USB SECURITY SYSTEM

Name & Enrollment No.:

Shobhit Gupta 091239

Shubham Agarwal 091251

Shubham Jain 091269

Divey Chugh 091273

Name of the Supervisor: Mr. Kapil Saini

May- 2013

Submitted in the partial fulfillment of the Degree of

Bachelor of Technology

Department Of Computer Science & Engineering

Jaypee University Of Information Technology, Waknaghat

CERTIFICATE

This is to certify that the work titled "**USB Security System**" submitted by **" Divey Chugh, Shobhit Gupta, Shubham Agarwal, Shubham Jain "** in partial fulfillment for the award of degree of **B.Tech** of **Jaypee University of Information Technology, Waknaghat** has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor ……..

Name of Supervisor Mr. Kapil Saini

Designation Senior Lecturer

Date ……..

ACKNOWLEDGEMENT

We hereby take this opportunity to thank all those who contributed directly or indirectly in shaping our project.

We would like to thank our **HOD, Brig(Retd) Satya Prakash Ghrera** for his encouragement and support. We are deeply indebted to our project guide **Mr. Kapil Saini** whose guidance and encouragement proved very valuable. We also express our sincerest thanks to all the staff members for their continuous support through the phases of this project.

Lastly we would like to acknowledge the many authors whose work has been quoted in this project.

Table of Contents

List of Figures

List of Tables

Abstract

Objectives:

To develop software that restricts the movement of data outside an organization; monitors and controls data exchange between the computer and the USB. The software will allow the transfer of legitimate data and data types to and from USB drives but any unsolicited activity like data theft; unwanted data/malware injection/planting etc. will be inhibited.

Implementation:

The software will employ software based encryption technique. All data leaving via the USB port into a flash disk will be encrypted with a specific key. While transferring data back to the company computer, only those files will be copied that had been encrypted using this specific key. The software will ensure that all transfers are made using the monitoring and encryption software.

1.Introduction

Transferring data from one computer system to another is nowadays a non-technical, highly efficient, inconspicuous task. This effectively puts corporations in harm's way, since the misuse of portable storage devices can expose corporate networks to a number of dangerous issues, which might have an impact on corporations in a variety of ways.

The uncontrolled use of portable storage devices by corporate insiders is a definite threat to the security and stability of every business. Malicious insiders and gullible employees who fall for social engineering practices are the weakest link in the corporate security chain. Relying on user's voluntary compliance to the corporate device usage policy is not a solution – one must deploy software countermeasures that thwart this risk.

Some of the major vulnerabilities of using USB sticks are Data Theft, Legal Liabilities, Productivity Loss and Corporate Network security breaches, of which the first and the last are the most notorious and damaging.

This is the idea behind developing software systems which mitigate, or at least minimize the risks involved in using USB sticks and hence the proposed system.

Secure USB flash drives protect the data stored on them from access by unauthorized users. USB flash drive products have been on the market since 2000, and their use is increasing exponentially. As both consumers and businesses have increased demand for these drives, manufacturers are producing faster devices with greater data storage.

An increasing number of portable devices are used in business, such as laptops, notebooks, universal serial bus (USB) flash drives, personal digital assistants (PDAs), advanced mobile phones and other mobile devices.

Companies in particular are at risk when sensitive data are stored on unsecured USB flash drives by employees who use the devices to transport data outside the office. The consequences of losing drives loaded with such information can be significant, and include the loss of customer data, financial information, business plans and other confidential information, with the associated risk of reputation damage.

Major dangers of USB drives

USB flash drives pose two major challenges to information system security: data leakage owing to their small size and ubiquity; system compromise through infection from computer virus and other malicious software.

Data Leakage

The large storage capacity of USB flash drives relative to their small size and low cost means that using them for data storage without adequate operational and logical controls can pose a serious threat to information confidentiality, integrity, and availability. The following factors should be taken into consideration for securing USB drives assets:

- **Storage:** USB flash drives are hard to track physically, being stored in bags, backpacks, laptop cases, jackets, trouser pockets, or left at unattended workstations.
- **Usage:** tracking corporate data stored on personal flash drives is a significant challenge; the drives are small, common, and constantly moving. While many enterprises have strict management policies toward USB drives, and some companies ban them outright to minimize risk, others seem unaware of the risks these devices pose to system security.

The average cost of a data breach from any source (not necessarily a flash drive) ranges from less than $100,000 to about $2.5 million.

A SanDisk survey characterized the data corporate end users most frequently copy:

1. customer data (25%)
2. financial information (17%)
3. business plans (15%)
4. employee data (13%)
5. marketing plans (13%)
6. intellectual property (6%)
7. source code (6%)

Examples of security breaches resulting from USB drives include:

- In the UK:
    - HM Revenue & Customs lost personal details of 6,500 private pension holders

- In the United States:
    - a USB drive was stolen with names, grades, and social security numbers of 6,500 former students [3]
    - USB flash drives with US Army classified military information were up for sale at a bazaar outside Bagram, Afghanistan.

Malware Infections

In the early days of computer viruses and malware the primary means of transmission and infection was the floppy disk. Today, USB flash drives perform the same data and software storage and transfer role as the floppy disk, often used for transferring files between computers which may be on different networks or in different offices, owned by different people; this has made USB flash drives a leading form of information system infection. When a piece of malware gets onto a USB flash drive it may infect the devices into which that drive is subsequently plugged.

The prevelance of malware infection by means of USB flash drive was documented in a 2011 Microsoft study [5] analyzing data from more than 600 million systems worldwide in the first half of 2011. The study found that 26 percent of all malware infections of Windows system were due to USB flash drives exploiting the AutoRun feature in Microsoft Windows. That finding was in line with other statistics, such as the monthly reporting of most commonly detected malware by antivirus company ESET, which lists abuse of autorun.inf as first among the top ten threats in 2011.

The Windows autorun.inf file contains information on programs meant to run automatically when removable media (often USB flash drives and similar devices) are accessed by a Windows PC user. The default Autorun setting in Windows versions prior to Windows 7 will automatically run a program listed in the autorun.inf file when you access many kinds of removable media. Many types of malware copy themselves to removable storage devices: while this is not always the program's primary distribution mechanism, malware authors often build in additional infection techniques.

Examples of malware spread by USB flash drives include:

- The Stuxnet worm.
- Flame modular computer malware.

Solutions

Since the security of the physical drive cannot be guaranteed without compromising the benefits of portability, security measures are primarily devoted to making the data on a compromised drive inaccessible to unauthorized users and unauthorized processes, such as may be executed by malware. One common approach is to encrypt the data for storage, and routinely scan drives for malware with an antivirus program, although other methods are possible.

Software

Software solutions such as FreeOTFE and TrueCrypt allow the contents of a USB drive to be encrypted automatically and transparently. Also, Windows 7 Enterprise and Ultimate Editions and Windows Server 2008 R2 provide USB drive encryption using BitLocker to Go. The Apple

Computer Mac OS X operating system has provided software for disc data encryption since Mac OS X Panther was issued in 2003 (see also: Disk Utility).

Additional software like USBCrypt or USB Secure can be installed on your USB/External drive to prevent access to your files in case your drive gets lost or stolen. Installing software on company computers may help track and minimize risk by recording the interactions between any USB drive and the computer and storing them in a centralized database.

Hardware

Some USB drives do have hardware encryption in which microchips within the USB drive do automatic and transparent encryption. For instance the company iStorage offer both flash and hard drives that require a pin code entering into a physical keypad on the drives to allow access to the drive, their products also contain all the features mentioned in this article. The cost of these USB drives can be significant but is starting to fall due to this type of USB drive gaining popualrity.

Hardware systems may offer additional features, such as the ability to automatically overwrite the contents of the drive if the wrong password is entered more than a certain number of times. This type of functionality cannot be provided by a software system since the encrypted data can simply be copied from the drive. However, this form of hardware security can result in data loss if activated accidentally by legitimate users, and strong encryption algorithms essentially make such functionality redundant.

As the encryption keys used in hardware encryption are typically never stored in the computer's memory, technically hardware solutions are less subject to "cold boot" attacks than software-based systems. In reality however, "cold boot" attacks pose little (if any) threat, assuming basic, rudimentary, security precautions are taken with software-based systems.

Compromised systems

The security of encrypted flash drives is constantly tested by individual hackers as well as professional security firms. At times (as in January 2010) data on flash drives that have been positioned as secure were found to have a bug that potentially could give access to data without knowledge of the correct password.

Flash drives that have been compromised (and fixed) include:

- SanDisk Cruzer Enterprise
- Kingston DataTraveler BlackBox
- Verbatim Corporate Secure USB Flash Drive
- Trek Technology ThumbDrive CRYPTO

All of the above companies reacted immediately. Kingston offered replacement drives with a different security architecture. SanDisk, Verbatim, and Trek released patches.

Management

In commercial environments, where most secure USB drives are used, a central management system may provide IT organizations with an additional level of IT asset control. This can include initial user deployment and ongoing management, password recovery, data backup, and termination of any issued secure USB drive. Such management systems are available as software as a service (where Internet connectivity is allowed) or as behind-the-firewall solutions.

1.1Purpose

The software restricts the movement of data outside an organization by monitoring and controlling data exchange between the computer and the USB. The software will allow the transfer of legitimate data and data types to and from USB drives but any unsolicited activity like data theft will be inhibited.

This document is meant to delineate the features of our software, so as to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other.

1.2 Scope

The features that are in the scope of the software to be developed are:

a. Controlling data flow in and out of the USB by encrypting files before writing to USB and identifying and decrypting only the encrypted files on USB.
b. Encrypting data with level specific keys to allow authorized access.
c. User authentication using password.

2. The Overall Description

Product Functions

The software should support the following use cases:

Use Case	Description of Use Case
Registration	Save username, password and authority level for a new user
File transfer	Transfer files to and from the USB via the software interface
Delete file	Erase files on the pen drive
Encryption	Encrypt the files before they are transferred from the system to the pen drive
Decryption	Decrypt the encrypted files present on the pen drive before transferring them to the system
Authentication and authorization	Verify username and password and check if the requested level key can be used
Access server DB	Get the encrypted AES key to encrypt/decrypt the file

Access client DB	Get the RSA key to decrypt the AES key and use it for encryption
Create DB	Populate the database with the keys and the associate level
Generate Keys	Generate the AES and RSA keys and encrypt the AES key with RSA key
Store keys	Store the keys at the server and client side

Use Case Descriptions

Table No. 1

3. Specific Requirements

3.1 Functional requirements

Functional requirements capture the intended behaviour of the system. This behaviour may be expressed as services, tasks or functions the system is required to perform

Req 1: Interactive GUI: To provide interactive interface to the user.
Req 2: Faster execution speed: Providing faster execution capability to the software.
Req 3: User Throughput: Execution speed experienced by the user.
Req 4 :Help and technical support: Providing help to the user.

3.2 Non-Functional requirements

Non-functional requirements impose constraints on the design or implementation (such as performance requirements, quality standards or design constraints).
Users have implicit expectations about how well the software will work. These characteristics include how easy the software is to use, how quickly it executes, how reliable it is, and how well it behaves when unexpected conditions arise.

Req 1: Performance: Turn Around Time(TAT),Memory Access Time, Response Time etc.
Req 2: Reliability: Decreasing failure probability.
Req 3: Fault Tolerance: Increasing immunity from unwarranted exceptions.

The proposed system uses a client-server architecture where a client invokes a method stored at the server side. Server in turn performs an exhaustive search on its database and returns the necessary information back to the client. The client then uses this information to decrypt the file. It uses private key cryptography for encrypting the user data and public key cryptography for transmitting the keys over a secure channel. Client maintains its own database that contains decryption keys for decrypting the encrypted private keys.

Since the system is server-centralized, it might fail if the server goes down, in which case the clients must either wait for the server to recover or use some other secure method for data transmission.

Presently, this software does not support drive encryption and has been developed keeping in mind the most popular operating system, Windows. But its capabilities may be expanded to support drive encryption as well as run on other platforms like Linux, Mac, and Solaris with minor design and implementation modifications.

