International Institute of Professional Studies

Software Design Document

[Version: 1]

for

Cryptic

http://cryptic.sourceforge.net/

Date: March 18, 2010

Prepared by:

Amber Jain
IC- 2K7 - 05
MCA (Semester-6) Section-A
IIPS, DAVV
ithinkminus@gmail

Ankit Vinayaka
IC- 2K7- 09
MCA (Semester-6) Section-A
IIPS, DAVV
vinayakaankit@gmail

Submitted to:

Mr. Shaligram Prajapat
Project Guide (Cryptic)
Faculty, SAD
IIPS, DAVV

Document Release notes:

- Project:
  Cryptic (http://cryptic.sourceforge.net/)
- Document version:
  1
- Description:
  Initial version (New SDD created)
- The latest version of this SDD is available online from project's website and is linked from: http://cryptic.sourceforge.net/documentation.html

Documentation Version Control:

Version	Date	Description	Page and item number	Previous page number	Reason for change
1	18/3/2010	Initial version	-	-	-

Document Conventions:

- *References are mentioned throughout the document in square bracket (e.g. [1])*
- *'Heading 1' is used for top level heading of main sections.*
- *'Heading' is used for sub-section heading.*
- URL are metalinked from the document. Ctrl-Click them in 'OpenOffice.org Writer' to open hyperlink.

Document outline:

1. Introduction

1.1 Background

Cryptic which is a cryptographic applications suite for encryption/decryption of data, calculating cryptographic hashes or message digests and securing user passwords using a master password in a password safe.

1.2 Purpose

This document provides all of the requirements for the Cryptic. Sections 1 and 2 are intended primarily for customers of the application, but will also be of interest to software engineers building or maintaining the software. Section 3 is intended primarily for software engineers, but will also be of interest to customers.

1.3 Scope

This document covers the requirements for release 0.1 of Cryptic. Mention will be made throughout this document of selected probable features of future releases. The purpose of this is to guide developers in selecting a design that will be able to accommodate the full-scale application.

1.4 Audience

This document is intended primarily for software engineers building or maintaining the software, but will also be of interest to users of application.

1.5 References

1. K&R style indentation: http://en.wikipedia.org/wiki/Indent_style#K.26R_style
2. Cryptic website: http://cryptic.sourceforge.net/
3. SRS of Cryptic: http://cryptic.sourceforge.net/files/srs.pdf
4. OpenSSL: http://www.openssl.org/
5. SQLite: http://www.sqlite.org/
6. Qt: http://qt.nokia.com/products
7. KDE HIG: http://techbase.kde.org/Projects/Usability/HIG
8. GNOME HIG: http://library.gnome.org/devel/hig-book/stable/
9. Unix philosophy: http://en.wikipedia.org/wiki/Unix_philosophy
10. ISC License: http://en.wikipedia.org/wiki/ISC_license
11. Cryptic documentation: http://cryptic.sourceforge.net/documentation.html
12. SQLite Is Self-Contained: http://www.sqlite.org/selfcontained.html
13. SQLite Is Serverless: http://www.sqlite.org/serverless.htmls
14. SQLite Is A Zero-Configuration Database: http://www.sqlite.org/zeroconf.html

1.6 Acronyms and abbreviations

- ANSI: American National Standards Institute
- BSD: Berkeley Software Distribution
- CUI: Character User Interface
- DB: Database
- FAQ: Frequently Asked Questions
- FIPS: Federal Information Processing Standards
- gcc: GNUC Compiler Collection
- GUI: Graphical User Interface
- GPL: General Public License
- HIG: Human Interface Guidelines
- I/O: Input/Output
- ISC: Internet Systems Consortium
- K&R: Kernighan and Ritchie
- KDE: K Dekstop Environment
- MB: Mega Byte
- OS: Operating System
- OSI: Open Source Initiative
- RAM: Random Access Memory
- SDD: System Design Document
- SDK: Software Development Kit
- SQL: Structured Query Language
- SRS: Software Requirements Specification
- SSL: Secure Sockets Layer
- UI: User Interface
- URL: Uniform Resource Locator

1.7 Overview

This document specifies SDD for Cryptic which is a cryptographic applications suite for encryption/decryption of data, calculating cryptographic hashes or message digests and securing user passwords using a master password in a password safe. The purpose for developing this type of application suite is to facilitate the users who want to secure their data that is expected either to be stored locally or to be transmitted over network.

2. System Description:

2.1 Objectives

The objective of design document is to give detail understand of the system which can be understand by managers and programmer.

2.2 System Architecture:

The system is a local application. The system components are listed below in subsystem architecture:

2.2.1 Subsystem architecture:

The system consists of following subsystems:

- Encryption:
  At the very basic level, the system will input unencrypted file name/location, algorithm, key. The system will output the encrypted file.

- Decryption:
  At the very basic level, the system will input encrypted file name/location, algorithm that was used to encrypt file, key. The system will output the decrypted file.

- Hashing:
  At the very basic level, the system will input a file and hash algorithm to be used. The system will then output the hash/message digest of the input file.

- Password safe:
  At the very basic level, the system will input list of user's account ID and passwords, a master password to lock password Database, algorithm to encrypt the password safe. This will call "Encryption" or decryption subsystem which in turn will output locked (encrypted) database or decrypted database.

