Software Requirements Specification Document

for

Cryptic

https://sourceforge.net/projects/cryptic/

Version- 1

Prepared by:

Amber Jain

IC-2k7-05

ithinkminus@gmail

Ankit Vinayaka

IC-2K7-09

vinayakaankit@gmail

Date: 02/10/2010

Table of Contents

Revision History

Name	Date	Description of Changes	Version
Amber Jain Ankit Vinayaka	02/10/10	New SRS document created (Initial version)	1

1. Introduction

1.1 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.

1.2 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 2' is used for sub-section heading.*
- *URL are metalinked from the document. Ctrl-Click them to open hyperlink.*

1.3 Product 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 References

1. *SQLite: http://www.sqlite.org/*
2. *OpenSSL: http://www.openssl.org/*
3. *ISC license: https://www.isc.org/software/license*
4. *KDE HIG: http://techbase.kde.org/Projects/Usability/HIG*
5. *GNOME HIG: http://library.gnome.org/devel/hig-book/stable/*
6. *Qt: http://qt.nokia.com/products*

7. *Cryptic official website: http://cryptic.sourceforge.net/*
8. *Cryptic on sourceforge: https://sourceforge.net/projects/cryptic/*
9. *Unix philosophy (Wikipedia): http://en.wikipedia.org/wiki/Unix_philosophy*
10. *SQLite Is Self-Contained: http://www.sqlite.org/selfcontained.html*
11. *SQLite Is A Zero-Configuration Database: http://www.sqlite.org/zeroconf.html*
12. *SQLite Is Serverless: http://www.sqlite.org/serverless.htmls*

1.5 Overview

This document specifies SRS 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. Overall Description

2.1 Product Perspective

2.1.1 User interfaces

There will be 2 interfaces:

- GUI for non-technical users
- CUI for users comfortable with typing commands

More details in section 3.1.

2.1.2 Hardware interfaces

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.

2.1.3 Software interfaces

Cryptic is intended to be operating system independent. Therefore no specific operating system is excluded.
More details in section 3.3 of this document.

2.1.4 Communication interfaces

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

2.1.5 Memory constraints

Cryptic is expected to use no more than 16 MB of RAM and 20 MB of external storage.

2.1.6 Operations

- Encrypt and decrypt data.

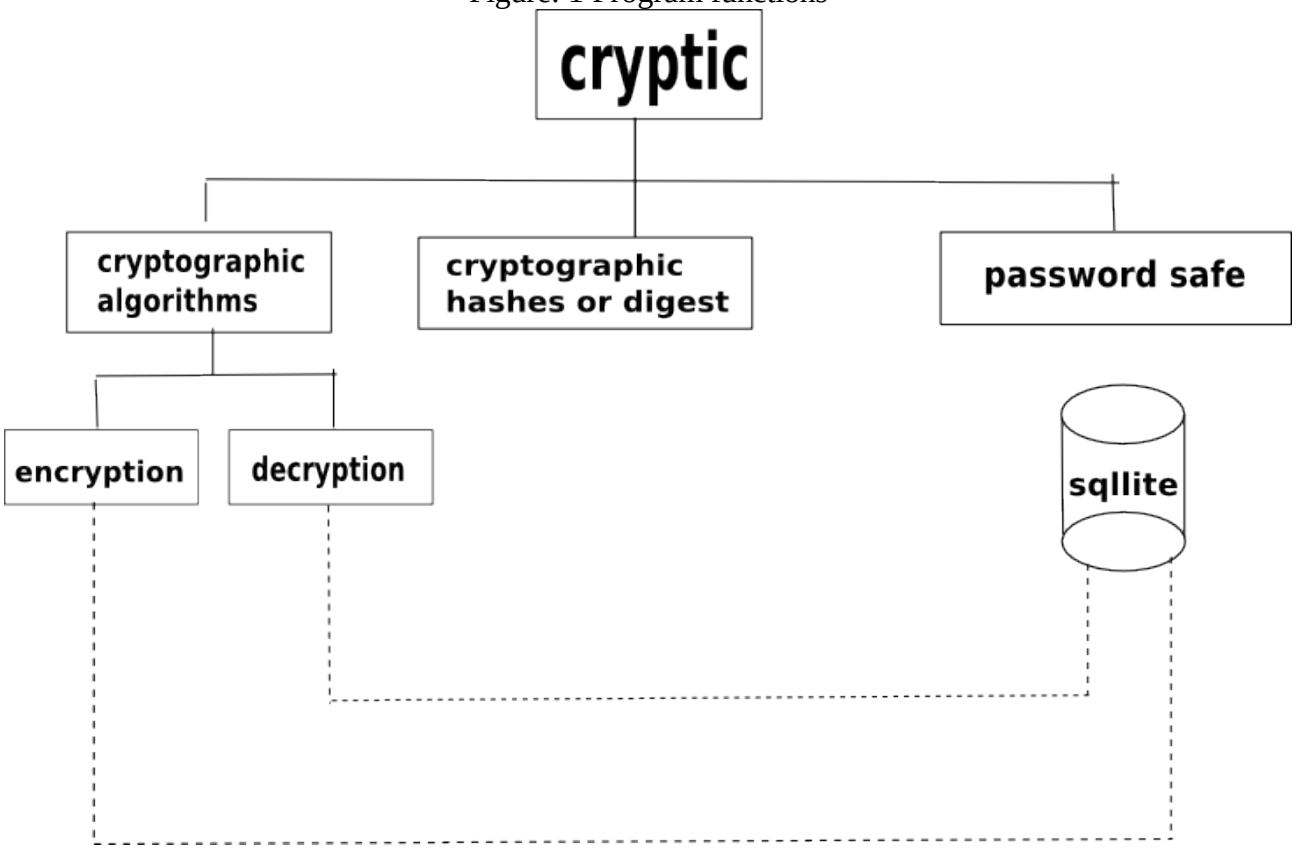
- Password safe locked with a master password.
- Calculating cryptographic hashes or digests of data.

