6-8-2020

Multilock: A Document Escrow Service

Dominic Lagorio

Steven Herman

SANTA CLARA UNIVERSITY

COMPUTER SCIENCE AND ENGINEERINGS DEPARTMENT

Date: June 8, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

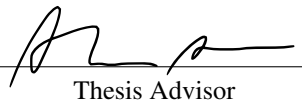
Dominic Lagorio
Steven Herman

ENTITLED

Multilock: A Document Escrow Service

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



Thesis Advisor

N. Ling

Department Chair

Multilock: A Document Escrow Service

by

Dominic Lagorio
Steven Herman

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 8, 2020

Table of Contents

List of Figures

List of Tables

Multilock: A Document Escrow Service

Dominic Lagorio
Steven Herman

Computer Science and Engineerings Department
Santa Clara University
June 8, 2020

ABSTRACT

The objective of this project is to create a system wherein it is no one person's responsibility to claim agency over a given file, but rather that the responsibility is distributed amongst a number of individuals who must decide as a unit to grant access to said files. There are a number of driving factors here, including removing culpability for individual involvement with the file, preventing compulsory access to files, and providing sophisticated security protocols for file access.

Chapter 1

Introduction

With the proliferation of cameras and Internet of Things devices, surveillance capabilities have skyrocketed in the last few years. Worse, government deployments of cameras are rapidly expanding, in the form of body cameras, city surveillance, and border security. While this surveillance may serve noble goals, in its current state, there is no oversight of its use. Without action, these changes endanger rights within our society, threatening the creation of real life panopticons throughout society.

Currently, exclusive ownership of this information is held by government or corporate entities. Entities like Amazon have shown their willingness to turn over sensitive data such as Ring surveillance video to the police, infringing on constitutionally protected rights. For the preservation of these rights, the public must be capable of preventing unauthorized use by malicious entities of personal or otherwise sensitive data.

While solutions exist for key management and secret management, few allow keys to be split among stakeholders, requiring a quorum for access. Those which do, like CloudFlare's Red October, have limited support for fine grained access control. Most work on the basis of delegating your key to the server for limited time or uses, rather than limiting which documents can be accessed. By requiring multiple actors to access to sensitive data, we can benefit from existing legal frameworks for evidence protection, access, and accountability for its alteration or destruction.

Our solution will act as a point of escrow, securely encrypting data and distributing shares of the key. A configurable amount of shares will be required to decrypt the data. Different types of information will be handled differently to ensure fine grained control. For example, video streams may be split by time allowing authorization of a limited segment of time. Additionally, each piece of information will be configurable to allow varying degrees of security. The data stakeholders and required shares will be set when the information enters the system. Upon entering the system, data will be encrypted using generated keys; Key fragments will then be securely distributed to the associated stakeholders, minimizing the time the data is vulnerable.

Although this still presents a single point of attack, the data isn't held by a single, omniscient entity. Because of the mathematical security of the encryption scheme, access requires a higher degree of stakeholder involvement than

current systems. Even if individuals aren't involved as stakeholders, this forces greater oversight into data access.

Chapter 2

Requirements

2.1 Functional Requirements

Functional requirements are those that describe what the system does. They refer to specific functionality and elaborate on the role that the system plays and add the context of what it can be used for.

2.1.1 Critical Requirements

These are requirements that stand for the core functionality of the system: they enumerate what, at a base level, the system actually does.

- The system must accept file uploads from users and encrypt them.
- The system must split and distribute encryption key fragments among specified stakeholders.
- The system must store encrypted files.
- The system will accept authorized client connections.
- The system must allow decryption of files for which a user has all keys.

2.1.2 Recommended Requirements

These are requirements that would ideally be implemented before shipping the system, but at a base level are not really necessary to the core functionality of the system.

- The system will notify users of requests for which they are stakeholders.
- The system will keep users anonymous.
- The system will split temporal files into chunks which are separately encrypted.

2.1.3 Suggested Requirements

These are functions that, while they would be nice, are not necessary, nor does their absence really detract from the system.

- The system will communicate over HTTPS.

2.2 Nonfunctional Requirements

Nonfunctional requirements lay out what the system is like: they elaborate on the experience the user will have while interfacing with the system.

2.2.1 Critical Requirements

At a base level, these are the attributes the system will seek to abide by.

- The system will be secure.
- The system does not expose the keys for longer than necessary.

2.2.2 Recommended Requirements

These requirements pertain to the system in so far as they would be ideally be features before it shipped, but if they are not, the system will still be usable.

- The system will be timely in processing requests.
- The system will be easy to use and looks “clean”.
- The system will be easily extensible to support future encryption algorithms.
- The system will be modular and easy to maintain.

2.2.3 Suggested Requirements

These features would be nice to implement, but all things being equal, are not in any way a necessity.

- User identifiers are tied to optional metadata.
- The system will be easily portable

2.3 Constraints

These are things that, for one reason or another, restrict the design space and/or scale of our project. While not technically requirements, they are considerations to be made.

- The system runs on Windows, Mac, and Linux

Chapter 3

Use Cases

3.1 Retrieve Metadata

Goal: There must be some way to know who to contact and how many shares are remaining before you can access a given file.

Actors: Shareholders

Pre-conditions: The actor has a share in the file.

Post-conditions: The actor receives metadata on a given file as text output.

Exceptions: N/A

3.2 Upload File

Goal: To upload a file and split the key.

Actors: Shareholders

Pre-conditions: The actor has a file that they would like to encrypt and split keys for, as well as all other shareholders specified.

Post-conditions: The actor receives a share of the key, and so do all shareholders.

Exceptions: N/A

3.3 Request Access

Goal: Access a given file.

Actors: Shareholders

Pre-conditions: The actor has a share in the file.

Post-conditions: The actor receives access to a file as a downloadable package.

Exceptions: N/A

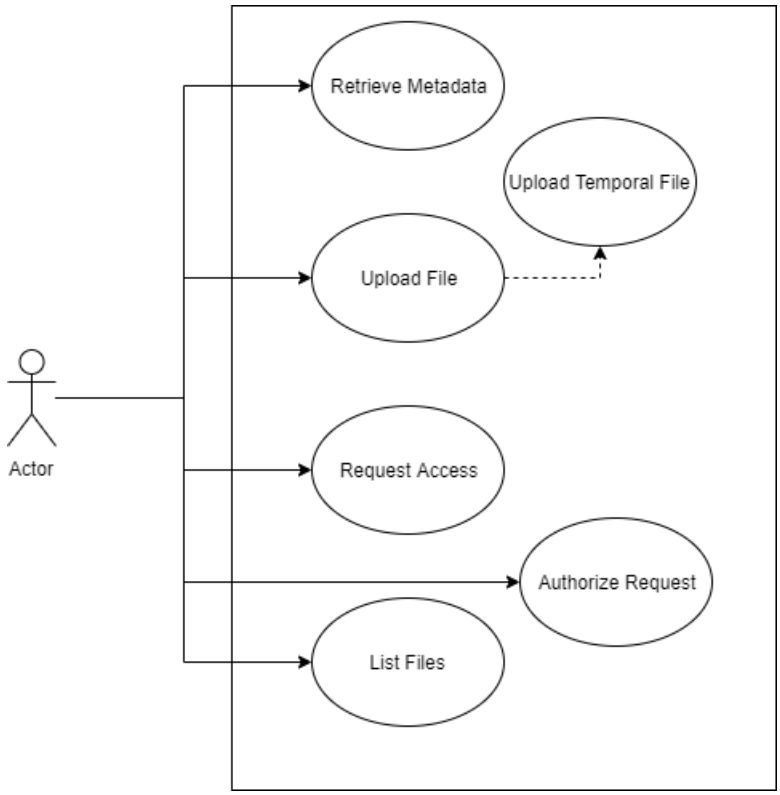


Figure 3.1: Use Case Diagram

3.4 Authorize Request

Goal: Give your share of the key to a shareholder who is requesting access to a particular file.

Actors: Shareholders

Pre-conditions: The actor has a share in the file.

Post-conditions: The actor gives up their share of the file to the requester.

Exceptions: N/A

3.5 List Files

Goal: See the files you have a share in.

Actors: Shareholders

Pre-conditions: The actor has a share some number of files.

Post-conditions: The actor receives a list of the files they have access to in the form of a text list.

Exceptions: N/A

Chapter 4

Activity Diagram

Shown in Figure 4.1 is the activity diagram for any user interacting with our document escrow service. These users could be humans but they could also be machines (ie security cameras). The client must authenticate with the server before they can perform any actions. Then they can choose an action to perform. After taking all necessary steps for that action, the user is able to choose another action.

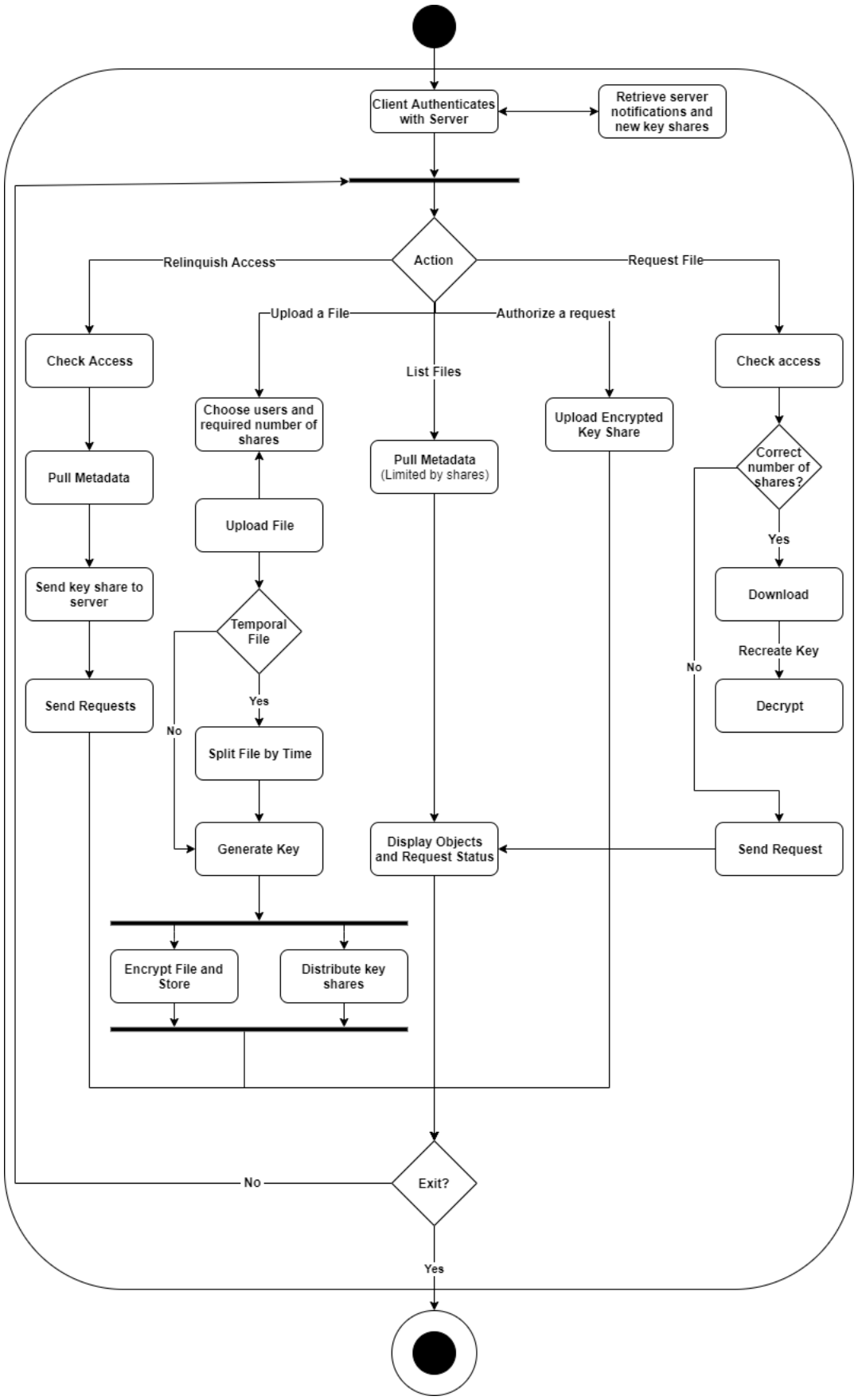


Figure 4.1: Activity Diagram

Chapter 5

Conceptual Models

Command Line Interface

List

The user should be able to list documents they are shareholders for with the following command:

```
escrow ls
```

This maintains similarity to Unix systems which should ease the user's learning process.

Authorize

The user will be able to give a specified user authorization for a document with the following command:

```
escrow authorize <document identifier> <user>
```

Any aliasing of users should be optional, not mandatory to protect the user's privacy.

Request

The user should be able to request a document using the document identifier. They may be able to alias the document, though for security reasons, the server should not know the name of the document.

```
escrow request <document identifier>
```

Secure

The user should be able to secure a file on their local machine. This will require a list of users to make shareholders.

```
escrow secure <filename> <user> [<user> ... <user>]
```

Chapter 6

Technologies Used

6.1 Golang

Golang was chosen main language for implementation our implementation because it has excellent support for concurency and a wide set of libraries for cryptography and backend development. Both the server and client (cli) will be implemented in Go.

6.2 Ed25519

Ed25519 is a fast elliptic curve algorithm. This will likely be used for identity within the system because it has small key lengths and signatures and is well suited for communication between machines. Ideally, this the security of the system won't depend on this algorithm. This may be replaced at some time with a post-quantum algorithm.

6.3 ChaChaPoly20/AES-GCM

Block ciphers are more efficient for encrypting large quantities of data. Neither ChaChaPoly20 nor AES are compromised by quantum computers in a meaningful way. They also are well tested which allows them to be trusted with securing all the data on the server.

6.4 Shamir Sharing Algorithm

Shamir's sharing algorithm is used for splitting any keys for distribution among different algorithms. This is post-quantum secure and shouldn't weaken any other sections of the products. Since this relies on defining an n-point curve, there should be no way to compromise the data without all (required) shareholder's consent.

6.5 S3 Compatible Object Storage (minio)

Since the escrow system is structured around different files/documents, there is no need for a complete file system. Object storage makes the most sense since each object can be considered independently. We choose an S3 compatible API because it makes it easier for individuals to deploy their own copies of the server and allows easy scalability. This doesn't compromise the security of the data because all the files are encrypted before they placed in the object store. Even if Amazon (or government entities) compromised the object store, they would still need to break the encryption which is the original source of security.

