Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system, but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure; theoretical advances, e.g., improvements in integer

factorization algorithms, and faster computing technology require these solutions to be continually adapted. There exist information-theoretically secure schemes that provably cannot be broken even with unlimited computing power—an example is the one-time pad—but these schemes are more difficult to use in practice than the best theoretically breakable but computationally secure mechanisms.

- The growth of cryptographic technology has raised a number of legal issues in the information age. Cryptography's potential for use as a tool for espionage and sedition has led many governments to classify it as a weapon and to limit or even prohibit its use and export. In some jurisdictions where the use of cryptography is legal, laws permit investigators to compel the disclosure of encryption keys for documents relevant to an investigation. Cryptography also plays a major role in digital rights management and copyright infringement of digital media. The first use of the term cryptograph (as opposed to cryptogram) dates back to the 19th century—originating from *The Gold-Bug*, a novel by Edgar Allan Poe.
- Until modern times, cryptography referred almost exclusively to encryption, which is the process of converting ordinary information (called plaintext) into unintelligible form (called ciphertext). Decryption is the reverse, in other words, moving from the unintelligible ciphertext back to plaintext. A cipher (or cypher) is a pair of algorithms that create the encryption and the reversing decryption. The detailed operation of a cipher is controlled both by the algorithm and in each instance by a "key". The key is a secret (ideally known only to the communicants), usually a short string of characters, which is needed to decrypt the ciphertext. Formally, a "cryptosystem" is the ordered list of elements of finite possible plaintexts, finite possible cyphertexts, finite possible keys, and the encryption and decryption algorithms which correspond to each key. Keys are important both formally and in actual practice, as ciphers without variable keys can be trivially broken with only the knowledge of the cipher used and are therefore useless (or even counter-productive) for most purposes. Historically, ciphers were often used directly for encryption or decryption without additional procedures such as authentication or integrity checks. There are two kinds of cryptosystems: symmetric and asymmetric. In symmetric systems the same key (the secret key) is used to encrypt and decrypt

a message. Data manipulation in symmetric systems is faster than asymmetric systems as they generally use shorter key lengths. Asymmetric systems use a public key to encrypt a message and a private key to decrypt it. Use of asymmetric systems enhances the security of communication. Examples of asymmetric systems include RSA (Rivest–Shamir–Adleman), and ECC (Elliptic Curve Cryptography). Symmetric models include the commonly used AES (Advanced Encryption Standard) which replaced the older DES (Data Encryption Standard).

- In colloquial use, the term "code" is often used to mean any method of encryption or concealment of meaning. However, in cryptography, code has a more specific meaning. It means the replacement of a unit of plaintext (i.e., a meaningful word or phrase) with a code word (for example, "wallaby" replaces "attack at dawn").
- Cryptanalysis is the term used for the study of methods for obtaining the meaning of encrypted information without access to the key normally required to do so; i.e., it is the study of how to crack encryption algorithms or their implementations. Some use the terms cryptography and cryptology interchangeably in English, while others (including US military practice generally) use cryptography to refer specifically to the use and practice of cryptographic techniques and cryptology to refer to the combined study of cryptography and cryptanalysis. English is more flexible than several other languages in which cryptology (done by cryptologists) is always used in the second sense above. RFC 2828 advises that steganography is sometimes included in cryptology.
- The study of characteristics of languages that have some application in cryptography or cryptology (e.g. frequency data, letter combinations, universal patterns, etc.) is called cryptolinguistics.
- Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key (or, less commonly, in which their keys are different, but related in an easily computable way). This was the only kind of encryption publicly known until June 1976.
- Symmetric key ciphers are implemented as either block ciphers or stream ciphers. A block cipher enciphers input in blocks of plaintext as opposed to individual characters, the input form used by a stream cipher.

- The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are block cipher designs that have been designated cryptography standards by the US government (though DES's designation was finally withdrawn after the AES was adopted). Despite its deprecation as an official standard, DES (especially its still-approved and much more secure triple-DES variant) remains quite popular; it is used across a wide range of applications, from ATM encryption to e-mail privacy and secure remote access. Many other block ciphers have been designed and released, with considerable variation in quality. Many, even some designed by capable practitioners, have been thoroughly broken, such as FEAL.
- Stream ciphers, in contrast to the 'block' type, create an arbitrarily long stream of key material, which is combined with the plaintext bit-by-bit or character-by-character, somewhat like the one-time pad. In a stream cipher, the output stream is created based on a hidden internal state that changes as the cipher operates. That internal state is initially set up using the secret key material. RC4 is a widely used stream cipher. Block ciphers can be used as stream ciphers.
- Cryptographic hash functions are a third type of cryptographic algorithm. They take a message of any length as input, and output a short, fixed length hash, which can be used in (for example) a digital signature. For good hash functions, an attacker cannot find two messages that produce the same hash.
- MD4 is a long-used hash function that is now broken; MD5, a strengthened variant of MD4, is also widely used but broken in practice. The US National Security Agency developed the Secure Hash Algorithm series of MD5-like hash functions: SHA-0 was a flawed algorithm that the agency withdrew; SHA-1 is widely deployed and more secure than MD5, but cryptanalysts have identified attacks against it; the SHA-2 family improves on SHA-1, but is vulnerable to clashes as of 2011; and the US standards authority thought it "prudent" from a security perspective to develop a new standard to "significantly improve the robustness of NIST's overall hash algorithm toolkit." Thus, a hash function design competition was meant to select a new U.S. national standard, to be called SHA-3, by 2012. The competition ended on October 2, 2012 when the NIST announced that Keccak would be the new SHA-3 hash algorithm. Unlike block and stream ciphers that are invertible, cryptographic

hash functions produce a hashed output that cannot be used to retrieve the original input data. Cryptographic hash functions are used to verify the authenticity of data retrieved from an untrusted source or to add a layer of security.

- Message authentication codes (MACs) are much like cryptographic hash functions, except that a secret key can be used to authenticate the hash value upon receipt; this additional complication blocks an attack scheme against bare digest algorithms, and so has been thought worth the effort.
- Symmetric-key cryptosystems use the same key for encryption and decryption of a message, although a message or group of messages can have a different key than others. A significant disadvantage of symmetric ciphers is the key management necessary to use them securely. Each distinct pair of communicating parties must, ideally, share a different key, and perhaps for each ciphertext exchanged as well. The number of keys required increases as the square of the number of network members, which very quickly requires complex key management schemes to keep them all consistent and secret.
- In a groundbreaking 1976 paper, Whitfield Diffie and Martin Hellman proposed the notion of public-key (also, more generally, called asymmetric key) cryptography in which two different but mathematically related keys are used—a public key and a private key. A public key system is so constructed that calculation of one key (the 'private key') is computationally infeasible from the other (the 'public key'), even though they are necessarily related. Instead, both keys are generated secretly, as an interrelated pair. The historian David Kahn described public-key cryptography as "the most revolutionary new concept in the field since polyalphabetic substitution emerged in the Renaissance".
- In public-key cryptosystems, the public key may be freely distributed, while its paired private key must remain secret. In a public-key encryption system, the public key is used for encryption, while the private or secret key is used for decryption. While Diffie and Hellman could not find such a system, they showed that public-key cryptography was indeed possible by presenting the Diffie–Hellman key exchange protocol, a solution that is now widely used in secure communications to allow two

parties to secretly agree on a shared encryption key. The X.509 standard defines the most commonly used format for public key certificates.

- Diffie and Hellman's publication sparked widespread academic efforts in finding a practical public-key encryption system. This race was finally won in 1978 by Ronald Rivest, Adi Shamir, and Len Adleman, whose solution has since become known as the RSA algorithm.
- The Diffie–Hellman and RSA algorithms, in addition to being the first publicly known examples of high-quality public-key algorithms, have been among the most widely used. Other asymmetric-key algorithms include the Cramer–Shoup cryptosystem, ElGamal encryption, and various elliptic curve techniques.
- A document published in 1997 by the Government Communications Headquarters (GCHQ), a British intelligence organization, revealed that cryptographers at GCHQ had anticipated several academic developments. Reportedly, around 1970, James H. Ellis had conceived the principles of asymmetric key cryptography. In 1973, Clifford Cocks invented a solution that very similar in design rationale to RSA. And in 1974, Malcolm J. Williamson is claimed to have developed the Diffie–Hellman key exchange.
- Public-key cryptography is also used for implementing digital signature schemes. A digital signature is reminiscent of an ordinary signature; they both have the characteristic of being easy for a user to produce, but difficult for anyone else to forge. Digital signatures can also be permanently tied to the content of the message being signed; they cannot then be 'moved' from one document to another, for any attempt will be detectable. In digital signature schemes, there are two algorithms: one for signing, in which a secret key is used to process the message (or a hash of the message, or both), and one for verification, in which the matching public key is used with the message to check the validity of the signature. RSA and DSA are two of the most popular digital signature schemes. Digital signatures are central to the operation of public key infrastructures and many network security schemes (e.g., SSL/TLS, many VPNs, etc.).
- Public-key algorithms are most often based on the computational complexity of "hard" problems, often from number theory. For example, the hardness of RSA is

related to the integer factorization problem, while Diffie– Hellman and DSA are related to the discrete logarithm problem. The security of elliptic curve cryptography is based on number theoretic problems involving elliptic curves. Because of the difficulty of the underlying problems, most public- key algorithms involve operations such as modular multiplication and exponentiation, which are much more computationally expensive than the techniques used in most block ciphers, especially with typical key sizes. As a result, public-key cryptosystems are commonly hybrid cryptosystems, in which a fast high-quality symmetric-key encryption algorithm is used for the message itself, while the relevant symmetric key is sent with the message, but encrypted using a public-key algorithm. Similarly, hybrid signature schemes are often used, in which a cryptographic hash function is computed, and only the resulting hash is digitally signed.

- The goal of cryptanalysis is to find some weakness or insecurity in a cryptographic scheme, thus permitting its subversion or evasion. It is a common misconception that every encryption method can be broken. In connection with his WWII work at Bell Labs, Claude Shannon proved that the one-time pad cipher is unbreakable, provided the key material is truly random, never reused, kept secret from all possible attackers, and of equal or greater length than the message. Most ciphers, apart from the one-time pad, can be broken with enough computational effort by brute force attack, but the amount of effort needed may be exponentially dependent on the key size, as compared to the effort needed to make use of the cipher. In such cases, effective security could be achieved if it is proven that the effort required (i.e., "work factor", in Shannon's terms) is beyond the ability of any adversary. This means it must be shown that no efficient method (as opposed to the time-consuming brute force method) can be found to break the cipher. Since no such proof has been found to date, the one-time-pad remains the only theoretically unbreakable cipher.
- There are a wide variety of cryptanalytic attacks, and they can be classified in any of several ways. A common distinction turns on what Eve (an attacker) knows and what capabilities are available. In a ciphertext-only attack, Eve has access only to the ciphertext (good modern cryptosystems are usually effectively immune to

ciphertext-only attacks). In a known-plaintext attack, Eve has access to a ciphertext and its corresponding plaintext (or to many such pairs). In a chosen-plaintext attack, Eve may choose a plaintext and learn its corresponding ciphertext (perhaps many times); an example is gardening, used by the British during WWII. In a chosen-ciphertext attack, Eve may be able to choose ciphertexts and learn their corresponding plaintexts. Finally in a man-in-the-middle attack Eve gets in between Alice (the sender) and Bob (the recipient), accesses and modifies the traffic and then forwards it to the recipient. Also important, often overwhelmingly so, are mistakes (generally in the design or use of one of the protocols involved).