2.1.7 Site adaptation requirements

User interface must exist in English. However, it should not be very difficult to translate Cryptic in other languages.

2.2 Product Functions

Figure: 1 Program functions



- It should be possible to encrypt and decrypt data.
- The passwords for different accounts should be secured in a paasword safe locked with a master password.
- It should be possible to calculate cryptographic hashes or digests of input text.

2.3 User Classes and Characteristics

This is a generic cryptographic application that is usable by all classes of users (who need to secure their data) with any level of technical expertise, education and experience. This suite is usable both for novice or non-technical users as well as expert technical users.

2.4 Operating Environment

Cryptic is intended to be operating system independent. Therefore no specific operating system is excluded. But it will run on Window/linux/unix/BSD distributions etc.

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 needed for software operation. Cryptic will use Qt [6] for GUI (application frontend) programming, SQLite [1] database for password safe and OpenSSL [2] for cryptography.

2.5 Design and Implementation Constraints

- SQLite [1] DB is used for password safe function because it is self-contained [10] and serverless [12] and also because it is zero-configuration [11] 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.

2.6 User Documentation

The following user documentation components will be delievered along with the software (All of the following will be linked from project website i.e. http://cryptic.sourceforge.net/ [7] once the documentation is available):

- Manual page(s)
- Project Wiki
- FAQs
- Online help in project forums (https://sourceforge.net/projects/Cryptic/forums/)

- Project Trackers: Bugs, Patches, Support requests, Feature requests (https://sourceforge.net/tracker/?group_id=304078)