3.3 Minimum Hardware/Software Requirements

1. Hardware Requirements:

- Processor Speed: 550 MHz
- Random Access Memory (RAM): 128 MB
- Peripheral Devices: Keyboard, Mouse
- Serial Ports

Recommended:

- Processor Speed: 800 MHz
- Random Access Memory (RAM): 256 MB

2. Software Requirements:

- Operating System: Windows
- Java Support: JDK 1.5 or later
- RDBMS, like MS Access, Oracle, MySQL, SQLServer Compliant JDBC Driver for vendor-specific RD

4. Literature Review

Our report describes the methodologies adopted to develop software that allows safe data transmission via the usb port by employing encryption techniques. The IEEE paper Encrypted key exchange: password based protocols secure against dictionary attack describes the use of a secret key to encrypt a public key. It introduces a novel combination of asymmetric (public-key) and symmetric (secret-key) cryptography that allows exchange of confidential and authenticated information over an insecure network.

Considering the hierarchical structure of management in the organizations that our software is targeted at, we have used level specific keys for encryption to ensure authorized access. A Cryptographic Key Generation Scheme for Multilevel Data Securityproposes a solution to the multilevel key generation problem while An Efficient Time-Bound Hierarchical Key Management Scheme for Secure Broadcasting proposes a hierarchical key management scheme where the number of encryption keys to be managed depends on the number of access control policies.

To study the existing systems we referred whitepapers such as Portable Panic- The evolution of USB Insecurity. The information in An Introduction to Cryptography and Digital Signatures and Symmetric Key Management Systems was of immense help.

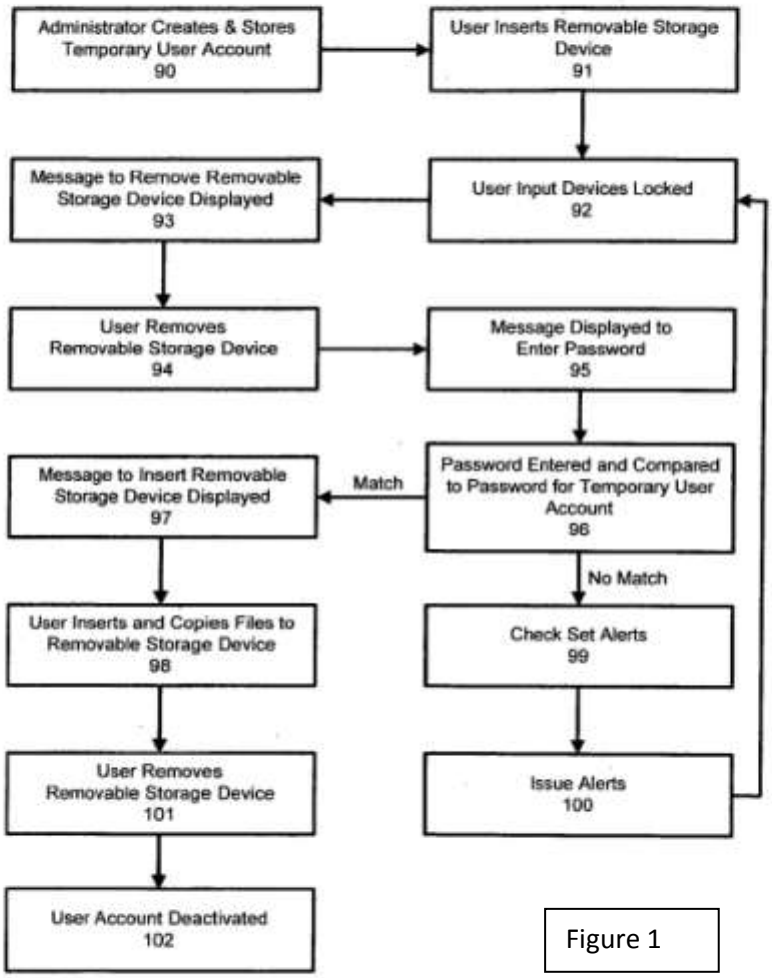

Figure 1

The given paper gave a skeleton to our project as it also talks about unauthorized transfer of legitimate data using a removable storage device and methods to prevent it.

5. System Analysis

5.1 Existing Systems

Some of the existing systems for encrypting files on the Thumb Drive are True Crypt, Microsoft's BitLocker, dmCrypt for Linux, and FileVault for MacOS. The following table lists the existing file encryption systems, and compares them comprehensively on parameters ranging from hidden containers to two-factor authentication.

Name	Hidden containers	Pre-boot authentication	Custom authentication	Multiple keys	Passphrase strengthening	Hardware acceleration	TPM	Filesystems	Two-factor authentication
BestCrypt	Yes	Yes	No	Yes[33]	Yes	No	No	Any supported by OS	Yes[34]
BitArmor DataControl	No	Yes	No	Yes	Yes	No	No	NTFS, FAT32 on non-system volumes	No
BitLocker Drive Encryption	No	Yes (With PIN or USB key)[35]	Yes[36]	Yes[35]	Yes (Recovery keys only)	No	Yes[35]	Chiefly NTFS †	Yes †
CGD	No	No	Yes[37]	Yes[38]	Yes[37]	No	No	Any supported by OS	Yes[37]
Checkpoint Full Disk Encryption	?	Yes	Yes	Yes	Yes	?	Yes[39]	?	Yes
CrossCrypt	No	No	No	No	No	No	No	?	No
CryptArchiver	No	No	No	No	?	No	No	?	?

Fig 2: Comparison of Existing Systems

5.2 Shortcomings of the existing systems

One of the major drawbacks of most of the above-discussed systems such as TrueCrypt and BitLocker is that they are pass-phrase dependant (for authentication) without offering a way for passphrase recovery in case they are lost, in which case the encrypted information is permanently lost.

The other major disadvantage of a majority of these systems is the amount of overhead involved in deploying and employing them. BitLocker, for example, requires a hard disk space of around 1.5 GB (just for itself) besides hogging up nearly 150-200 MB of RAM all alone, which takes a heavy toll on the system on which it has been installed.

5.3 Need / Motivation For The Proposed System

Keeping in view the above anomalies of the enlisted file/drive encryption systems, a new "transparent light-weight" encryption system is proposed.

"Transparent" here means that the user, should, at no point of time of using the software, be bothered with the intricacies and complexities involved with the software's functioning. Also the GUI of the system should be simple enough to enable naïve users to be able to use the software effectively without any hassles as such.

A "light-weight" software is one which fulfils its specifications without monopolizing resources.

The proposed system provides security from threats outside as well as inside the organization as the system ensures that the data is read only by those who are authorized to.

5.4 Feasibility Study Report

Technical feasibility

The system uses AES for encrypting user data and RSA Encryption algorithms for encrypting AES private key. These are widely accepted standard algorithms and are fairly robust. The user interfaces can be implemented with ease and are user friendly. Hence the project is technically feasible.

Economic feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of the new system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs.

In case of this project no special investment is needed to manage the tool. No specific training is required for employees to use the tool. Investment requires only once at the time of installation. The software used in this project is freeware so the cost of developing the tool is minimal.

Operational feasibility

The proposed software solves the problems identified with the existing system and takes advantage of the opportunities identified during scope definition. It satisfies the requirements identified in the requirements analysis phase of system development.

Schedule feasibility

Schedule feasibility is a measure of how reasonable the project timetable is. The software is an added asset for the organization and not an integral part of its system. Hence project deadlines are not mandatory but desirable. This ensures a certain degree of flexibility in the schedule and makes the project feasible in terms of the time required.

Resource feasibility

The time available to build the new software is ample and the software can be built and installed any time as it does not interfere with normal business operations. There is no need to train the personnel to use the software as it is built to be very user friendly.