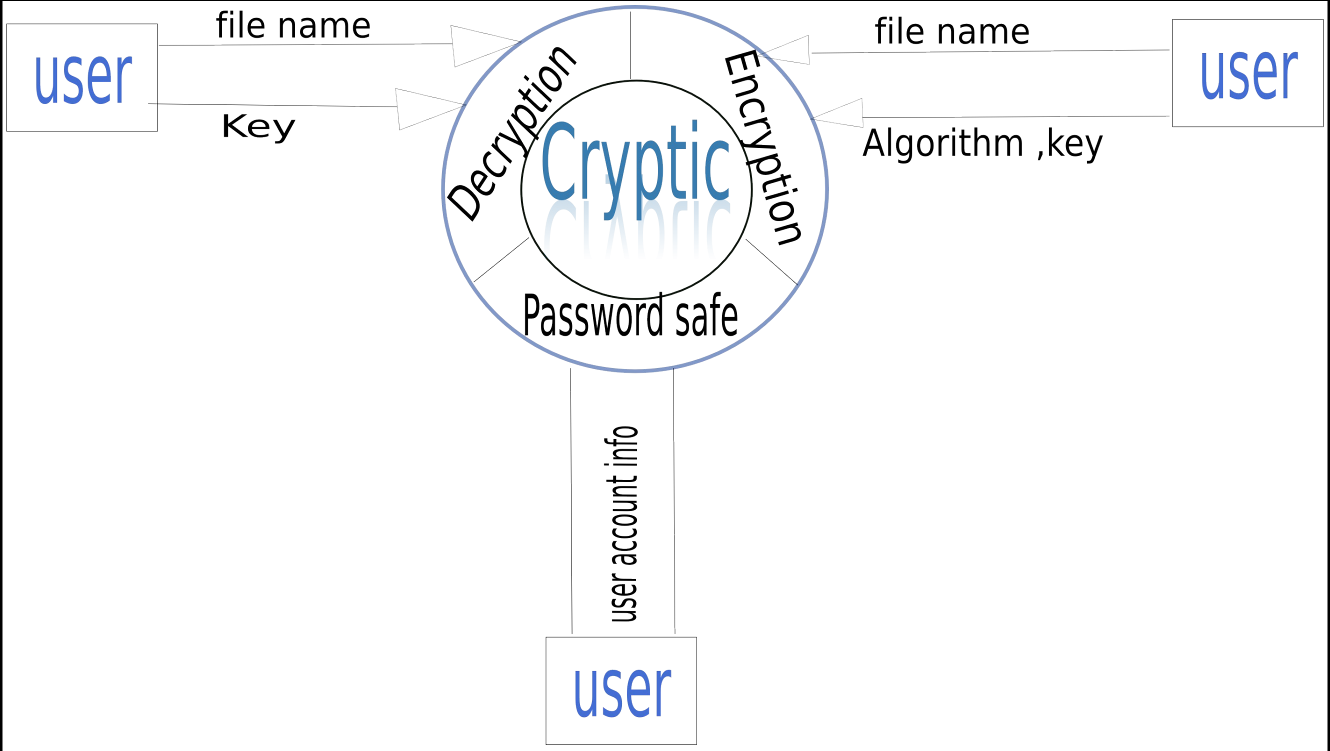

Figure: Zero Level Cryptic DFD Part -1

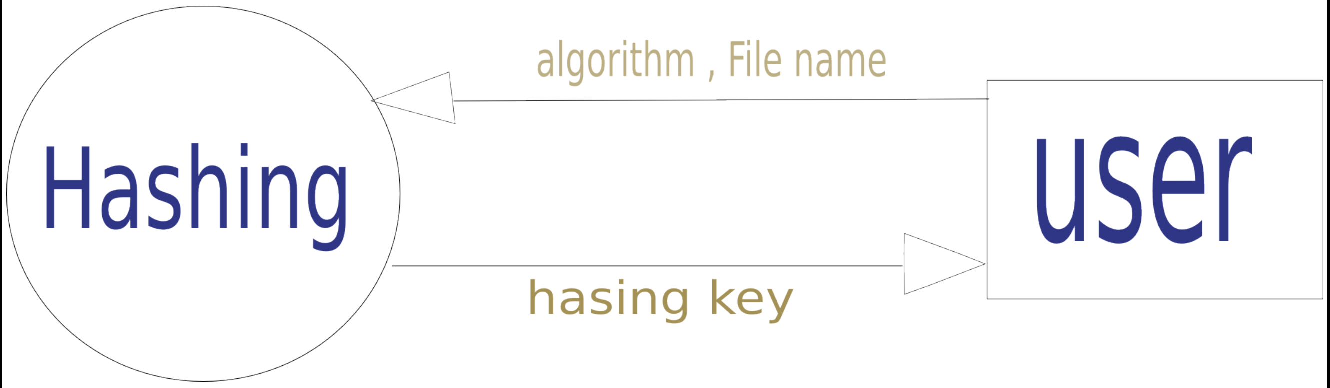

Figure: Zero Level Cryptic DFD Part – 2

2.3 Architectural strategies:

Here are the design decision and strategy that will effect the overall organisation of the system and its higher level structures:

- We will use C for core cryptographic code, OpenSSL[4] library for cryptographic algorithms, Qt for program GUI and SQLite[5] for password database functionality.

- We chose C programming language for our core code because C is a general purpose programming language that fits nicely into our requirement for encryption/decryption program. Moreover , OpenSSL[4] library that we plan to use is written in C.

- We chose Qt[6] for our program GUI because it is cross platform that make our program portable.

- We chose SQLite[5] for password database because SQLite[5] is a C cross platform software library that implements a self-contained, serverless, zero-configuration,transactional SQL database engine that fits nicely into our need for a loacalized password DB.

- We chose OpenSSL[4] because the core library (written in the C programming language) implements the basic cryptographic functions and provides various utility functions. Moreover, OpenSSL is one of the few open source programs to be validated under the FIPS 140-2 computer security standard.