Chapter 7

Architecture

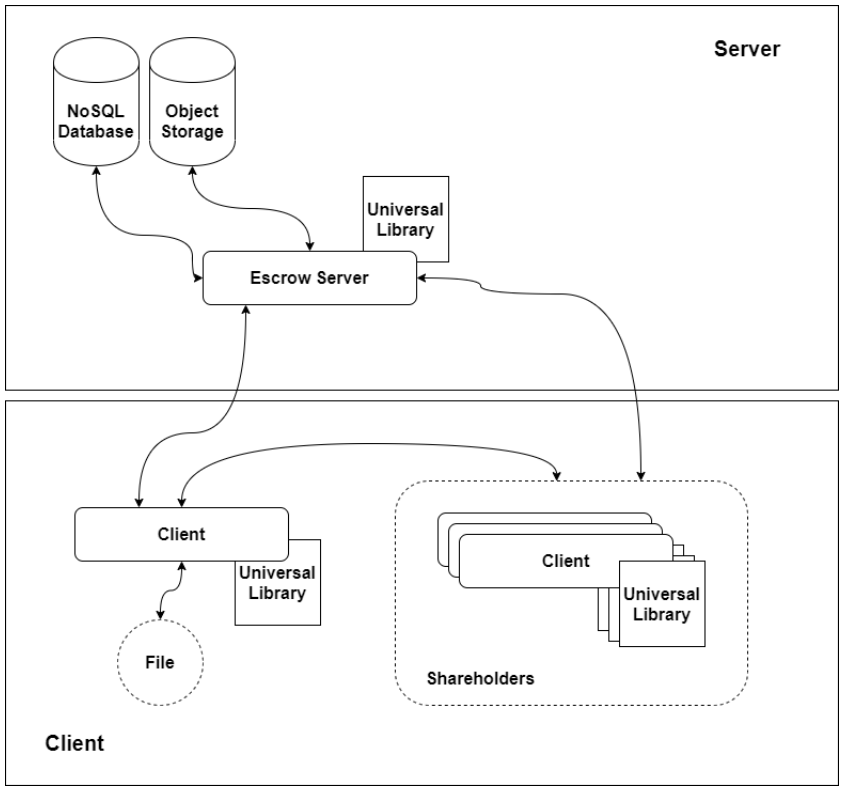


Figure 7.1: Architecture Diagram

7.1 Server Architecture - Service Oriented Architecture

We plan to build the server portion of the architecture around a service oriented model. This will make it easily deployable and auditable to ensure no insecure transfer of data. Additionally, this will likely help with scaling out in deployments since you could easily add load balancers and data replication. The isolation of different components makes it so in an ideal case, each component could be located separately, distributing the limited trust.

7.2 Client Architecture - Client-Server Architecture

From the perspective of the client, the entire system represents a client-server architecture. The server is responsible for mediating requests between clients and protecting client identity. However, this shouldn't be taken to say the server has complete control because the server shouldn't be able to access the data either. The server functions as an untrusted intermediary and place for storage of encrypted files.

Notifications from the messaging queue aren't real time. Rather the clients have to poll for them to eliminate risk of tracking internet traffic. While this might allow an eavesdropper to determine whether an individual uses the service, they shouldn't be able to identify that individual by making them a stakeholder.

Chapter 8

Design Rationale

8.1 Architecture

We chose a service oriented architecture because it makes each independent component easy to separate. This will allow easier development and reduce the risk of interconnection. The architectural limitations of this model will help enforce security practices.

8.2 Technologies

Golang was chosen because it is powerful while being easy to use. While we initially considered Python, the limited control means we would have to use C/C++ cryptography libraries. Additionally, the global interpreter lock would have prevented effective multi-threading. Go's concurrency patterns helped us achieve efficient concurrency and it has many readily available cryptographic libraries which have been audited. This simplified the process since our project uses a lot different cryptographic algorithms.

Other choices are subject to further decision, but we mostly look to utilize well tested and used software. This should help mitigate the risk of vulnerabilities and provide a production-ready software product.

Chapter 9

Testing

All parts should have adequate written unit tests to ensure they work as designed. These should include both positive (functionality) tests as well as negative tests to ensure vulnerabilities are not present. We extensively used golang's built in testing features which enabled easy and regular testing. All code was also run through static and dynamic analysis tools to minimize the potential for insecure code.

9.1 Core Library

Since the core client and server functionality is implemented as a single library, the same implementation can be tested multiple ways. We should have unit tests for many available functions to ensure we don't break this code. Additionally, we can set tests to execute before committing or before allowing pull requests to be merged. This will help ensure every change to the library has minimal risk of contaminating the client or server.

9.2 Server

The server was tested with both integration and unit tests. After initial development, we should always have a working version of the server both for demos and for testing. This testing will help ensure we have the stability needed for this application.

We also did several rounds of fuzzing to test the server for crashes. While this isn't sufficient to ensure the server adequately handles any input, it provides assurances that the server won't crash or otherwise mishandle bad input.

Chapter 10

Risk Analysis

Table 10.1: Risk Analysis Table

Risk	Consequences	Probability	Severity	Impact	Mitigation
Bugs	Project progress is delayed. Team members will have to work overtime to fix the issues.	0.9	4	3.6	Clean code; consistent style; unit testing; peer code review
Time	Project is not completed on time.	0.35	9	3.15	Set deadlines and prioritize features
Illness	Project progress is delayed	0.2	5	1.0	Wash hands; take frequent breaks; start tasks early.
Loss of Dev Environment	Loss of productive time, loss of data	0.1	5	0.5	Make backups, use virtualenv, document development environment
Data loss	Delays in production	0.02	9	0.18	Use version control for all code. Store code off device
Changes in requirements	Delays in production, Extra work for team to complete project	0.2	6	1.2	Document requirements clearly in design documents. Solicit feedback early

Chapter 11

Development Timeline

While the timeline for the first quarter was fairly set in stone, the rest of the project is more fluid, and as such is more subject to change. This is reflected in the blank spaces therein.

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Fall Quarter**					
1.1	**Conception Phase**				0	100%
1.1.1	Deliverable-Problem Statement	Team	9/27/19	10/11/19	14	100%
1.2	**Elaboration Phase, Part 1**				0	11%
1.2.1	Deliverable-Design Report	Team	10/25/19	12/6/19	41	11%
1.2.1.1	Dev Timeline	Dominic	10/25/19	12/6/19	41	90%
1.2.1.2	Risks	Steven	10/25/19	12/6/19	41	0%
1.2.1.3	Use Cases	Dominic	10/25/19	12/6/19	41	50%
1.2.1.4	Conceptual Models	Steven	10/25/19	12/6/19	41	0%
1.2.1.5	List of Requirements	Dominic	10/25/19	12/6/19	41	0%
1.2.1.6	Sequence Diagram	Steven	10/25/19	12/6/19	41	0%
1.2.1.7	Architectural Diagram	Dominic	10/25/19	12/6/19	41	0%
1.2.1.8	Design Rationale	Steven	10/25/19	12/6/19	41	0%
1.2.1.9	Technologies Used	Dominic	10/25/19	12/6/19	41	0%
1.2.1.10	Component State Chart(s)	Steven	10/25/19	12/6/19	41	0%
1.2.1.11	Test Plan	Dominic	10/25/19	12/6/19	41	0%
1.2.1.12	User Manual	Steven	10/25/19	12/6/19	41	0%
1.2.1.13	Test/Experimental Results	Dominic	10/25/19	12/6/19	41	0%
1.3	**Disccretionary Component**	Team	9/27/19	12/7/19	70	0%

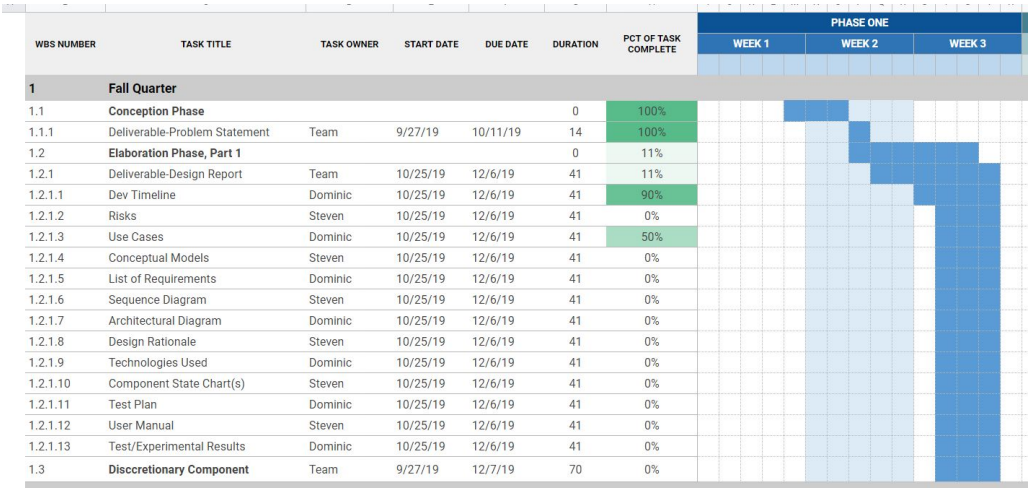


Figure 11.1: Initial Gantt Chart Phase 1

2	**Winter Quarter**					
2.1	**Elaboration Phase, Part 2**		1/6/19	1/31/19	25	0%
2.1.1	Deliverable-Revised DR	Team	1/6/19	1/31/19	25	0%
2.1.1.1	System Prototype		1/6/19	1/31/19	25	0%
2.1.1.2	Collaboration Diagrams		1/6/19	1/31/19	25	0%
2.1.1.3	Class Diagram/Schematic		1/6/19	1/31/19	25	0%
2.1.1.4	Risk Resolution/Mitigation		1/6/19	1/31/19	25	0%
2.2	**Construction Phase**		1/31/19	3/13/19	43	0%
2.2.1	Deliverable-Operational System	Team	1/31/19	3/13/19	43	0%
2..2.1.1	Source Code		1/31/19	3/13/19	43	0%
2..2.1.2	Deployment Scripts		1/31/19	3/13/19	43	0%
2..2.1.3	Deployment Instructions		1/31/19	3/13/19	43	0%
2..2.1.4	Test Cases/Implementation		1/31/19	3/13/19	43	0%
2..2.1.5	Test Results		1/31/19	3/13/19	43	0%
2.3	**Discretionary Component**	Team	1/6/19	3/13/19	67	0%

Figure 11.2: Initial Gantt Chart Phase 2

3	**Spring Quarter**			
3.1	**Delivery Phase**		0	0%
3.1.1	Deliverable-Senior Thesis	Team	0	0%
3.1.1.1	User Manual		0	0%
3.1.1.2	API Documentation		0	0%
3.1.1.3	Maintenance Guide		0	0%
3.1.1.4	Suggested Changes		0	0%
3.1.1.5	Experiences/Lessons		0	0%
3.1.1.6	API Documentation		0	0%

Figure 11.3: Initial Gantt Chart Phase 3

Chapter 12

Suggested Changes and Next Steps

12.1 Suggested Changes

12.1.1 File Name Aliasing

One of the most important changes for human usability is to the user interface. Currently, for authorization, requesting, and listing files, as well as giving up their key share, the user will need or get the UUID of a file rather than a human-readable name. It is our recommendation that the most immediate next step then is to get file name aliasing working for the whole system. This would allow users to better track what they are requesting or authorizing, as well as allowing them to better use the list and relinquish commands, leading to a better user experience overall.

12.1.2 Client Data Caching

Several design decisions were made assuming the server would not even hold information about who owns the file on the server. When working on this, this complicated the design extensively because it required caching and storage of data on the user's computer. Ensuring this was both correct and up to date posed additional challenges outside the scope of this project. The authors would recommend this system be implemented both to add additional security to the design as well as make the system more efficient.

12.2 Next Steps

12.2.1 Web User-Interface

Something that we had talked about doing but decided was outside of the scope of the project was building a web-based user interface. Something more graphically appealing than a command line tool could be a very welcome addition to the project, especially something that organizes your files in some way that makes human sense.

Building out this UI is possible in two ways: first, one could use Javascript or other web native languages to interact directly with the server. Alternatively, the core library could be compiled to WebAssembly and used directly.

12.2.2 Temporal-Based File Access Control

As mentioned previously, there is huge need for this kind of software to be used for video capture. Automatically splitting and securing the file would be simple enough to implement though we ran out of time to implement this additional feature.

Chapter 13

Ethical Implications

13.1 Data Ownership and Legality

In the past, data ownership has been kind of one dimensional, with the owners of data being fairly straightforward. However, with the changing dynamic of data collection and processing, single-party ownership of data may be a dated concept. Enter this data escrow service. While it might stand as a sort of seaway between the current status quo of data ownership and its future, the nature of this data escrow service raises quite a few interesting ethical questions.

First is the question of legality; whether the system follows the letter of the law or the spirit of it. From a purely legal standpoint, this kind of multiparty encryption service does nothing inherently wrong. It merely removes ownership of data from a single entity for favor of multiparty access. While this might muddy the waters of who exactly is responsible for what part of the data, it does not strictly subvert any legalities about who can hold ownership of the data: the owners of the keys are still collectively responsible for the data. Though they may not individually be able to access the files they upload, as a collective they hold sovereignty over access to said files. All of this is to say that a group might be individually inculpable while collectively responsible.

13.2 Security

This brings up a second ethical question regarding the sense of security provided by the system, and not regarding the security of files, but rather of people. That is to say that a person might think him or herself invulnerable to some level of persecution for what's on a file being that they have no personal responsibility for it. However, that's not necessarily the case. There is likely still going to be some level of colloquial association of person to file, aided by the fact that there is an innate user-to-file association system. This begs the question, "has the system created a false sense of security for its users," being that it promises multiparty ownership but still associates user to file. The fact of the matter here is that the system is labeled as an escrow service, not one that can magically erase culpability. While there is a certain level of privacy and anonymity that can be associated with the system, it never promises any of that; it merely promises to hold a file in stasis for a time.

Furthermore, this presumed discretion leads to a question about how the user will interact with the system and exactly what kind of information will be stored using it. People tend to store things that they'd rather most other people not see in private or anonymous places, sometimes with like-minded people. It is possible that the system could provide a platform for temporary, protected storage for unsavory data. That being said, that kind of data would be no more accessible than any other data the system stores. This implies that the interested parties would have to go through the same key request and collection process that everyone does. Presumably, that would be riskier than it would be for the everyday user, due to the nature of what they'd be hiding therein and the paranoia surrounding the access of that data. In that way, it makes less sense for a group to use a service like the system to store their less ethical materials.