Work Breakdown Structure

6. System Design

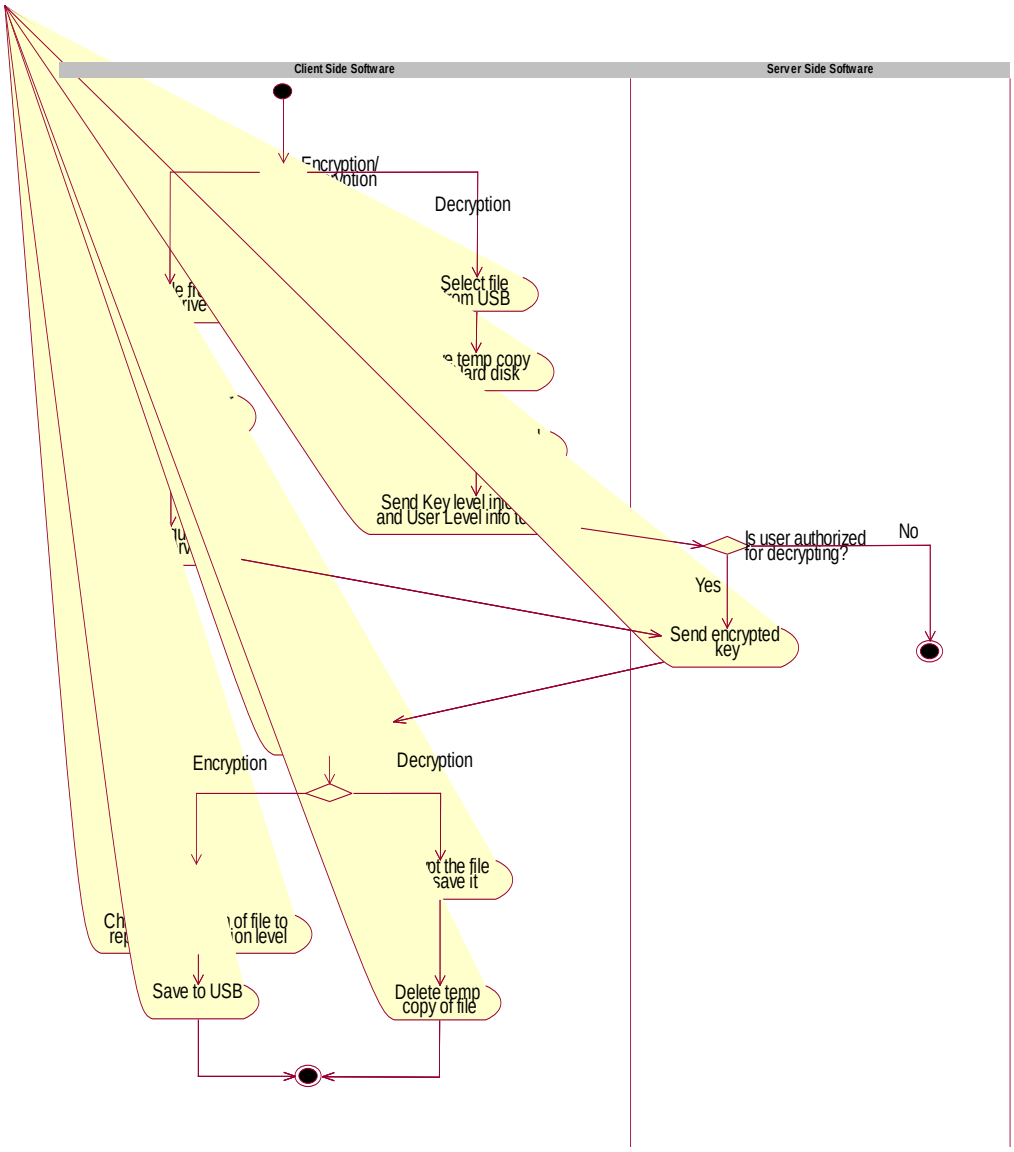

Fig. 4 Activity diagram

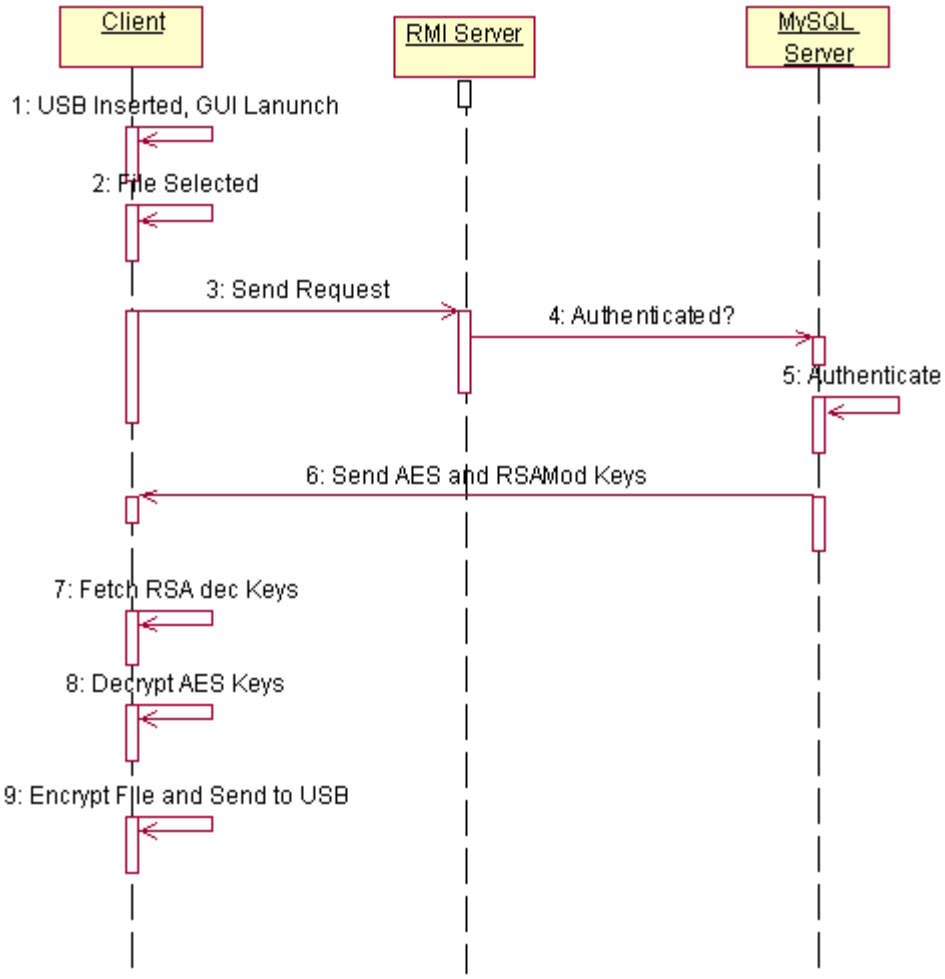

Fig. 5 Sequence diagram

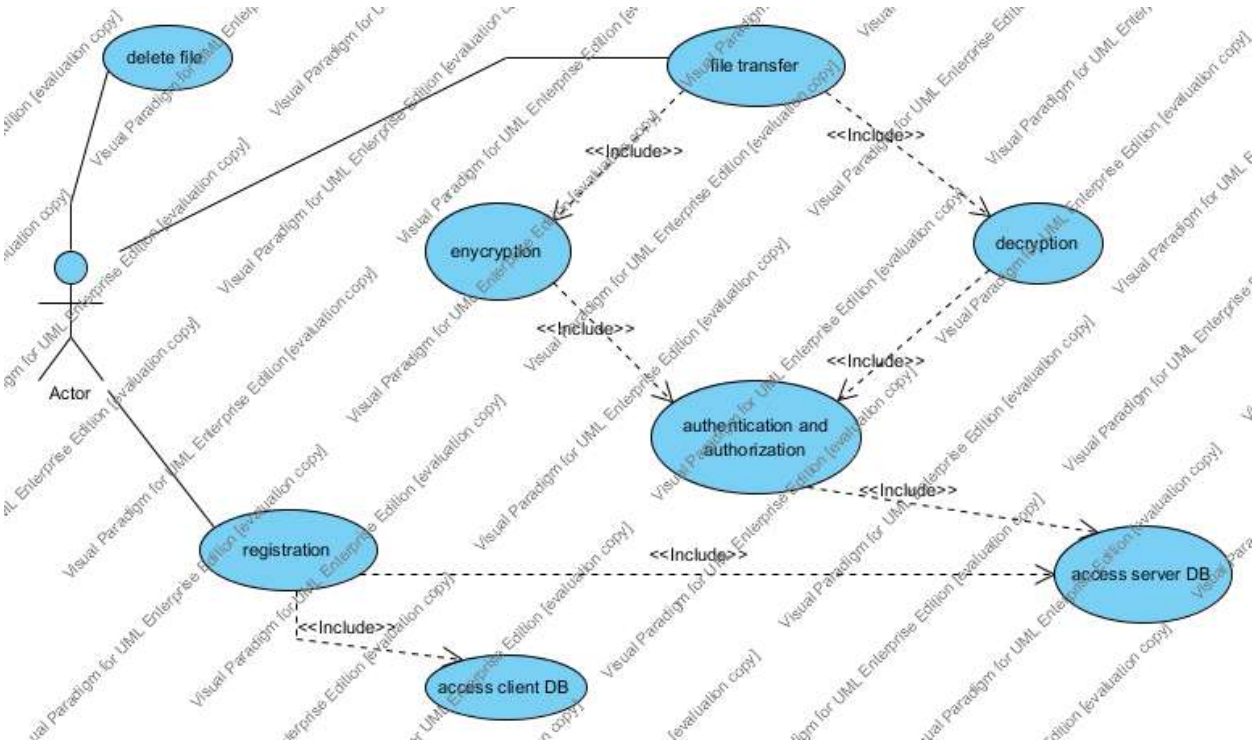

Fig 6 Use Case Diagram

DFD-LEVEL 0

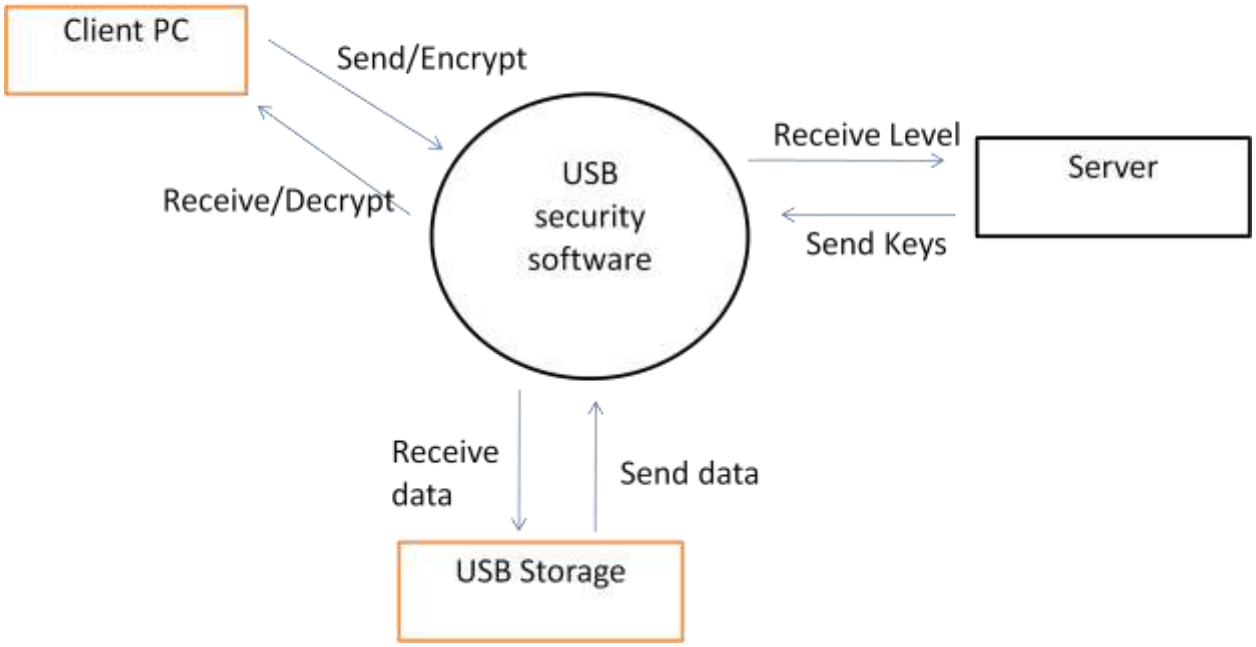

Fig. 7

DFD-LEVEL 1

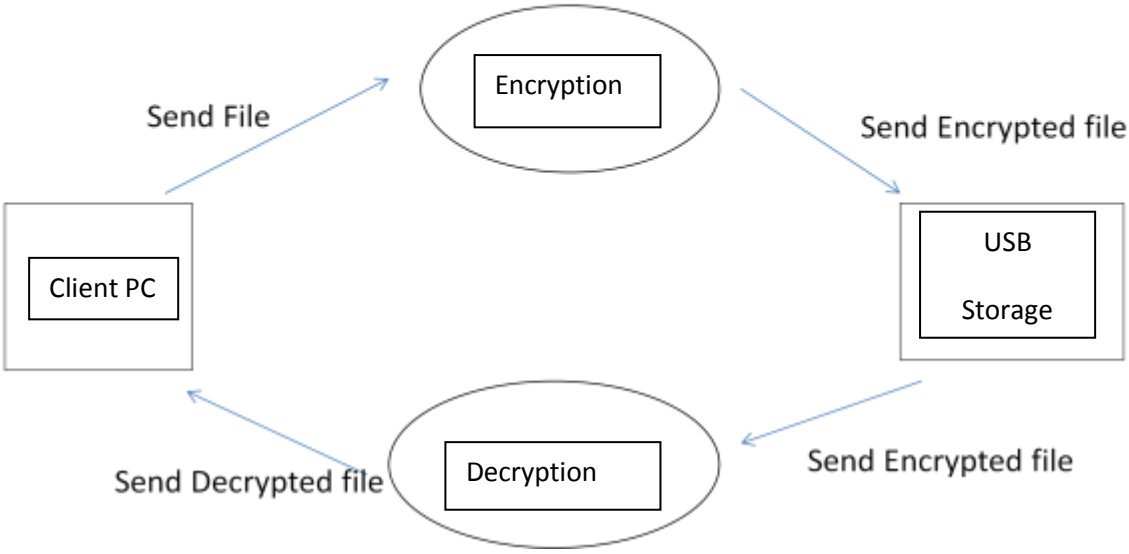

Fig. 8

7. TECHNIQUES USED

7.1 Software Model Used

Waterfall model

The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance.

The waterfall development model originates in the manufacturing and construction industries; highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.

The first known presentation describing use of similar phases in software engineering was held by Herbert D. Benington at Symposium on advanced programming methods for digital computers on 29 June 1956. This presentation was about the development of software for SAGE. In 1983 the paper was republished with a foreword by Benington pointing out that the process was not in fact performed in strict top-down, but depended on a prototype.

The first formal description of the waterfall model is often cited as a 1970 article by Winston W. Royce,[4] [5] although Royce did not use the term "waterfall" in this article. Royce presented this model as an example of a flawed, non-working model. This, in fact, is how the term is generally used in writing about software development—to describe a critical view of a commonly used software development practice.

Model

In Royce's original waterfall model, the following phases are followed in order:

1. Requirements specification
2. Design
3. Construction (implementation or coding)
4. Integration
5. Testing and debugging (validation)
6. Installation
7. Maintenance

Thus the waterfall model maintains that one should move to a phase only when its preceding phase is completed and perfected. Various modified waterfall models (including Royce's final

model), however, can include slight or major variations on this process.