- Future plans to extend Cryptic include implementing many algorithms available under OpenSSL[4]. Moreover, there are also plans to include functions for compressing, splitting, synchronising and archiving files to make this application a 'data security and management application' rather than just 'data security application'.

- **User interface paradigms:** Special effort is put into this project to create an intuitive, simple and streamlined graphical interface for non-technical users. The project adapts many tips from GNOME Human Interface Guidelines (HIG) [8] and KDE HIG [7] . All cryptographic functions, hashing and password safe functionalities are accessible from GUI.

  The Graphical UI (GUI) will include support for keyboard shortcuts for quick access to program options.

  The Graphical UI will also include all standard menus/buttons/options found in other programs such as menu bar (File, Edit, Help etc.), status bar etc.

  For advanced users, CUI design for Cryptic should conform with Unix philosophy[9] to support reuse of Cryptic code and extending functionality:
  http://en.wikipedia.org/wiki/Unix_philosophy#McIlroy:_A_Quarter_Century_of_Unix
  http://en.wikipedia.org/wiki/Unix_philosophy#Eric_Raymond

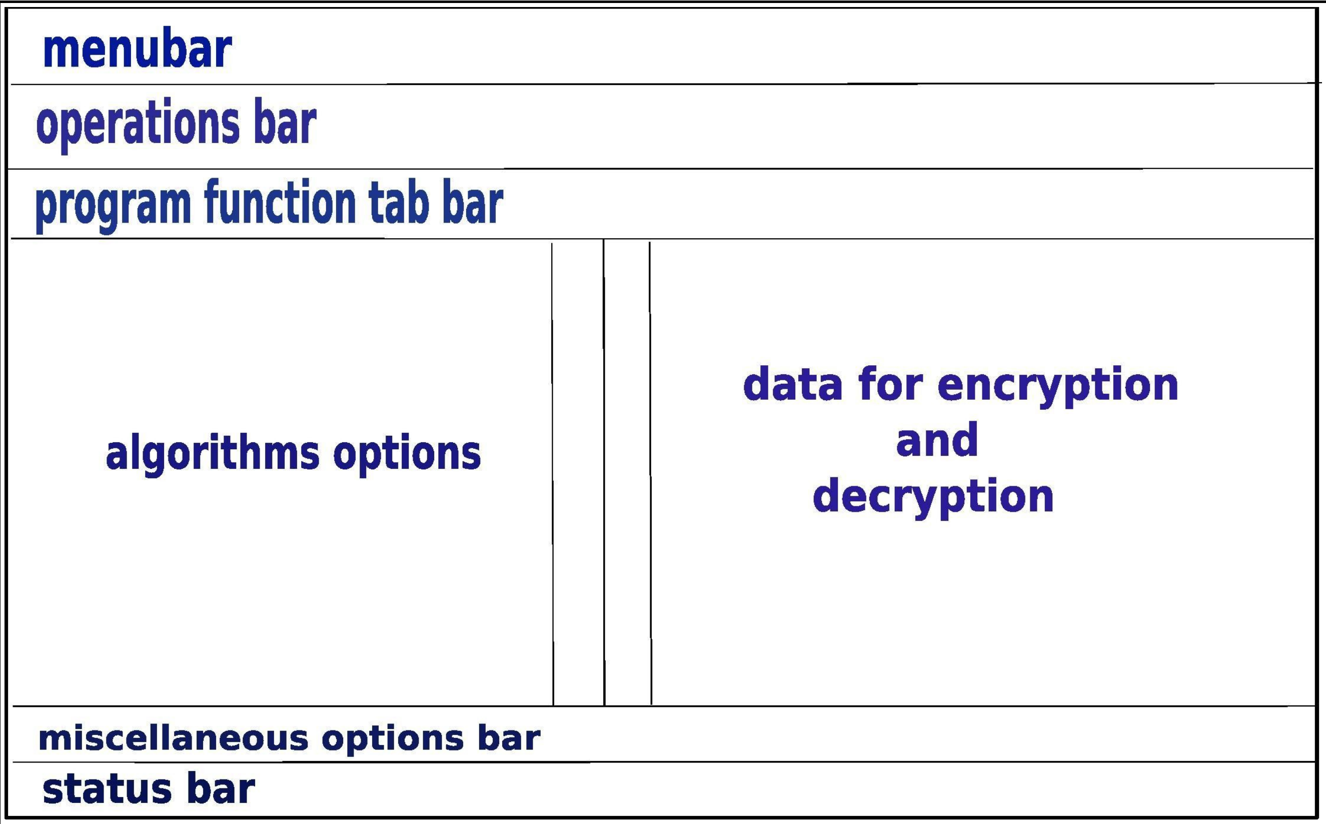

Figure: Graphical User Interface mockup

2.4 Hardware Environment:

- Cryptic is intended to be platform independent. Therefore no specific hardware is excluded. But it will atleast work on x86 systems without any additional porting efforts. Moreover, no special hardware is needed for software operation.
- **Minimum hardware configuration:** Cryptic is expected to use no more than 16 MB of RAM and 20 MB of external storage.

2.5 Software Environment:

- We require C compiler gcc ,Qt SDK [6], SQLite(DBMS) [5] and OpenSSL [4].
- The program releases will be OS portable. Ideally, they should run on all major OSes including Unixes, BSDs, Linuxes, Windows family etc.

2.6 Network Environment:

The system will use:

- Network protocols for program update information.
- System I/O protocols for local file access.

