Levi Amen

Quentin Covert

Mark Hernandez

Cameron Johnson

Collin Victor



CSCE 361

Dr. Cohen

**Testing Document**



## Testing Strategy

The system was tested in increments. The system was tested each time a feature was implemented. This strategy made it easier to integrate features by allowing us to focus on specific issues as they arose. Many features were cut during the implementation of the program, so testing focused on the core functionality. Our criteria for a functional system were the basic operations it would be expected to perform: new files, modifying files, deleting files.

A final benchmark test was done using the git repository. The project’s visible directories were synced to a remote location and then synced back to another, different location. This test was executed to ensure that the system could process a large amount of files at once, while maintaining a complex file structure between two locations.

The following requirements were tested:

1. General Requirements

1. The program shall allow the user to specify a remote file space for files to be synced to.
   1. This was tested by selecting a remote file space with the GUI, syncing to that location through the program, and then checking that location on the computer.
2. The program shall automatically sync changed files under root directories to the remote filesystem.
   1. This was tested by running the GUI with a specified root directory and a remote directory. Files and directories were added inside the root directory, and the remote was checked for changes.
3. The program shall provide the user the ability to choose how files are synced to the remote filesystem:
   1. The program shall encrypt files and sync to the remote.
      1. This was tested by syncing files and verifying that the contents of the file in the remote directory were encrypted.
4. The program shall provide a record, to the user, of each file watched by the program, which contains:
   1. The file’s path relative to its root directory.
   2. Time of latest modification of file.
   3. Time of latest sync of the file.
      1. To test all of these, the program was run on a system with files and directories selected from the GUI. The GUI displays a window that shows all the relevant information about the selected file or directory, this window was checked against the database to validate that the displayed information was correct
5. The program shall update local files if the corresponding files on the remote filesystem has a more recent modification time.
   1. This was tested as a part of the sync functionality. When the program is initialized database on the remote system is tested against the local filesystem, then only the files with a different modification date from the remote will be replaced with their newer, remote counterparts.

## 2. Crypto Requirements

1. The crypto module shall provide a method to generate a new 32-bit symmetric key in the fernet format.
   1. This was tested simply by having the crypto module generate a key and then checking to ensure the key is a valid 32-bit symmetric key in the fernet format.
2. The crypto module shall automatically produce a 32-bit symmetric, fernet format, key for the user if none is provided.
   1. This was tested by running the GUI without specifying a key. After running, a location was checked to ensure that a key was created in that location and that it conformed to the standards of the crypto module.
3. The crypto module shall provide a method to encrypt files.
   1. This method will be automatically executed by the program.
4. The crypto module shall provide a method to decrypt files.
   1. This method will be automatically executed by the program.

## 3. GUI Requirements

1. The program shall present the user with a GUI once the user runs the program executable.
2. The GUI shall provide means for the user to exit.
3. The GUI shall provide means for root directories to be added the the program’s watch list.
4. The GUI shall provide means for files or directories, under watched root directories, to be ignored and not automatically synced.
5. The GUI shall provide means for auto-encryption to be enabled for files or directories under watched root directories.
6. The GUI shall provide means for auto-encryption to be disabled for files or directories under watched root directories.
7. The GUI shall display to the user relevant information of a selected file contained within a watched root directory.
8. The GUI shall provide means for the user to specify a target remote file system for the program to sync.

The GUI requirements were tested by simply running the program. The GUI automatically starts with the rest of the system. Some features of the GUI are not yet tied with the system (such as ignoring files or setting encryption options.) These options are still in the GUI, as future versions of the software could include these features.