2.7 Assumptions and Dependencies

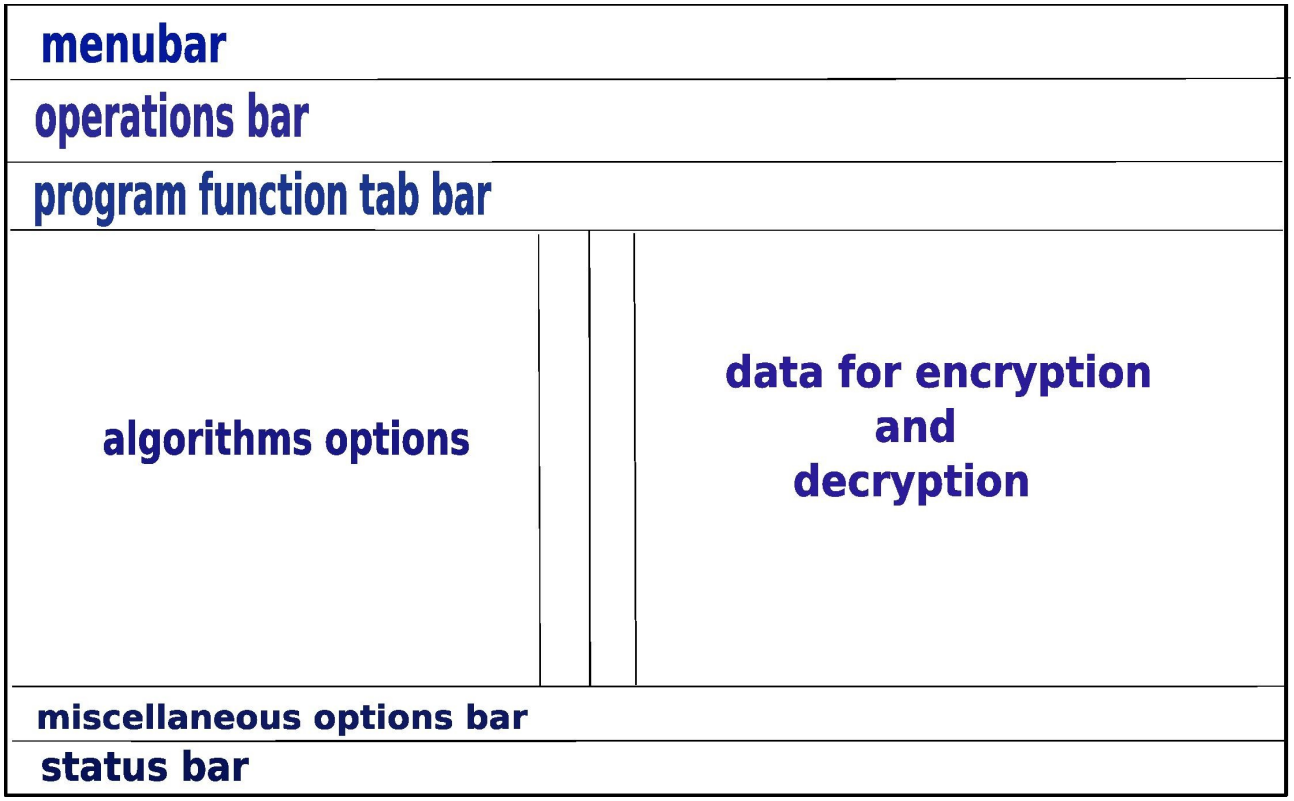
Future versions of Cryptic should use OpenSSL [2] library for cryptographic functions completely.

3. Specific requirements

3.1 External Interface Requirements

3.1.1 User interfaces

Figure-2 Program GUI mockup



- *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) [5] and KDE HIG [4] . All cryptographic functions, hashing and password safe functionalities are accessible from GUI.*
  The GUI will include support for keyboard shortcuts for quick access to program options.
  The GUI 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 for Cryptic is designed with Unix philosophy [9] in mind especially:*
  http://en.wikipedia.org/wiki/Unix_philosophy#McIlroy:_A_Quarter_Century_of_Unix
  http://en.wikipedia.org/wiki/Unix_philosophy#Eric_Raymond

3.1.2 Hardware interfaces

Section 2.1.2 of this document already provides all details about hardware interfaces.

3.1.3 Software interfaces

Crytic will interface with following software components:

- OpenSSL [2] : Cryptic will use cryptographic functions and cryptographic hashing functions from OpenSSL library.
- Qt SDK [6] : Cryptic will use QtSDK to create GUI frontend for program functions.
- SQLite [1] : Cryptic will use SQLite to store user's login info for his/her many accounts securely in a table in a DB.

3.1.4 Communication interfaces

Section 2.1.4 of this document already provides all details about communication interfaces.

3.2 System Features (or functional requirements)

3.2.1 Cryptographic function: Encryption

- **Description and Priority**

  This system feature involves encrypting the data using one of many symmetric or asymmetric cryptographic ciphers for data security.

  Priority: High

- **Stimulus/Response Sequences**

  User selects the cipher used to encrypt data.

  User selects data to be encrypted

  User confirms encryption to be performed by program.