- Cryptanalysis of symmetric-key ciphers typically involves looking for attacks against the block ciphers or stream ciphers that are more efficient than any attack that could be against a perfect cipher. For example, a simple brute force attack against DES requires one known plaintext and 255 decryptions, trying approximately half of the possible keys, to reach a point at which chances are better than even that the key sought will have been found. But this may not be enough assurance; a linear cryptanalysis attack against DES requires 243 known plaintexts (with their corresponding ciphertexts) and approximately 243 DES operations. This is a considerable improvement over brute force attacks.
- Public-key algorithms are based on the computational difficulty of various problems. The most famous of these are the difficulty of integer factorization of semiprimes and the difficulty of calculating discrete logarithms, both of which are not yet proven to be solvable in polynomial time using only a classical Turing-complete computer. Much public-key cryptanalysis concerns designing algorithms in P that can solve these problems, or using other technologies, such as quantum computers. For instance, the best known algorithms for solving the elliptic curve-based version of discrete logarithm are much more time-consuming than the best known algorithms for factoring, at least for problems of more or less equivalent size. Thus, other things being equal, to achieve an equivalent strength of attack resistance, factoring-based encryption techniques must use larger keys than elliptic curve techniques. For this reason, public-key cryptosystems based on elliptic curves have become popular since their invention in the mid-1990s.

- While pure cryptanalysis uses weaknesses in the algorithms themselves, other attacks on cryptosystems are based on actual use of the algorithms in real devices, and are called side-channel attacks. If a cryptanalyst has access to, for example, the amount of time the device took to encrypt a number of plaintexts or report an error in a password or PIN character, he may be able to use a timing attack to break a cipher that is otherwise resistant to analysis. An attacker might also study the pattern and length of messages to derive valuable information; this is known as traffic analysis and can be quite useful to an alert adversary. Poor administration of a cryptosystem, such as permitting too short keys, will make any system vulnerable, regardless of other virtues. Social engineering and other attacks against humans (e.g., bribery, extortion, blackmail, espionage, torture, ...) are usually employed due to being more cost-effective and feasible to perform in a reasonable amount of time compared to pure cryptanalysis by a high margin.
- Much of the theoretical work in cryptography concerns cryptographic primitives—algorithms with basic cryptographic properties—and their relationship to other cryptographic problems. More complicated cryptographic tools are then built from these basic primitives. These primitives provide fundamental properties, which are used to develop more complex tools called cryptosystems or cryptographic protocols, which guarantee one or more high-level security properties. Note however, that the distinction between cryptographic primitives and cryptosystems, is quite arbitrary; for example, the RSA algorithm is sometimes considered a cryptosystem, and sometimes a primitive. Typical examples of cryptographic primitives include pseudorandom functions, one-way functions, etc.
- One or more cryptographic primitives are often used to develop a more complex algorithm, called a cryptographic system, or cryptosystem. Cryptosystems (e.g., El-Gamal encryption) are designed to provide particular functionality (e.g., public key encryption) while guaranteeing certain security properties (e.g., chosen-plaintext attack (CPA) security in the random oracle model). Cryptosystems use the properties of the underlying cryptographic primitives to support the system's security properties. As the distinction between primitives and cryptosystems is somewhat arbitrary, a sophisticated cryptosystem can be derived from a

combination of several more primitive cryptosystems. In many cases, the cryptosystem's structure involves back and forth communication among two or more parties in space (e.g., between the sender of a secure message and its receiver) or across time.

CHAPTER 2

LITERATURE SURVEY

2.1. OVER USAGE OF SAME PASSWORD, NO PROTECTION FOR INFORMATION SHARING AND IT'S STORAGE:

The over usage among all the internet users is found to be a common threat to their privacy leakages, also most users share critical information directly on the internet. It's understandable that humans can't memorize many information on a day to day basis, and we tend to forget many things. Being a social animal, we still fall under the limit of Dunbar's number, which means we can't maintain more than 150 close friendship. To ease our troubles most of us put same password everywhere. This may sound convenient to most of us, but this makes user's private data highly vulnerable to cyber-attacks. In today's age where we have more than one device connected to internet and we ourselves have more than one online account. Putting same password everywhere makes it's easier for the attacker to target one website then use the gathered information, like user's password. He could attack other websites more easily, and user could lose a lot of personal data in a single night.

User's also share private messages on chat or via simple email, this does look secure at the first glance but it's not. There could be a malware sitting on the website reading all user text while you type them, not only it could read what you have written but it would also know to whom you are sending this message. This may not affect directly to user; it could potentially give attacker more information about the user and his habits. This information could then be used against him in some or the other way. Giving away such information is bad and it's quite difficult to trust the platform to save you from such attack. Phishing attacks are quite common and a normal user wouldn't know until the users actually loses something via such attacks. Such attack is not always harmful from economical perspective but attacker may attack on user's image in society using this information. A common user won't think much before sharing such information on any online platform that he thinks is protected, but no guarantee of being so. There are some

advance attack techniques as well, which can take a lot from the user than they ever would want to allow. This kind of information is stored as an image. User's now a days prefer to post images online; all such images fall in public domain. Users are not bothered by any person who sees them as they have allowed this to happen, but they wouldn't want their images to be seen if they prefer to share it among friends and family. An attacker could get to user's system and steal user's images that are stored in user's system un protected, And, even if we keep them on some online cloud, these images are still vulnerable to cyber- attacks for previously mentioned reasons.

Images makes a big part of the user's personality on online platform, and thus they sometime are the prime target of attackers. As new machine learning models are being created in the world. The interest of companies and hackers have taken a spike in and around gathering data about users who uses online service on a day to day biases. As we give out our data willing or unknowingly. There might be someone who would be mapping our habits our visual models, how we change, what has changed. All of this information might be used to create and predicted user's action even before user make them. User's pictures could be fed to neural network to create deep fakes of the person. These are some real concerns of today's life and a normal person is not aware of it, and those who are aware are unable to do anything as their social network life is on cloud and lives on cloud. The mistrust in cloud service is the reason any aware user feels unsafe even with the best protection. For any general user, their system that they work on and do all their daily work will feel safer than any cloud service that is being provided to them. In this report I have mentioned that how much a common internet user is unaware about the data being stole or the potential risk of the user's data being stole or taken away by some third party, legally or illegally, or simply losing it via a cyber- attack. Even such cyber-attack is not recognized upfront till, something terrible happens to user's data and it comes under his attention. Hence, it's more essential for all internet user to safe guard the primary modes of information transfer which are passwords, messages and images.

Interaction between individuals that aim to estimate an underlying state of nature. In this context, the following questions are addressed: how can self-interested agents that

interact via social learning achieve a tradeoff between individual privacy and reputation of the social group? How can protocols be designed to prevent data incest in online reputation blogs where individuals make recommendations? How can sensing by individuals that interact with each other be used by a global decision maker to detect changes in the underlying state of nature? When individual agents possess limited sensing, computation, and communication capabilities, can a network of agents achieve sophisticated global behavior? Social and game-theoretic learning are natural settings for addressing these questions. This article presents an overview, insights, and discussion of social learning models in the context of data incest propagation, change detection, and coordination of decision-making.

2.2. SECURITY THREAT DESPITE ONLINE PROTECTION:

In September 2016, Yahoo confirmed a massive security breach that hackers swiped personal information associated with at least 500 million account that took in place in 2014, disclosed that a different attack in 2013 compromised more than 1 billion accounts. "This is the biggest data breach ever," said well-known cryptologist Bruce Schneier. The incident is a big deal, since so many have a Yahoo account of some type or other for email or finance or fantasy sports and so on. In the official statement given by Yahoo, The account information may have included names, email addresses, telephone numbers, dates of birth, hashed passwords (the vast majority with bcrypt) and, in some cases, encrypted or unencrypted security questions and answers. The ongoing investigation suggests that stolen information did not include unprotected passwords, payment card data, or bank account information; payment card data and bank account information are not stored in the system that the investigation has found to be affected. Based on the ongoing investigation, Yahoo believes that information associated with at least 500 million user accounts was stolen and the investigation has found no evidence that the state-sponsored actor is currently in Yahoos network.¹ Yahoo, as one of the internets busiest sites with one billion monthly users that is one of the oldest free email services, have many users built their digital identities around it, from their bank accounts to photo albums and even medical information.

Even though it is too early to say what impact the breach might have on Yahoo and its users because many questions remain, including the identity of the state-sponsored hackers behind it. But there are already some impact to Yahoo users and stakeholders. As Yahoo confirmed the stolen user information was being used primarily for spamming, i.e., sending spam to the people whose information was stolen. But since such info can often be passed around widely among criminal hackers, its always possible it could be used for more nefarious purposes. In the meantime, hackers posted to underground forums and online marketplaces what they claimed was stolen Yahoo data and offering a large collection of stolen Yahoo credentials including user names, easily cracked passwords, birth dates, ZIP codes and email addresses on a underground site where hackers can buy and sell stolen data. An infamous cybercriminal named Peace claimed on a website that he was selling credentials of 200 million Yahoo users from 2012 on the dark web for just over \$1,800. And the underground site uses Tor, the anonymity software, and Bitcoin, the digital currency, to hide the identities of buyers, sellers and administrators who are trading attack methods and stolen data.² Whats more, It will take Yahoo months and years' time before it regain users trust. Since Yahoo is a major webmail provider, there's one extra problem: any further service which has password reset emails sent to a Yahoo Mail account should also be considered compromised, and passwords accordingly changed.

No one knew the severity of the breach cloud-based file sharing giant Dropbox announced back in 2012. In fact, it wasn't until four years later that we learned what really happened. Hackers tapped into more than 68 million user accounts – email addresses and passwords included – representing nearly 5 gigabytes of data. But there's more! Those stolen credentials reportedly made their way to a dark web marketplace – the price for them was bitcoins. At the time, this was equivalent to roughly \$1,141. Dropbox responded by requesting a site-wide password reset from the user base. They also went into some generic spiel about its ongoing commitment to data security.

Some guys have all the luck – or not. Business-focused social networking site LinkedIn felt the sting of cyber criminals when some 6 million user passwords were stolen

then published on a Russian forum in 2012. Unfortunately, its streak of bad luck was just getting started. In May 2016, hackers stole and posted for sale on the dark web an estimated 167 million LinkedIn email addresses and passwords. In addition to changing their passwords, LinkedIn implemented two- way authentication, an optional feature that makes you enter a pin code on your mobile device prior to logging in to the network.

Hence it does seem justified we see mistrust in cloud services by many of online users.

2.3. CONCLUSION:

One of the most hated security policies is that of asking users to have different passwords for each different service they make use of. Many see this as unnecessary because they reason that if their chosen password is strong enough then having multiple passwords is an unnecessary precaution. Worse still, it is not just everyday users that ignore this best practice as we have seen by the HBGray federal compromise; even senior managers, up to the CEO, of a security company had been using the same password on multiple services.