13.3 Accessibility

Also, the fact that the system is available publicly means that literally anyone with internet access and Windows or Linux could have access to it. This was a choice made in an era where every other "free" app has a pay-walled feature; that this should be something that anyone who needed or wanted should have access to. This is because, quite simply, people should be able to protect their data. While yes, as stated before, there could be unsavory data hidden, it is our belief that the vast majority of users would just want a secure platform to store data they'd rather keep private. It's no company's or country's right to access a person or party's data without their consent (or in the case of a country perhaps a warrant). So why, then, should there be any problem with people protecting their data? Multiparty ownership just allows said party another avenue to do that in a way that can help avoid the compulsory relinquishing of data.

Overall the system raises some interesting ethical questions surrounding legality, user culpability, and the nature of the data that will be stored to name a few. If this represents the future of data ownership, these questions, among many others, will need to be answered.

Chapter 14

Lessons Learned

14.1 External Challenges

This is a hard section to write, not only because the lessons learned were particularly painful ones to have learned, but because many of them came from forces outside of the project.

Both members of the project experienced life-changing events during the course of this project. This required massive changes be made to the scope of the project. While we really felt certain things, like a web user interface for instance, would have brought a lot to the project, there simply was not enough time to get it all done before the presentation. Through this, we learned to identify what was important and what was less so. This skill then let us prioritize those things we identified as more important. Both members learned the importance of clear communication, especially with increasing remote work. The code base reflects many of these changes and some uncompleted goals. However the initial design and ideas, as well as the project we present, are still something we are happy with given the circumstances.

14.2 Tooling Challenges

The actual implementation also presented a bevvy of unique challenges. Golang relies on imports to be located by URI, an issue since the repository wasn't publicly available at the time. This prevented one member from having easy testing and execution of the project due to issues with their git installation. This highlights that tools, especially version control, can be useless if not configured properly.

14.3 Concluding Lessons

The overarching theme here is really that we learned to cope with stress, and more generally to problem-solve under pressure. Without these two skills, we never would have completed the project. All of the work-arounds, re-prioritizing, and other work done for this project was really a teaching point for these two things.

Appendices

Appendix A

User Manual

A.1 Initial Setup

A.1.1 Download

The code is available at `github.com/multilock`.

A.1.2 Installation

The user will need to have go ¿ 1.13 installed. Instructions for installing go can be found at `https://golang.org/doc/install`. After installing go, the the user should run

```
go build cmd/escrow.go
```

This will generate the required executable using go modules to download dependencies.

A.1.3 Hosting a Server

While the executable contains everything necessary to run the client, the server requires a data storage backend. This can be either an S3 compatible server or minio running locally.

The access information for the data storage backend can be set for the server using the environment variables:

- STORAGE_ADDR (ie ”localhost:8000”)
- KEY_ID
- ACCESS_KEY

The server can then be started by running.

```
escrow serve
```

A.1.4 Notes

Users should note that this program is a proof of concept. If they want to deploy this program, they should conduct their own security audits and risk analysis.

A.2 CLI Documentation

The following commands include more details available when using the ”-h” flag on any command. There are a variety of optional flags not described here.

A.2.1 List

The user can list documents they are shareholders for with the following command:

```
escrow ls
```

This maintains similarity to Unix systems which should ease the user’s learning process.

A.2.2 Authorize

The user is able to give a specified user authorization for a document with the following command:

```
escrow authorize <document identifier> <user>
```

Any aliasing of users is optional, not mandatory to protect the user’s privacy.

A.2.3 Request

The user is able to request a document using the document identifier. They are able to alias the document, though for security reasons, the server does not know the name of the document.

```
escrow request <document identifier>
```

A.2.4 Secure

The user is able to secure a file on their local machine. This requires a list of users to make shareholders.

```
escrow secure <filename> <user> [<user> ... <user>]
```

A.2.5 Relinquish

The user is able to relinquish access to a document. This will place their share in the ownership of the server.

```
escrow relinquish <document identifier>
```

Appendix B

Code Listing

B.1 Online

All the code for this project can be found at `github.com/multilock`. This will be the most up to date version of the code and may include changes implemented after the senior design project is finished.

B.2 Code Archive

The remaining pages of this PDF contain the code used for this project as of June 8, 2020

B.2.1 cmd/escrow.go

```
package main

import (
	"os"

	"github.com/multilock/multilock/internal/cmd"
)

func main() {
	cmd.Main(os.Args)
}
```

B.2.2 internal/client/client.go

```
package client

import (
	"bytes"
	"context"
	"encoding/base64"
	"encoding/json"
	"fmt"
	"io"
	"log"
	"mime/multipart"
```

```
	"net/http"
	"time"

	"github.com/multilock/multilock/internal/debug"
	"github.com/multilock/multilock/internal/encryption"
	api "github.com/multilock/multilock/internal/http"
	"github.com/multilock/multilock/internal/shamir"
	"github.com/multilock/multilock/internal/store"
	"github.com/multilock/multilock/internal/user"
	"github.com/multilock/multilock/internal/uuid"
)

// EscrowClient ..
type EscrowClient struct {
	serverAddr string
	user *user.SecretIdentity
	httpClient *http.Client
}

// NewEscrowClient ..
func NewEscrowClient(server string, user *user.SecretIdentity) (*EscrowClient, error) {
	client := EscrowClient{
		serverAddr: fmt.Sprintf("http://%s", server),
		user: user,
		httpClient: &http.Client{Timeout: 10 * time.Second},
	}

	return &client, nil
}

// Close ..
func (e *EscrowClient) Close() {
}

// Do ..
func (e *EscrowClient) Do(req *http.Request) (*http.Response, error) {
	err := e.user.Authenticate(req)

	if err != nil {
		return nil, err
	}

	resp, err := e.httpClient.Do(req)

	if err != nil {
		return nil, fmt.Errorf("not found: %v", err)
	} else if resp.StatusCode != http.StatusOK {
		return nil, fmt.Errorf("bad status: %d", resp.StatusCode)
	}

	return resp, nil
}

// Identity returns the identity of the client user
```

```
func (e *EscrowClient) Identity() user.Identity {
	return e.user.Identity()
}

// GetRequests ..
func (e *EscrowClient) GetRequests(ctx context.Context) ([]store.EscrowRequest, error) {
	req, err := http.NewRequestWithContext(ctx, http.MethodGet, e.serverAddr+"/v1/api/
	    "+"requests", nil)

	if err != nil {
		return nil, fmt.Errorf("failed␣%v", err)
	}

	resp, err := e.Do(req)

	if err != nil {
		return nil, fmt.Errorf("failed␣to␣connect␣to␣server␣%v", err)
	}

	var requests []store.EscrowRequest
	err = json.NewDecoder(resp.Body).Decode(&requests)

	if err != nil {
		return nil, fmt.Errorf("failed␣to␣read␣response␣%v", err)
	}

	return requests, nil
}

// Authorize ..
func (e *EscrowClient) Authorize(ctx context.Context, documentID uuid.UUID, user user.
    Identity) error {
	routeGetKey := fmt.Sprintf("%s/v1/api/documents/%v", e.serverAddr, documentID)
	routePutShare := fmt.Sprintf("%s/v1/api/requests/%v", e.serverAddr, documentID)

	req, err := http.NewRequestWithContext(ctx, http.MethodGet, routeGetKey, nil)

	if err != nil {
		return fmt.Errorf("failed␣authorize␣%v", err)
	}

	resp, err := e.Do(req)

	if err != nil {
		return fmt.Errorf("failed␣authorize␣%v", err)
	}

	var mdata store.EscrowMetadata
	err = json.NewDecoder(resp.Body).Decode(&mdata)

	if err != nil {
		return fmt.Errorf("failed␣authorize␣%v", err)
	} else if mdata.Key == nil {
		return fmt.Errorf("no␣key␣for␣doc␣%v", documentID)
```

```
	}

	// Handle decrypt & re-encrypt
	key, err := e.user.Unseal(mdata.Owner, mdata.Key)

	if err != nil {
		return err
	}

	encryptedKey, err := e.user.Seal(user, key)

	if err != nil {
		return err
	}

	auth := api.AuthorizationResponse{
		To: user,
		Key: encryptedKey,
	}

	authVal, err := json.Marshal(auth)

	if err != nil {
		log.Println(err)
	}

	req, err = http.NewRequestWithContext(ctx, http.MethodPost, routePutShare, bytes.
	    NewBuffer(authVal))

	_, err = e.Do(req)

	if err != nil {
		log.Println(err)
	}

	return err
}

// Ls ..
func (e *EscrowClient) Ls(ctx context.Context) ([]store.EscrowDocument, error) {
	req, err := http.NewRequestWithContext(ctx, http.MethodGet, e.serverAddr+"/v1/api/
	    "+"documents", nil)

	if err != nil {
		return nil, fmt.Errorf("failed␣%v", err)
	}

	resp, err := e.Do(req)

	if err != nil {
		return nil, fmt.Errorf("failed␣to␣connect␣to␣server␣%v", err)
	}

	var documentList []store.EscrowDocument
```

```
	err = json.NewDecoder(resp.Body).Decode(&documentList)

	if err != nil {
		return nil, fmt.Errorf("failed to read response %v", err)
	}

	return documentList, nil
}

// Secure ..
func (e *EscrowClient) Secure(ctx context.Context, users []user.Identity, object io.
    Reader, alias string) (uuid.UUID, error) {
	key, err := encryption.GenerateKey()

	if err != nil {
		return uuid.Zero, fmt.Errorf("couldn't generate key: %w", err)
	}

	buf := new(bytes.Buffer)

	_, err = encryption.Encrypt(buf, object, key)

	if err != nil {
		return uuid.Zero, fmt.Errorf("couldn't encrypt data: %w", err)
	}

	shares, err := shamir.Split(key, len(users), len(users))
	debug.Printf("Master Key: %s\n", base64.StdEncoding.EncodeToString(key))

	if err != nil {
		return uuid.Zero, fmt.Errorf("couldn't split key: %w", err)
	}

	// handles distribution of keys
	var escrowShares []store.EscrowShare
	for n, share := range shares {
		debug.Printf("Key (%s): %v\n", users[n].Short(), base64.StdEncoding.
		    EncodeToString(share))

		encryptedUserShare, err := e.user.Seal(users[n], share)

		if err != nil {
			return uuid.Zero, err
		}

		escrowShares = append(escrowShares, store.EscrowShare{
			User: users[n],
			Key: encryptedUserShare,
		})
	}
	body := new(bytes.Buffer)
	writer := multipart.NewWriter(body)

	part, err := writer.CreateFormFile("file", alias)
```

```
        if err != nil {
                return uuid.Zero, err
        }
        io.Copy(part, buf)

        part, err = writer.CreateFormField("shares")
        if err != nil {
                return uuid.Zero, err
        }

        json.NewEncoder(part).Encode(api.DocumentEscrowShares{
                Shares: escrowShares,
        })

        writer.Close()

        req, _ := http.NewRequestWithContext(ctx, http.MethodPost, e.serverAddr+"/v1/api/"
            +"documents", body)
        req.Header.Set("Content-Type", writer.FormDataContentType())

        resp, err := e.Do(req)

        if err != nil {
                return uuid.Zero, fmt.Errorf("failed to post %v", err)
        }

        var documentID uuid.UUID
        err = json.NewDecoder(resp.Body).Decode(&documentID)

        return documentID, err
}

// Metadata ..
func (e *EscrowClient) Metadata(ctx context.Context, documentID uuid.UUID) (store.
    EscrowMetadata, error) {
        routeGetMetadata := fmt.Sprintf("%s/v1/api/documents/%v", e.serverAddr, documentID
            )
        req, _ := http.NewRequestWithContext(ctx, http.MethodGet, routeGetMetadata, nil)
        resp, err := e.Do(req)

        if err != nil {
                return store.EscrowMetadata{}, fmt.Errorf("failed to get request %v", err)
        }

        var mdata store.EscrowMetadata
        err = json.NewDecoder(resp.Body).Decode(&mdata)

        if err != nil {
                return store.EscrowMetadata{}, err
        }

        return mdata, nil
}
```

```
// Request ..
func (e *EscrowClient) Request(ctx context.Context, documentID uuid.UUID) (io.Reader,
    error) {
       // ------------------------------------------------------------------------------
       var shares [][]byte = make([][]byte, 0)

       routeGetAuthorizations := fmt.Sprintf("%s/v1/api/requests/%v", e.serverAddr,
           documentID)
       routePutRequest := fmt.Sprintf("%s/v1/api/requests", e.serverAddr)
       routeDownload := fmt.Sprintf("%s/v1/api/%v", e.serverAddr, documentID)

       mdata, err := e.Metadata(ctx, documentID)

       if err != nil {
              return nil, err
       }

       // add in own key if it exists
       if mdata.Key != nil {
              share, err := e.user.Unseal(mdata.Owner, mdata.Key)

              if err != nil {
                     return nil, err
              }

              shares = append(shares, share)
       }

       req, _ := http.NewRequestWithContext(ctx, http.MethodGet, routeGetAuthorizations,
           nil)
       resp, err := e.Do(req)

       if err != nil {
              return nil, fmt.Errorf("failed␣to␣get␣request␣%v", err)
       }

       var authorizations []store.EscrowShare
       _ = json.NewDecoder(resp.Body).Decode(&authorizations)

       for _, auth := range authorizations {

              // decrypt shares
              share, err := e.user.Unseal(auth.User, auth.Key)

              if err != nil {
                     return nil, err
              }

              shares = append(shares, share)
       }

       if mdata.HeldShares < mdata.Threshold {
              r := api.AuthorizationRequest{
                     Message: "NA",
```

```
                Document: documentID,
        }

        body := new(bytes.Buffer)
        _ = json.NewEncoder(body).Encode(r)

        req, _ = http.NewRequestWithContext(ctx, http.MethodPut, routePutRequest,
            body)
        _, err = e.Do(req)

        if err != nil {
                return nil, err
        }

        return nil, ErrInsufficientShares(mdata.Threshold - mdata.HeldShares)
    }

    // ------------------------------------------------------------------------

    key, err := shamir.Combine(shares)
    if err != nil {
        log.Fatal(err)
    }

    req, _ = http.NewRequestWithContext(ctx, http.MethodGet, routeDownload, nil)
    resp, err = e.Do(req)

    if err != nil {
        return nil, fmt.Errorf("couldn't␣download␣object:␣%w", err)
    }

    buf := new(bytes.Buffer)
    _, err = encryption.Decrypt(buf, resp.Body, key)

    if err != nil {
        return nil, fmt.Errorf("couldn't␣decrypt␣object:␣%w", err)
    }

    return buf, nil
}
```