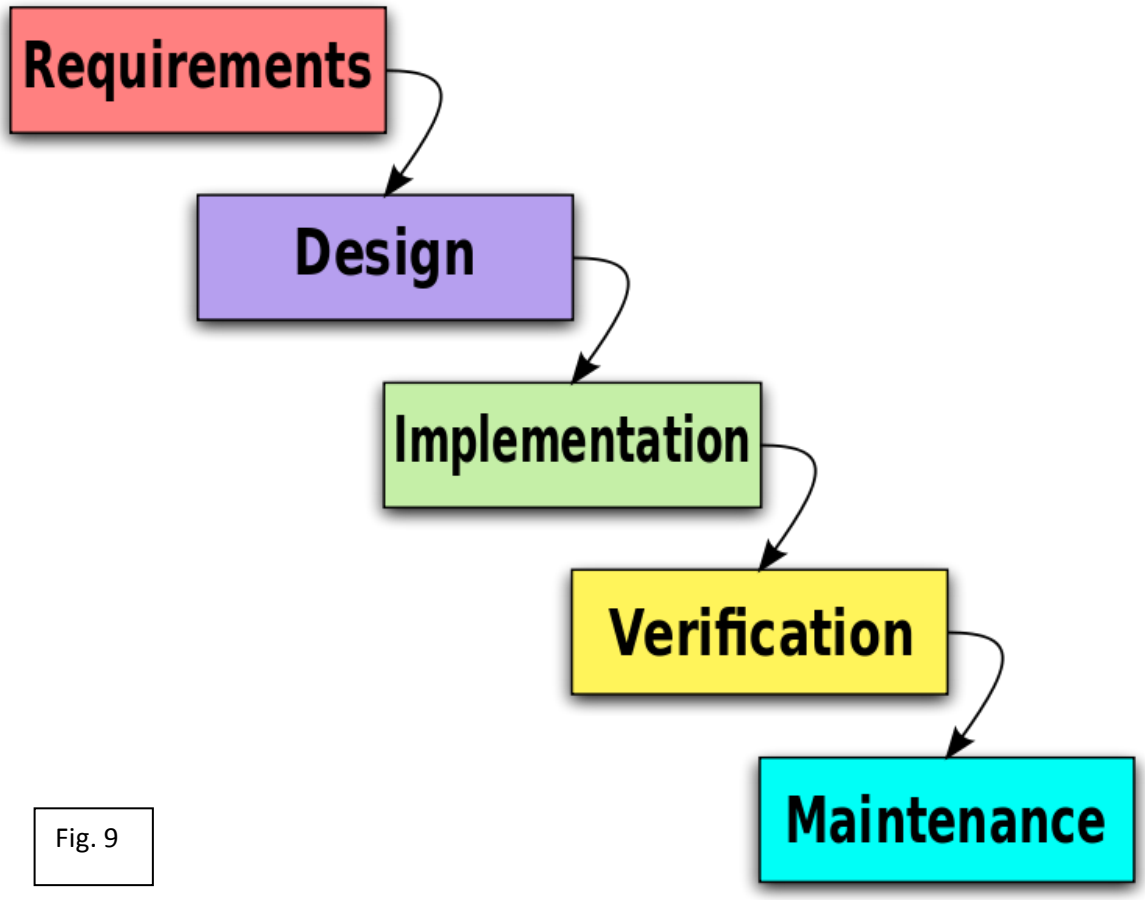

Fig. 9

Supporting arguments

Time spent early in the software production cycle can lead to greater economy at later stages. McConnell shows that a bug found in the early stages (such as requirements specification or design) is cheaper in money, effort, and time to fix than the same bug found later on in the process.[8] To take an extreme example, if a program design turns out to be impossible to implement, it is easier to fix the design at the design stage than to realize months later, when program components are being integrated, that all the work done so far has to be scrapped because of a broken design.

This is the central idea behind Big Design Up Front and the waterfall model: time spent early on making sure requirements and design are correct saves much time and effort later. Thus, the thinking of those who follow the waterfall process goes, make sure each phase is 100% complete and absolutely correct before proceeding to the next phase. Program requirements should be set in stone before design begins (otherwise work put into a design based on incorrect requirements is wasted). The program's design should be perfect before people begin to implement the design (otherwise they implement the wrong design and their work is wasted), etc.

A further argument for the waterfall model is that it places emphasis on documentation (such as requirements documents and design documents) as well as source code. In less thoroughly designed and documented methodologies, knowledge is lost if team members leave before the project is completed, and it may be difficult for a project to recover from the loss. If a fully working design document is present (as is the intent of Big Design Up Front and the waterfall model), new team members or even entirely new teams should be able to familiarize themselves by reading the documents.

Some waterfall proponents prefer the waterfall model for its simple approach and argue that it is more disciplined. The waterfall model provides a structured approach; the model itself progresses linearly through discrete, easily understandable and explainable phases and thus is easy to understand; it also provides easily identifiable milestones in the development process. It is perhaps for this reason that the waterfall model is used as a beginning example of a development model in many software engineering texts and courses.

It is argued that the waterfall model and Big Design up Front in general can be suited to software projects that are stable (especially those projects with unchanging requirements, such as with shrink wrap software) and where it is possible and likely that designers will be able to fully predict problem areas of the system and produce a correct design before implementation is started. The waterfall model also requires that implementers follow the well-made, complete design accurately, ensuring that the integration of the system proceeds smoothly.

Criticism

Advocates of Agile software development argue the waterfall model is a bad idea in practice—believing it impossible for any non-trivial project to finish a phase of a software product's lifecycle perfectly before moving to the next phases and learning from them.

For example, clients may not know exactly what requirements they need before reviewing a working prototype and commenting on it. They may change their requirements constantly. Designers and programmers may have little control over this. If clients change their requirements after the design is finalized, the design must be modified to accommodate the new requirements. This effectively means invalidating a good deal of working hours, which means increased cost, especially if a large amount of the project's resources has already been invested in Big Design Up Front.

Designers may not be aware of future implementation difficulties when writing a design for an unimplemented software product. That is, it may become clear in the implementation phase that a particular area of program functionality is extraordinarily difficult to implement. In this case, it is better to revise the design than persist in a design based on faulty predictions, and that does not account for the newly discovered problems.

Modified models

In response to the perceived problems with the pure waterfall model, many modified waterfall models have been introduced. These models may address some or all of the criticisms of the pure

waterfall model. Many different models are covered by Steve McConnell in the "lifecycle planning" chapter of his book Rapid Development: Taming Wild Software Schedules.

While all software development models bear some similarity to the waterfall model, as all software development models incorporate at least some phases similar to those used in the waterfall model, this section deals with those closest to the waterfall model. For models that apply further differences to the waterfall model, or for radically different models seek general information on the software development process

7.2. SQL Server and Database Encryption Keys

SQL Server encrypts data with a hierarchical encryption and key management infrastructure. Each layer encrypts the layer below it by using a combination of certificates, asymmetric keys, and symmetric keys. Asymmetric keys and symmetric keys can be stored outside of SQL Server in an Extensible Key Management (EKM) module.

The following illustration shows that each layer of the encryption hierarchy encrypts the layer beneath it, and displays the most common encryption configurations. The access to the start of the hierarchy is usually protected by a password.

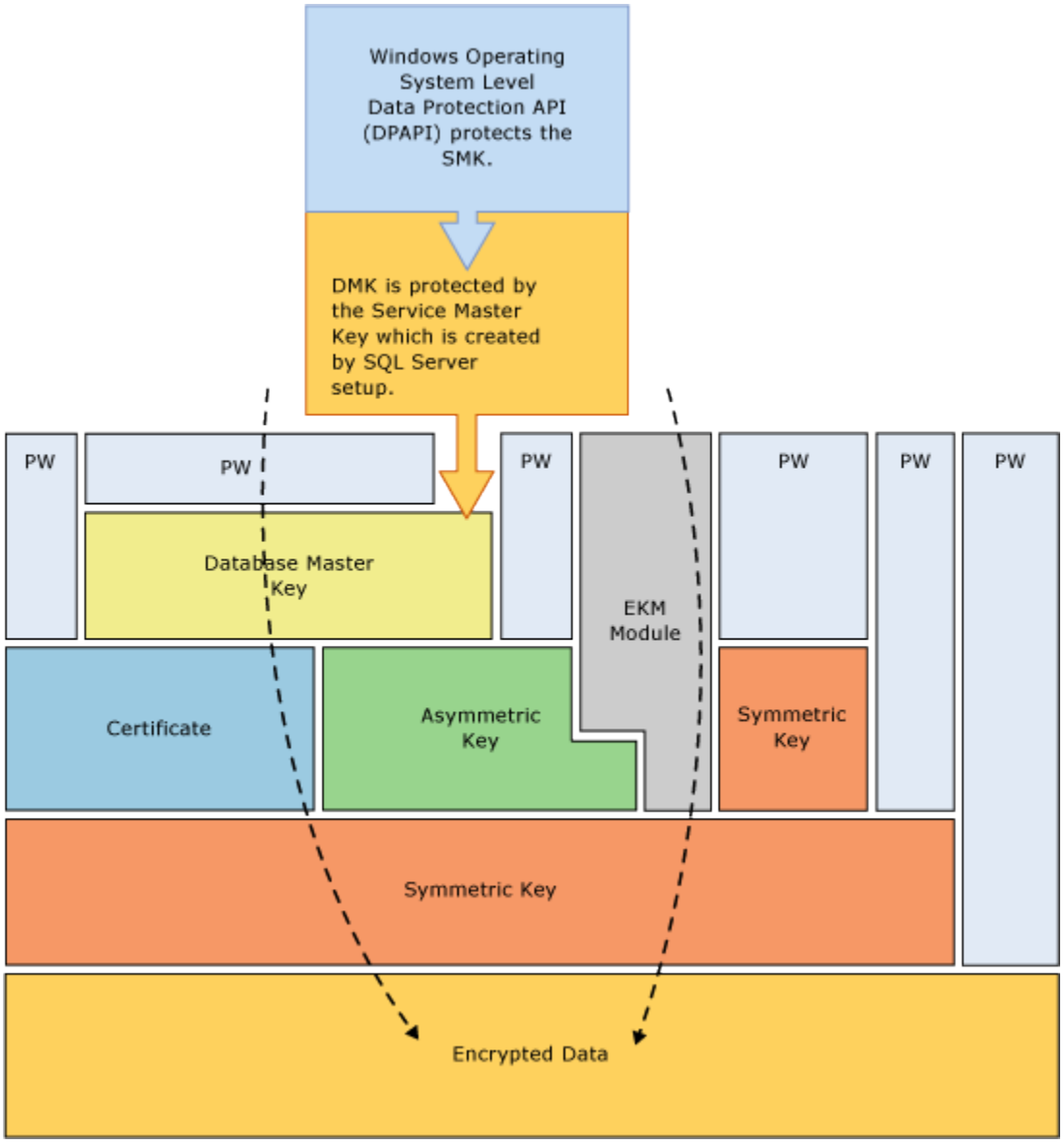

Fig. 10

Keep in mind the following concepts:

- For best performance, encrypt data using symmetric keys instead of certificates or asymmetric keys.
- Database master keys are protected by the Service Master Key. The Service Master Key is created by SQL Server setup and is encrypted with the Windows Data Protection API (DPAPI).
- Other encryption hierarchies stacking additional layers are possible.
- An Extensible Key Management (EKM) module holds symmetric or asymmetric keys outside of SQL Server.

- Transparent Data Encryption (TDE) must use a symmetric key called the database encryption key which is protected by either a certificate protected by the database master key of the master database, or by an asymmetric key stored in an EKM.
- The Service Master Key and all Database Master Keys are symmetric keys.

SQL Server provides the following mechanisms for encryption:

- Transact-SQL functions
- Asymmetric keys
- Symmetric keys
- Certificates
- Transparent Data Encryption

Transact-SQL Functions

Individual items can be encrypted as they are inserted or updated using Transact-SQL functions. For more information, see ENCRYPTBYPASSPHRASE (Transact-SQL) and DECRYPTBYPASSPHRASE (Transact-SQL).

Certificates

A public key certificate, usually just called a certificate, is a digitally-signed statement that binds the value of a public key to the identity of the person, device, or service that holds the corresponding private key. Certificates are issued and signed by a certification authority (CA). The entity that receives a certificate from a CA is the subject of that certificate. Typically, certificates contain the following information.

- The public key of the subject.
- The identifier information of the subject, such as the name and e-mail address.
- The validity period. This is the length of time that the certificate is considered valid.

  A certificate is valid only for the period of time specified within it; every certificate contains Valid From and Valid To dates. These dates set the boundaries of the validity period. When the validity period for a certificate has passed, a new certificate must be requested by the subject of the now-expired certificate.

- Issuer identifier information.
- The digital signature of the issuer.

  This signature attests to the validity of the binding between the public key and the identifier information of the subject. (The process of digitally signing information entails transforming the information, as well as some secret information held by the sender, into a tag called a signature.)

A primary benefit of certificates is that they relieve hosts of the need to maintain a set of passwords for individual subjects. Instead, the host merely establishes trust in a certificate issuer, which may then sign an unlimited number of certificates.

When a host, such as a secure Web server, designates an issuer as a trusted root authority, the host implicitly trusts the policies that the issuer has used to establish the bindings of certificates it issues. In effect, the host trusts that the issuer has verified the identity of the certificate subject. A host designates an issuer as a trusted root authority by putting the self-signed certificate of the issuer, which contains the public key of the issuer, into the trusted root certification authority certificate store of the host computer. Intermediate or subordinate certification authorities are trusted only if they have a valid certification path from a trusted root certification authority.