Thinking that a single strong password is effective enough protection is flawed. A strong password might make it very difficult or even impossible for an attacker to crack it, but that's not the only risk that can result with your password being compromised.

We use many different services, from forums to games, and in each of these we are asked to create accounts for authentication. We have no way of knowing how these work, how secure they are or even if they are legitimate. What if you sign up with a web hosting service, use your really strong 20 character password (which even includes symbols!) and then it turns out that this service is storing that password as plain text?

It might seem highly unlikely but unfortunately it happens, as reported by The Register where InterWorx, a web hosting control panel system, suffered a data breach and they admitted that their system was storing client credentials in plain text. Those clients who used the same credentials on all servers and whose credentials were stolen

have figuratively opened their organization's (and home) doors to these hackers until they change their credentials.

It is also safe to assume there will always be a sizeable gap between identifying an attack, informing the users about it and the users changing their passwords on all their systems, thus giving attackers plenty of time to gain a foothold before the situation is resolved. Hence, having multiple password could save user from a lot of data loss.

E-mail is known to us all as a modern-day form of communication. In companies and organizations e-mail is often the main communication tool for internal use and when communicating with customers. Too often it is overlooked how insecure e-mail actually is – in reality e-mail is as easily read by outsiders as a postcard, and encrypting the message inside will help from any kind of miss hap that might happen. As even if the hacker get's your email he won't be able to read anything from it. Simply put, encryption is the method by which the communication can be scrambled in a way that no one else, except the intended recipient of the message, can intercept it or read its contents. However, this isn't entirely a foolproof solution, and any organization, armed with the right set of software tools, can un-encrypt a message that was not encrypted using end-to-end encryption. End-to-end encryption adds another layer of security, making it harder to be snooped on. While your favourite instant messenger app may be doing its best to secure your chats, there may still be some weak links. As a basic condition for secure chats, end-to-end encrypted messages require that both ends of the conversation should be using the same system for the chain to complete itself. if you tend to save or share screenshots of conversations, those shares are not secure, leading to information leaking out.

Finally, there is the whole privacy debate as well about apps themselves sharing user data with third parties. Instant messaging app WhatsApp is currently in the dock for sharing user names, phone numbers, device details, etc., with Facebook, after the popular social media network acquired WhatsApp. For all the complications in that debate, it is the sort of data sharing that as users, we have no control over.

Images are encrypted for many reasons, including identifying the creator of an image, protecting copyright information, deterring piracy, and blocking images from being viewed by users who shouldn't have access to them. By encrypting images, you can send them through email or over the Internet without worrying about your images are being viewed by people that you don't want to see them. Encrypting images on your home computer will also give you a measure of security in case a hacker gains access to your hard drive.

CHAPTER 3

METHODOLOGY

3.1 OBJECTIVE OF THE PROJECT

The objectives of the system is Design a system-based application Window to enter requirements like employee's web ID details and Password details. Well-designed database to store User information. A user friendly front-end environment for the user to interact.

3.2 EXISTING SYSTEM

Most of the information security system are present in cloud or as some kind of lock system. These either have been under a purchase window or under some subscription either monthly or annually. Some cloud service is free but may steal the user's data.

3.3 PROPOSED SYSTEM

In order to help common people in protecting their password offline on their own system which act like a dry system, user could feel much more safe and at much more power of such control. This also provides the user to safeguard their message either into their own system or they could share it with a new password. This protection is leveraged to images as well, user could do the same with images. They can have a password to store images as an unreadable format on their own system or share it with other using a different password.

3.4 MODIFICATION SYSTEM

I also have made several layers of complexity built into the software rather than having only one strong security wall. This ensure that even if some malicious person grab holds the user's data it will still take him far more longer than a simple computer can process this extra layers protects the data not only offline, but also online. Having these

many layers to all the aspects makes it a single software with all the feature of protection, i.e. password, message, and Image protection.

3.5 ADVANTAGES

- Works without internet
- Provide hassle free updates
- All 3 features in a single software

3.6 PLANS

Planning is important before starting the project. It ensures that there's a proper plan for executing on strategic goals. The plans created before starting the project helps to manage time, cost, quality, change, risk and issues.

The plans for this project are as follows:

1. Analyzing the project's objectives
2. Collection of requirements for the project
3. Designing the Application Window
4. Coding using VisualCsharp
5. Adding interactive designs using Buttons and Textbox
6. Testing

3.7 REQUIREMENTS

The materials and components needed for the project are gathered or collected. It consists of three primary steps: taking Window in right order and placement of textboxes, identifying which additional ones are needed and then scheduling their production.

The requirements for this project are as follows:

1. Csharp
2. SQLite
3. XAML
4. Visual studio toolkit
5. InkScape
6. UI Library
7. Update Library

3.8 DESIGN

The purpose of this project, "Design" means the visual design of the Window as well as the implementation details that underlie it. The design is document should describe what the finished Window will look like and how a person will make it work. An important part of the design will be a set transition from one Window to another. The final product of the design phase is a connected Window that contains a structured essay - a combination of code and Database. The Main Window should contain these components :-

1. MainWindow: Project begins from this Window.
2. Mini-Window: More windows appear when button is pressed, custom prompts
3. Side-Bar: This is an animated bar which helps in knowing the button.
4. Setting: This menu gives the options for updating the software .
5. Update : This is an Update system for the software.
6. SQLite : The database used for the application.

The User will start at “Main Window”, Then He can either SignUp if he is a new user, or continue with login window if he is an already existing user. At SignUp window the software do some validation about the given input and will guide user to put correct credentials.

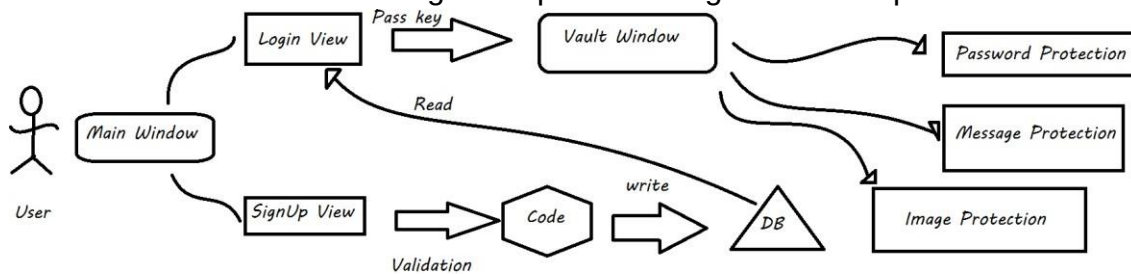


Fig 1, Architecture of Secure Vault

After that this will be fed to the backend code. Code will write a new entry in the database. Then user can go to login window, enter his credentials, if Pass keys are correct. User will enter into the vault and will get the options of protection. It provides Password, Message and Image protection.

3.9 DEVELOPMENT TOOLS

Building a software application has many coding languages, but for this have chosen Visual csharp. Microsoft Visual C# (often abbreviated to MSVCSHARP) is an integrated development environment (IDE) product from Microsoft for the C, C#, and C#/CLI programming languages. MSVCSHARP is proprietary software; it was originally a standalone product but later became a part of Visual Studio and made available in both trialware and freeware Window. It features tools for developing and debugging C# code, especially code written for the Windows API, DirectX and .Net.

3.10 TESTING

Application Testing in simple terms is checking the application for potential bugs before its made live or before code is moved into the production environment. Heavy Testing has been done on many host systems for this project. Testing all buttons in the Window working correctly and making sure there are no broken buttons. Testing the site Navigation, Menus, buttons or Links to different pages on the page and ensure that it should be easily visible and consistent on all Window.

3.11 VISUAL C#

The Visual C# Win32 Application Wizard allows a person to create any of four types of projects. In each case, a person can specify additional options that are appropriate for the type of project a person open. The following table indicates which options are available for each application type. A person can use a project template to create the basic program structure, menus, toolbars, icons, references, and using statements that are appropriate for the kind of project a person want to create. Visual Studio includes several kinds of Visual C# project templates and provides wizards for many of them so that a person can customize a person's projects as a person creates them. Immediately after a person create a project, a person can build it and run the application; it's good practice to build intermittently as a person develop a person's application.

A person don't have to use a template to create a project, but in most cases it's more efficient to do so because it's easier to modify the provided project files and structure than it is to create them from scratch. C#/CX is a set of extensions to the C# language that enable the creation of Windows apps and Windows Runtime components in an idiom that is as close as possible to modern C#. Use C#/CX to write Windows apps and components in native code that easily interact with Visual C#, Visual Basic, and JavaScript, and other languages that support the Windows Runtime. In those rare cases that require direct access to the raw COM interfaces, or non-exceptional code, a person can use the Windows Runtime C# Template Library (WRL). By using C#/CX, a person can create: C# Universal Windows Platform (UWP) apps that use XAML to define the user interface and use the native stack. For more information, see [Create a "hello world" app in C# \(UWP\)](#). C# Windows Runtime components that can be consumed by JavaScript-based Windows apps. For more information, see [Creating Windows Runtime Components in C#](#). Windows DirectX games and graphics-intensive apps. For more information, see [Create a simple UWP Game with DirectX](#).

3.12 SQLITE3

SQLite is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Wideness's daughter, and "SQL", the abbreviation for Structured Query Language. The SQLite development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. SQLite was owned and sponsored by a single for-profit firm, the Swedish company SQLite AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality. SQLite is a central component of the LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, SQLite, Perl/PHP/Python". Applications that use the SQLite database include: TYPO3, MODx, Joomla, WordPress, Simple Machines Forum, phpBB, MyBB, and Drupal. SQLite is also used in many high-profile, large-scale websites, including Google. SQLite is written in C and C#. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. SQLite works on many system platWindow, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of SQLite to OpenVMS also exists.

The SQLite software itself and the client libraries use dual-licensing distribution. They are offered under GPL version 2, beginning from 28 June 2000(which in 2009 has been extended with a FLOSS License Exception) or to use a proprietary license. Support can be obtained from the official manual. Free support additionally is available in different IRC channels and forums. Oracle offers paid support via its SQLite Enterprise products. They differ in the scope of services and in price. Additionally, a number of third-party organizations exist to provide support and services, including MariaDB and Percona. SQLite has received positive reviews, and reviewers noticed it "per Window extremely well in the average case" and that the "developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good". It has also been tested to be a "fast, stable and true multi-user, multi- threaded sql database ".

3.13 AES ALGORITHM

AES is another The Advanced Encryption Standard (AES), also known by its original name Rijndael, is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

AES is a subset of the Rijndael block cipher developed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen, who submitted a proposal to NIST during the AES selection process. Rijndael is a family of ciphers with different key and block sizes. For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits. AES has been adopted by the U.S. government and is now used worldwide. It supersedes the Data Encryption Standard (DES), which was published in 1977. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data. In the United States, AES was announced by the NIST as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001. This announcement followed a five-year standardization process in which fifteen competing designs were presented and evaluated, before the Rijndael cipher was selected as the most suitable (see Advanced Encryption Standard process for more details). AES became effective as a federal government standard on May 26, 2002, after approval by the Secretary of Commerce. AES is included in the ISO/IEC 18033-3 standard. AES is available in many different encryption packages, and is the first (and only) publicly accessible cipher approved by the National Security Agency (NSA) for top secret information when used in an NSA approved cryptographic module.