B.2.3 internal/client/errors.go

```
package client

import (
    "fmt"
)

// ErrInsufficientShares ..
type ErrInsufficientShares int

func (e ErrInsufficientShares) Error() string {
    return fmt.Sprintf("insufficient␣shares,␣missing␣%d␣shares", e)
```

```
}
```

B.2.4 internal/cmd/serve.go

```
package cmd

import (
	"log"
	"time"

	"github.com/minio/minio-go"
	"github.com/multilock/multilock/internal/debug"
	"github.com/multilock/multilock/internal/http"
	"github.com/multilock/multilock/internal/store"
	bolt "go.etcd.io/bbolt"
)

func init() {
	registerCommand("serve", Command{
		Function: (*Escrow).Serve,
		Help: "escrow␣serve",
	})
}

// Serve ..
func (e *Escrow) Serve(args ...string) {
	defer debug.Timing("Escrow::Serve", time.Now())

	boltdb, err := bolt.Open(e.internalConfig.DbLocation, 0666, &bolt.Options{Timeout:
	     1 * time.Second})

	if err != nil {
		log.Fatalln("error␣getting␣database:", err)
	}

	db := store.BoltDatastore{DB: boltdb}
	defer db.Close()

	mc, err := minio.New(
		e.internalConfig.StorageAddress,
		e.internalConfig.StorageKeyID,
		e.internalConfig.StorageAccessKey,
		false, // USE HTTPS
	)

	if err != nil {
		log.Fatalln("error␣getting␣minio␣client:", err)
	}

	router := http.Routes(db, mc)
	log.Fatal(http.NewServer(router, nil).ListenAndServe())
}
```

B.2.5 internal/cmd/secure.go

```
package cmd

import (
	"bytes"
	"context"
	"fmt"
	"log"
	"os"
	"strings"
	"time"

	"github.com/fatih/color"
	"github.com/multilock/multilock/internal/debug"
	"github.com/multilock/multilock/internal/user"
	"github.com/multilock/multilock/internal/utils"
	"github.com/multilock/multilock/internal/uuid"
)

func init() {
	registerCommand("secure", Command{
		Function: (*Escrow).Secure,
		Help: "escrow␣secure␣<filename>␣[<user>␣...␣<user>]",
		MinArgs: 1,
		Refresh: true,
	})
}

// Secure ..
func (e *Escrow) Secure(args ...string) {
	defer debug.Timing("Escrow::Secure", time.Now())

	var users []user.Identity
	for _, u := range args[1:] {
		userIdentity, err := e.ResolveUser(u)

		if err != nil {
			log.Fatalf("Failed␣to␣parse␣user␣%v␣with␣error␣%v\n", u, err)
		}

		users = append(users, userIdentity)
	}

	// Allow securing to the current user if there are no specified users
	if len(users) == 0 {
		users = append(users, e.client.Identity())
	}

	filename := args[0]

	file, err := os.Open(filename)
	defer file.Close()

	if err != nil {
		log.Fatalln(err)
```

```
	}

	documentID, err := e.client.Secure(context.TODO(), users, file, filename)

	if err != nil {
		log.Fatalln(err)
	}

	fmt.Println(formatSecuredFile(filename, documentID, users, e.client.Identity()))
}

func formatSecuredFile(filename string, documentID uuid.UUID, users []user.Identity,
    currUser user.Identity) string {
	threshold := len(users)
	owned := false
	usersString := make([]string, len(users))

	for i, user := range users {
		if bytes.Compare(user, currUser) == 0 {
			usersString[i] = color.GreenString("\t" + user.String())
			owned = true
		} else {
			usersString[i] = "\t" + user.String()
		}
	}

	return fmt.Sprintf(
		"%s:\t%v\nShare␣Owners␣(%d/%d):\n%s",
		color.HiWhiteString(filename),
		color.HiWhiteString(documentID.String()),
		utils.BoolToInt(owned), threshold,
		strings.Join(usersString, "\n"),
	)
}
```

B.2.6 internal/cmd/request.go

```
package cmd

import (
	"bytes"
	"context"
	"flag"
	"fmt"
	"log"
	"os"
	"time"

	"github.com/multilock/multilock/internal/client"
	"github.com/multilock/multilock/internal/debug"
)

func init() {
	registerCommand("request", Command{
```

```
                Function: (*Escrow).Request,
                Help: "escrow request <document id>",
                MinArgs: 1,
                Refresh: true,
        })
}

// Request ..
func (e *Escrow) Request(args ...string) {
        defer debug.Timing("Escrow::Request", time.Now())

        flags := flag.NewFlagSet("escrow ls", flag.ExitOnError)
        wait := flags.Bool("i", false, "exits immediately if not enough shares are aquired
            ")
        flags.Parse(args)

        docID, err := e.ResolveDocument(flags.Arg(0))

        if err != nil {
                log.Fatalln(err)
        }

        fmt.Println("Requesting: ", docID)

        buffer, err := e.client.Request(context.TODO(), docID)

        iteration := 0

        for err != nil {
                switch et := err.(type) {
                case client.ErrInsufficientShares:
                        if !*wait {
                                mdata, merr := e.client.Metadata(context.TODO(), docID)

                                if merr != nil {
                                        log.Fatalf("Failed to check metadata: %v", err)
                                }

                                if mdata.HeldShares < mdata.Threshold {
                                        fmt.Printf("\r%s\t%d/%d", spinner(iteration), mdata.
                                            HeldShares, mdata.Threshold)
                                        iteration++
                                        time.Sleep(500 * time.Millisecond)
                                } else {
                                        fmt.Printf("\r")
                                        buffer, err = e.client.Request(context.TODO(), docID)
                                }
                        } else {
                                fmt.Println(err.Error())
                                os.Exit(0)
                        }
                default:
                        log.Fatalf("Unexpected error (type %v) requesting: %v\n", et, err)
                }
```

```
	}

	buf := new(bytes.Buffer)
	buf.ReadFrom(buffer)

	fmt.Println(buf)
}

func spinner(i int) string {
	spinnerFrames := []string{"", "", "", "", "", "", "", "", "", ""}
	return spinnerFrames[i%len(spinnerFrames)]
}
```

B.2.7 internal/cmd/ls.go

```
package cmd

import (
	"context"
	"flag"
	"fmt"
	"log"
	"time"

	"github.com/fatih/color"
	"github.com/multilock/multilock/internal/debug"
	"github.com/multilock/multilock/internal/store"
	"github.com/multilock/multilock/internal/terminal"
	"github.com/multilock/multilock/internal/uuid"
)

func init() {
	registerCommand("ls", Command{
		Function: (*Escrow).Ls,
		Help: "escrow␣ls",
		Refresh: true,
	})
}

// Constant Symbols for better presentation
const (
	lock = ""
	unlock = ""
)

// Ls ..
func (e *Escrow) Ls(args ...string) {
	defer debug.Timing("Escrow::Ls", time.Now())

	flags := flag.NewFlagSet("escrow␣ls", flag.ExitOnError)
	long := flags.Bool("l", false, "outputs␣long␣form␣for␣all␣documents␣given")
	unique := flags.Bool("u", false, "outputs␣documents␣with␣unique␣strings␣for␣
	    filename␣user")
	flags.Parse(args)
```

```
	var documents []interface{}

	// NOTE: if no-refresh, this will be empty
	if flags.NArg() == 0 {
		documents = e.docTrie.PrefixMatch("")
	} else {
		documents = e.docTrie.PrefixMatch(flags.Arg(0))
	}

	var maxLength int
	var tmpstring string
	listdisplay := make([]string, 0)

	for _, document := range documents {
		switch d := document.(type) {
		case string:
			tmpstring = document.(string)
			docID, err := uuid.FromString(tmpstring)

			if err == nil {
				mdata, _ := e.client.Metadata(context.TODO(), docID)

				if *long {
					tmpstring = formatDocumentLong(docID, mdata)
				} else if *unique {
					tmpstring = getColorFormat(mdata)(mdata.Alias + ":" + e
					    .docTrie.UniquePrefix(tmpstring))
				} else {
					tmpstring = formatDocumentShort(docID, mdata)
				}
			}

			maxLength = max(maxLength, len(tmpstring))
			listdisplay = append(listdisplay, tmpstring)

		default:
			log.Println("Error, unexpected type", d, "for", document)
		}
	}

	w, _, err := terminal.GetSize()

	if err != nil {
		log.Fatalf("error getting terminal %v", err)
	}

	terminal.PrintColumns(
		listdisplay,
		w,
		maxLength+2,
		columnNumber(*long),
	)
}
```

```
func columnNumber(long bool) int {
	if long {
		return 1
	}

	return 0
}

func max(x, y int) int {
	if x >= y {
		return x
	}

	return y
}

func getColorFormat(mdata store.EscrowMetadata) func(string, ...interface{}) string {
	colorFunc := fmt.Sprintf

	if mdata.HeldShares >= mdata.Threshold {
		colorFunc = color.GreenString
	} else if mdata.HeldShares != 0 {
		colorFunc = color.YellowString
	} else {
		colorFunc = color.RedString
	}

	return colorFunc
}

func formatDocumentLong(documentID uuid.UUID, mdata store.EscrowMetadata) string {
	return getColorFormat(mdata)("%v\t%d/%d\t%s", documentID, mdata.HeldShares, mdata.
	    Threshold, mdata.Alias)
}

func formatDocumentShort(documentID uuid.UUID, mdata store.EscrowMetadata) string {
	if mdata.Alias != "" {
		return getColorFormat(mdata)("%v", mdata.Alias)
	}

	return getColorFormat(mdata)("%v", documentID)
}
```

B.2.8 internal/cmd/utils.go

```
package cmd

import (
	"context"
	"fmt"
	"log"
	"time"
```

```
	"github.com/multilock/multilock/internal/client"
	"github.com/multilock/multilock/internal/config"
	"github.com/multilock/multilock/internal/datastructures/trie"
	"github.com/multilock/multilock/internal/debug"
	"github.com/multilock/multilock/internal/keyring"
	"github.com/multilock/multilock/internal/user"
	"github.com/multilock/multilock/internal/uuid"
)

// Escrow command class
type Escrow struct {
	ring keyring.Storage
	client *client.EscrowClient
	docTrie *trie.Trie
	userTrie *trie.Trie
	internalConfig config.EscrowConfig
}

func (e *Escrow) refresh() {
	msgs, err := e.client.GetRequests(context.TODO())

	if err != nil {
		log.Printf("Failed␣to␣refresh:␣%v\n", err)
	} else {
		for _, msg := range msgs {
			fmt.Printf("%s␣requested␣%v\n", msg.From, msg.DocumentID)

			// NOTE: Add aliasing of users for users in message
			e.userTrie.Insert(msg.From)
		}
	}

	// DocumentID Completion
	doclist, err := e.client.Ls(context.TODO())

	if err != nil {
		log.Println(err)
	}

	for _, doc := range doclist {
		e.docTrie.Insert(doc.DocumentID.String())
	}
}

func loadIdentity(userStr string) *user.SecretIdentity {
	kr, err := keyring.NewOSKeyring()

	if err != nil {
		log.Fatalf("Failed␣to␣load␣keyring:␣%v", err)
	}

	seed, err := kr.Get(userStr)

	if err != nil || len(seed) != 32 {
```

```
                identity := user.NewSecretIdentity()
                err = kr.Set(userStr, identity.PrivateKey.Seed())
                return identity
        }

        return user.SecretIdentityFromSeed(seed)

}

// Load ..
func (e *Escrow) Load(config config.EscrowConfig, refresh bool) {
        defer debug.Timing("Escrow::Load", time.Now())
        var err error

        e.internalConfig = config

        // Setup Keyring
        e.ring, err = keyring.NewOSKeyring()

        if err != nil {
                log.Fatal(err)
        }

        e.client, err = client.NewEscrowClient(config.ServerAddress, loadIdentity(config.
            User))

        e.docTrie = trie.NewTrie()
        e.userTrie = trie.NewTrie()

        if !config.NoRefresh && refresh {
                e.refresh()
        }

        // Load local identities into doctrie
        identities, err := e.localIdentity()

        for k, v := range identities {
                identitystring := v.Identity().String()
                e.userTrie.InsertVal(k, identitystring)
                e.userTrie.InsertVal(identitystring, identitystring)
        }

}

// ResolveDocument takes a string which is some portion of a uuid, filename, or filename
    + uniqueness.
// It resolves this to a document id (uuid) in the order of uuid, partial uuid, filename,
     partial filename
func (e *Escrow) ResolveDocument(document string) (uuid.UUID, error) {
        // Resolve if UUID full
        documentID, err := uuid.FromString(document)

        if err == nil {
                return documentID, nil
```

```
	}

	// Resolve partial UUIDs
	possibleDocumentIDs := e.docTrie.PrefixMatch(document)

	if len(possibleDocumentIDs) == 1 {
		return uuid.FromString(possibleDocumentIDs[0].(string))
	} else if len(possibleDocumentIDs) > 1 {
		return uuid.Zero, fmt.Errorf("MultipleResults")
	}

	return uuid.Zero, fmt.Errorf("NotImplemented")
	// Resolve filenames

	// Resolve partial filenames
}

// ResolveUser takes a string which is some portion of an identity (base64 encoded) or a
    name to search locally
// It returns the matching identity or throws an error if not found
func (e *Escrow) ResolveUser(userid string) (user.Identity, error) {
	// Resolve partial base64 encoded string OR names
	possibleUsers := e.userTrie.PrefixMatch(userid)

	if len(possibleUsers) == 1 {
		userid = possibleUsers[0].(string)
		// Handle decoding later
	} else if len(possibleUsers) > 1 {
		return user.Identity{}, fmt.Errorf("MultipleResults")
	}

	// Resolve full base64 encoded string
	return user.IdentityFromString(userid)
}

// Close ..
func (e *Escrow) Close() {
	defer debug.Timing("Escrow::Close", time.Now())
	e.client.Close()
}
```