2.7 Design constraints:

- SQLite [5] DB is used for password safe function because it is self-contained [12] and serverless [13] and also because it is zero-configuration [14] SQL DB engine. That matches our requirement of small, quick, single file based, zero configuration password DB.

- Public key or asymmetric cryptographic algorithms usually take a lot of computation time and memory as compared to symmetric ciphers. In most cases, symmetric ciphers are recommended for data encryption (and decryption) unless the user is willing to spend very large amount of computational effort to encryption/decryption using public key cipher. Public key ciphers are recommended to be used only in situations such as authentication or to confidentially distribute symmetric keys.

- Due to large scale use of ANSI C89 standard in professional world, we will be following C89 rather than C99.

- **Performance requirements:** Most public key ciphers rely on high computational cost operations. Therefore, keeping performance considerations in mind, in most cases symmetric key ciphers are recommended for data encryption/decryption (unless the user is willing to spend very large amount of computational effort to encryption/decryption using public key cipher). Public key ciphers are recommended to be used only in situations such as authentication or to confidentially distribute symmetric keys.

- **Security requirements:** This software package uses strong cryptography, so even if it is created, maintained and distributed from liberal countries (where it is legal to do this), it falls under certain export/import and/or use restrictions in some other parts of the world.

  Please remember that export/import and/or use of strong cryptography software, providing cryptography hooks or even just communicating technical details about cryptography software is illegal in some parts of the world. So, when you import this package to your country, re-distribute it from there or even just email technical suggestions or even source patches to the author or other people you are strongly advised to pay close attention to any export/import and/or use laws which apply to you. The authors of Cryptic (or any other code/libraries that Cryptic uses) are not liable for any violations you make here. So be careful, it is your responsibility.

- **Safety requirements:** This application suite is provided under ISC license [10]. So, there are chances of possible loss/damage to data, or harm that could result from the use of the this suite.

  User is required to remember his password that he/she used to encrypt data (or lock password safe) because most of secure cryptographic algorithms implemented in this suite are secure enough so that no algorithms better than brute-force can be used to recover lost password.

2.8 Goals and Guidelines:

Cryptic must conform with following guidelines:

- ***Adaptability and reusability***: *The suite should be simple (and intuitive) enough that it should not be difficult to adapt it to users needs. The code should be standards compliant and therefore should be easily reusable in other applications.*

- ***Availability:*** *The complete code for Cryptic is released under OSI approved, GPL* *compatible, free and open source ISC license [10] (http://en.wikipedia.org/wiki/ISC_license). So, the code is available free on project's website.*

- ***Portability:*** *The code should be ANSI C89 compliant and so, it should be easily portable to different architectures and operating systems.*
- ***Reliability****: Serious attempts should be made to make sure code is reliable and of enterprise quality.*
- ***Usability:*** When the development starts, the code should be usable as soon as possible

2.9 Policies and tactics:

- We'll be using gcc compiler for compiling C code.
- We'll be using SQLite database for password database functionality.
- We will use OpenSSL for cryptographic algorithms.
- We will follow K&R style indentation[1].
- The program suite will be thoroughly tested before any release because this program is concerned with user's data security.

3. Database Description:

3.1 Design Decision:

S. No.	Common Field Name	Data type and Length	Default value	Allow NULL	Primary key	Constraints/ Comments
1	ID	INTEGER	-	No	Yes	-
2	account_name	VARCHAR	CURRENT_TIME	No	-	-
3	username	VARCHAR	-	No	-	-
4	password	VARCHAR	-	No	-	-
5	comment	VARCHAR	CURRENT_DATE	Yes	-	-

3.2 Database Design

Schema: safe(ID, account_name, username, password, comment)

Entity-Relationship Diagram:

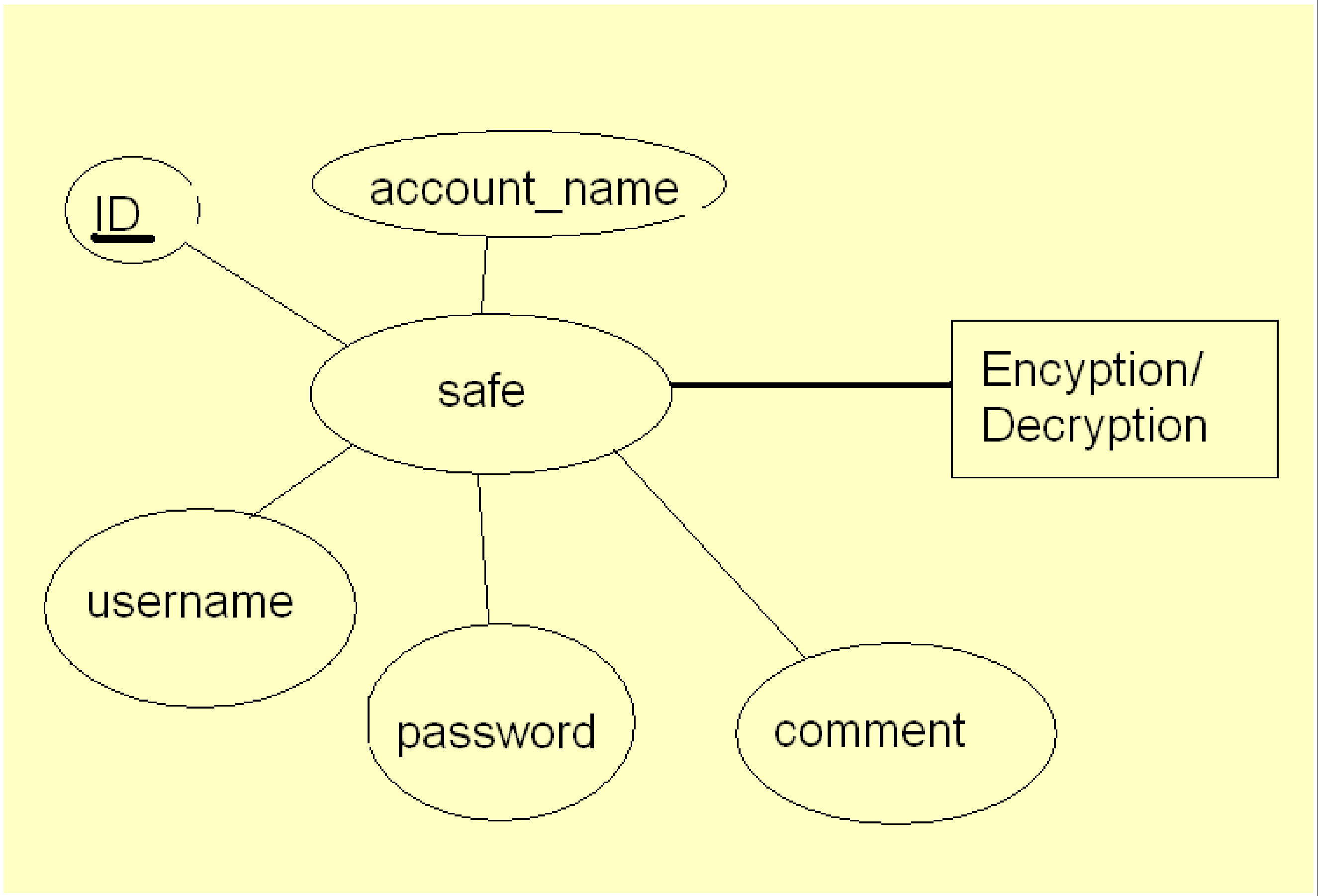

Figure: Entity- Relationship Diagram for Cryptic's Password Safe Feature

4. Data Migration Strategy:

- Data is entered to Cryptic in the form of input file (in Encryption/Decryption/Hashing), which is chosen by the user.

5. Input Output Specifications:

5.1 Input Description:

Cryptic accepts inputs in any format because core component of program deals with bit (and/or bytes, words etc.) rather than file formats.

5.2 Output Description:

Cryptic will output data in:

- .CRY extension files in Encryption/Decryption function.
- .sqlite.CRY extension files in Password safe functionality.
- .txt to store output hashes in hashing function.

5.3 Help facilities:

There will be a Help guide as well as FAQs/Quick Start guide available online from (program's "Help" menu) to the user. Other help resources are accessible from:

http://cryptic.sourceforge.net/documentation.html [11]

http://cryptic.sourceforge.net/help.html

5.4 Error messages:

The error messages will be shown in a separate error window. Optionally, a short error message description will be displayed in program status bar. There is also an "Error Console" option available from Cryptic 'Tools' menu that will show a list of all error message for this session.

5.5 Navigation:

See Annexure 'A' for a pictorial representation of menu navigation.

6. Detailed System Design:

Cryptic consists of 4 subsystems:

1. Encryption
2. Decryption
3. Password Safe
4. Hashing

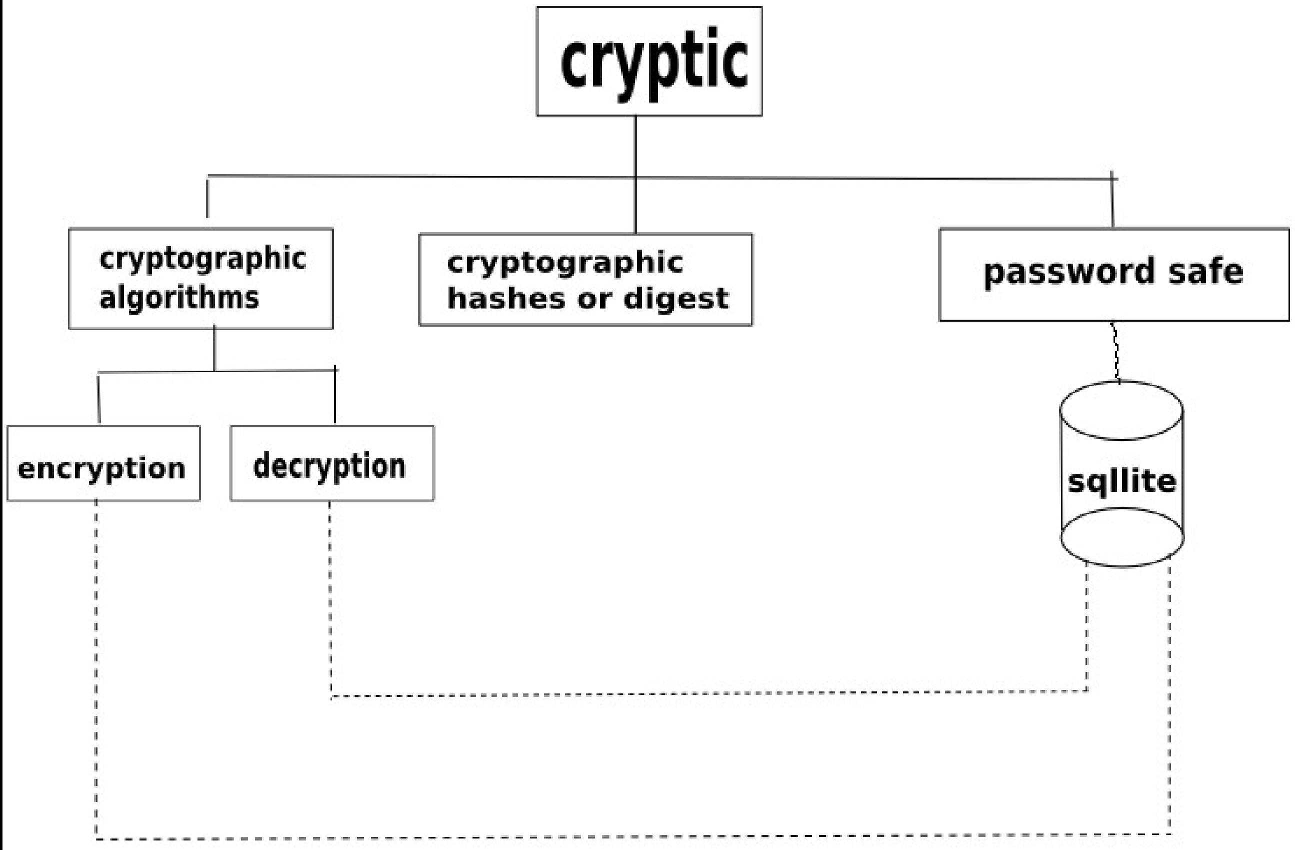

Figure: Program Subsystems Outline

6.1 Encryption:

SRS Process Reference: The SRS version-1 for Cryptic lists functional requirements of this subsystem. Please refer to Section- 3.2.1 of Cryptic SRS [version-1] for details about this software component. [3]

Classification: This component is classified as a separate module or subsytem.

Definition: In cryptography, encryption is the process of transforming information (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key.

6.2 Decryption:

SRS Process Reference: The SRS version-1 for Cryptic lists functional requirements of this subsystem. Please refer to Section- 3.2.2 of Cryptic SRS [version-1] for details about this software component. [3]

Classification: This component is classified as a separate module or subsytem.

Definition: Decryption is the process to make the encrypted information readable again (i.e. to make it unencrypted).

6.3 Hashing:

SRS Process Reference: The SRS version-1 for Cryptic lists functional requirements of this subsystem. Please refer to Section- 3.2.3 of Cryptic SRS [version-1] for details about this software component. [3]

Classification: This component is classified as a separate module or subsytem.

Definition: A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an accidental or intentional change to the data will change the hash value.

6.4 Password Safe:

SRS Process Reference: The SRS version-1 for Cryptic lists functional requirements of this subsystem. Please refer to Section- 3.2.4 of Cryptic SRS [version-1] for details about this software component. [3]

Classification: This component is classified as a separate module or subsystem.

Definition: A password safe stores user's login info of many accounts in a database and then encrypts it with a master password which can be later decrypted with the same master key.

Resources: The system components 'Encryption' and 'Decryption' are part of this component.