AES is symmetric block cipher algorithm. AES stands for Advanced Encryption Standard. AES uses 128 bit block size and the keys range from 128, 192, 256 bits. AES is highly recommended algorithm and is approved by NIST. In order to understand AES it is very important to understand the basics of Cryptography. If you need a refresher check out my previous blog on Cryptography concepts [Read Here](#). AES128 algorithm uses block size of 128 bits and key size of 128 bits. The 128 bit key is stretched via Key stretching function PRG into 11 keys of 128 bits each. Suppose we have a message M of arbitrary length.

The message is first divided into 128 bit blocks. Each block is then run through the round function for 10 iterations. Each round involves multiple steps as described below. Let's consider block m_0 which goes through the following steps to get the corresponding cipher block c_0 .

Step 1: 128 bits of block m_0 are bitwise XORed with 128 bit key k_0 .

Step 2: The resulting 128 bit block is then input into a substitution function s_0 which uses some mapping table to produce 128 bit of output.

Step 3: Some permutation function p_0 is then applied to the previous output to get an intermediate value m_{01} .

Step 4: This block m_{01} is then XORed with the key k_1 and the steps 1–3 are applied for 10 iterations one with each key.

Step 5: The output of the last iteration is then XORed with k_{10} to get resulting cipher block c_0 .

Step 6: Steps 1–5 are repeated for all the message blocks $m_0, m_1, \dots, m_n$ to get the resulting cipher $c_0, c_1, c_3, \dots, c_n$.

AES128 provides cryptographic strength of 2^{128} , which is above the minimum required strength of 2^{112} . However AES256 offers key size of 256 bits and thus offers far more strength of 2^{256} . In the age of quantum computing AES256 will be able to survive because quantum computers are considered to solve the problem in square root time instead of linear time, this will make AES256 key strength to 2^{128} which is considerably higher than 2^{112} . Although it will make DES obsolete, if you would like to know more about DES check my blog. AES is a subset of the Rijndael block cipher developed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen, who submitted a proposal to NIST during the AES selection process. . The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

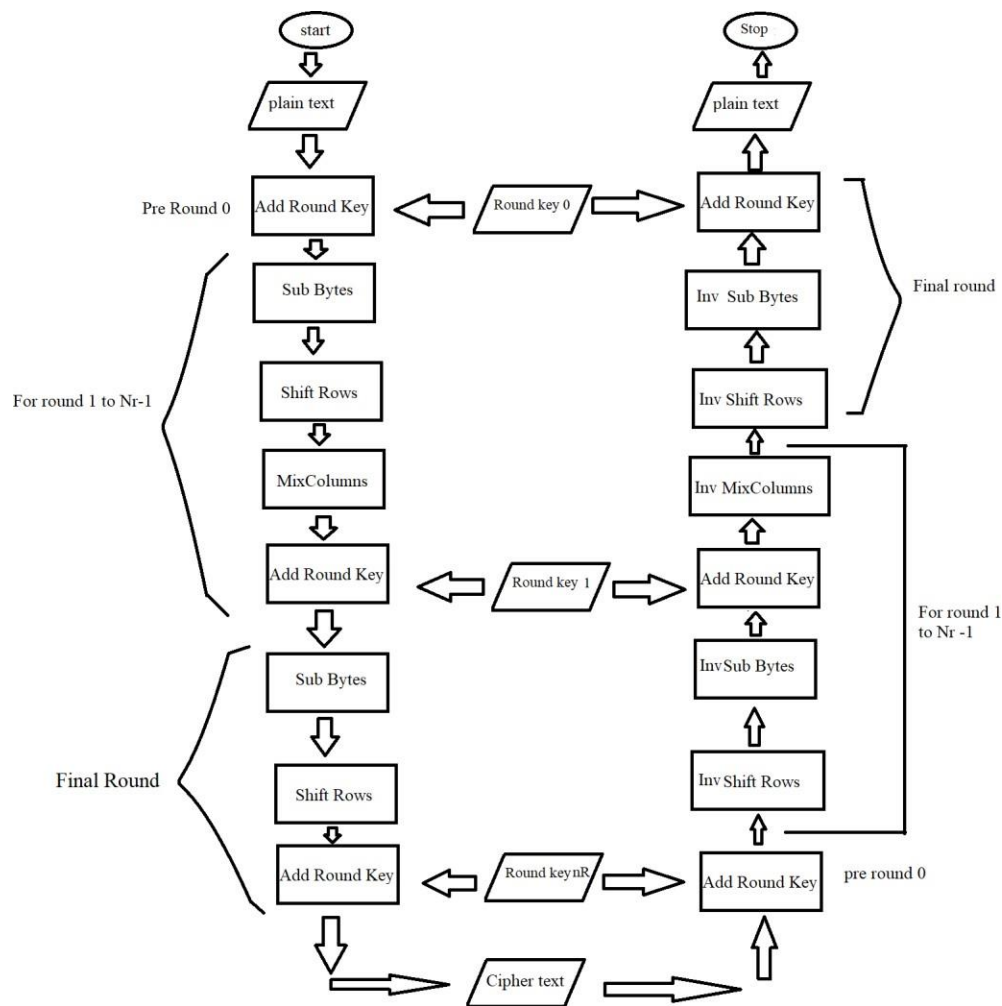


Fig. 3.2 AES Flow Chart

AES128 provides cryptographic strength of 2^{128} , which is above the minimum required strength of 2^{112} . However AES256 offers key size of 256 bits and thus offers far more strength of 2^{256} . In the age of quantum computing AES256 will be able to survive because quantum computers are considered to solve the problem in square root time instead of linear time, this will make AES256 key strength to 2^{128} which is considerably higher than 2^{112} . Although it will make DES obsolete, if you would like to know more about DES check my blog.

3.14 RSA ALGORITHM

RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is

public and distinct from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the “factoring problem”. The acronym RSA is the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1977. Clifford Cocks, an English mathematician working for the British intelligence agency Government Communications Headquarters (GCHQ), had developed an equivalent system in 1973, which was not declassified until 1997.

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but only someone with knowledge of the prime numbers can decode the message. Breaking RSA encryption is known as the RSA problem. Whether it is as difficult as the factoring problem is an open question. There are no published methods to defeat the system if a large enough key is used.

RSA is a relatively slow algorithm, and because of this, it is less commonly used to directly encrypt user data. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption-decryption operations at much higher speed. The idea of an asymmetric public-private key cryptosystem is attributed to Whitfield Diffie and Martin Hellman, who published this concept in 1976. They also introduced digital signatures and attempted to apply number theory. Their formulation used a shared-secret-key created from exponentiation of some number, modulo a prime number. However, they left open the problem of realizing a one-way function, possibly because the difficulty of factoring was not well-studied at the time.

Ron Rivest, Adi Shamir, and Leonard Adleman at the Massachusetts Institute of Technology, made several attempts over the course of a year to create a one-way function that was hard to invert. Rivest and Shamir, as computer scientists, proposed many potential functions, while Adleman, as a mathematician, was responsible for finding their weaknesses. They tried many approaches including “knapsack-based” and “permutation polynomials”. For a time, they thought what they wanted to achieve was impossible due to contradictory requirements. In April 1977, they spent Passover at the

house of a student and drank a good deal of Manischewitz wine before returning to their homes at around midnight. Rivest, unable to sleep, lay on the couch with a math textbook and started thinking about their one-way function. He spent the rest of the night formalizing his idea, and he had much of the paper ready by daybreak. The algorithm is now known as RSA – the initials of their surnames in same order as their paper.

Clifford Cocks, an English mathematician working for the British intelligence agency Government Communications Headquarters (GCHQ), described an equivalent system in an internal document in 1973. However, given the relatively expensive computers needed to implement it at the time, RSA was considered to be mostly a curiosity and, as far as is publicly known, was never deployed. His discovery, however, was not revealed until 1997 due to its top-secret classification.

Kid-RSA (KRSA) is a simplified public-key cipher published in 1997, designed for educational purposes. Some people feel that learning Kid-RSA gives insight into RSA and other public-key ciphers, analogous to simplified DES.

The keys for the RSA algorithm are generated in the following way: Choose two distinct prime numbers p and q . For security purposes, the integers p and q should be chosen at random, and should be similar in magnitude but differ in length by a few digits to make factoring harder. Prime integers can be efficiently found using a primality test.

p and q are kept secret. Compute $n = pq$.

n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.

n is released as part of the public key.

Compute $\lambda(n)$, where λ is Carmichael's totient function. Since $n = pq$, $\lambda(n) = \text{lcm}(\lambda(p), \lambda(q))$, and since p and q are prime, $\lambda(p) = \phi(p) = p - 1$ and likewise $\lambda(q)$

$= q - 1$. Hence $\lambda(n) = \text{lcm}(p - 1, q - 1)$.

$\lambda(n)$ is kept secret.

Choose an integer e such that $1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1$; that is, e and

$\lambda(n)$ are coprime.

E having a short bit-length and small Hamming weight results in more efficient encryption – the most commonly chosen value for e is $2^{16} + 1 = 65,537$. The smallest (and fastest) possible value for e is 3, but such a small value for e has been shown to be less secure in some settings.

E is released as part of the public key.

Determine d as $d \equiv e^{-1} \pmod{\lambda(n)}$; that is, d is the modular multiplicative inverse of e modulo $\lambda(n)$.

This means: solve for d the equation $d \cdot e \equiv 1 \pmod{\lambda(n)}$. D can be computed efficiently by using the Extended Euclidean algorithm.

D is kept secret as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e .

The private key consists of the private (or decryption) exponent d , which must be kept secret. P , q , and $\lambda(n)$ must also be kept secret because they can be used to calculate d . In fact, they can all be discarded after d has been computed.

In the original RSA paper, the Euler totient function $\phi(n) = (p - 1)(q - 1)$ is used instead of $\lambda(n)$ for calculating the private exponent d . Since $\phi(n)$ is always divisible by $\lambda(n)$ the algorithm works as well. The keys for the RSA algorithm are generated in the following way: Choose two distinct prime numbers p and q . For security purposes, the integers p and q should be chosen at random, and should be similar in magnitude but differ in length by a few digits to make factoring harder. Prime integers can be efficiently found using a primality test.