B.2.9 internal/cmd/main.go

```
package cmd

import (
	"fmt"
	"log"
	"os"
	"time"

	"github.com/multilock/multilock/internal/config"
	"github.com/multilock/multilock/internal/debug"
)
```

```
// Main ..
func Main(args []string) {
	defer debug.Timing("Main", time.Now())
	cfg := config.ParseConfig(args[1:])

	if cfg.NArg < 1 {
		fmt.Println("Usage: escrow <command>")
		for command := range registeredFunctions {
			fmt.Println("              ", command)
		}
		os.Exit(1)
	}

	client := Escrow{}
	defer client.Close()

	c, ok := registeredFunctions[cfg.Args[0]]

	if ok {
		if cfg.NArg < c.MinArgs+1 {
			fmt.Println(c.Help)
			os.Exit(1)
		}
		client.Load(cfg, c.Refresh)
		c.Function(&client, cfg.Args[1:]...)

	} else {
		log.Println("Unknown command:", cfg.Args[0])
	}

}

// CommandFunction ..
type CommandFunction func(*Escrow, ...string)

// Command ..
type Command struct {
	Function CommandFunction
	Help string
	Subcommands []string
	MinArgs int
	Refresh bool
}

var registeredFunctions map[string]Command

func registerCommand(name string, command Command) {
	if registeredFunctions == nil {
		registeredFunctions = make(map[string]Command)
	}
	registeredFunctions[name] = command
}
```

B.2.10 internal/cmd/authorize.go

```
package cmd

import (
	"context"
	"fmt"
	"log"
	"time"

	"github.com/fatih/color"
	"github.com/multilock/multilock/internal/debug"
)

func init() {
	registerCommand("authorize", Command{
		Function: (*Escrow).Authorize,
		Help: "escrow␣authorize␣<document␣identifier>␣<user>",
		MinArgs: 2,
		Refresh: true,
	})
}

// Authorize ..
func (e *Escrow) Authorize(args ...string) {
	defer debug.Timing("Escrow::Authorize", time.Now())

	docID, err := e.ResolveDocument(args[0])

	if err != nil {
		log.Fatalln(err)
	}

	fmt.Println(color.HiWhiteString("Authorize:␣%v", docID))

	for _, user := range args[1:] {
		pk, err := e.ResolveUser(user)

		if err != nil {
			fmt.Printf("Unknown␣user:␣%s\n", user)
			continue
		}

		err = e.client.Authorize(context.TODO(), docID, pk)

		if err != nil {
			log.Printf("Failed␣to␣authorize␣%v␣for␣%s␣user\n", docID, pk.Short()
			    )
			log.Println(err)
		} else {
			fmt.Printf("\t%v\n", pk)
		}
	}
```

```
}
```

B.2.11 internal/cmd/identity.go

```
package cmd

import (
	"fmt"
	"log"
	"time"

	"github.com/fatih/color"
	"github.com/multilock/multilock/internal/debug"
	"github.com/multilock/multilock/internal/user"
)

func init() {
	registerCommand("identity", Command{
		Function: (*Escrow).Identity,
		Help: "escrow␣identity",
		Refresh: false,
	})
}

// Identity ..
func (e *Escrow) Identity(args ...string) {
	defer debug.Timing("Escrow::Identity", time.Now())

	identities, err := e.localIdentity()

	if err != nil {
		log.Fatalln(err)
	}

	fmt.Println(color.HiWhiteString("Local␣Identities:"))
	for k, v := range identities {
		fmt.Printf("%-25s␣%v\n", color.GreenString(k), v.Identity())
	}

}

func (e *Escrow) localIdentity() (map[string]user.SecretIdentity, error) {
	var identityMap = make(map[string]user.SecretIdentity, 0)
	users, err := e.ring.List()

	if err != nil {
		return nil, err
	}

	for _, name := range users {
		seed, err := e.ring.Get(name)

		if err != nil {
			return identityMap, err
```

```
                }

                // De-serialize identities from keyring
                identity := user.SecretIdentityFromSeed(seed)
                identityMap[name] = *identity
        }

        return identityMap, nil
}
```

B.2.12 internal/http/router.go

```
package http

import (
        "fmt"

        "github.com/go-chi/chi"
        "github.com/go-chi/chi/middleware"
        "github.com/minio/minio-go"
        "github.com/multilock/multilock/internal/store"
        "github.com/multilock/multilock/internal/user"
)

const (
        APIVersionString = "v1"
        APIPrefix = "api"
)

// Routes ..
func Routes(db store.Datastore, mc *minio.Client) *chi.Mux {
        router := chi.NewRouter()

        router.Use(
                user.AuthenticateMiddleware,
                middleware.Logger,
                middleware.RedirectSlashes,
                middleware.Recoverer,
        )

        router.Route(fmt.Sprintf("/%s/%s", APIVersionString, APIPrefix), func(r chi.Router
            ) {
                r.Get("/documents", ListOwnedDocuments(db))
                r.Get("/documents/{documentID}", DocumentsMetadata(db))
                r.Get("/requests", ListRequests(db))
                r.Get("/requests/{documentID}", RequestStatus(db))

                r.Get("/{documentID}", FileDownloadHandler(mc))
                r.Post("/documents", MultipartEscrowHandler(db, mc))

                r.Post("/requests/{documentID}", AuthorizeHandler(db))
                r.Put("/requests", RequestHandler(db))
        })
```

```
	return router
}
```

B.2.13 internal/http/headers.go

```
package http

// Headers used by default in multilock
const (
	Authorization = "Authorization"
	ContentEncoding = "Content-Encoding"
	ContentLanguage = "Content-Language"
	ContentLength = "Content-Length"
	ContentMD5 = "Content-Md5"
	ContentRange = "Content-Range"
	ContentType = "Content-Type"
	Date = "Date"
)
```

B.2.14 internal/http/handlers.go

```
package http

import (
	"bytes"
	"encoding/json"
	"fmt"
	"io"
	"log"
	"net/http"

	"github.com/go-chi/chi"
	"github.com/go-chi/render"
	"github.com/minio/minio-go"
	"github.com/multilock/multilock/internal/store"
	"github.com/multilock/multilock/internal/user"
	"github.com/multilock/multilock/internal/utils"
	"github.com/multilock/multilock/internal/uuid"
)

// DummyHandler ..
func DummyHandler() http.HandlerFunc {
	return func(w http.ResponseWriter, r *http.Request) {
		fmt.Fprintln(w, "{}")
	}
}

// ListOwnedDocuments ..
func ListOwnedDocuments(db store.Datastore) http.HandlerFunc {
	return func(w http.ResponseWriter, r *http.Request) {
		user := r.Context().Value(user.UserContextKey).(user.Identity)
		render.JSON(w, r, db.ListDocuments(user))
	}
}
```

```
// ListRequests ..
func ListRequests(db store.Datastore) http.HandlerFunc {
	return func(w http.ResponseWriter, r *http.Request) {
		user := r.Context().Value(user.UserContextKey).(user.Identity)
		render.JSON(w, r, db.GetRequests(user))
	}
}

// DocumentsMetadata ..
func DocumentsMetadata(db store.Datastore) http.HandlerFunc {
	return func(w http.ResponseWriter, r *http.Request) {
		user := r.Context().Value(user.UserContextKey).(user.Identity)
		docID := chi.URLParam(r, "documentID")
		documentID, err := uuid.FromString(docID)

		if err != nil {
			http.Error(w, http.StatusText(404), 404)
			return
		}

		mdata := db.GetDocumentMetadata(documentID)
		mdata.Key = db.GetDocumentKey(user, documentID)
		mdata.HeldShares = db.CountAuthorizations(user, documentID) + utils.
		    BoolToInt(mdata.Key != nil)

		render.JSON(w, r, mdata)
	}
}

// RequestStatus ..
func RequestStatus(db store.Datastore) http.HandlerFunc {
	return func(w http.ResponseWriter, r *http.Request) {
		user := r.Context().Value(user.UserContextKey).(user.Identity)
		docID := chi.URLParam(r, "documentID")
		documentID, err := uuid.FromString(docID)

		if err != nil {
			http.Error(w, http.StatusText(404), 404)
		}

		render.JSON(w, r, db.GetAuthorizations(user, documentID))
	}
}

// AuthorizeHandler ..
func AuthorizeHandler(db store.Datastore) http.HandlerFunc {
	return func(w http.ResponseWriter, r *http.Request) {
		user := r.Context().Value(user.UserContextKey).(user.Identity)
		var resp AuthorizationResponse
		err := json.NewDecoder(r.Body).Decode(&resp)

		if err != nil {
			log.Println("err:␣", err)
			http.Error(w, http.StatusText(404), 404)
```

```
                return
        }

        docID := chi.URLParam(r, "documentID")
        documentID, err := uuid.FromString(docID)

        db.PutAuthorization(resp.To, documentID, store.EscrowShare{
                User: user,
                Key: resp.Key,
        })
    }
}

// RequestHandler ..
func RequestHandler(db store.Datastore) http.HandlerFunc {
    return func(w http.ResponseWriter, r *http.Request) {
        user := r.Context().Value(user.UserContextKey).(user.Identity)
        var req AuthorizationRequest
        err := json.NewDecoder(r.Body).Decode(&req)

        if err != nil {
                http.Error(w, http.StatusText(404), 404)
                return
        }

        documentMetadata := db.GetDocumentMetadata(req.Document)

        for _, owner := range documentMetadata.Users {
                if bytes.Compare(owner, user) != 0 {
                        db.PutRequest(user, owner, req.Document, req.Message)
                }
        }

    }
}

const (
    fileMultipartName = "file"
    shareMultipartName = "shares"
)

// MultipartEscrowHandler ..
func MultipartEscrowHandler(db store.Datastore, mc *minio.Client) http.HandlerFunc {
    return func(w http.ResponseWriter, r *http.Request) {
        owner := r.Context().Value(user.UserContextKey).(user.Identity)
        // TOOD: File Size?
        reader, err := r.MultipartReader()

        if err != nil {
                log.Println(err)
                http.Error(w, err.Error(), http.StatusBadRequest)
                return
        }
```

```
            documentID, _ := uuid.NewUUID()

            var fileMetadata store.EscrowMetadata

            for part, err := reader.NextPart(); err == nil; part, err = reader.NextPart
                () {
                   name := part.FormName()
                   if name == fileMultipartName {
                          filename := part.FileName()
                          fileMetadata.Alias = filename

                          _, err = mc.PutObject("multilock", documentID.String(), part,
                              -1, minio.PutObjectOptions{
                                 ContentType: "application/octet-stream",
                                 ContentEncoding: "base64",
                          })

                          if err != nil {
                                 log.Println(err)
                          }
                   } else if name == shareMultipartName {
                          var shares DocumentEscrowShares
                          err := json.NewDecoder(part).Decode(&shares)

                          if err != nil {
                                 log.Println(err)
                                 http.Error(w, err.Error(), http.StatusBadRequest)
                          }

                          var users []user.Identity
                          for _, share := range shares.Shares {
                                 users = append(users, share.User)
                                 db.PutDocumentKey(share.User, documentID, share.Key)
                          }

                          fileMetadata.Shares = uint(len(shares.Shares))
                          fileMetadata.Threshold = uint(len(shares.Shares))
                          fileMetadata.Users = users
                          fileMetadata.Owner = owner
                   }

                   part.Close()

            }

            db.PutDocumentMetadata(documentID, &fileMetadata)

            render.JSON(w, r, documentID)
      }
}

// FileDownloadHandler ..
func FileDownloadHandler(mc *minio.Client) http.HandlerFunc {
      return func(w http.ResponseWriter, r *http.Request) {
```

```
                docID := chi.URLParam(r, "documentID")
                documentID, err := uuid.FromString(docID)

                if err != nil {
                        log.Println(err)
                        http.Error(w, err.Error(), http.StatusBadRequest)
                        return
                }

                obj, err := mc.GetObjectWithContext(r.Context(), "multilock", documentID.
                    String(), minio.GetObjectOptions{})

                if err != nil {
                        log.Println(err)
                        http.Error(w, err.Error(), http.StatusBadRequest)
                        return
                }

                io.Copy(w, obj)
        }
}
```

B.2.15 internal/http/api.go

```
package http

import (
        "github.com/multilock/multilock/internal/store"
        "github.com/multilock/multilock/internal/user"
        "github.com/multilock/multilock/internal/uuid"
)

// AuthorizationResponse ..
type AuthorizationResponse struct {
        To user.Identity
        Key []byte
}

// AuthorizationRequest ..
type AuthorizationRequest struct {
        Document uuid.UUID
        Message string
}

// DocumentEscrowShares ..
type DocumentEscrowShares struct {
        Shares []store.EscrowShare
}
```

B.2.16 internal/http/server.go

```
package http

import (
        "crypto/tls"
```

```
	"net/http"
	"time"
)

// Server ..
type Server struct {
	http.Server
}

const (
	// DefaultMaxHeaderBytes - default maximum HTTP header size in bytes.
	DefaultMaxHeaderBytes = 1 << 20 // 1 MB
	// DefaultReadTimeout ..
	DefaultReadTimeout = 10 * time.Second
	// DefaultWriteTimeout ..
	DefaultWriteTimeout = 10 * time.Second
)

// Secure Go implementations of modern TLS ciphers
// The following ciphers are excluded because:
// - RC4 ciphers: RC4 is broken
// - 3DES ciphers: Because of the 64 bit blocksize of DES (Sweet32)
// - CBC-SHA256 ciphers: No countermeasures against Lucky13 timing attack
// - CBC-SHA ciphers: Legacy ciphers (SHA-1) and non-constant time
// implementation of CBC.
// (CBC-SHA ciphers can be enabled again if required)
// - RSA key exchange ciphers: Disabled because of dangerous PKCS1-v1.5 RSA
// padding scheme. See Bleichenbacher attacks.
var defaultCipherSuites = []uint16{
	tls.TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305,
	tls.TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305,
	tls.TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
	tls.TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
	tls.TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
	tls.TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
}

// Go only provides constant-time implementations of Curve25519 and NIST P-256 curve.
var secureCurves = []tls.CurveID{tls.X25519, tls.CurveP256}

// GetCertificateFunc ..
type GetCertificateFunc func(*tls.ClientHelloInfo) (*tls.Certificate, error)

// NewServer ..
func NewServer(handler http.Handler, getCert GetCertificateFunc) *Server {

	var tlsConfig *tls.Config

	if getCert != nil {
		tlsConfig = &tls.Config{
			PreferServerCipherSuites: true,
			CipherSuites: defaultCipherSuites,
			CurvePreferences: secureCurves,
			MinVersion: tls.VersionTLS12,
```

```
		}
		tlsConfig.GetCertificate = getCert
	}

	httpServer := &Server{
		// TOOD: What does our server need to store
	}

	httpServer.Addr = ":8000"
	httpServer.Handler = handler
	httpServer.TLSConfig = tlsConfig
	httpServer.ReadTimeout = DefaultReadTimeout
	httpServer.WriteTimeout = DefaultWriteTimeout
	httpServer.MaxHeaderBytes = DefaultMaxHeaderBytes

	return httpServer
}
```