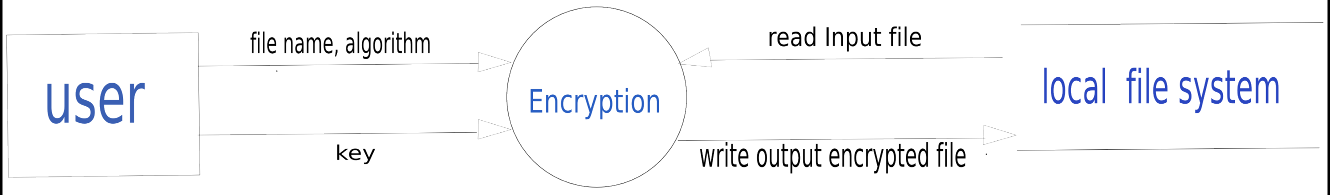

Figure: Level ONE DFD for Encryption

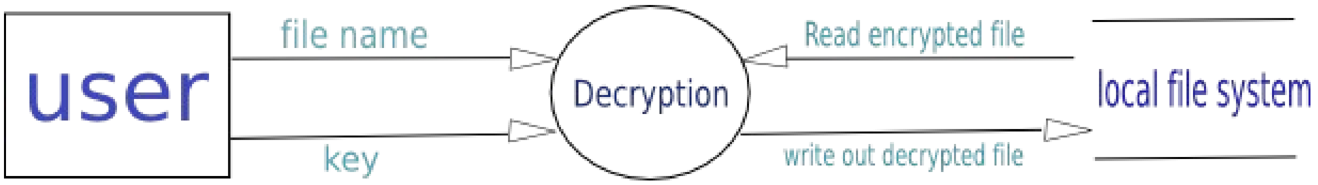

Figure: Level ONE DFD for Decryption

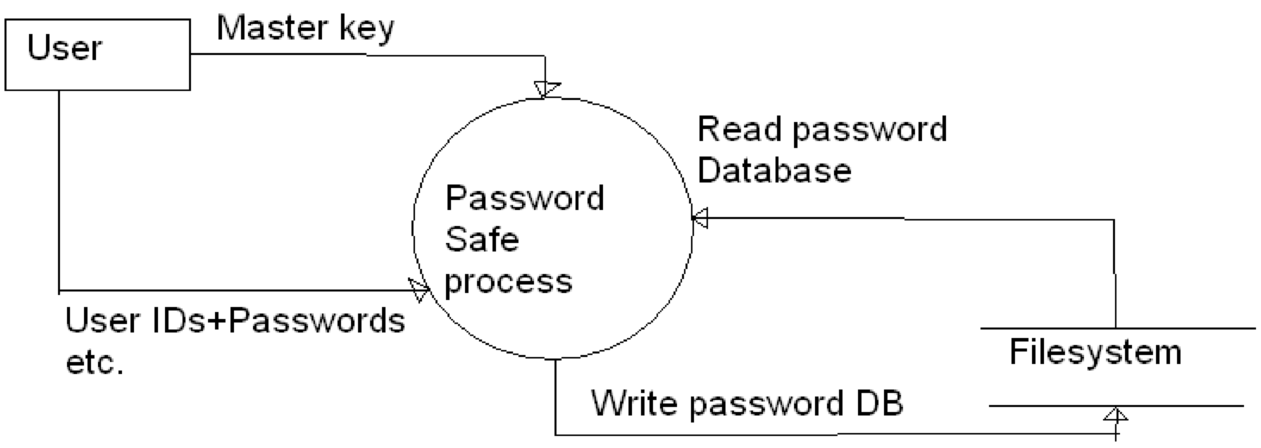

Figure: Level ONE DFD for Password safe process

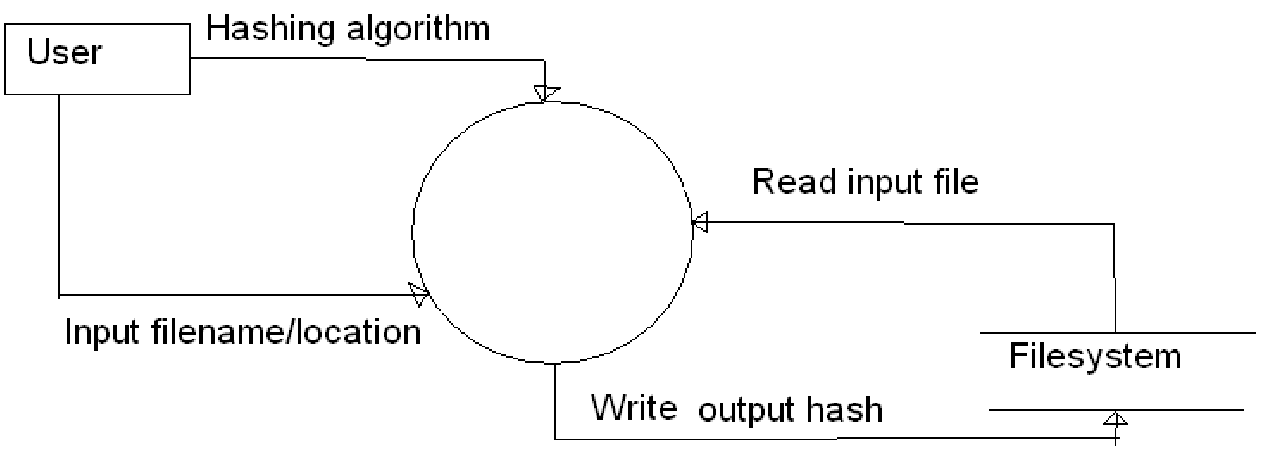

Figure: Level ONE DFD for Hashing

7. Operational Controls:

7.1 Startup, Shutdown and Restart:

There are no specific restart,startup,shutdown for normal operation of this program.

8. Security:

8.1 Security Strategy:

- **Operating system level security:** Any user with sufficient execute privileges in the directory that contains the program binary executable will be able to run this program normally.
- **Database Level Security:** No specific database security is needed and implemented for the SQLite database that we use in password safe, but operating system read/write permissions should be kept in mind when using Cryptic password safe DB.
- **Application Level Security:** No specific application security is implemented.
- **Menu Level Security:** No specific menu level security is implemented.