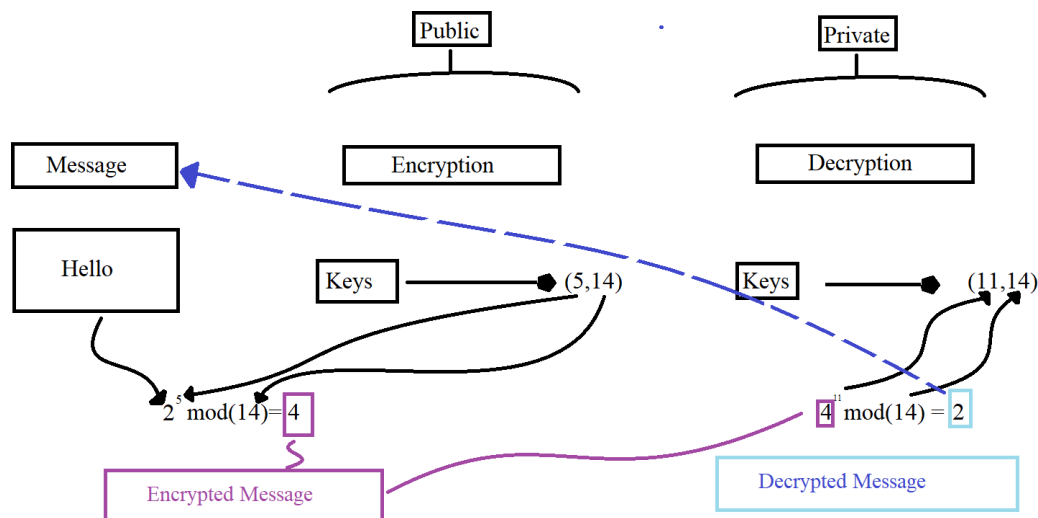


Fig. 3.3 RSA Work Flow

That the Euler totient function can be used can also be seen as a consequence of the Lagrange's theorem applied to the multiplicative group of integers modulo pq . Thus any d satisfying $d \cdot e \equiv 1 \pmod{\phi(n)}$ also satisfies $d \cdot e \equiv 1 \pmod{\lambda(n)}$. However, computing d modulo $\phi(n)$ will sometimes yield a result that is larger than necessary (i.e. $d > \lambda(n)$). Most of the implementations of RSA will accept exponents generated using either method (if they use the private exponent d at all, rather than using the optimized decryption method based on the Chinese remainder theorem described below), but some standards like FIPS 186-4 may require that $d < \lambda(n)$. Any "oversized" private exponents not meeting that criterion may always be reduced modulo $\lambda(n)$ to obtain a smaller equivalent exponent.

Since any common factors of $(p - 1)$ and $(q - 1)$ are present in the factorisation of $n - 1 = pq - 1 = (p - 1)(q - 1) + (p - 1) + (q - 1)$, it is recommended that $(p - 1)$ and $(q - 1)$ have only very small common factors, if any besides the necessary 2.

Note: The authors of the original RSA paper carry out the key generation by choosing d and then computing e as the modular multiplicative inverse of d modulo $\phi(n)$, whereas most current implementations of RSA, such as those following PKCS#1, do the reverse (choose e and compute d). Since the chosen key can be small whereas the computed key normally is not, the RSA paper's algorithm optimizes decryption compared

to encryption, while the modern algorithm optimizes encryption instead.

3.15 XAML

Extensible Application Markup Language (XAML) is a declarative XML-based language developed by Microsoft that is used for initializing structured values and objects. It is available under Microsoft's Open Specification Promise. The acronym originally stood for Extensible Avalon Markup Language, Avalon being the code-name for Windows Presentation Foundation (WPF).

XAML is used extensively in .NET Framework 3.0 & .NET Framework 4.0 technologies, particularly Windows Presentation Foundation (WPF), Silverlight, Windows Workflow Foundation (WF), Windows Runtime XAML Framework and Windows Store apps. In WPF, XAML forms a user interface markup language to define UI elements, data binding, events, and other features. In WF, workflows can be defined using XAML. XAML can also be used in Silverlight applications, Windows 10 Mobile (previously Windows Phone) and Universal Windows Platform apps, also called Windows Store apps.

XAML elements map directly to Common Language Runtime object instances, while XAML attributes map to Common Language Runtime properties and events on those objects. XAML files can be created and edited with visual design tools like Microsoft Expression Blend, Microsoft Visual Studio, and the hostable Windows Workflow Foundation visual designer. They can also be created and edited with a standard text editor, a code editor like XAMLPad, or a graphical editor like Vector Architect.

Anything that is created or implemented in XAML can be expressed using a more traditional .NET language, such as C# or Visual Basic .NET. However, a key aspect of the technology is the reduced complexity needed for tools to process XAML, because it is based on XML. Consequently, a variety of products are emerging, particularly in the WPF space, which create XAML-based applications. As XAML is simply based on XML, developers and designers are able to share and edit content freely amongst themselves without requiring compilation. XAML also benefits from being a declarative definition of the UI rather than procedural code to generate it.

A XAML file can be compiled into a .BAML file (Binary Application Markup Language), which may be inserted as a resource into a .NET Framework assembly. At run-time, the framework engine extracts the .BAML file from assembly resources, parses it, and creates a corresponding WPF visual tree or workflow.

When used in Windows Presentation Foundation, XAML is used to describe visual user interfaces. WPF allows for the definition of both 2D and 3D objects, rotations, animations, and a variety of other effects and features.

When used in Windows Workflow Foundation contexts, XAML is used to describe potentially long-running declarative logic, such as those created by process modeling tools and rules systems. The serialization format for workflows was previously called XOML, to differentiate it from UI markup use of XAML, but now they are no longer distinguished. However, the file extension for files containing the workflow markup is still "XOML".

This can be integrated into a Web page if WPF is installed using XBAPs (XAML Browser Applications) that are compiled applications running in a sandboxed environment hosted within the browser. Another way is to use the Silverlight plugin. The code cannot be included directly in an HTML page; rather it must be loaded into the page via JavaScript. If .NET 3.0 or later is installed, loose XAML files can also be viewed on their own in a compatible Web browser (including Internet Explorer and Firefox) in conjunction with the .NET Framework 3.0, without the need for the Silverlight plugin. Loose XAML files are markup-only files limited to defining the visual content to be rendered. They are not compiled with an application.

3.16 INKSCAPE

Inkscape is a free and open-source vector graphics editor. This software can be used to create or edit vector graphics such as illustrations, diagrams, line arts, charts, logos, business cards, book covers, icons, CD/DVD covers, and complex paintings. Inkscape's primary vector graphics format is Scalable Vector Graphics (SVG); however, many other formats can be imported and exported.

Inkscape can render primitive vector shapes (e.g. rectangles, ellipses, polygons, arcs, spirals, stars and 3D boxes) and text. These objects may be filled with solid colors, patterns, radial or linear color gradients and their borders may be stroked, both with adjustable transparency. Embedding and optional tracing of raster graphics is also supported, enabling the editor to create vector graphics from photos and other raster sources. Created shapes can be further manipulated with transformations, such as moving, rotating, scaling and skewing.

Inkscape workflow is based around vector objects. Included tools allow manipulating primitive vector shapes—e.g. rectangles, ellipses and arcs—and more complex objects, like 3D boxes with adjustable perspectives, stars, polygons and spirals. Inkscape also has rendering feature that can create objects like barcodes, calendars, grids, gears or roulette curves (using spirograph tool). These objects may be filled with solid colors, patterns, radial or linear color gradients and their borders may be stroked, both with adjustable transparency. All of those can be further edited by transformations—such as moving, rotating, scaling and skewing—or by editing paths.

Inkscape is able to write and edit text with tools available for changing font, spacing, kerning, rotation, flowing along the path or into a shape. Text can be converted to paths for further editing. The program also has a layer (as well as an objects) feature that allows the user to organize objects in a preferred stacking order in the canvas. Objects can be made visible/invisible and locked/unlocked through these features. Symbol libraries enable Inkscape to use existing symbols like logic-gate symbols or DOT pictograms. Additional libraries can be included by user. Inkscape supports image tracing, the process of extracting vector graphics from raster sources. Clones are child objects of an original parent object. Different transformations can be applied to them, such as: size, position, rotation, blur, opacity, color and symmetry. Clones are updated live whenever the parent object changes.

Every object in the drawing can be subjected to arbitrary affine transformations: moving, rotating, scaling, skewing and a configurable matrix. Transformation parameters can be specified numerically. Transformations can snap to angles, grids, guidelines and

nodes of other objects, or be aligned in specified direction, spaced equally, scattered at random. Objects can be grouped together. Groups of objects behave similarly to objects. Objects in a group can be edited without having to ungroup them first. The Z-order determines the order in which objects are drawn on the canvas. Objects with a high Z-order are drawn on top of objects lower in the Z-order. Order of objects can be managed either using layers, or by manually moving the object up and down in the Z-order. Layers can be locked or hidden, preventing modifying and accidental selection.

Create Tiled Clones tool allows to create symmetrical or grid-like drawings using various plane symmetries. Appearance of objects can be further changed by using masks and clipping paths, which can be created from arbitrary objects, including groups. The style attributes are 'attached' to the source object, so after cutting/copying an object onto the clipboard, the style's attributes can be pasted to another object. Inkscape's primary format is Scalable Vector Graphics (SVG) version 1.1, meaning that it can create and edit with the abilities and within the constraints of this format. Any other format must either be imported (converted to SVG) or exported (converted from SVG). The SVG format is using the Cascading Style Sheets (CSS) standard internally.

Inkscape supports text editing for both regular multi-line text and flowed text. Text rendering is based on the Pango library, which allows Inkscape to support several complex scripts including Hebrew, Arabic, Thai, Tibetan, etc. Kerning and letter-spacing can be adjusted on a per-glyph basis using keyboard shortcuts. Putting text on path is also supported, and both the text and the path remain editable. Inkscape supports italicized and bold, as well as super- and subscript character attributes, but underlining is not yet implemented.

3.17 GUI LIBRARY

The graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation.

GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs), which require commands to be typed on a computer keyboard.

The actions in a GUI are usually performed through direct manipulation of the graphical elements. Beyond computers, GUIs are used in many handheld mobile devices such as MP3 players, portable media players, gaming devices, smartphones and smaller household, office and industrial controls. The term GUI tends not to be applied to other lower-display resolution types of interfaces, such as video games (where head-up display (HUD) is preferred), or not including flat screens, like volumetric displays because the term is restricted to the scope of two-dimensional display screens able to describe generic information, in the tradition of the computer science research at the Xerox Palo Alto Research Center.

3.18 UPDATE LIBRARY

An Update Library is an external code added to the project so that it could communicate with the internet. This library looks for the current version of the software that is being run on the user's system. After gathering the version information of the project, it sends a get request to an a predefined xml file which is stored in the cloud. The xml file will have all the information needed by the update library. This information is then compared with the version of the current software being used by the user. If the update library find that a new version is available and it could be downloaded it will prompt the user with an update window. The library allows us to make the download either mandatory or skippable. Once downloaded it will update all the dependency of the project and will do a reboot of the software. Once everything is done it will prompt the user that the software has been updated successfully. Otherwise it will inform the user that update had failed and he would need to do it again.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 WINDOWS

Main Window



Fig:- 4.1 – Main Window

SecureVault Window is the main Window from which the software launches, it provides a person four buttons to make a proper use of the software. Buttons are Signin, Register, Vault and Future Feature. It also provides a Settings button, and along with a side animated bar for more information. View the Fig:- 4.1 present above.

Sign In Window



Fig:- 4.2 – Login Window

In this Window a person can SignIn to login in a person's Vault, a person can't enter the Vault without logging In. This is the first security layer provided by the Secure Vault. The window is well descriptive about the details needed to enter, With a very appealing UI. View the Fig:- 4.2.