B.2.17 internal/user/user.go

```
package user

import (
	"context"
	"crypto"
	"crypto/ed25519"
	"crypto/rand"
	"encoding/base64"
	"fmt"
	"log"
	"net/http"
	"strings"

	"golang.org/x/crypto/chacha20poly1305"
	"golang.org/x/crypto/curve25519"
)

// Constants ..
const (
	AuthenticationType = "PK-XED25519"
	shortIdentityLength = 16
)

// UserContextKey ..
var UserContextKey userContextKey = "pk-user-context-key"

type userContextKey string

// Identity ..
type Identity ed25519.PublicKey

func (identity Identity) String() string {
	return base64.StdEncoding.EncodeToString(identity)
}
```

```
// Short ..
func (identity Identity) Short() string {
	return identity.String()[:shortIdentityLength]
}

// IdentityFromString ..
func IdentityFromString(b64 string) (Identity, error) {
	var identity Identity
	identity, err := base64.StdEncoding.DecodeString(b64)

	if err != nil {
		return nil, err
	}

	return identity, nil
}

// Verify ..
func (identity Identity) Verify(message, signature []byte) bool {
	return ed25519.Verify(ed25519.PublicKey(identity), message, signature)
}

// SecretIdentity ..
type SecretIdentity struct {
	PrivateKey ed25519.PrivateKey
}

// NewSecretIdentity ..
func NewSecretIdentity() *SecretIdentity {
	var (
		identity SecretIdentity
		err error
	)

	_, identity.PrivateKey, err = ed25519.GenerateKey(rand.Reader)

	if err != nil {
		log.Fatalf("Failed to generate identity: %v", err)
	}

	return &identity
}

// SecretIdentityFromSeed ..
func SecretIdentityFromSeed(seed []byte) *SecretIdentity {
	var identity SecretIdentity
	identity.PrivateKey = ed25519.NewKeyFromSeed(seed)
	return &identity
}

// Identity ..
func (identity *SecretIdentity) Identity() Identity {
	var publicIdentity Identity = make([]byte, ed25519.PublicKeySize)
```

```
	copy(publicIdentity, identity.PrivateKey[32:])
	return publicIdentity
}

// Sign ..
func (identity *SecretIdentity) Sign(message []byte) ([]byte, error) {
	return identity.PrivateKey.Sign(rand.Reader, message, crypto.Hash(0))
}

// Authenticate ..
func (identity *SecretIdentity) Authenticate(req *http.Request) error {
	authString := AuthenticationString(identity)
	req.Header.Set("Authorization", authString)
	return nil
}

// Seal ..
func (identity *SecretIdentity) Seal(To Identity, msg []byte) ([]byte, error) {
	x25519Public := ed25519PublicKeyToCurve25519(ed25519.PublicKey(To))
	x25519Private := ed25519PrivateKeyToCurve25519(identity.PrivateKey)

	shared, err := curve25519.X25519(x25519Private, x25519Public)

	if err != nil {
		return nil, err
	}

	aead, err := chacha20poly1305.NewX(shared)

	if err != nil {
		return nil, err
	}

	// Decryption
	// Select a random nonce, and leave capacity for the ciphertext.
	nonce := make([]byte, aead.NonceSize(), aead.NonceSize()+len(msg)+aead.Overhead())

	if _, err := rand.Read(nonce); err != nil {
		return nil, err
	}

	// Encrypt message and append to nonce
	return aead.Seal(nonce, nonce, msg, nil), nil
}

// Unseal ..
func (identity *SecretIdentity) Unseal(From Identity, msg []byte) ([]byte, error) {
	x25519Public := ed25519PublicKeyToCurve25519(ed25519.PublicKey(From))
	x25519Private := ed25519PrivateKeyToCurve25519(identity.PrivateKey)

	shared, err := curve25519.X25519(x25519Private, x25519Public)

	if err != nil {
		return nil, err
```

```
	}

	aead, err := chacha20poly1305.NewX(shared)

	if err != nil {
		return nil, err
	}

	// Split nonce and ciphertext
	nonce, ciphertext := msg[:aead.NonceSize()], msg[aead.NonceSize():]

	plaintext, err := aead.Open(nil, nonce, ciphertext, nil)

	if err != nil {
		return nil, err
	}

	return plaintext, nil
}

// AuthenticationString generates an authentication header given a SecretIdentity. This
    is similar to Amazon's signatures
// and should be easy to use in most cases. It is chosen over OAuth because we ensure
    authentication and identity are tied
// to private keys by the user. This could easily be centralized using OAuth or other
    options.
// Authorization <AuthenticationType>
//   Credentials=<public key>,
//  Nonce=<128-bit nonce>
//  Signature=<signature bytes base64,
func AuthenticationString(identity *SecretIdentity) string {
	var nonce [128]byte
	n, err := rand.Read(nonce[:])

	if err != nil || n != 128 {
		log.Fatalf("Failed␣random␣read:␣%v", err)
	}

	message := make([]byte, 0)
	message = append(message, identity.Identity()...)
	message = append(message, nonce[:]...)

	signature, err := identity.Sign(message)

	if err != nil {
		log.Fatalf("Failed␣signing:␣%v", err)
	}

	return fmt.Sprintf(
		"%s␣Credentials=%s,␣Nonce=%s,␣Signature=%s",
		AuthenticationType,
		identity.Identity().String(),
		base64.StdEncoding.EncodeToString(nonce[:]),
		base64.StdEncoding.EncodeToString(signature),
```

```
	)
}

// ParseAuthenticationString ..
func ParseAuthenticationString(auth string) (Identity, bool) {
	parts := strings.Split(auth, "␣")

	if len(parts) != 4 {
		return nil, false
	}

	if parts[0] != AuthenticationType {
		return nil, false
	}

	// Credentials=<base64 public key>,
	var publicKey Identity
	publicKey, err := IdentityFromString(parts[1][12 : len(parts[1])-1])

	if err != nil {
		return nil, false
	}

	// Nonce=<base64 nonce(128 byte)>
	nonce, err := base64.StdEncoding.DecodeString(parts[2][6 : len(parts[2])-1])

	if err != nil {
		return nil, false
	}

	// Signature=<base64 signature>
	signature, err := base64.StdEncoding.DecodeString(parts[3][10:])

	if err != nil {
		return nil, false
	}

	message := make([]byte, 0)
	message = append(message, publicKey...)
	message = append(message, nonce[:]...)

	return publicKey, publicKey.Verify(message, signature)
}

// AuthenticateMiddleware ..
func AuthenticateMiddleware(next http.Handler) http.Handler {
	return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
		userAuth := r.Header.Get("Authorization")
		pk, ok := ParseAuthenticationString(userAuth)

		if !ok {
			w.WriteHeader(http.StatusUnauthorized)
			return
		}
```

```
                ctx := context.WithValue(r.Context(), UserContextKey, pk)

                next.ServeHTTP(w, r.WithContext(ctx))
        })
}
```

B.2.18 internal/user/crypto.go

```
package user

import (
        "crypto/ed25519"
        "crypto/sha512"
        "math/big"

        "golang.org/x/crypto/curve25519"
)

// Tentatively this should be provided eventually by golang.org/x/crypto
// see https://github.com/golang/go/issues/20504 for updates?
// Currently this code is pulled from the age project

var curve25519P, _ = new(big.Int).SetString("
    57896044618658097711785492504343953926634992332820282019728792003956564819949", 10)

func ed25519PublicKeyToCurve25519(pk ed25519.PublicKey) []byte {
        // ed25519.PublicKey is a little endian representation of the y-coordinate,
        // with the most significant bit set based on the sign of the x-coordinate.
        bigEndianY := make([]byte, ed25519.PublicKeySize)
        for i, b := range pk {
                bigEndianY[ed25519.PublicKeySize-i-1] = b
        }
        bigEndianY[0] &= 0b0111_1111

        // The Montgomery u-coordinate is derived through the bilinear map
        // u = (1 + y) / (1 - y)
        // See https://blog.filippo.io/using-ed25519-keys-for-encryption.
        y := new(big.Int).SetBytes(bigEndianY)
        denom := big.NewInt(1)
        denom.ModInverse(denom.Sub(denom, y), curve25519P) // 1 / (1 - y)
        u := y.Mul(y.Add(y, big.NewInt(1)), denom)
        u.Mod(u, curve25519P)

        out := make([]byte, curve25519.PointSize)
        uBytes := u.Bytes()
        for i, b := range uBytes {
                out[len(uBytes)-i-1] = b
        }

        return out
}

func ed25519PrivateKeyToCurve25519(pk ed25519.PrivateKey) []byte {
```

```
	h := sha512.New()
	h.Write(pk.Seed())
	out := h.Sum(nil)
	return out[:curve25519.ScalarSize]
}
```

B.2.19 internal/shamir/shamir.go

```
package shamir

import (
	"fmt"

	"github.com/corvus-ch/shamir"
)

// Split ..
func Split(secret []byte, parts int, threshold int) ([][]byte, error) {
	if threshold <= 0 {
		return nil, fmt.Errorf("can't split secret to threshold 0")
	} else if threshold == 1 {
		if parts != threshold {
			return nil, fmt.Errorf("Unimplemented: Threshold 1, parts %d", parts
			    )
		}

		return [][]byte{secret}, nil
	} else {
		shareMap, err := shamir.Split(secret, parts, threshold)

		if err != nil {
			return nil, err
		}

		shares := make([][]byte, 0)

		for n, share := range shareMap {
			shares = append(shares, append(share, n))
		}

		return shares, nil
	}
}

// Combine ..
func Combine(shares [][]byte) ([]byte, error) {
	if len(shares) <= 0 {
		return nil, fmt.Errorf("Can't combine 0 shares")
	} else if len(shares) == 1 {
		return shares[0], nil
	} else {
		shareMap := make(map[byte][]byte)

		for _, share := range shares {
```

```
                shareMap[share[len(share)-1]] = share[:len(share)-1]
        }

        secret, err := shamir.Combine(shareMap)

        if err != nil {
                return nil, err
        }

        return secret, nil
    }
}
```

B.2.20 internal/encryption/main.go

```
package encryption

import (
    "crypto/rand"
    "io"

    "github.com/minio/sio"
)

// KeySize is fixed at 256 bits
const KeySize = 32

// NOTE: Currently avoiding implementing file encryption/decryption
// using sio & DARE format. For now this is also fixed to CHACHA20_POLY1305

// Encrypt ..
func Encrypt(dst io.Writer, src io.Reader, key []byte) (int64, error) {
    return sio.Encrypt(dst, src, sio.Config{
        Key: key,
        CipherSuites: []byte{sio.CHACHA20_POLY1305},
    })
}

// Decrypt ..
func Decrypt(dst io.Writer, src io.Reader, key []byte) (int64, error) {
    return sio.Decrypt(dst, src, sio.Config{
        Key: key,
        CipherSuites: []byte{sio.CHACHA20_POLY1305},
    })
}

// GenerateKey ..
func GenerateKey() ([]byte, error) {
    key := make([]byte, KeySize)
    _, err := rand.Read(key)

    if err != nil {
        return nil, err
    }
```

```
	return key, nil
}
```

B.2.21 internal/debug/logging_debug.go

```
// +build pedantic

package debug

import (
	"fmt"
	"log"
	"time"
)

// Timing ..
func Timing(name string, start time.Time) {
	log.Printf("%s took %s\n", name, time.Since(start))
}

// Printf ..
func Printf(str string, args ...interface{}) {
	fmt.Printf(str, args...)
}
```

B.2.22 internal/debug/logging_release.go

```
// +build !pedantic

package debug

import (
	"time"
)

// Timing ..
func Timing(_ string, _ time.Time) {}

// Printf ..
func Printf(args ...string) {}
```

B.2.23 internal/config/config.go

```
package config

import "flag"

// EscrowConfig ..
type EscrowConfig struct {
	ServerAddress string
	StorageAddress string
	StorageKeyID string
	StorageAccessKey string
	DbLocation string
```

```
	// NOTE Should be extracted?
	NoRefresh bool
	User string
	NArg int
	Args []string
}

// ParseConfig ..
func ParseConfig(arguments []string) EscrowConfig {
	flags := flag.NewFlagSet("escrow", flag.ExitOnError)

	server := flags.String(
		"server",
		LookupEnvOrString("SERVER_ADDR", "localhost:8000"),
		"public-key␣server␣(address:port)",
	)

	storage := flags.String(
		"storage-addr",
		LookupEnvOrString("STORAGE_ADDR", "localhost:9000"),
		"s3/minio␣storage␣server␣(address:port)",
	)

	storageKeyID := flags.String(
		"keyid",
		LookupEnvOrString("KEY_ID", "minioadmin"),
		"storage␣server␣key␣id",
	)

	storageAccessKey := flags.String(
		"accesskey",
		LookupEnvOrString("ACCESS_KEY", "minioadmin"),
		"storage␣server␣access␣key",
	)

	dbLocation := flags.String(
		"db",
		LookupEnvOrString("DB_FILE", "/tmp/kvdb.bolt"),
		"file␣to␣use␣for␣loading␣bolt␣database",
	)

	user := flags.String(
		"user",
		LookupEnvOrString("PK_USER", "default-user"),
		"user-id␣to␣run␣the␣program",
	)

	norefresh := flags.Bool(
		"no-refresh",
		LookupEnvOrBool("PK_NOREFRESH", false),
		"don't␣refresh␣current␣data␣on␣load",
	)
```

```
	// NOTE: Handle errors?
	_ = flags.Parse(arguments)

	args := flags.Args()
	nargs := flags.NArg()

	return EscrowConfig{
		ServerAddress: *server,
		StorageAddress: *storage,
		StorageKeyID: *storageKeyID,
		StorageAccessKey: *storageAccessKey,
		DbLocation: *dbLocation,
		User: *user,
		NoRefresh: *norefresh,
		NArg: nargs,
		Args: args,
	}
}
```