The issuer can revoke a certificate before it expires. Revocation cancels the binding of a public key to an identity that is asserted in the certificate. Each issuer maintains a certificate revocation list that can be used by programs when they are checking the validity of any given certificate.

The self-signed certificates created by SQL Server follow the X.509 standard and support the X.509 v1 fields.

Asymmetric Keys

An asymmetric key is made up of a private key and the corresponding public key. Each key can decrypt data encrypted by the other. Asymmetric encryption and decryption are relatively resource-intensive, but they provide a higher level of security than symmetric encryption. An asymmetric key can be used to encrypt a symmetric key for storage in a database.

Symmetric Keys

A symmetric key is one key that is used for both encryption and decryption. Encryption and decryption by using a symmetric key is fast, and suitable for routine use with sensitive data in the database.

Transparent Data Encryption

Transparent Data Encryption (TDE) is a special case of encryption using a symmetric key. TDE encrypts an entire database using that symmetric key called the database encryption key. The database encryption key is protected by other keys or certificates which are protected either by the database master key or by an asymmetric key stored in an EKM module. For more information.

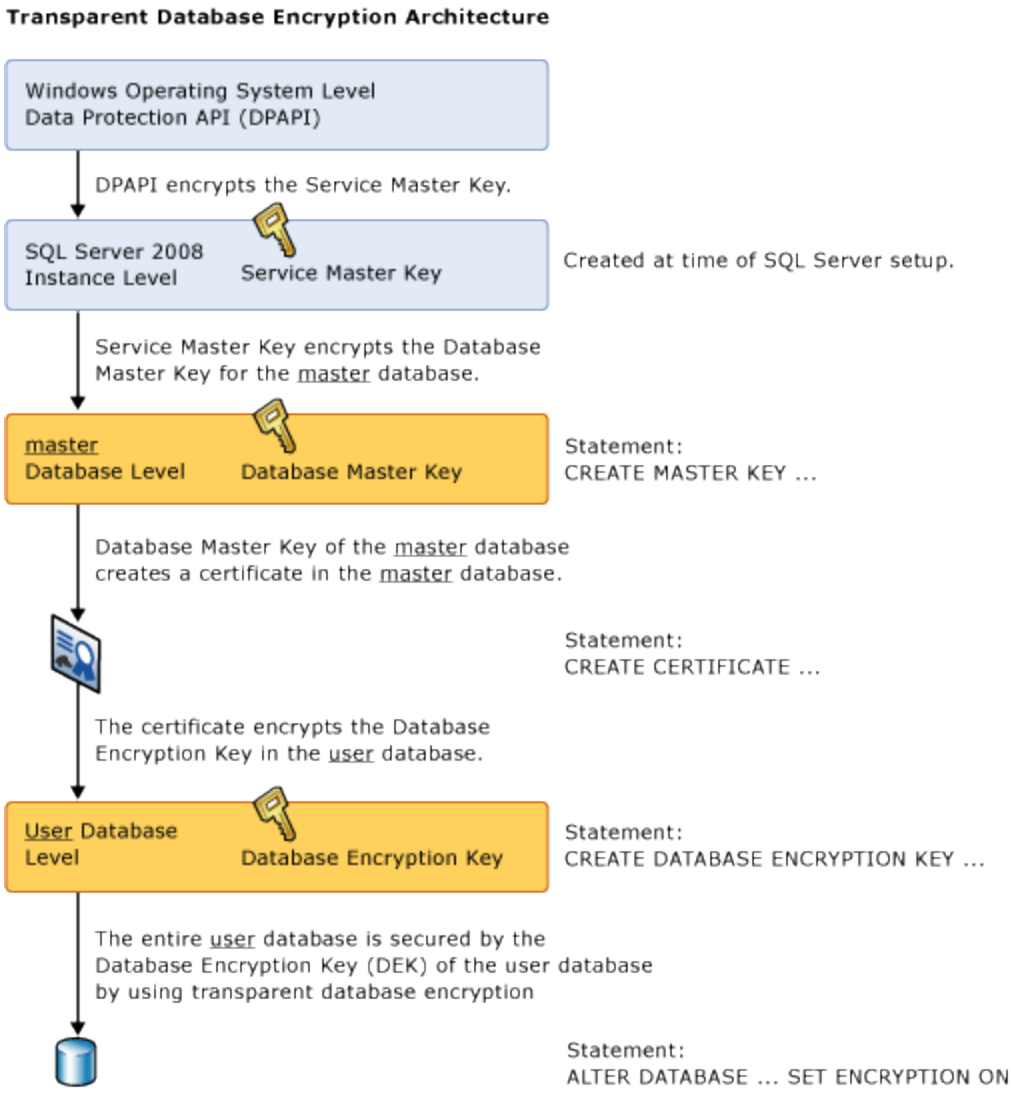

Fig. 11

7.3 Applications for SQL Server and Database Keys

SQL Server has two primary applications for keys: a service master key (SMK) generated on and for a SQL Server instance, and a database master key (DMK) used for a database.

The SMK is automatically generated the first time the SQL Server instance is started and is used to encrypt a linked server password, credentials, and the database master key. The SMK is encrypted by using the local computer key using the Windows Data Protection API (DPAPI). The DPAPI uses a key that is derived from the Windows credentials of the SQL Server service account and the computer's credentials. The service master key can only be decrypted by the service account under which it was created or by a principal that has access to the machine's credentials.

The database master key is a symmetric key that is used to protect the private keys of certificates and asymmetric keys that are present in the database. It can also be used to encrypt data, but it has length limitations that make it less practical for data than using a symmetric key.

When it is created, the master key is encrypted by using the Triple DES algorithm and a user-supplied password. To enable the automatic decryption of the master key, a copy of the key is encrypted by using the SMK. It is stored in both the database where it is used and in the master system database.

The copy of the DMK stored in the master system database is silently updated whenever the DMK is changed. However, this default can be changed by using the DROP ENCRYPTION BY SERVICE MASTER KEY option of the ALTER MASTER KEY statement. A DMK that is not encrypted by the service master key must be opened by using the OPEN MASTER KEY statement and a password.

Managing SQL Server and Database Keys

Managing encryption keys consists of creating new database keys, creating a backup of the server and database keys, and knowing when and how to restore, delete, or change the keys. To manage symmetric keys, you can use the tools included in SQL Server to do the following:

- Back up a copy of the server and database keys so that you can use them to recover a server installation, or as part of a planned migration.
- Restore a previously saved key to a database. This enables a new server instance to access existing data that it did not originally encrypt.
- Delete the encrypted data in a database in the unlikely event that you can no longer access encrypted data.
- Re-create keys and re-encrypt data in the unlikely event that the key is compromised. As a security best practice, you should re-create the keys periodically (for example, every few months) to protect the server from attacks that try to decipher the keys.
- Add or remove a server instance from a server scale-out deployment where multiple servers share both a single database and the key that provides reversible encryption for that database.

7.4 The AES (Advanced Encryption Standard) Algorithm

The Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001. Based on the Rijndael cipher developed by two Belgian cryptographers, Joan Daemen andVincent Rijmen, who submitted a proposal which was evaluated by the NIST during the AES selection process.

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). It 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 publicly accessible and open cipher approved by the National Security Agency (NSA) for top secret information when used in an NSA approved cryptographic module

The name *Rijndael* (Dutch pronunciation: [ˈrɛindaːl]) is a play on the names of the two inventors (Joan Daemen and Vincent Rijmen). Strictly speaking, the AES standard is a variant of Rijndael where the block size is restricted to 128 bits.

7.5 Description Of The Cipher

AES is based on a design principle known as a substitution-permutation network, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification *per se* is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

AES operates on a 4×4 column-major order matrix of bytes, termed the *state*, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field.

The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext. The number of cycles of repetition are as follows:

- 10 cycles of repetition for 128-bit keys.
- 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys.

Each round consists of several processing steps, each containing five similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

7.6 High-Level Description Of the Algorithm

1. **Key Expansion**—round keys are derived from the cipher key using key schedule.
2. **Initial Round**
    1. **AddRoundKey**—each byte of the state is combined with the round key using bitwise xor.
3. **Rounds**
    1. **SubBytes**—a non-linear substitution step where each byte is replaced with another according to a lookup table.
    2. **ShiftRows**—a transposition step where each row of the state is shifted cyclically a certain number of steps.

3. **MixColumns**—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
4. **AddRoundKey**

4. **Final Round (no MixColumns)**
    1. **SubBytes**
    2. **ShiftRows**
    3. **AddRoundKey**

The SubBytes Step

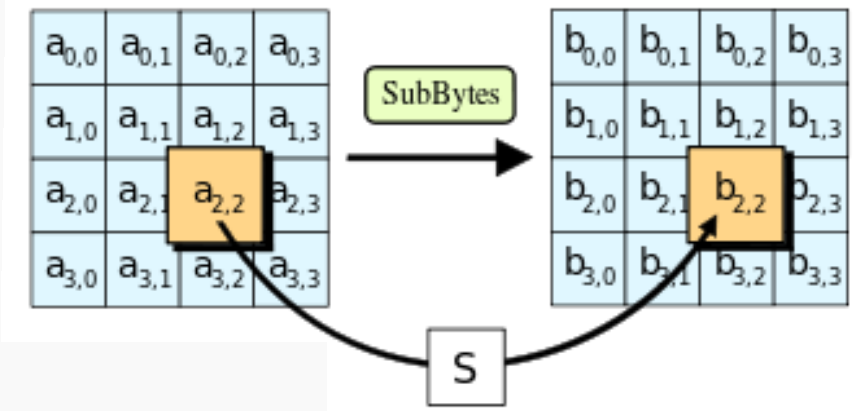

Fig. 12

In the SubBytes step, each byte in the *state* matrix is replaced with a SubByte using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF(2^8), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), and also any opposite fixed points.

The ShiftRows Step

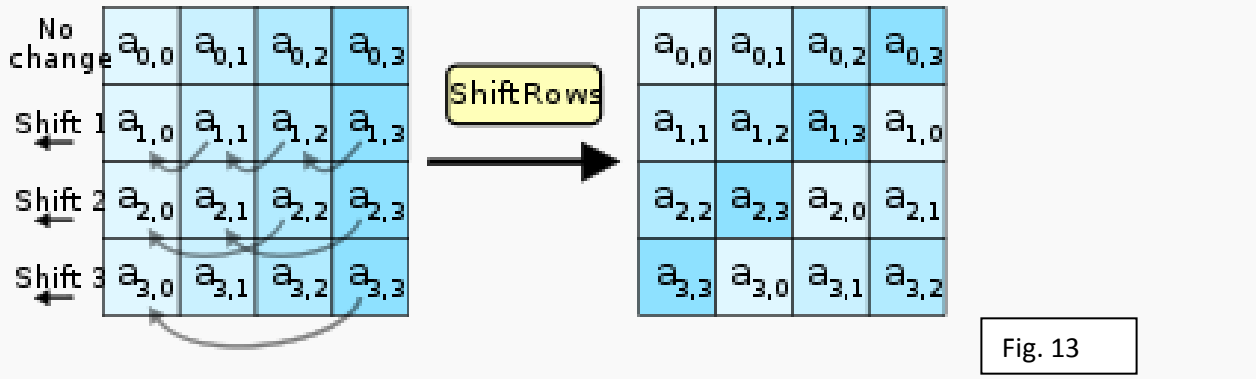

Fig. 13

The ShiftRows step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For blocks of sizes 128 bits and 192 bits, the shifting pattern is the same. Row n is shifted left circular by n-1 bytes. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. (Rijndael variants with a larger block size have slightly different offsets). For a 256-bit block, the first row is unchanged and the shifting for the second, third and fourth row is 1 byte, 3 bytes and 4 bytes respectively—this change only applies for the Rijndael cipher when used with a 256-bit block, as AES does not use 256-bit blocks. The importance of this step is to make columns not linear independent If so, AES becomes four independent block ciphers.