9. ANNEXURE- A: Menu Navigation

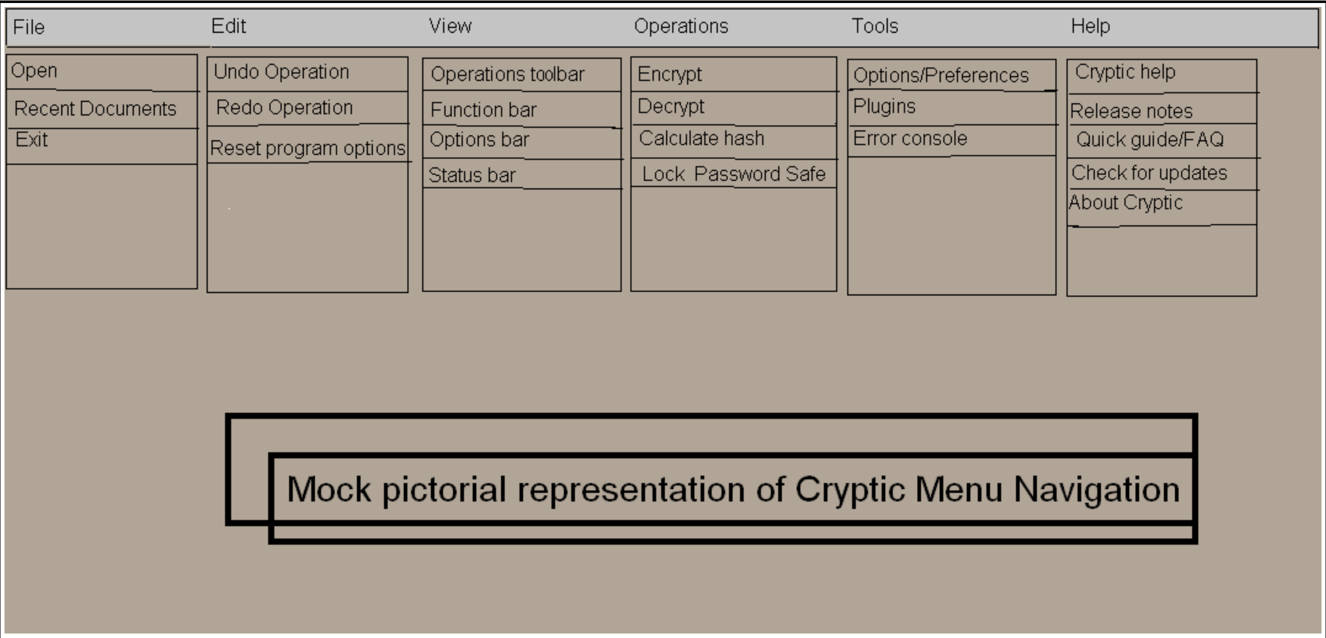

Figure: Mock pictorial representation of Cryptic Menu Navigation

10. ANNEXURE- B: Screen Layout

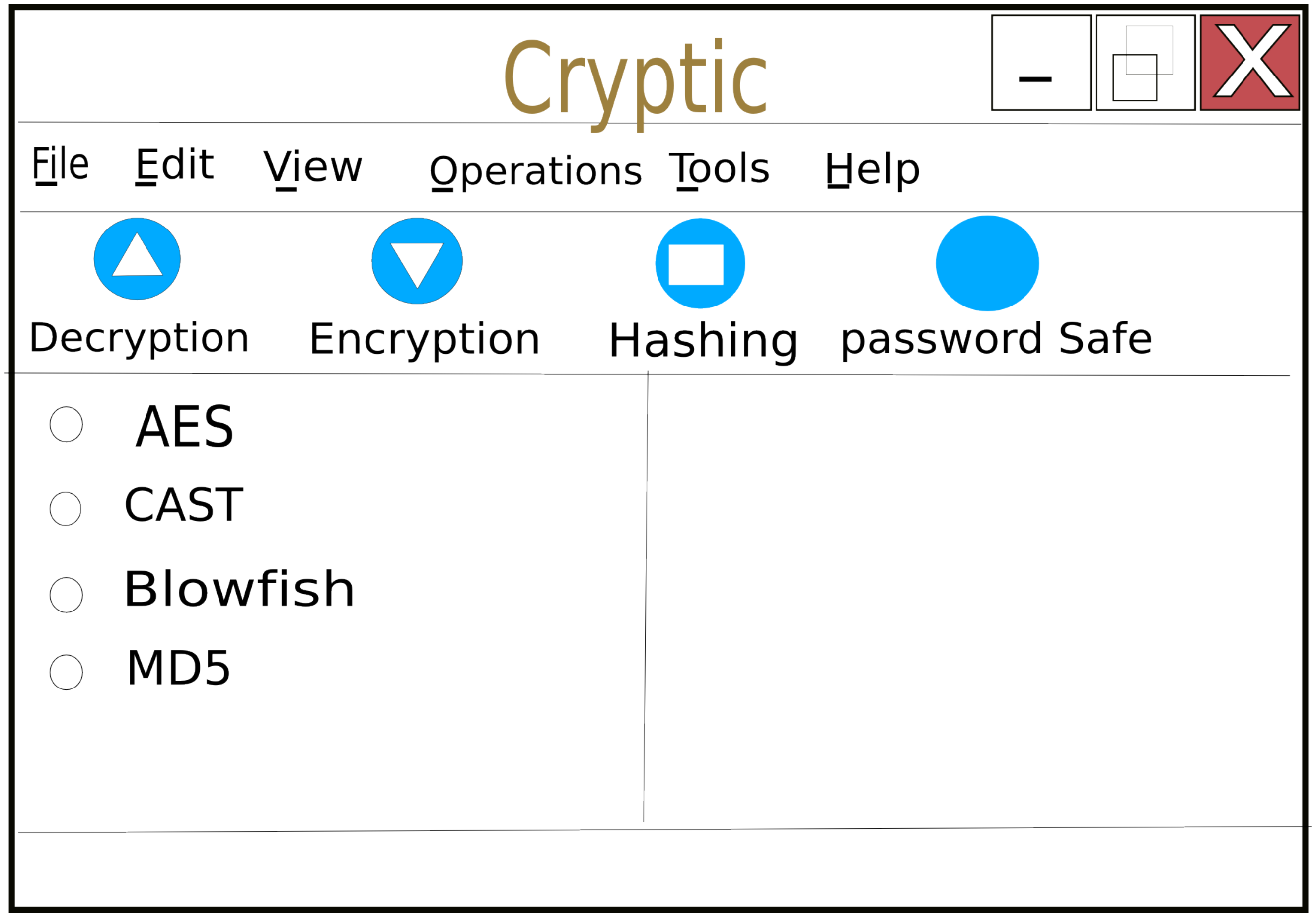

Figure: Program screen layout

11. Bibliography

1. Wikipedia articles
2. Applied Cryptography (2e) by Bruce Scheiner
3. http://dict.mizoram.gov.in/download/books/Software%20Design%20Document%20Guidelines.pdf
4. http://en.wikipedia.org/wiki/Software_design_document
5. http://blog.slickedit.com/2007/05/how-to-write-an-effective-design-document/
6. http://www.cmcrossroads.com/bradapp/docs/sdd.html
7. http://www.dlhoffman.com/classnotes/csci490-f06/designDocFormat.html
8. http://www.cs.purdue.edu/homes/apm/courses/cs490D-2001/design-doc-format.html
9. http://www.cse.iitk.ac.in/JaloteSEbook/CaseStudies/CaseStudy1/SDDesignDoc.pdf
10. An integrated approach to softwae engineering by Pankaj Jalote
11. SAD by E. M. Awad

** END OF SDD **