B.2.24 internal/config/env.go

```
package config

import (
	"os"
	"strconv"
)

// LookupEnvOrString ..
func LookupEnvOrString(key string, standard string) string {
	if val, ok := os.LookupEnv(key); ok {
		return val
	}
	return standard
}

// LookupEnvOrBool ..
func LookupEnvOrBool(key string, standard bool) bool {
	if sval, ok := os.LookupEnv(key); ok {
		val, err := strconv.ParseBool(sval)
		if err != nil {
			return standard
		}
		return val
	}
	return standard
}
```

B.2.25 internal/keyring/keyring.go

```
package keyring

import "github.com/99designs/keyring"

// Storage ..
```

```
type Storage interface {
	Set(key string, value []byte) error
	Get(key string) ([]byte, error)
	Delete(key string) error
	List() ([]string, error)
}

// OSKeyring ..
type OSKeyring struct {
	ring keyring.Keyring
}

// NewOSKeyring ..
func NewOSKeyring() (*OSKeyring, error) {
	// NOTE: ServiceName cannot include '-' or '_' otherwise it will create duplicates
	// https://github.com/99designs/keyring/issues/44
	ring, err := keyring.Open(keyring.Config{
		ServiceName: "publiclock",
	})

	if err != nil {
		return nil, err
	}

	return &OSKeyring{ring}, nil
}

// Set ..
func (r *OSKeyring) Set(key string, value []byte) error {
	return r.ring.Set(keyring.Item{
		Key: key,
		Data: []byte(value),
	})
}

// Get ..
func (r *OSKeyring) Get(key string) ([]byte, error) {
	item, err := r.ring.Get(key)

	if err != nil {
		return []byte{}, err
	}

	return item.Data, nil
}

// Delete ..
func (r *OSKeyring) Delete(key string) error {
	return r.ring.Remove(key)
}

// List ..
func (r *OSKeyring) List() ([]string, error) {
	return r.ring.Keys()
```

```
}
```

B.2.26 internal/datastructures/trie/trie.go

```
// Trie is a trie of runes with string keys and interface{} values.
// Note that internal nodes have nil values so a stored nil value will not
// be distinguishable and will not be included in Walks.
/*
 * MinIO Cloud Storage, (C) 2014, 2015, 2016, 2017 MinIO, Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

// Package trie implements a simple trie tree for minio server/tools borrows
// idea from - https://godoc.org/golang.org/x/text/internal/triegen.
package trie

// Node trie tree node container carries value and children.
type Node struct {
	exists bool
	value interface{}
	child map[rune]*Node // runes as child.
}

// newNode create a new trie node.
func newNode() *Node {
	return &Node{
		exists: false,
		value: nil,
		child: make(map[rune]*Node),
	}
}

// Trie is a trie container.
type Trie struct {
	root *Node
	size int
}

// Root returns root node.
func (t *Trie) Root() *Node {
	return t.root
}
```

```
// Insert insert a key.
func (t *Trie) Insert(key string) {
	t.InsertVal(key, key)
}

// InsertVal ..
func (t *Trie) InsertVal(key, value string) {
	curNode := t.root
	for _, v := range key {
		if curNode.child[v] == nil {
			curNode.child[v] = newNode()
		}
		curNode = curNode.child[v]
	}

	if !curNode.exists {
		// increment when new rune child is added.
		t.size++
		curNode.exists = true
	}
	// value is stored for retrieval in future.
	curNode.value = value
}

// PrefixMatch - prefix match.
func (t *Trie) PrefixMatch(key string) []interface{} {
	node, _ := t.findNode(key)
	if node != nil {
		return t.Walk(node)
	}
	return []interface{}{}
}

// UniquePrefix - finds the shortest unique prefix for a full key
func (t *Trie) UniquePrefix(fullkey string) string {
	var index int
	curNode := t.root

	for k, v := range fullkey {
		if curNode.child[v] == nil {
			return fullkey
		}

		if len(curNode.child) > 1 {
			index = k + 1
		}

		curNode = curNode.child[v]
	}

	return fullkey[:index]
}

// Walk the tree.
```

```
func (t *Trie) Walk(node *Node) (ret []interface{}) {
	if node.exists {
		ret = append(ret, node.value)
	}
	for _, v := range node.child {
		ret = append(ret, t.Walk(v)...)
	}
	return
}

// find nodes corresponding to key.
func (t *Trie) findNode(key string) (node *Node, index int) {
	curNode := t.root
	f := false
	for k, v := range key {
		if f {
			index = k
			f = false
		}
		if curNode.child[v] == nil {
			return nil, index
		}
		curNode = curNode.child[v]
		if curNode.exists {
			f = true
		}
	}

	if curNode.exists {
		index = len(key)
	}

	return curNode, index
}

// NewTrie create a new trie.
func NewTrie() *Trie {
	return &Trie{
		root: newNode(),
		size: 0,
	}
}
```

B.2.27 internal/datastructures/trie/trie_test.go

```
/*
 * MinIO Cloud Storage, (C) 2016, 2017 MinIO, Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
```

```
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package trie

import (
	"testing"
)

// Simply make sure creating a new tree works.
func TestNewTrie(t *testing.T) {
	trie := NewTrie()

	if trie.size != 0 {
		t.Errorf("expected size 0, got: %d", trie.size)
	}
}

// Ensure that we can insert new keys into the tree, then check the size.
func TestInsert(t *testing.T) {
	trie := NewTrie()

	// We need to have an empty tree to begin with.
	if trie.size != 0 {
		t.Errorf("expected size 0, got: %d", trie.size)
	}

	trie.Insert("key")
	trie.Insert("keyy")

	// After inserting, we should have a size of two.
	if trie.size != 2 {
		t.Errorf("expected size 2, got: %d", trie.size)
	}
}

// Ensure that PrefixMatch gives us the correct two keys in the tree.
func TestPrefixMatch(t *testing.T) {
	trie := NewTrie()

	// Feed it some fodder: only 'minio' and 'miny-os' should trip the matcher.
	trie.Insert("minio")
	trie.Insert("amazon")
	trie.Insert("cheerio")
	trie.Insert("miny-o's")

	matches := trie.PrefixMatch("min")
	if len(matches) != 2 {
		t.Errorf("expected two matches, got: %d", len(matches))
	}
```

```
	if matches[0] != "minio" && matches[1] != "minio" {
		t.Errorf("expected␣one␣match␣to␣be␣'minio',␣got:␣'%s'␣and␣'%s'", matches
		    [0], matches[1])
	}
}
```

B.2.28 internal/uuid/uuid_test.go

```
package uuid

import (
	"bytes"
	"encoding/json"
	"testing"
)

// Ensure creating a UUID works
func TestNewUUID(t *testing.T) {
	id, err := NewUUID()

	if id == Zero {
		t.Error(err)
	}
}

func TestString(t *testing.T) {
	id := UUID{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
	idStr := "00010203-0405-0607-0809-0a0b0c0d0e0f"

	if id.String() != idStr {
		t.Errorf("recieved␣%v␣instead␣of␣%s", id, idStr)
	}

}

func TestFromString(t *testing.T) {
	truth := UUID{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
	str := "00010203-0405-0607-0809-0a0b0c0d0e0f"

	id, err := FromString(str)

	if err != nil {
		t.Error(err)
	}

	if id != truth {
		t.Errorf("%v␣doesn't␣match␣%v", id, truth)
	}

	_, err = FromString(str + "#")

	if err == nil {
		t.Errorf("decoded␣invalid␣string␣%s", str+"#")
```

```
	}

	_, err = FromString(str + "aa")

	if err == nil {
		t.Errorf("decoded␣wrong␣length")
	}

}

func TestFromByteSlice(t *testing.T) {
	truth := UUID{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
	slice := []byte{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

	id := FromByteSlice(slice)

	if id != truth {
		t.Errorf("%v␣doesn't␣match␣%v", id, truth)
	}
}

func TestJSON(t *testing.T) {
	truth := UUID{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

	var result UUID
	jsonBuf := bytes.NewBuffer(nil)

	errEncode := json.NewEncoder(jsonBuf).Encode(truth)

	if errEncode != nil {
		t.Errorf("failed␣to␣encode:␣%v", errEncode)
	}

	errDecode := json.NewDecoder(jsonBuf).Decode(&result)

	if errDecode != nil {
		t.Errorf("failed␣to␣decode:␣%v", errDecode)
	}

	if result != truth {
		t.Errorf("decoded␣%v␣doesn't␣match␣%v", result, truth)
	}
}
```

B.2.29 internal/uuid/uuid.go

```
package uuid

import (
	"crypto/rand"
	"encoding/hex"
	"encoding/json"
	"fmt"
	"strings"
```

```
)

// Length ..
const Length = 16

// UUID represents a uuid in an efficient method
type UUID [Length]byte

// Zero = 00000000-0000-0000-0000-000000000000
var Zero UUID = UUID{0}

// NewUUID generates a UUID
// Returns an error if rand.Read fails
func NewUUID() (UUID, error) {
	var u UUID
	n, err := rand.Read(u[:])

	if err != nil || n < Length {
		return Zero, fmt.Errorf("NewUUID␣Error:␣%w␣(n=%d)", err, n)
	}

	return u, nil
}

// FromString converts a UUID in hex encoded format, with '-' delimiters into a UUID
func FromString(u string) (UUID, error) {
	u = strings.Replace(u, "-", "", 4)
	slice, err := hex.DecodeString(u)

	if err != nil {
		return UUID{}, fmt.Errorf("FromString␣Error:␣%w", err)
	}

	if len(slice) != Length {
		return UUID{}, fmt.Errorf("FromString␣Error:␣bad␣length␣n=%d", len(slice))
	}

	var res UUID
	copy(res[:], slice)
	return res, nil
}

// FromByteSlice converts a byte slice into a UUID without emmiting any errors.
// If the byte slice is too short, it will be extended with zeros. If the byte slice
// is too long, it will be cut off at 16 chars
func FromByteSlice(slice []byte) UUID {
	uuid := UUID{0}
	copy(uuid[:], slice)
	return uuid
}

// String returns a UUID as a hex-encoded string in
// format ########-####-####-####-############
func (u UUID) String() string {
```

```
	result := hex.EncodeToString(u[:])
	return result[:8] + "-" + result[8:12] + "-" + result[12:16] + "-" + result[16:20]
		+ "-" + result[20:]
}

// MarshalJSON customizes the json expression representing a uuid
func (u UUID) MarshalJSON() ([]byte, error) {
	return json.Marshal(u.String())
}

// UnmarshalJSON customizes the json expression representing a uuid
func (u *UUID) UnmarshalJSON(data []byte) error {
	// Strip off quotes at start and end
	tmp, err := FromString(string(data[1 : len(data)-1]))
	copy(u[:], tmp[:])
	return err
}
```

B.2.30 internal/utils/utils.go

```
package utils

// BoolToInt ..
func BoolToInt(val bool) uint {
	if val {
		return 1
	}

	return 0
}
```