The MixColumns Step

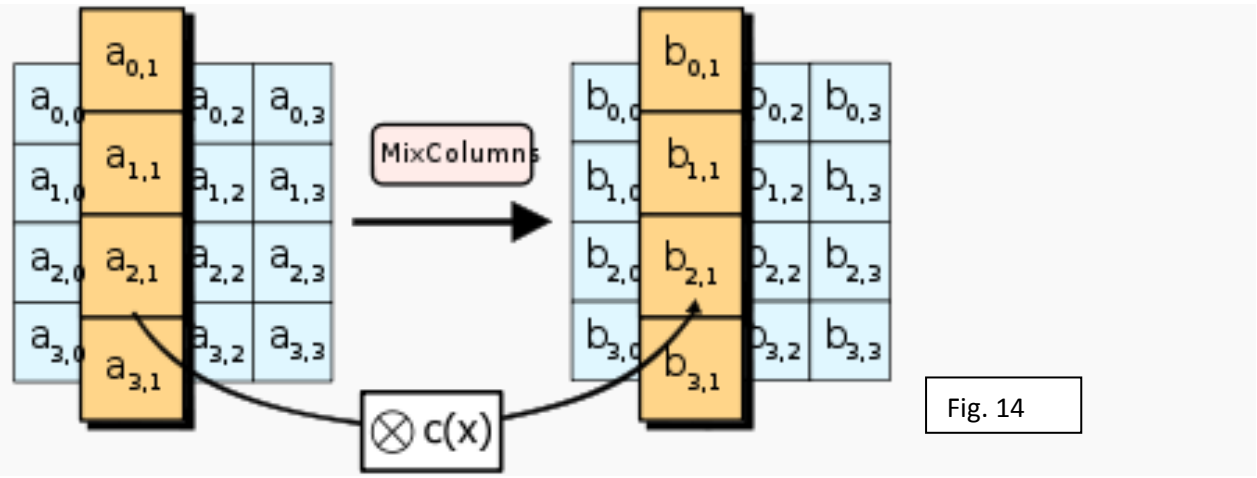

Fig. 14

In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. TheMixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher.

During this operation, each column is multiplied by the known matrix that for the 128-bit key is:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

The multiplication operation is defined as: multiplication by 1 means no change, multiplication by 2 means shifting to the left, and multiplication by 3 means shifting to the left and then performing xor with the initial unshifted value. After shifting, a conditional xor with 0x11B should be performed if the shifted value is larger than 0xFF.

In more general sense, each column is treated as a polynomial over GF(2^8) and is then multiplied modulo x^4+1 with a fixed polynomial $c(x) = 0x03 \cdot x^3 + x^2 + x + 0x02$. The coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from GF(2)[x]. The MixColumns step can also be viewed as a multiplication by a particular MDS matrix in a finite field. This process is described further in the article Rijndael mix columns.

The AddRoundKey Step

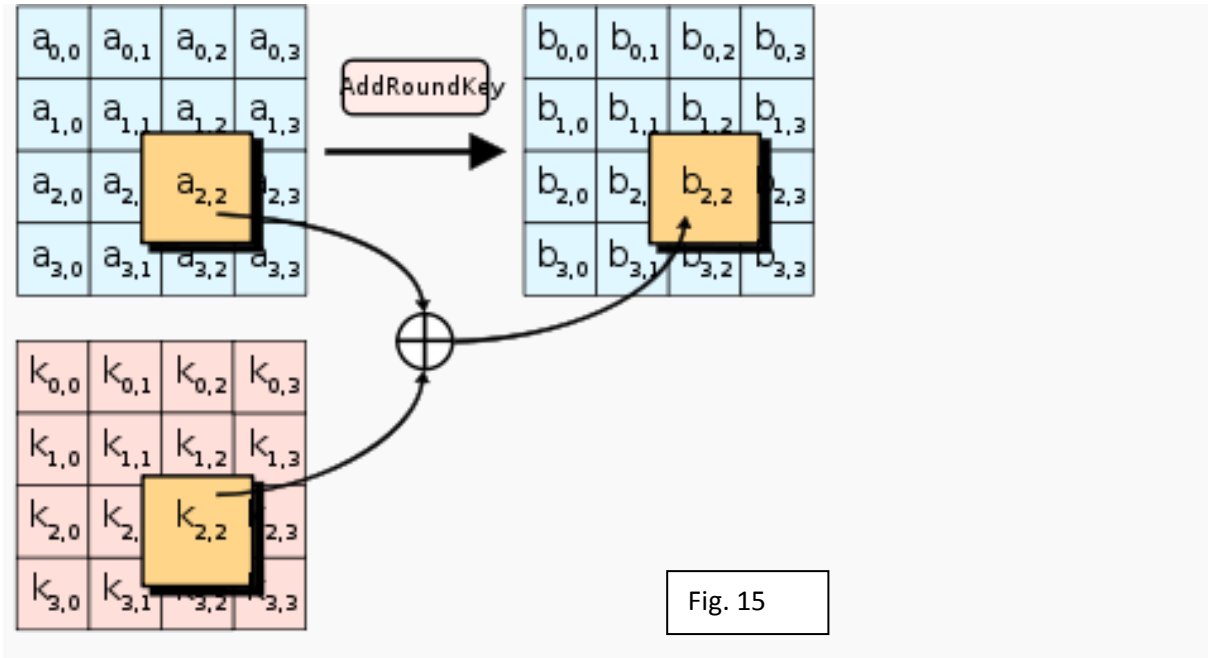

Fig. 15

In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.

7.7 Optimization Of The Cipher

On systems with 32-bit or larger words, it is possible to speed up execution of this cipher by combining the SubBytes andShiftRows steps with the MixColumns step by transforming them into a sequence of table lookups. This requires four 256-entry 32-bit tables, and utilizes a total of four kilobytes (4096 bytes) of memory — one kilobyte for each table. A round can then be done with 16 table lookups and 12 32-bit exclusive-or operations, followed by four 32-bit exclusive-or operations in the AddRoundKey step.

If the resulting four-kilobyte table size is too large for a given target platform, the table lookup operation can be performed with a single 256-entry 32-bit (i.e. 1 kilobyte) table by the use of circular rotates.

Using a byte-oriented approach, it is possible to combine the SubBytes, ShiftRows, and MixColumns steps into a single round operation.

8. IMPLEMENTATION

8.1 CODE for LOGIN AUTHENTICATION:

```
package sg;

import java.awt.Dimension;

import java.awt.GridLayout;

import java.awt.Toolkit;

import java.awt.event.*;

import javax.swing.*;

import java.sql.*;


public class authentication implements ActionListener{

JFrame fr;

JTextField userid;

JPasswordField userpass;

JButton submit;

JLabel uid;

JLabel pass;

static UserInterface ui;

String ed ,query;

ResultSet rs;

Connection con;

Statement stmt;

boolean authentic = false;


authentication(String externalDirectory){
```

```
        ed = externalDirectory;
        fr=new JFrame("Authentication");
        Toolkit tk = Toolkit.getDefaultToolkit();
        Dimension dim = tk.getScreenSize();
        int frwidth=dim.width;
        int frheight=dim.height;
        fr.setSize(frwidth,frheight);
        int xPos = (dim.width / 2) - (frwidth/ 2);
        int yPos = (dim.height / 2) - (frheight/ 2);
        fr.setLocation(xPos, yPos);

        fr.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        fr.setUndecorated(true);
        fr.getRootPane().setWindowDecorationStyle(JRootPane.NONE);
        fr.setLayout(null) ;

        uid = new JLabel("User id");
        pass = new JLabel ("Password");
        userid = new JTextField();
        userpass = new JPasswordField();
        submit = new JButton("submit");
        submit.addActionListener(this);

        uid.setBounds(350,100,100,20);
    userid.setBounds(450,100,200,20);
```

```
    pass.setBounds(350,130,100,20);
    userpass.setBounds(450,130,200,20);
    submit.setBounds(450,160,100,20);

        fr.add(uid);
        fr.add(pass);
        fr.add(userid);
        fr.add(userpass);
        fr.add(submit);
        fr.setVisible(true);
}

public void jdbc_con()
{
        try{
        Class.forName("com.mysql.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost/projectdb","root","root");
        stmt = con.createStatement();
        query = "select * from empinfo" ;
        //query = "select * from empinfo where empid = "'+userid.getText().toString()+"'";
        rs=stmt.executeQuery(query);
        System.out.println(rs);
}
catch (Exception e) {
        // TODO: handle exception
```

```
        e.printStackTrace();
}
}

public void check_authentication()
{
        try {
                        if("banana".equals(userpass.getText()))
                                authentic = true;

                        if(!authentic)
                        {
                                JOptionPane.showMessageDialog(null,"Password and username does'nt match","warning message", JOptionPane.INFORMATION_MESSAGE);

                        }
                //till this point
                }catch (Exception e) {
                // TODO Auto-generated catch block
                System.out.println("database read error");
        }

        }
@Override
public void actionPerformed(ActionEvent e) {
```

```
        if(e.getSource().equals(submit)){
                //jdbc_con();
                check_authentication();
                if(authentic)
                {
                        System.out.println("authentic");
                        ui = new UserInterface(ed);
                }
        }


}
public void closeWindow()
{
fr.dispose();
ui.closeWindow();
}
}
```

8.2 Code For Disabling Key Strokes

```
package sg;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import com.sun.jna.platform.win32.Kernel32;
import com.sun.jna.platform.win32.User32;
import com.sun.jna.platform.win32.WinDef.HMODULE;
import com.sun.jna.platform.win32.WinDef.LRESULT;
import com.sun.jna.platform.win32.WinDef.WPARAM;
import com.sun.jna.platform.win32.WinUser.HHOOK;
import com.sun.jna.platform.win32.WinUser.KBDLLHOOKSTRUCT;
import com.sun.jna.platform.win32.WinUser.LowLevelKeyboardProc;
import com.sun.jna.platform.win32.WinUser.MSG;
public class KeyHook {
    private static HHOOK hhk;
    private static LowLevelKeyboardProc keyboardHook;
    private static User32 lib;
    private static boolean working = true;
    public static void blockWindowsKey() {
        if (isWindows()) {
            new Thread(new Runnable() {
             @Override
                public void run() {
                    lib = User32.INSTANCE;
                    HMODULE hMod = Kernel32.INSTANCE.GetModuleHandle(null);
```

```
            keyboardHook = new LowLevelKeyboardProc() {
                public LRESULT callback(int nCode, WPARAM wParam,
KBDLLHOOKSTRUCT info) {
                    if (nCode >= 0) {
                        switch (info.vkCode){
                            case 0x5B:
                            case 0x5C:
                                return new LRESULT(1);
                            default: //do nothing
                        }
                    }
                    return lib.CallNextHookEx(hhk, nCode, wParam, info.getPointer());
                }
            };
            hhk = lib.SetWindowsHookEx(13, keyboardHook, hMod, 0);
        // This bit never returns from GetMessage
            int result;
            MSG msg = new MSG();
            while ((result = lib.GetMessage(msg, null, 0, 0)) != 0) {
                if (result == -1) {
                    break;
                } else {
                    lib.TranslateMessage(msg);
                    lib.DispatchMessage(msg);
                }
```

```
            }
            lib.UnhookWindowsHookEx(hhk);
          }
        }).start();
    }
}

public static void altTabStopper()
{
        new Thread(new Runnable()
        {
                public void run()
                {
        try
    {
        Robot robot = new Robot();
        while (working)
        {
            robot.keyRelease(KeyEvent.VK_ALT);
            robot.keyRelease(KeyEvent.VK_TAB);
            //robot.keyRelease(KeyEvent.VK_DELETE);
           // robot.keyRelease(KeyEvent.VK_CONTROL);
            }
        }
            catch (Exception e) { e.printStackTrace(); System.exit(-1); }
```

```
 }
       }).start();
}

public static void taskbar_stopper()
{
       new Thread(new Runnable()
       {
                public void run()
                {
        try
    {
        Robot rbt = new Robot();
         while (working)
     {
         rbt.keyRelease(KeyEvent.VK_CONTROL);
          rbt.keyRelease(KeyEvent.VK_ESCAPE);
          rbt.keyRelease(KeyEvent.VK_SHIFT);
          //robot.keyRelease(KeyEvent.VK_TAB);
          //rbt.keyRelease(KeyEvent.VK_DELETE);

     }
    }
     catch (Exception e)
     {
```

```
            e.printStackTrace();
            System.exit(-1);
            }
    }
    }).start();
}
public static void keysUnhook()
    {
        working=false;
    }
public static void unblockWindowsKey() {
        if (isWindows() && lib != null) {
            lib.UnhookWindowsHookEx(hhk);
                    KeyHook.keysUnhook();
        }
    }
public static boolean isWindows(){
        String os = System.getProperty("os.name").toLowerCase();
        return (os.indexOf( "win" ) >= 0);
    }
}
```