Register Window

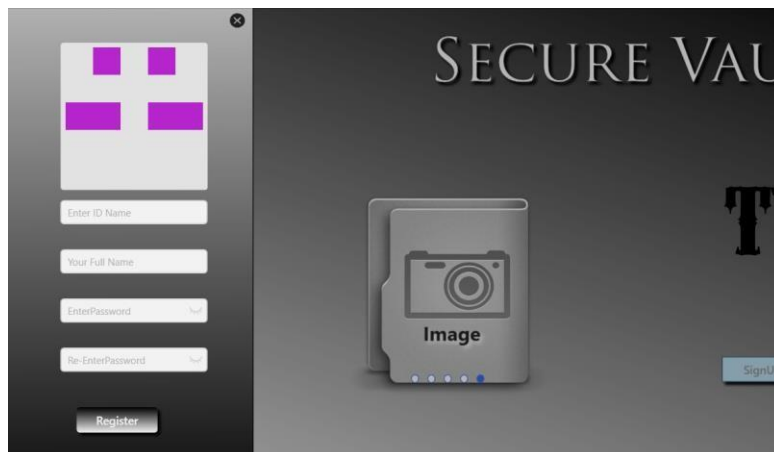


Fig:- 4.3 – SignUp Window

Register Window Is to do one-time SignUp, Enter the needed details along with a person's own chosen VaultID, and a 10 Digit password. View the Fig- : 4.3

Vault Window

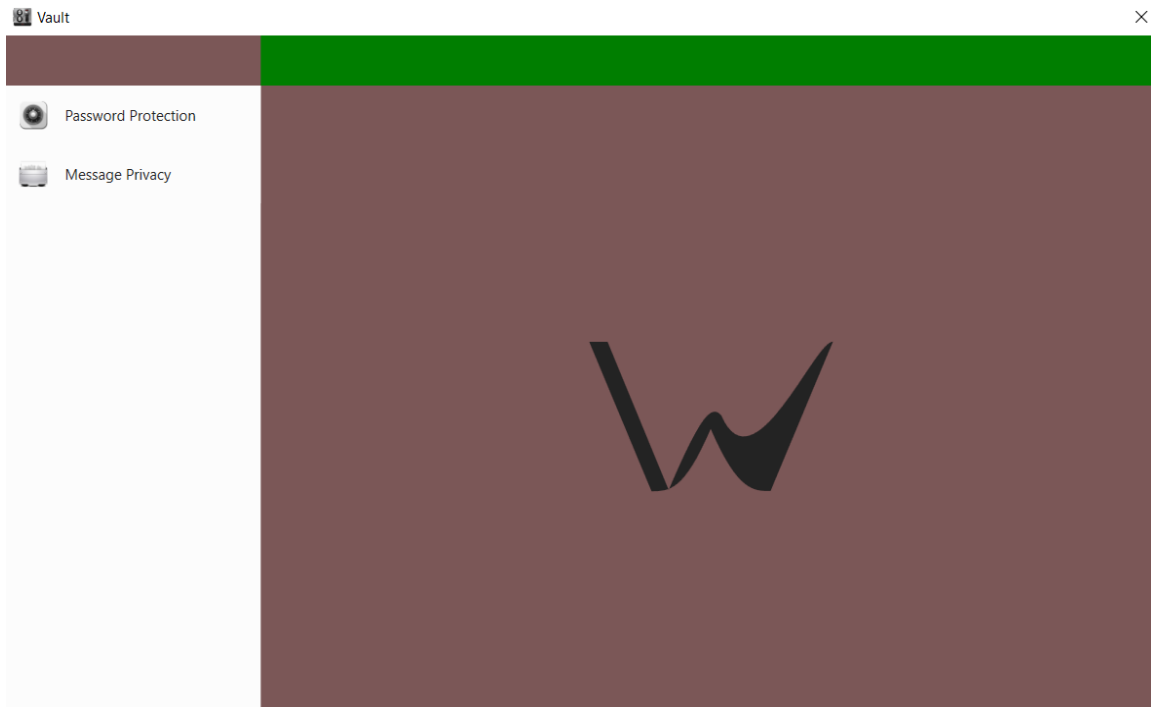


Fig:- 4.4 – Entry Window

Vault Window, in this Window we enter the web id along with password and load them, if a person want to know the password A person unload by entering a person's WebID.Fig:-4.4

This like a portal to enter all other different windows present in this software, user can click the side menu on the respective images to do the respective task regarding the protection he needs at the moment. Password protection Message protection has sub division in themselves along with image protection which appears as a separate window chose by the user.

Settings Window

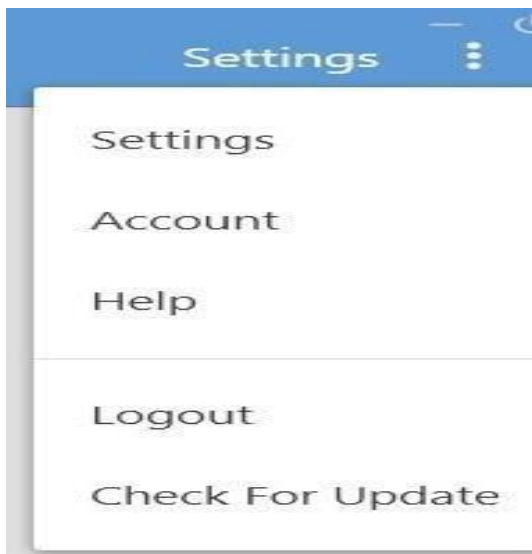


Fig:- 4.5 – Settings Window

This is to get the Various options for the Software and Update Button. See Fig:- 4.5.

Update Window



Fig:- 4.6 – Update Window

Update window is used for updating the software, user can skip, later and update the software with the given options. Fig:-4.6

Hence, we concluded that all the dialog boxes work well and doesn't crash while using any of the windows. It had successfully added a user account and provide a smooth login experience to the user. after a successful login, vault window opens as expected and user was able to store passwords in the window. These passwords were successfully retrieved by the user after wards. Message encryption process was also great and the recovery of message was 100%. Image encryption took a bit of time and processing power but was successful. Update prompt was also working while checking of the latest build available from the web. Once the application was ready it got restarted automatically. Over all the project ends with a successful implementation of the intended features and is capable of handling any crash or failures, and is a reliable software for a user.

4.2 PERFORMANCE

The Performance was satisfactory as the CPU consumption was very less and it proved that the software can handle heavy computation without putting much pressure on underlining hardware. The maximum resultant memory usage in Debug mode was less than 170MB. This doesn't mean that the software used all the memory. As we can see in this Picture.

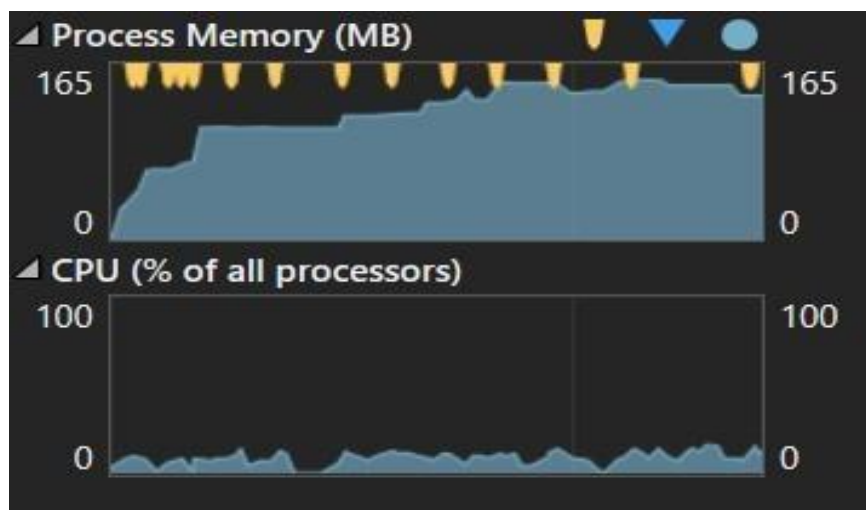


Fig:- 4.7 – Debug Performance

The Performance in Deployment cycle was also great as the Cap limit never exceeded the 100MB mark and CPU consumption was also very less. This proves that the software is deployment ready and it also achieves what it was intended for; The Results for Password protection and Message protection are promising, some optimization could be done in image protection side.

Name	Status	19% CPU	73% Memory	45% Disk	0% Network
Apps (6)					
> Java(TM) Platform SE binary		0%	22.5 MB	0 MB/s	0 Mbps
> Microsoft Visual Studio 2019 (32 bit) (21)		3.4%	852.6 MB	0.1 MB/s	0 Mbps
> Peek Through		0%	1.6 MB	0 MB/s	0 Mbps
> SecureVaultV2		4.6%	95.6 MB	0 MB/s	0 Mbps
> Task Manager		0.3%	27.0 MB	0 MB/s	0 Mbps
> Windows Explorer		0.2%	68.7 MB	0 MB/s	0 Mbps

Fig:- 4.8 – Production Performance

CHAPTER- 5

CONCLUSION & FUTURE WORKS

5.1 SUMMARY

The Project of SecureVault is made in order to give protection to the users and to provide a secure option, of having various password, to all the registered users of SecureVault. This software is easy to use and just needs one-time SignUp. It has a feature of viewing the number of ppl registered in the software, hence it's not restricted to any one user. It has a secure login page, which can be used to display or enter a new password. This software also comes with a Support menu that describes the functionality of the software.

5.2 CONCLUSION

Password protection is a main concern for any person who wants his/her data to be safe all the time and away from any second person. This software has been created in order to meet this demand, by having multiple password on different web ID one can assure themselves that having multiple password will increase their web security to high level. This software has a unique User ID and a unique password for every password and Web ID.

5.2 FUTURE WORKS

Although the project has all the features it still needs optimization for images encryption and sound protection could be added. The project does consist of an update system so the distribution of the software will not be a problem in future for existing users. This could be further expanded to provide cloud storage for those who are willing to trust cloud services via secure vault system.

REFERENCE

- [1] Daniel (Yue) Zhang, Dong Wang, Nathan Vance, Yang Zhang, and Steven Mike. On Scalable and Robust Truth Discovery in Big Data Social Media Sensing Applications. Citation information: DOI 10.1109/TBDATA.2018.2824812, IEEE Transactions on Big Data 2332-7790 (c) 2018 IEEE.
- [2] Dong. X. L, Berti-Equille. L, and Srivastava D. Integrating conflicting data: the role of source dependence. In Proceedings of the VLDB Endowment, pages 550–561, 2009.
- [3] etal. X.X. Towards confidence in the truth: A boot strapping based truth discovery approach. In Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD '16, 2016.
- [4] Les Servi, Sara Beth Elson. The Mathematical Approach Gauging Influence by Identifying Shifts in the Emotions of Social Media Users. IEEE Transactions on computational social system (volume1, issue 4, dec 2014) pages180-190. IEEE-2014.
- [5] Miaomiao Wen, Diyi Yang, Carolyn Penstein Rosé. Sentiment Analysis in MOOC Discussion Forums: What does it tell us?
- [6] Mohamed M. Mostafa. More than words: Social networks' text mining for consumer brand sentiments. volume 40, issue10, august 2013, page4241-4251.
- [7] Pasternack. J and Roth.D. Knowing what to believe (when you already know something). Association for Computational Linguistics, pages 877–885, 2010.
- [8] Sitaram Asur, Bernardo A. Huberman. Predicting the Future with Social Media. 2010 IEEE/WIC/ACM International conference on web intelligent agent technology. volume1 papers-641. IEEE-2010.