- **Functional requirements**

  1. User selects the radio button corresponding to cipher he wants to be used to encrypt data.
  1. User inputs master key to lock file in 'Master password' text field.

2. User selects the file (input) that is to be encrypted.
3. User clicks on 'Apply' button on 'Operations bar' to apply changes.
4. Output encrypted file with ciphertext is automatically created in input file directory.

3.2.2 Cryptographic function: Decryption

- **Description and Priority**

  This system feature involves decrypting the data using one of many symmetric or asymmetric cryptographic ciphers for data security.

  Priority: High

- **Stimulus/Response Sequences**

  User selects the cipher used to decrypt data.

  User selects data to be decrypted

  User confirms decryption to be performed by program.

- **Functional requirements**

  2. User selects the radio button corresponding to cipher he wants to be used to decrypt data.
  2. User inputs master key to lock file in 'Master password' text field.
  1. User selects the file (input) that is to be decrypted.
  2. User clicks on 'Apply' button on 'Operations bar' to apply changes.
  3. Output decrypted file with plain text is automatically created in input file directory.

3.2.3 Cryptographic Hashes or digests

- **Description and Priority**

  This system feature involves calculating the cryptographic hash or digest of the data using one of many cryptographic hashing algorithms for data integrity.

  Priority: High

- **Stimulus/Response Sequences**

  User selects the hashing algorithm used.

  User selects data whose digest is to be calculated.

  User confirms calculation of digest to be performed by program.

- **Functional requirements**

    1. User selects the radio button corresponding to hashing algorithm he wants to be used.
    2. User selects the file (input) that is to be processed.
    3. User clicks on 'Apply' button on 'Operations bar' to apply changes.
    4. Output hash is displayed in output hash text field in program window.

3.2.4 Password safe

- **Description and Priority**

  This system feature involves securing a set of user's login info (i.e. username/password + additional useful info) by storing them in a table in DB and then encrypting it.

  Priority: Medium

- **Stimulus/Response Sequences**

  User inputs login info (i.e. username/password + additional useful info) into table using program GUI.

  User selects cipher to be used to encrypt password safe table.

  User confirms encryption of password safe table to be performed by program.

- **Functional requirements**

    1. User selects the radio button corresponding to cipher he wants to be used to encrypt data.
    2. User inputs the data to be stored in table that is to be encrypted.
    3. User inputs master key to lock table in 'Master password' text field.
    4. User clicks on 'Apply' button on 'Operations bar' to apply changes.
    5. Output encrypted password (safe) DB is automatically created in current file directory.

4. Other Nonfunctional Requirements

4.1 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.

4.2 Safety Requirements

This application suite is provided under ISC license [3]. 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.

4.3 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.

4.4 Software Quality Attributes

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

- *Availability: The complete code for Cryptic is released under OSI approved, GPL compatible, free and open source ISC license [3] (http://en.wikipedia.org/wiki/ISC_license). So, the code is available free on project's website.*

- *Portability: The code is ANSI C89 compliant and so, it should be easily portable to different architectures and operating systems.*

- *Reliability: Serious attempts are made to make sure code is reliable and of enterprise quality. Still, the code is provided under ISC license and authors of Cryptic (and other code/libraries that Cryptic uses) are not responsible for any damage or data loss. More details in section 4.2 and 4.3 of this document.*

- *Usability: The project is still in planning stage as of SRS version-1, but when the development starts, the code is not expected to be usable atleast until it's first official ALPHA release is tagged in repository.*

4.5 Database Requirements

SQLite [1] (http://www.sqlite.org/) database will be used for password safe. The user will store their login information for various accounts in a table in a SQLite database. The SQLite table will then be encrypted using user-defined cryptographic function using a master password.

Appendix A: Glossary

- ANSI – American National Standards Institute
- BSD – Berkeley Software Distribution
- CUI – Character User Interface
- FAQs – Frequently Asked Questions
- GUI – Graphical User Interface
- GPL – General Public License
- HIG – Human Interface Guidelines
- I/O - Input/Output
- ISC – Internet Systems Consortium
- KDE – K Desktop Environment
- MB - Megabyte
- OS – Operating System
- OSI – Open Source International
- OpenSSL – Open Secure Sockets Layer
- RAM – Random Access Memory
- SRS – Software Requirements Specification

***** End Of SRS *****