Fig. 16

Authentication Window

8.3 Code For Interface Design

```
package sg;

import javax.swing.*;

import java.awt.* ;

import java.awt.event.*;

import java.io.File;

import java.io.FileNotFoundException;

import javax.swing.JButton;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.filechooser.FileSystemView;


public class UserInterface implements ActionListener

{

        static int frwidth;

        static int frheight;

        Jbutton d2u , u2d;

        Jbutton refresh, exit;

        Jlabel diskLabel,usbLabel;

        FileTreeFrame diskbrowsePanel,usbbrowsePanel;

        Jframe frame;

        //JtextField details;

        ImageIcon pendrive;

        Jlabel imgLabel,details;
```

```
        static String ed;
        static String drivesList[] = new String[26];
        static String selectedDrive = "c:\\" ;
        JcomboBox<String> c;
        public static void main(String[] args){
        new UserInterface("d:\\");
        }
public UserInterface(String externalDirectory )
{
        //frame settings
        ed=externalDirectory;
        frame = new Jframe("usbframe");
        frame.setTitle("USB Security Software");
        Toolkit tk = Toolkit.getDefaultToolkit();
        Dimension dim = tk.getScreenSize();
        frwidth=dim.width;
        frheight=dim.height;
        frame.setSize(frwidth,frheight);
        int xPos = (dim.width / 2) – (frwidth/ 2);
        int yPos = (dim.height / 2) – (frheight/ 2);
        frame.setLocation(xPos, yPos);
        frame.setLayout(null) ;
        frame.setDefaultCloseOperation(Jframe.DO_NOTHING_ON_CLOSE);
        frame.setUndecorated(true);
        frame.getRootPane().setWindowDecorationStyle(JrootPane.NONE);
```

```
//disk-Label
diskLabel=new Jlabel(“Local Disk”);
diskLabel.setBounds(80, 10, 100, 20);
diskLabel.setFont(new Font(“Serif”, Font.PLAIN, 16));
frame.add(diskLabel);

//usb-Label
usbLabel=new Jlabel(“USB Disk”);
usbLabel.setBounds(864, 10, 100, 20);
usbLabel.setFont(new Font(“Serif”, Font.PLAIN, 16));
frame.add(usbLabel);

//Label – usb security
details = new Jlabel(“USB Security Software”);
details.setBounds(362, 5, 300, 30);
details.setFont(new Font(“Serif”, Font.BOLD, 30));
frame.add(details);

//local disks
diskbrowsePanel= new FileTreeFrame(selectedDrive);
diskbrowsePanel.setBounds(10, 30, 250, frheight-60);
frame.add(diskbrowsePanel);

//combo box for list of drives
c = new JcomboBox<String>();
```

```
listDrives();
for(int i=0;i<drivesList.length;i++)
{
        c.addItem(drivesList[i]);
}
c.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
                        selectedDrive=(String) ((JcomboBox<?>)
                        e.getSource()).getSelectedItem();
                        System.out.println("selected drive:"+selectedDrive);
                        diskbrowsePanel.setVisible(false);
                        diskbrowsePanel= new FileTreeFrame(selectedDrive);
                        diskbrowsePanel.setBounds(10, 30, 250, frheight-60);
                        frame.add(diskbrowsePanel);
          }
   });
c.setBounds(260, 100, 100, 20);
frame.addI;

//disk to usb button
d2u = new Jbutton("Send To USB");
d2u.setBounds(412, 230, 200, 40);
frame.add(d2u);
d2u.addActionListener(this);
```

```
        //image
        pendrive = new ImageIcon("PENDRIVE.jpg");
        imgLabel = new Jlabel(pendrive);
        imgLabel.setBounds(412,284,200,200);
        frame.add(imgLabel);

        //usb to disk button
        u2d = new Jbutton("Send To Disk");
        u2d.setBounds(412, 500, 200, 40);
        u2d.addActionListener(this);
        frame.add(u2d);

        //refresh button
        refresh=new Jbutton("refresh");
        refresh.addActionListener(this);
        refresh.setBounds(658, 100, 100, 30);
        frame.add(refresh);

        //usb drive
        usbbrowsePanel = new FileTreeFrame(ed);
        usbbrowsePanel.setBounds(764, 30, 250, frheight-60);
        frame.add(usbbrowsePanel);
        frame.setVisible(true);
}
```

```
public void actionPerformed(ActionEvent e){
        if(e.getSource() == d2u){
                try{
                Encryption encrypt = new Encryption(diskbrowsePanel.getFilePath());
                encrypt.enc();}
                catch(NullPointerException ne)
                {
                        JoptionPane.showMessageDialog(null,"Please Select a file to
                        Encrypt!!","Warning", JoptionPane.WARNING_MESSAGE);
                }
                }

if(e.getSource() == refresh){
        diskbrowsePanel.setVisible(false);
        diskbrowsePanel= new FileTreeFrame(selectedDrive);
        diskbrowsePanel.setBounds(10, 30, 250, frheight-60);
        frame.add(diskbrowsePanel);
        usbbrowsePanel.setVisible(false);
        usbbrowsePanel = new FileTreeFrame(ed);
        usbbrowsePanel.setBounds(764, 30, 250, frheight-60);
        frame.add(usbbrowsePanel);
        frame.setVisible(true);
        }

if(e.getSource() == u2d)
```

```
{
        try{
        Encryption decrypt = new Encryption(usbbrowsePanel.getFilePath());
        decrypt.dec();}
        catch(NullPointerException ne)
        {
                JoptionPane.showMessageDialog(null,"Please Select a file to
                Decrypt!!","Warning", JoptionPane.WARNING_MESSAGE);
        }

        }
}

public void closeWindow()
{
frame.dispose();
}

public void listDrives()
{
        int drives=0;
        File[] roots = File.listRoots();
          FileSystemView fsv = FileSystemView.getFileSystemView();

        try{
```

```
            for (int i=0;i<roots.length;i++)
{
    //Print out each drive/partition
            if(roots[i].canRead()&&(!fsv.isFloppyDrive(roots[i])))
            {
                    String str = roots[i].toString();
                    drivesList[drives]=str+"\\";
                    drives++;
            }
}
          int i=0;
 while(drivesList[i]!=null)
 {
            System.out.println(drivesList[i]);
            i++;
 }
          }
          catch(NullPointerException e)
          {
          }
          }}
```

8.4 Code For File Tree Frame

```
package sg;
import java.awt.Dimension;
import java.awt.Toolkit;
import java.io.File;
import java.util.Iterator;
import java.util.Vector;
import javax.swing.Jframe;
import javax.swing.Jpanel;
import javax.swing.JscrollPane;
import javax.swing.JsplitPane;
import javax.swing.JtextArea;
import javax.swing.Jtree;
import javax.swing.ScrollPaneConstants;
import javax.swing.event.TreeModelEvent;
import javax.swing.event.TreeModelListener;
import javax.swing.event.TreeSelectionEvent;
import javax.swing.event.TreeSelectionListener;
import javax.swing.tree.TreeModel;
import javax.swing.tree.TreePath;
 public class FileTreeFrame extends Jpanel {
  private Jpanel panel;
  private Jtree fileTree;
  private FileSystemModel fileSystemModel;
  File file;
```

```
  public FileTreeFrame(String directory) {
          panel = new Jpanel();
   fileSystemModel = new FileSystemModel(new File(directory));
   fileTree = new Jtree(fileSystemModel);
   fileTree.addTreeSelectionListener(new TreeSelectionListener() {
     public void valueChanged(TreeSelectionEvent event) {
       file = (File) fileTree.getLastSelectedPathComponent();
     }
   });
   Toolkit tk = Toolkit.getDefaultToolkit();
        Dimension dim = tk.getScreenSize();
   panel.add(fileTree);
   panel.setSize(250,dim.height);
   panel.setVisible(true);
   this.add(panel);
   this.setSize(250,dim.height);
   JscrollPane scroll = new JscrollPane (panel);
   scroll.setViewportView(panel);
   scroll.setVerticalScrollBarPolicy (
ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED );

scroll.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_N
EEDED);
   this.add(scroll);
   this.setVisible(true);
  } public String getFilePath()
```

```
  {
        String str = file.getPath();
        return str;
  }
}
 class FileSystemModel implements TreeModel {
  private File root;
  private Vector listeners = new Vector();
 public FileSystemModel(File rootDirectory) {
    root = rootDirectory;
  }
  public Object getRoot() {
    return root;
  }
 public Object getChild(Object parent, int index) {
    File directory = (File) parent;
    String[] children = directory.list();
    return new TreeFile(directory, children[index]);
  }
  public int getChildCount(Object parent) {
    File file = (File) parent;
    if (file.isDirectory()) {
      String[] fileList = file.list();
      if (fileList != null)
        return file.list().length;
```

```
    }
    return 0;
  }
 public 65oolean isLeaf(Object node) {
    File file = (File) node;
    return file.isFile();
  }
  public int getIndexOfChild(Object parent, Object child) {
    File directory = (File) parent;
    File file = (File) child;
    String[] children = directory.list();
    for (int I = 0; I < children.length; i++) {
      if (file.getName().equals(children[i])) {
        return I;
      }
    }
    return -1;
  }
 public void valueForPathChanged(TreePath path, Object value) {
    File oldFile = (File) path.getLastPathComponent();
    String fileParentPath = oldFile.getParent();
    String newFileName = (String) value;
    File targetFile = new File(fileParentPath, newFileName);
    oldFile.renameTo(targetFile);
    File parent = new File(fileParentPath);
```

```
    int[] changedChildrenIndices = { getIndexOfChild(parent, targetFile) };
    Object[] changedChildren = { targetFile };
    fireTreeNodesChanged(path.getParentPath(), changedChildrenIndices, changedChildren);
  }
  private void fireTreeNodesChanged(TreePath parentPath, int[] indices, Object[] children) {
    TreeModelEvent event = new TreeModelEvent(this, parentPath, indices, children);
    Iterator iterator = listeners.iterator();
    TreeModelListener listener = null;
    while (iterator.hasNext()) {
      listener = (TreeModelListener) iterator.next();
      listener.treeNodesChanged(event);
    }
  }
  public void addTreeModelListener(TreeModelListener listener) {
    listeners.add(listener);
  }
  public void removeTreeModelListener(TreeModelListener listener) {
    listeners.remove(listener);
  }
  private class TreeFile extends File {
    public TreeFile(File parent, String child) {
      super(parent, child);
    }
  public String toString() {
      return getName(); }}}
```

Fig. 17

User Interface

Fig. 18

User interface Showing Available Drives

8.5 Code For Drive Detection

```
package sg;

import java.io.*;

public class FindDrive

{

        static authentication au;

public static void main(String[] args)

{

String[] letters = new String[]{ "A", "B", "C", "D", "E", "F", "G", "H",
"I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z"};

File[] drives = new File[letters.length];

boolean[] isDrive = new boolean[letters.length];

for ( int i = 0; i < letters.length; ++i )

{

drives[i] = new File(letters[i]+":/");

isDrive[i] = drives[i].canRead();

}

System.out.println("FindDrive: waiting for devices...");

while(true)

{

for ( int i = 0; i < letters.length; ++i )

{

boolean pluggedIn = drives[i].canRead();

if ( pluggedIn != isDrive[i] )

        {
```

```
if ( pluggedIn )
        {
        System.out.println("Drive "+letters[i]+" has been plugged in");
        au = new authentication(letters[i]+":\\");
        }
else
        {
        try{
        au.closeWindow();
        }
        catch(java.lang.NullPointerException e)
        {}
        System.out.println("Drive "+letters[i]+" has been unplugged");
        }
isDrive[i] = pluggedIn;
}
}
try { Thread.sleep(100); }
catch (InterruptedException e) {  }
}}}
```