[9] Yin. X, Han J, and Yu. P. S. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796–808, June 2008.

[10] Zhang. D. Y, Wang. D, and Zhang. Y. Constraint-aware dynamic truth discovery in big data social media sensing. In *Big Data (Big Data)*, 2017 IEEE International Conference on, pages 57–66. IEEE, 2017.

APPENDICES

SAMPLE Code XAML

a) Main Window

```
<Window x:Class="SecureVault.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:SecureVault"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        mc:Ignorable="d" Height="600" Width="1080" Foreground="White"
        ResizeMode="NoResize" WindowStartupLocation="CenterScreen" WindowStyle="None"
>

<!--Open/Close Menu Animation -->
<Window.Resources>

    <Storyboard x:Key="OpenMenu">
        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(FrameworkElement.Width)" Storyboard.TargetName="GridMenu">
            <EasingDoubleKeyFrame KeyTime="0" Value="70"/>
            <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="200"/>
        </DoubleAnimationUsingKeyFrames>
    </Storyboard>

    <Storyboard x:Key="CloseMenu">
        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(FrameworkElement.Width)" Storyboard.TargetName="GridMenu">
            <EasingDoubleKeyFrame KeyTime="0" Value="200"/>
            <EasingDoubleKeyFrame KeyTime="0:0:0.5" Value="70"/>
        </DoubleAnimationUsingKeyFrames>
    </Storyboard>

</Window.Resources>

<!--Open/Close Menu Trigger -->

<Window.Triggers>
    <EventTrigger RoutedEvent="ButtonBase.Click" SourceName="ButtonOpenMenu">
        <BeginStoryboard Storyboard="{StaticResource OpenMenu}"/>
    </EventTrigger>
    <EventTrigger RoutedEvent="ButtonBase.Click" SourceName="ButtonCloseMenu">
        <BeginStoryboard Storyboard="{StaticResource CloseMenu}"/>
    </EventTrigger>
</Window.Triggers>
```

```

<Grid x:Name="GridMain" HorizontalAlignment="Right" VerticalAlignment="Bottom"
Height="540" Width="1010"/>

<Grid Height="60" VerticalAlignment="Top" Background="#FF5C99D6">
    <Grid.Effect>
        <DropShadowEffect BlurRadius="1" ShadowDepth="1" Color="#FFACACAC"/>
    </Grid.Effect>
    <Grid Margin="1030 0 3 40" HorizontalAlignment="Right">
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>

        <Button Grid.Column="0" x:Name="MinButton" Width="30" Height="23"
Background="{x:Null}" BorderBrush="{x:Null}" Padding="0" Foreground="LightGray"
Click="Min_Click">
            <materialDesign:PackIcon Kind="Minus"/>
        </Button>

        <Button Grid.Column="1" x:Name="PowerButton" Width="33" Height="23"
Background="{x:Null}" BorderBrush="{x:Null}" Padding="0" Foreground="LightGray"
Click="Pow_Click">
            <materialDesign:PackIcon Kind="Power"/>
        </Button>

    </Grid>
    <StackPanel Orientation="Horizontal" HorizontalAlignment="Right">
        <TextBlock Text="Settings" VerticalAlignment="Center" FontSize="16"/>

        <!--PopupBox 7-->
        <materialDesign:PopupBox PlacementMode="BottomAndAlignRightEdges"
StaysOpen="False" Margin="10">
            <StackPanel Width="150">
                <Button Content="Settings" Click="Settings_Click" />
                <Button Content="Account" Click="Account_Click"/>
                <Button Content="Help" Click="Help_Click"/>
                <Separator/>
                <Button Click="Logout_Click" Content="Logout"/>
                <Button Content="Check For Updates" Click="Update_Click"/>
            </StackPanel>
        </materialDesign:PopupBox>

    </StackPanel>
    <TextBlock HorizontalAlignment="Center" VerticalAlignment="Center"
FontSize="22">Secure Vault</TextBlock>
</Grid>

<!--Grid Menu { open menu } 7-->

<Grid x:Name="GridMenu" Width="70" HorizontalAlignment="Left"
Background="#FF31577E">
    <Grid.Effect>
        <DropShadowEffect BlurRadius="2" ShadowDepth="1" Color="#FFAEAEAE"/>
    </Grid.Effect>

```

```

<Grid.RenderTransform>
    <TransformGroup>
        <ScaleTransform/>
        <SkewTransform/>
        <RotateTransform/>
        <TranslateTransform/>
    </TransformGroup>
</Grid.RenderTransform>

<!--Animation Control Buttons -->
<Button x :Name= »ButtonOpenMenu » Height= »60 » Width= »60 »
Margin= »5 » HorizontalAlignment= »Right » VerticalAlignment= »Top »
Style= »{StaticResource MaterialDesignFloatingActionMiniAccentButton} »
Background= »{x :Null} » BorderBrush= »{x :Null} » Foreground= »#FF31577E »
Click= »ButtonOpenMenu_Click »>
    <materialDesign:PackIcon Kind="Menu" Width="25" Height="25"/>
</Button>
<Button x:Name="ButtonCloseMenu" Visibility="Collapsed" Height="60"
Width="60" Margin="5" HorizontalAlignment="Right" VerticalAlignment="Top"
Style="{StaticResource MaterialDesignFloatingActionMiniAccentButton}"
Background="{x:Null}" BorderBrush="{x:Null}" Foreground="#FF31577E"
Click="ButtonCloseMenu_Click">
    <materialDesign:PackIcon Kind="ArrowLeft" Width="25"
Height="25"/>
</Button>
<!-- -->

<Image Height="80" VerticalAlignment="Bottom"/>
</Grid>
<ListView x:Name="ListViewMenu" Foreground="#FF5C99D6"
ScrollViewer.HorizontalScrollBarVisibility="Disabled"
SelectionChanged="ListViewMenu_SelectionChanged">
    <ListViewItem x:Name="ItemLogin" Height="60">
        <StackPanel Orientation="Horizontal">
            <materialDesign:PackIcon Kind="ViewDashboard" Height="25"
Width="25" Margin="10"/>
            <TextBlock Text="Login" VerticalAlignment="Center"
Margin="20 10"/>
        </StackPanel>
    </ListViewItem>
    <ListViewItem x:Name="ItemRegister" Height="60">
        <StackPanel Orientation="Horizontal">
            <materialDesign:PackIcon Kind="Pencil" Height="25"
Width="25" Margin="10"/>
            <TextBlock Text="Register" VerticalAlignment="Center"
Margin="20 10"/>
        </StackPanel>
    </ListViewItem>
    <ListViewItem x:Name="ItemRecord" Height="60">
        <StackPanel Orientation="Horizontal">
            <materialDesign:PackIcon Kind="Ticket" Height="25"
Width="25" Margin="10"/>

```

```

        </ListItem>
        <ListItem Height="60">
            <StackPanel Orientation="Horizontal">
                <materialDesign:PackIcon Kind="Message" Height="25"
Width="25" Margin="10"/>
                <TextBlock Text="Future Feature" VerticalAlignment="Center"
Margin="20 10"/>
            </StackPanel>
        </ListItem>
    </ListView>
</StackPanel>
</Grid>

<!--Ends here -->

</Grid>

</Window>

```

b) Tab1

```

<UserControl x:Class="SecureVault.Tab1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
xmlns:local="clr-namespace:SecureVault"
mc:Ignorable="d"
d:DesignHeight="540" d:DesignWidth="1010"
Background="Blue"
>
<Grid>
    <Image Width="1010" Height="540" Stretch="UniformToFill">
        <Image.Source>
            <BitmapImage DecodePixelWidth="280" UriSource="Source/HandKey.jpg"/>
        </Image.Source>
    </Image>
    <Grid Margin=" 100 90 100 90" Background="Black" Opacity="0.3"/>
    <Grid Name="Center_Margin" Margin=" 100 90 100 90" >
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"></ColumnDefinition>
            <ColumnDefinition Width="2*"></ColumnDefinition>
            <ColumnDefinition Width="1*"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <StackPanel VerticalAlignment="Center" Grid.Column="1" >
            <TextBlock Text="Welcome User" Foreground="LightBlue" Padding="65
0 65 35" FontFamily="Hobo Std" FontSize="24" TextDecorations="Underline" Margin="2,0,-
2.2,0">
                <TextBlock.Effect>
                    <DropShadowEffect ShadowDepth="2" Color="#FF853939"/>
                </TextBlock.Effect>
            </TextBlock>
        </StackPanel>
    </Grid>
</Grid>

```

```

        <TextBlock Text="Login" Padding="100 0 70 0"
FontFamily="Simplified Arabic Fixed" FontSize="27px" Foreground="GreenYellow"/>
        <TextBox Name="Logintxt" MaxLength="27"
materialDesign:HintAssist.Hint="Enter Your VaultId" Background="Transparent"
Height="40" materialDesign:HintAssist.IsFloating="True" Margin="10" FontSize="16"
Foreground="LightBlue"/>

        <TextBlock Text="Password" FontWeight="Bold" Padding="80 0 70 0"
FontFamily="Simplified Arabic Fixed" FontSize="27px" Foreground="GreenYellow">
            <TextBlock.Effect>
                <DropShadowEffect ShadowDepth="3" BlurRadius="6"/>
            </TextBlock.Effect>
        </TextBlock>
        <TextBox Name="passtxt" MaxLength="27"
materialDesign:HintAssist.Hint="Enter Your Vault Unique Password"
Background="Transparent" Height="41" materialDesign:HintAssist.IsFloating="True"
Margin="10" FontSize="16" />
        <Button Name="LoginBut" Margin="0 20" Click="LoginBut_Click"
Content="SignIn" FontFamily="Charlemagne Std" FontSize="20px"
Foreground="LightGoldenrodYellow">
            <Button.Effect>
                <DropShadowEffect/>
            </Button.Effect>
        </Button>

    </StackPanel>
</Grid>

<Grid Name="Left_Margin" Margin="0 0 700 0">

</Grid>
<Grid Name="Top_Margin" Margin="100 0 0 350" >

</Grid>
<Grid Name="Right_Margin" Margin=" 700 100 0 0">

</Grid>
<Grid Name="Bottom_Margin" Margin="100 350 100 0 ">

</Grid>

</Grid>
</UserControl>

