William Kennington (kenning1)
UIUC CS 242 Fa 12

LibSync

Motivation

I find that I often want to securely synchronize sensitive files across my machines automatically and have changes pushed almost immediately. Dropbox takes care of most of these issues but doesn't actually offer a built in way of encrypting file content. There are workarounds like Secretsync, however the client feels a little unstable and requires a userland process to be started upon login. I am seeking a tool that allows me to synchronize my work across multiple machines with user specified encryption keys. More importantly I need a tool which lets me control my own storage platform, and even allows future developers to add extensions for other platforms.

Overview

LibSync is aimed to represent a stable set of tools which will empower end users to safely synchronize their computers by utilizing pre-egress encryption. It will utilize symmetric encryption keys derived from a passphrase the user specifies to the application. Libsync will attempt to watch the filesystem and network communication channel for changes, making sure files stay in sync between all machines. It will also feature an extensible interface, so that submodules can be easily developed for other networking protocols and storage subsystems. Libsync will be initially be developed for GNU/Linux, however, my aim is to isolate as many system specific libraries as I can, so that the code would not take too much effort to port to another operating system. All of the components will be made available under the GPLv3 and work will be pushed to the git repository:
https://github.com/libsync/libsync

Features

- Encryption by leveraging the freely available Openssl library.
- Synchronization over the network using a well defined open protocol.
- A network server responsible for storing the user data and pushing changes to clients
- A network client responsible for waiting for changes and pushing them to the server
- A library which is easily extensible and leveraged by the provided network client and server.

Technologies

- C++ as the programming language for the library and daemons
- Potentially a 3rd party configuration parser
- Openssl for cryptographic operations
- Standard POSIX socket interface and threading interface.
- Inotify for filesystem change notifications.
- Win32 standard functions for the port later in the project

Challenges

- File and Memory consistency, due to the multiple client nature of the program.
- Recovering from errors during transmission and the proper handling of transactions.
- Implementing a mechanism to handle compression and file deltas before encryption is applied.
- Integration and full project testing will be hard since it requires network servers and clients.
- Learning the interfaces to the numerous 3rd party libraries I will be leveraging.

Schedule

Week	Goal	Objectives
1	Basic Foundation	• Establish the data protocol for communication • Familiarize with tcp networking in c++ • Create Abstractions for threads, sockets, crypto, filesystem. • Setup the network server and client with generic callback methods • Define and create a configuration parser and language • Setup the testing suite for the networked protocols • Setup a basic file metadata memory structure • Basic file synchronization (Single Client Sync)
2	Multiple Clients	• Extend the previous metadata entries for transaction awareness • Define a protocol for pushing changes to clients • Extend the server and client to support proper file locking mechanisms • Add transaction awareness to the clients so that they properly interact and sync responsibly. • Extend test suite to simulate real world consistency issues
3	Encryption and Deltas	• Add encryption support to the file transfer stream • Add encryption support to the metadata layer • Add block based delta support using the rsync algorithm http://rsync.samba.org/tech_report/
4	Operating System Ports	• Figure out how to compile the shared libraries for Windows • Port components like file monitoring to BSD / OSX • Port all of the POSIX calls and file monitoring to Win32 • Port testing suite to work accordingly for these platforms • Make a build cluster which tests compatibility between platforms