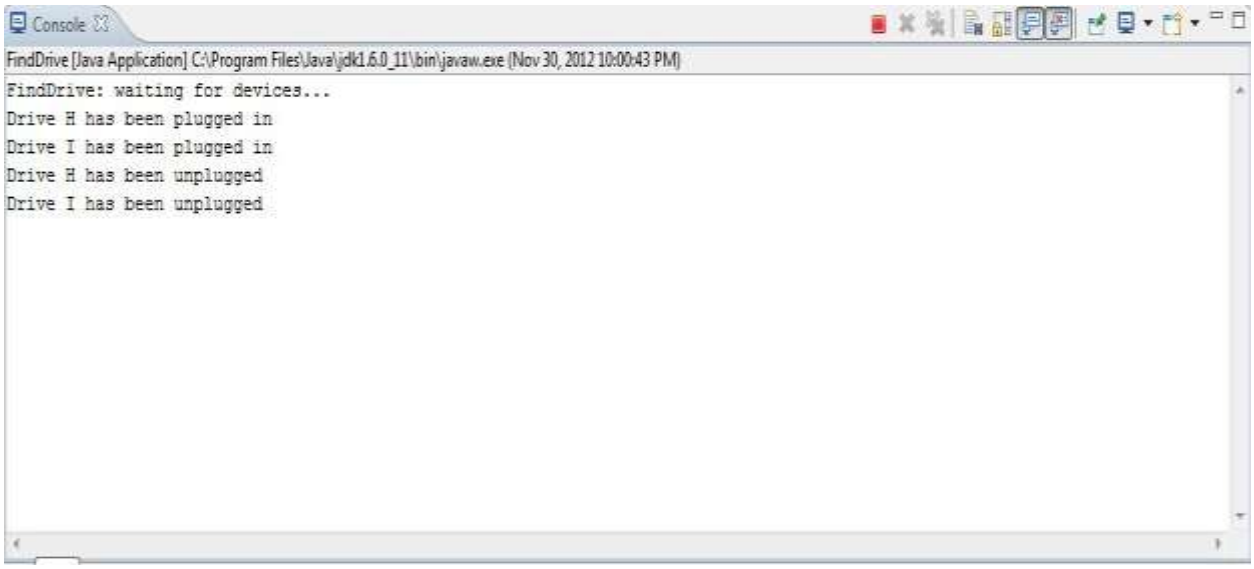

Fig. 19

Drive Detection

8.6 Code For Encryption- Decryption Using AES:

```
package sg;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.File;

import java.io.InputStream;

import java.io.OutputStream;

import java.security.SecureRandom;

import javax.crypto.Cipher;

import javax.crypto.CipherInputStream;

import javax.crypto.CipherOutputStream;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.SecretKeySpec;

import javax.swing.JOptionPane;


public class Encryption {

        private static final int IV_LENGTH=16;

        static String fileNm;

        String fileName;

        String tempFileName;

        String resultFileName;

        boolean check=true ;
```

```
static boolean en=false,dn = false;
 public Encryption(String fn)
{
        fileName=fn;
        tempFileName=EncNameAlt(fileName)+".ENC";
        resultFileName=DecNameAlt(fileName)+".DEC";
        System.out.println("temp file name:"+tempFileName);
        System.out.println("result file name:"+resultFileName);
        try{
             File file = new File(fileName);
            if(!file.exists()){
                System.out.println("No file "+fileName);
                check=false;
            }
            if(check)
            {
            File file2 = new File(tempFileName);
            File file3 = new File(resultFileName);
            if(file2.exists())
                en = true;
            if( file3.exists()){
                System.out.println("File for encrypted temp file or for the result decrypted
                file already exists. Please remove it or use a different file name");
                dn = true;
            }
```

```
                }
        }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
        public static void encrypt(InputStream in, OutputStream out, String password) throws
Exception{
if(!en)
{
                SecureRandom r = new SecureRandom();
                byte[] iv = new byte[IV_LENGTH];
                r.nextBytes(iv);
                out.write(iv); //write IV as a prefix
                out.flush();
                //System.out.println(">>>>>>>>>written"+Arrays.toString(iv));
                Cipher cipher = Cipher.getInstance("AES/CFB8/NoPadding");
        //"DES/ECB/PKCS5Padding";"AES/CBC/PKCS5Padding"
                SecretKeySpec keySpec = new SecretKeySpec(password.getBytes(), "AES");
                IvParameterSpec ivSpec = new IvParameterSpec(iv);
                cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSpec);
                out = new CipherOutputStream(out, cipher);
                byte[] buf = new byte[1024];
                int numRead = 0;
                while ((numRead = in.read(buf)) >= 0) {
```

```
                out.write(buf, 0, numRead);

            }

            out.close();

            JOptionPane.showMessageDialog(null,"Encryption Sucessful","Information",
JOptionPane.INFORMATION_MESSAGE);

        }

        }

public static void decrypt(InputStream in, OutputStream out, String password) throws
Exception{

if(!dn)

{

            byte[] iv = new byte[IV_LENGTH];

            in.read(iv);

            //System.out.println(">>>>>>>>>red"+Arrays.toString(iv));

            Cipher cipher = Cipher.getInstance("AES/CFB8/NoPadding");
        //"DES/ECB/PKCS5Padding";"AES/CBC/PKCS5Padding"

            SecretKeySpec keySpec = new SecretKeySpec(password.getBytes(), "AES");

            IvParameterSpec ivSpec = new IvParameterSpec(iv);

            cipher.init(Cipher.DECRYPT_MODE, keySpec, ivSpec);

            in = new CipherInputStream(in, cipher);

            byte[] buf = new byte[1024];

            int numRead = 0;

            while ((numRead = in.read(buf)) >= 0) {

                out.write(buf, 0, numRead);

            }

            out.close();
```

```
            JOptionPane.showMessageDialog(null,"Decryption Sucessful","Information",
JOptionPane.INFORMATION_MESSAGE);

        }

        }

        public static void copy(int mode, String inputFile, String outputFile, String password)
throws Exception {

                BufferedInputStream is = new BufferedInputStream(new
                FileInputStream(inputFile));

                BufferedOutputStream os = new BufferedOutputStream(new
                FileOutputStream(outputFile));

                if(mode==Cipher.ENCRYPT_MODE){

                        encrypt(is, os, password);

                }

                else if(mode==Cipher.DECRYPT_MODE){

                        decrypt(is, os, password);

                }

                else throw new Exception("unknown mode");

                is.close();

                os.close();

        }


public void enc()

{

        try {

                copy(Cipher.ENCRYPT_MODE, fileName, tempFileName,
"password12345678");

        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        }
public void dec()
{
        try {
            copy(Cipher.DECRYPT_MODE, fileName, resultFileName,
"password12345678");
        } catch (FileNotFoundException e) {

                JOptionPane.showMessageDialog(null,"Decryption
Sucessful","Information", JOptionPane.INFORMATION_MESSAGE);

        }
        catch(Exception e)
        {}
        }
        public String EncNameAlt(String file)
        {
            return UserInterface.ed + (String) file.subSequence(2, file.length());
        }
        public String DecNameAlt(String file)
        {
            return UserInterface.selectedDrive + (String) file.subSequence(2, file.length());
        }
}
```

Fig. 20

Selection Of File For Encryption

Fig. 21

Encryption Successful

Fig. 22

Encrypted File Created

Fig. 23

Selection Of File For Decryption

Fig. 24

Decryption Successful

Fig. 25

Decrypted file Created

9. System Testing

Case 1: File Transfer to USB

Description: Involves authentication, encryption and transfer of file to USB

Precondition: User is registered with default username and password.

Inputs: Existing username, password, file to be transferred

Post-condition: File is successfully encrypted and transferred to pendrive

User: Final end user

Sno	Steps to be executed	User's expected result	Developer's expected result	Actual Result	Pass/Fail	Defect	Remarks
1.	Pen drive inserted	User Interface starts automatically	User Interface starts automatically	User Interface starts automatically	Pass	-	-
2.	No file selected and 'Send to USB' button clicked	Error prompt- "Please select a file/folder first"	Error prompt- "Please select a file/folder first"	Error prompt- "Please select a file/folder first"	Pass	-	-
3.	File selected and 'Send to USB' button clicked	Authentication dialog appears	Authentication dialog appears	Authentication dialog appears	Pass	-	-
4.	Single field entered in authentication dialog	Error prompt- "Access denied"	Error prompt- "Access denied"	Error prompt- "Access denied"	Pass	-	-
5.	Wong value entered in any field in authentication dialog	Error prompt- "Access denied"	Error prompt- "Access denied"	Error prompt- "Access denied"	Pass	-	-
6.	Valid entries in authentication dialog	File is successfully encrypted and sent to pendrive	File is successfully encrypted and sent to pendrive	File is successfully encrypted and sent to pendrive	Pass	-	-

Table No. 2: Testing File Transfer to USB

Case 2: File Transfer to Disk

Description: Involves authentication, decryption and transfer of file to disk

Precondition: User is registered with default username and password.

Inputs: Existing username, password, file to be transferred

Post-condition: File is successfully decrypted and transferred to disk

User: Final end user

Sno	Steps to be executed	User's expected result	Developer's expected result	Actual Result	Pass/Fail	Defect	Remarks
1.	Pen drive inserted	User Interface starts automatically	User Interface starts automatically	User Interface starts automatically	Pass	-	-
2.	No file selected and 'Send to Disk' button clicked	Error prompt- "Please select a file/folder first"	Error prompt- "Please select a file/folder first"	Error prompt- "Please select a file/folder first"	Pass	-	-
3.	File selected and 'Send to Disk' button clicked	Authentication dialog appears	Authentication dialog appears	Authentication dialog appears	Pass	-	-
4.	Single field entered in authentication dialog	Error prompt- "Access denied"	Error prompt- "Access denied"	Error prompt- "Access denied"	Pass	-	-
5.	Wong value entered in any field in authentication dialog	Error prompt- "Access denied"	Error prompt- "Access denied"	Error prompt- "Access denied"	Pass	-	-
6.	Valid entries in authentication dialog	File is successfully decrypted and sent to disk	File is successfully decrypted and sent to disk	File is successfully decrypted and sent to disk	Pass	-	-

Table No. 3: Testing File Transfer to Disk

10. Result & Analysis

The software restricts the movement of data outside an organization; monitors and controls data exchange between the computer and the USB. The software also allows the transfer of legitimate data and data types to and from USB drives but any unsolicited activity like data theft; unwanted data/malware injection/planting etc. is inhibited.

The software employs software based encryption technique. All data leaving via the USB port into a flash disk is encrypted with a specific key. While transferring data back to the company computer, only those files are copied that had been encrypted using this specific key. The software ensures that all transfers are made using the monitoring and encryption software.

11. Future Scope

- Customize for personal home use.
- Protection against malwares.
- Time bound or varying key.

12. References

IEEE Papers:

- *An Efficient Time-Bound Hierarchical Key Management Scheme for Secure Broadcasting:* Elisa Bertino, Ning Shang, and Samuel Wagstaff Jr.
- *Encrypted Key Exchange:* Steven M. Bellovin, Michael Merritt
- *Multi-level Key Encryption:* Lein Harn and Hung-Yu Lin
- W.G. Tzeng, *"A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy,"* IEEE Trans. Knowledge and Data Eng., Proc. Sixth ACM Symp. Access Control Models and Technologies (SACMAT '01), vol. 14, no. 1, pp. 182-188, Jan./Feb. 2002.
- X. Yi, *"Security of Chien's Efficient Time-Bound Hierarchical Key Assignment Scheme,"* IEEE Trans. Knowledge and Data Eng.,vol. 17, no. 9, pp. 1298-1299, Sept. 2005.
- X. Yi and Y. Ye, *"Security of Tzeng's Time-Bound Key Assignment Scheme for Access Control in a Hierarchy,"* IEEE Trans. Knowledge and Data Eng., vol. 15, no. 4, pp. 1054-1055, July/Aug. 2003.

White Papers:

- *An Introduction to Cryptography and Digital Signatures*- Ian Curry, Entrust
- *Portable Panic- The evolution of USB Insecurity*, Lumension White Paper
- *Symmetric Key Management Systems,* Arshad Noor

Summits & Conferences:

- *Current Legal Framework for Liability Associated with Data Loss,* Alan Paller, Ben Wright [2006]

Web Sites/Pages:

- *PGP Technology, AES, DES Algorithms, USB Encryption Mechanisms*
- Wikipedia pages: *Public Key Cryptography, Hybrid CryptoSystems*
- Other sites: *docs.google.com, ieee.org*

Multimedia Resources:

- *Powerpoint Presentation on USB Security*, Slideshare.net
- *USB Flash Drives- Protecting Data and Enhancing Storage,* Steffen Hellmold
- *Video on Cold Boot Attacks on Encryption Keys,* Princeton University