```

B) SCREEN SHOTS

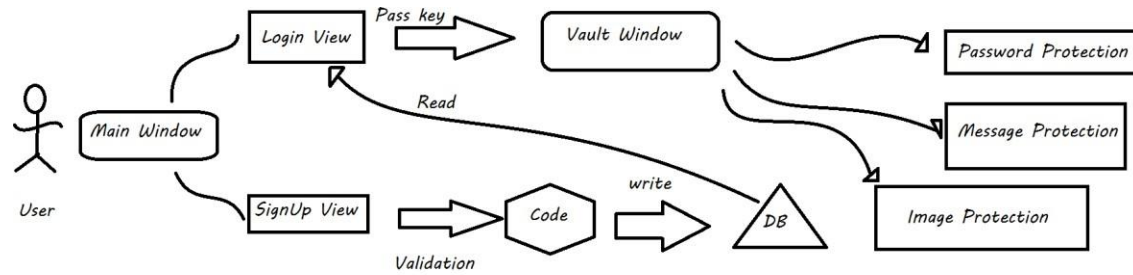
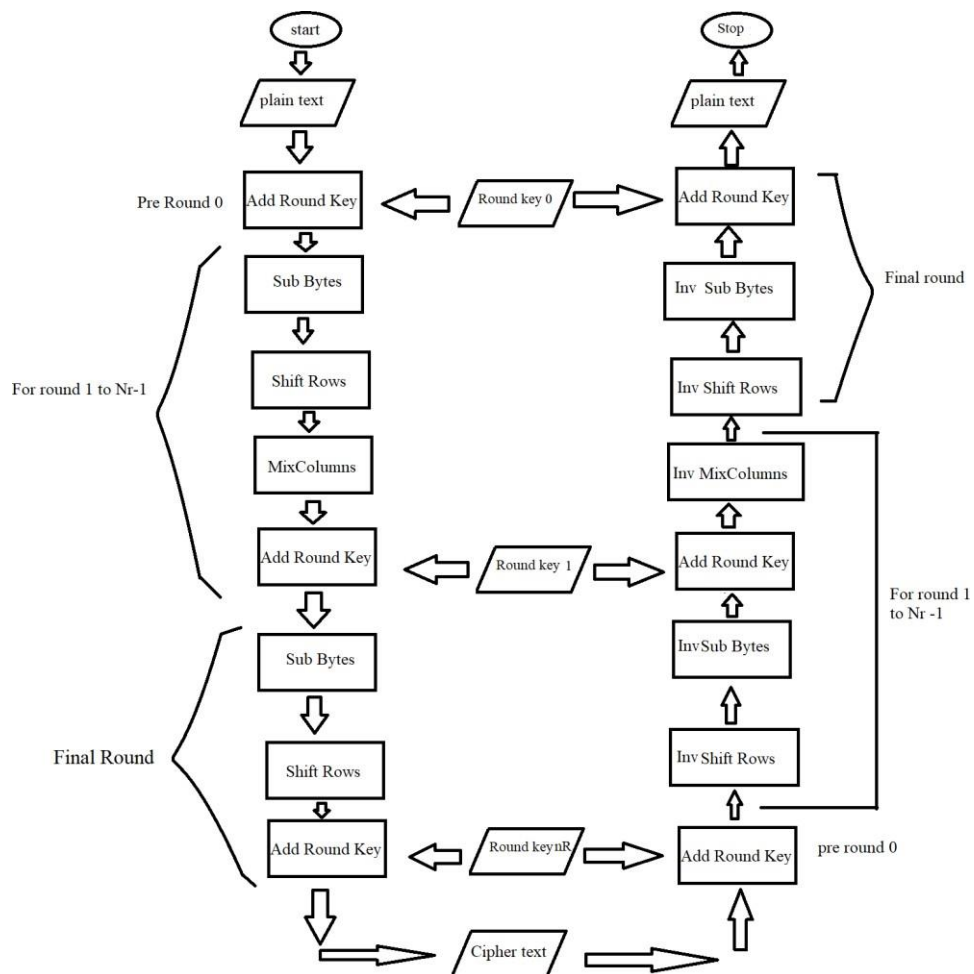
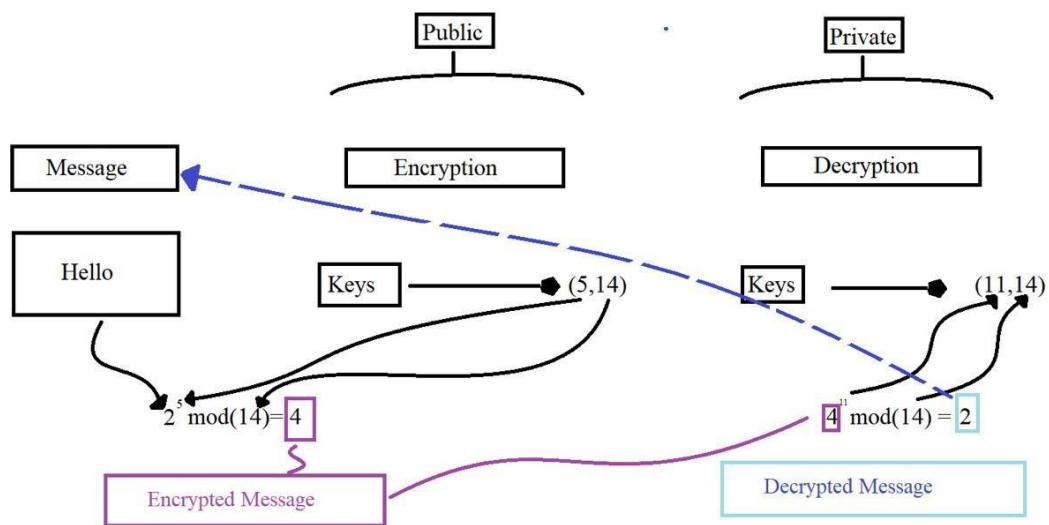
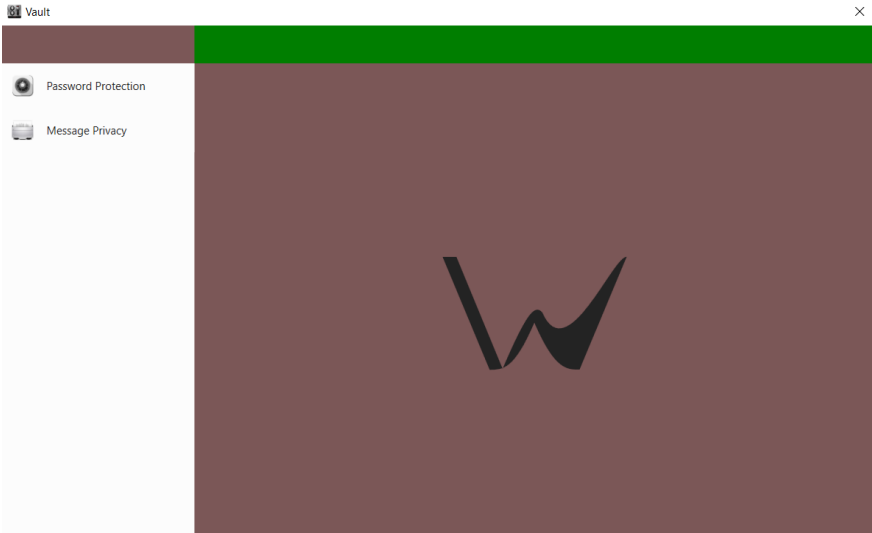
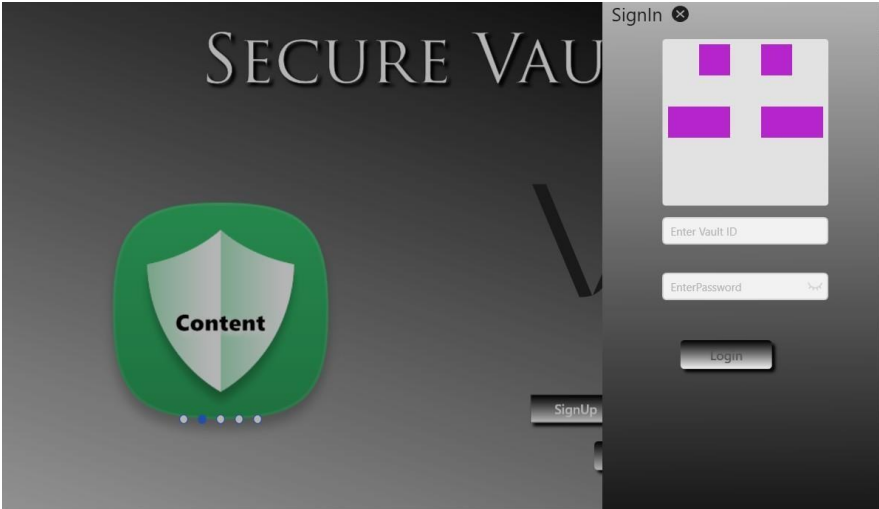
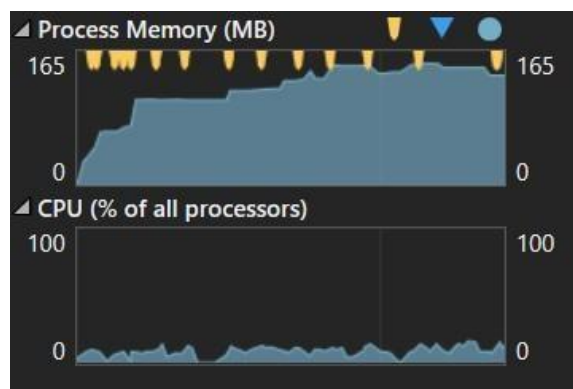
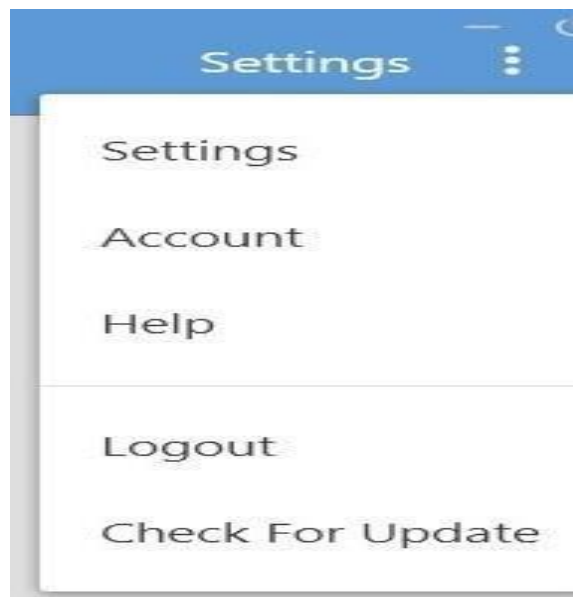


Fig 1, Architecture of Secure Vault









Name	Status	19% CPU	73% Memory	45% Disk	0% Network
Apps (6)					
> Java(TM) Platform SE binary		0%	22.5 MB	0 MB/s	0 Mbps
> Microsoft Visual Studio 2019 (32 bit) (21)		3.4%	852.6 MB	0.1 MB/s	0 Mbps
> Peek Through		0%	1.6 MB	0 MB/s	0 Mbps
> SecureVaultV2		4.6%	95.6 MB	0 MB/s	0 Mbps
> Task Manager		0.3%	27.0 MB	0 MB/s	0 Mbps
> Windows Explorer		0.2%	68.7 MB	0 MB/s	0 Mbps

SecureVault 0.2.0.0 is available!

A new version of SecureVault is available!

SecureVault 0.2.0.0 is now available. You have version 0.1.0.0 installed. Would you like to download it now?

Skip this version
Remind me later
Update