B.2.31 internal/store/datastore.go

```
package store

import (
	"bytes"
	"encoding/base64"
	"encoding/json"
	"log"

	"github.com/multilock/multilock/internal/user"
	libuser "github.com/multilock/multilock/internal/user"
	"github.com/multilock/multilock/internal/uuid"
	bolt "go.etcd.io/bbolt"
)

// Datastore ..
type Datastore interface {
	Close() error
	GetRequests(user user.Identity) []EscrowRequest
	PutRequest(user, owner user.Identity, documentID uuid.UUID, message string)
	ListDocuments(user user.Identity) []EscrowDocument
	GetDocumentKey(user user.Identity, documentID uuid.UUID) []byte
	PutDocumentKey(user user.Identity, documentID uuid.UUID, key []byte)
```

```
	GetDocumentMetadata(documentID uuid.UUID) EscrowMetadata
	PutDocumentMetadata(documentID uuid.UUID, mdata *EscrowMetadata)
	GetAuthorizations(user user.Identity, documentID uuid.UUID) []EscrowShare
	CountAuthorizations(user user.Identity, documentID uuid.UUID) uint
	PutAuthorization(user user.Identity, documentID uuid.UUID, share EscrowShare)
}

/* Schema Description for BoltDB
[USERS, REQUESTS, AUTHS] could be replaced by a message queue
	USERS
		<user>
			documentid:key
	REQUESTS
		<user>
			documentid|requesting_user:message
	AUTHS
		<user>
			documentid|sending_user:key
	DOCUMENTS
		documentid:info
*/

// BoltDatastore ..
type BoltDatastore struct {
	*bolt.DB
}

const (
	userBucket = "USERS"
	requestBucket = "REQUESTS"
	documentBucket = "DOCUMENTS"
	authorizationBucket = "AUTHS"
)

// GetRequests ..
func (db BoltDatastore) GetRequests(user user.Identity) []EscrowRequest {
	requests := make([]EscrowRequest, 0)

	err := db.View(func(tx *bolt.Tx) error {
		var b *bolt.Bucket
		if b = tx.Bucket([]byte(requestBucket)); b == nil {
			return nil
		}

		if b = b.Bucket([]byte(user)); b == nil {
			return nil
		}

		b.ForEach(func(key, value []byte) error {
			from := key[16:]
			documentID := key[:16]

			var ba *bolt.Bucket
```

```
                        // ignore authorizations which have already been updated
                        if ba = tx.Bucket([]byte(authorizationBucket)); ba != nil {
                                if ba = ba.Bucket([]byte(from)); ba != nil {
                                        val := ba.Get(bytes.Join([][]byte{documentID, user},
                                            nil))

                                        if val != nil {
                                                return nil
                                        }
                                }
                        }

                        requests = append(requests, EscrowRequest{
                                From: base64.StdEncoding.EncodeToString(from),
                                DocumentID: uuid.FromByteSlice(documentID),
                                Message: string(value),
                        })

                        return nil
                })
                return nil
        })
        if err != nil {
                log.Printf("db failed to get requests: %v", err)
                return nil
        }

        return requests
}

// PutRequest ..
func (db BoltDatastore) PutRequest(user user.Identity, owner user.Identity, documentID
    uuid.UUID, message string) {
        err := db.Update(func(tx *bolt.Tx) error {
                b, _ := tx.CreateBucketIfNotExists([]byte(requestBucket))
                b, _ = b.CreateBucketIfNotExists([]byte(owner))
                err := b.Put(bytes.Join([][]byte{documentID[:], []byte(user)}, nil), []byte
                    (message))
                return err
        })
        if err != nil {
                log.Printf("db failed to put requests: %v", err)
                return
        }
}

// ListDocuments ..
func (db BoltDatastore) ListDocuments(user user.Identity) []EscrowDocument {
        documentList := make([]EscrowDocument, 0)

        err := db.View(func(tx *bolt.Tx) error {
                var b *bolt.Bucket
                if b = tx.Bucket([]byte(userBucket)); b == nil {
                        return nil
```

```
		}

		if b = b.Bucket([]byte(user)); b == nil {
			return nil
		}

		b.ForEach(func(key, value []byte) error {
			documentList = append(documentList, EscrowDocument{
				DocumentID: uuid.FromByteSlice(key),
			})
			return nil
		})
		return nil
	})

	if err != nil {
		log.Printf("db failed to get document: %v", err)
		return nil
	}

	return documentList
}

// GetDocumentKey ..
func (db BoltDatastore) GetDocumentKey(user user.Identity, documentID uuid.UUID) []byte {
	var key []byte
	err := db.View(func(tx *bolt.Tx) error {
		var b *bolt.Bucket
		if b = tx.Bucket([]byte(userBucket)); b == nil {
			return nil
		}

		if b = b.Bucket([]byte(user)); b == nil {
			return nil
		}

		result := b.Get(documentID[:])

		if result != nil {
			key = make([]byte, len(result))
			copy(key, result)
		}

		return nil
	})

	if err != nil {
		log.Printf("db failed to get document: %v", err)
		return nil
	}

	return key
}
```

```
// PutDocumentKey ..
func (db BoltDatastore) PutDocumentKey(user user.Identity, documentID uuid.UUID, key []
    byte) {
       err := db.Update(func(tx *bolt.Tx) error {
              b, _ := tx.CreateBucketIfNotExists([]byte(userBucket))
              b, _ = b.CreateBucketIfNotExists([]byte(user))
              err := b.Put(documentID[:], key)
              return err
       })

       if err != nil {
              log.Printf("db␣failed␣to␣put␣document:␣%v", err)
              return
       }
}

// GetDocumentMetadata ..
func (db BoltDatastore) GetDocumentMetadata(documentID uuid.UUID) EscrowMetadata {
       var mdata *EscrowMetadata = &EscrowMetadata{}
       err := db.View(func(tx *bolt.Tx) error {
              if b := tx.Bucket([]byte(documentBucket)); b != nil {
                     metadata := b.Get(documentID[:])
                     _ = json.Unmarshal(metadata, mdata)
              }
              return nil
       })

       if err != nil {
              log.Printf("db␣failed␣to␣get␣document␣metadata:␣%v", err)
              return EscrowMetadata{}
       }

       return *mdata
}

// PutDocumentMetadata ..
func (db BoltDatastore) PutDocumentMetadata(documentID uuid.UUID, mdata *EscrowMetadata)
    {
       metadata, err := json.Marshal(mdata)

       if err != nil {
              log.Printf("failed␣to␣marshal␣metadata␣%v:␣%v", mdata, err)
       }

       err = db.Update(func(tx *bolt.Tx) error {
              b, _ := tx.CreateBucketIfNotExists([]byte(documentBucket))
              err := b.Put(documentID[:], metadata)
              return err
       })

       if err != nil {
              log.Printf("db␣failed␣to␣put␣document␣metadata:␣%v", err)
              return
       }
```

```
}

// GetAuthorizations ..
func (db BoltDatastore) GetAuthorizations(user user.Identity, documentID uuid.UUID) []
    EscrowShare {
       shareList := make([]EscrowShare, 0)

       err := db.View(func(tx *bolt.Tx) error {
              var b *bolt.Bucket
              if b = tx.Bucket([]byte(authorizationBucket)); b == nil {
                     return nil
              }

              if b = b.Bucket([]byte(user)); b == nil {
                     return nil
              }

              c := b.Cursor()

              prefix := documentID[:]
              for key, val := c.Seek(prefix); key != nil && bytes.HasPrefix(key, prefix);
                   key, val = c.Next() {
                     sendingUser := key[uuid.Length:]
                     shareList = append(shareList, EscrowShare{
                            User: libuser.Identity(sendingUser),
                            Key: val,
                     })
              }
              return nil
       })

       if err != nil {
              log.Printf("db␣failed␣to␣get␣authorizations:␣%v", err)
              return nil
       }

       return shareList
}

// CountAuthorizations ..
func (db BoltDatastore) CountAuthorizations(user user.Identity, documentID uuid.UUID)
    uint {
       var count uint = 0

       err := db.View(func(tx *bolt.Tx) error {
              var b *bolt.Bucket
              if b = tx.Bucket([]byte(authorizationBucket)); b == nil {
                     return nil
              }

              if b = b.Bucket([]byte(user)); b == nil {
                     return nil
              }
```

```
                c := b.Cursor()

                prefix := documentID[:]
                for key, _ := c.Seek(prefix); key != nil && bytes.HasPrefix(key, prefix);
                    key, _ = c.Next() {
                        count++
                }
                return nil
        })

        if err != nil {
                log.Printf("db failed to check authorizations: %v", err)
                return 0
        }

        return count
}

// PutAuthorization ..
func (db BoltDatastore) PutAuthorization(user user.Identity, documentID uuid.UUID, share
    EscrowShare) {
        err := db.Update(func(tx *bolt.Tx) error {
                b, _ := tx.CreateBucketIfNotExists([]byte(authorizationBucket))
                b, _ = b.CreateBucketIfNotExists([]byte(user))
                err := b.Put(bytes.Join([][]byte{documentID[:], []byte(share.User)}, nil),
                    share.Key)
                return err
        })

        if err != nil {
                log.Printf("db failed to put requests: %v", err)
                return
        }
}
```

B.2.32 internal/store/datatypes.go

```
package store

import (
        "github.com/multilock/multilock/internal/user"
        "github.com/multilock/multilock/internal/uuid"
)

// EscrowDocument ..
type EscrowDocument struct {
        DocumentID uuid.UUID
}

// EscrowRequest ..
type EscrowRequest struct {
        From string
        DocumentID uuid.UUID
        Message string
```

```
}

// EscrowMetadata ..
type EscrowMetadata struct {
	Alias string
	Users []user.Identity
	Owner user.Identity
	Threshold uint
	Shares uint
	Key []byte `json:",omitempty"`
	HeldShares uint `json:",omitempty"`
}

// EscrowShare ..
type EscrowShare struct {
	Key []byte
	User user.Identity
}
```

B.2.33 internal/terminal/terminal.go

```
package terminal

import (
	"fmt"
	"runtime"
	"strings"

	"golang.org/x/crypto/ssh/terminal"
)

// GetSize ..
func GetSize() (width int, height int, err error) {
	switch runtime.GOOS {
	case "windows":
		return 120, 28, nil
	default:
		return terminal.GetSize(0)
	}
}

// PrintColumns ..
// Will shorten strings if they exceded max width
func PrintColumns(vals []string, terminalWidth int, maxWidth int, columns int) {
	if columns <= 0 || (columns*maxWidth > terminalWidth) {
		columns = max(1, terminalWidth/maxWidth)
	}

	for n, str := range vals {
		if n != 0 && (n)%columns == 0 {
			fmt.Println()
		}
		fmt.Print(str[:min(len(str), maxWidth)] + strings.Repeat("␣", max(0,
		    maxWidth-len(str))))
```

```
	}

	fmt.Println()
}

func min(x, y int) int {
	if x <= y {
		return x
	}

	return y
}

func max(x, y int) int {
	if x >= y {
		return x
	}

	return y
}
```

B.2.34 docker-compose.yml

```
version: '3'
services:
  storage:
    image: minio/minio
    command: server /data
    ports:
      - 9000:9000
    volumes:
      - ./.data/storage:/data
    environment:
      MINIO_ACCESS_KEY: minioadmin
      MINIO_SECRET_KEY: minioadmin
```

B.2.35 go.mod

```
module github.com/multilock/multilock

go 1.13

require (
	github.com/99designs/keyring v1.1.5
	github.com/corvus-ch/shamir v1.0.0
	github.com/fatih/color v1.9.0
	github.com/go-chi/chi v4.1.1+incompatible
	github.com/go-chi/render v1.0.1
	github.com/go-ini/ini v1.56.0 // indirect
	github.com/minio/minio-go v6.0.14+incompatible
	github.com/minio/sio v0.2.0
	github.com/niemeyer/pretty v0.0.0-20200227124842-a10e7caefd8e // indirect
	github.com/smartystreets/goconvey v1.6.4 // indirect
	github.com/stretchr/testify v1.5.1 // indirect
	go.etcd.io/bbolt v1.3.4
```

```
	golang.org/x/crypto v0.0.0-20200510223506-06a226fb4e37
	golang.org/x/net v0.0.0-20200520004742-59133d7f0dd7 // indirect
	gopkg.in/check.v1 v1.0.0-20200227125254-8fa46927fb4f // indirect
	gopkg.in/ini.v1 v1.56.0 // indirect
	gopkg.in/yaml.v2 v2.3.0 // indirect
)
```

B.2.36 go.sum

```
github.com/99designs/keyring v1.1.5 h1:wLv7QyzYpFIyMSwOADq1CLTF9KbjbBfcnfmOGJ64aO4=
github.com/99designs/keyring v1.1.5/go.mod h1:7hsVvt2qXgtadGevGJ4ujg+u8m6SpJ5TpHqTozIPqf0
    =
github.com/corvus-ch/shamir v1.0.0 h1:k7Ijyhcdeel+eXpkz4lkKGO5yiyrYB0QL8lRrhfq4Gs=
github.com/corvus-ch/shamir v1.0.0/go.mod h1:yM+u+OT89j/Sx5Lynoy5b47HqlxhABr8p5NWvEW8H7o=
github.com/danieljoos/wincred v1.0.2 h1:zf4bhty2iLuwgjgpraD2E9UbvO+fe54XXGJbOwe23fU=
github.com/danieljoos/wincred v1.0.2/go.mod h1:SnuYRW9lp1oJrZX/
    dXJqr0cPK5gYXqx3EJbmjhLdK9U=
github.com/davecgh/go-spew v1.1.0/go.mod h1:J7Y8YcW2NihsgmVo/mv3lAwl/skON4iLHjSsI+c5H38=
github.com/davecgh/go-spew v1.1.1 h1:vj9j/u1bqnvCEfJOwUhtlOARqs3+rkHYY13jYWTU97c=
github.com/davecgh/go-spew v1.1.1/go.mod h1:J7Y8YcW2NihsgmVo/mv3lAwl/skON4iLHjSsI+c5H38=
github.com/dvsekhvalnov/jose2go v0.0.0-20180829124132-7f401d37b68a h1:mq+
    R6XEM6lJX5VlLyZIrUSP8tSuJp82xTK89hvBwJbU=
github.com/dvsekhvalnov/jose2go v0.0.0-20180829124132-7f401d37b68a/go.mod h1:7
    BvyPhdbLxMXIYTFPLsyJRFMsKmOZnQmzh6Gb+uquuM=
github.com/fatih/color v1.9.0 h1:8xPHl4/q1VyqGIPif1F+1V3Y3lSmrq01EabUW3CoW5s=
github.com/fatih/color v1.9.0/go.mod h1:eQcE1qtQxscV5RaZvpXrrb8Drkc3/DdQ+uUYCNjL+zU=
github.com/go-chi/chi v4.1.1+incompatible h1:MmTgB0R8Bt/jccxp+t6S/1VGIKdJw5J74CK/c9tTfA4=
github.com/go-chi/chi v4.1.1+incompatible/go.mod h1:eB3wogJHnLi3x/kFX2A+
    IbTBlXxmMeXJVKy9tTv1XzQ=
github.com/go-chi/render v1.0.1 h1:4/5tis2cKaNdnv9zFLfXzcquC9HbeZgCnxGnKrltBS8=
github.com/go-chi/render v1.0.1/go.mod h1:pq4Rr7HbnsdaeHagklXub+p6Wd16Af5l9koip1OvJns=
github.com/go-ini/ini v1.56.0 h1:6HjxSjqdmgnujDPhlzR4a44lxK3w03WPN8te0SoUSeM=
github.com/go-ini/ini v1.56.0/go.mod h1:ByCAeIL28uOIIG0E3PJtZPDL8WnHpFKFOtgjp+3Ies8=
github.com/godbus/dbus v0.0.0-20190726142602-4481cbc300e2 h1:ZpnhV/YsD2/4cESfV5+Hoeu/
    iUR3ruzNvZ+yQfO03a0=
github.com/godbus/dbus v0.0.0-20190726142602-4481cbc300e2/go.mod h1:bBOAhwG1umN6/6
    ZUMtDFBMQR8jRg9O75tm9K00oMsK4=
github.com/gopherjs/gopherjs v0.0.0-20181017120253-0766667cb4d1 h1:EGx4pi6eqNxGaHF6qqu48+
    N2wcFQ5qg5FXgOdqsJ5d8=
github.com/gopherjs/gopherjs v0.0.0-20181017120253-0766667cb4d1/go.mod h1:
    wJfORRmW1u3UXTncJ5qlYoELFm8eSnnEO6hX4iZ3EWY=
github.com/gsterjov/go-libsecret v0.0.0-20161001094733-a6f4afe4910c h1:6rhixN/
    i8ZofjG1Y75iExal34USq5p+wiN1tpie8IrU=
github.com/gsterjov/go-libsecret v0.0.0-20161001094733-a6f4afe4910c/go.mod h1:
    NMPJylDgVpX0MLRlPy15sqSwOFv/U1GZ2m21JhFfek0=
github.com/jtolds/gls v4.20.0+incompatible h1:xdiiI2gbIgH/gLH7ADydsJ1uDOEzR8yvV7C0MuV77Wo
    =
github.com/jtolds/gls v4.20.0+incompatible/go.mod h1:QJZ7F/aHp+rZTRtaJ1ow/lLfFfVYBRgL+9
    YlvaHOwJU=
github.com/keybase/go-keychain v0.0.0-20190712205309-48d3d31d256d h1:Z+RDyXzjKE0i2sTjZ/
    b1uxiGtPhFy34Ou/Tk0qwN0kM=
github.com/keybase/go-keychain v0.0.0-20190712205309-48d3d31d256d/go.mod h1:JJNrCn9otv/2
    QP4D7SMJBgaleKpOf66PnW6F5WGNRIc=
github.com/kr/pretty v0.1.0 h1:L/CwN0zerZDmRFUapSPitk6f+Q3+0za1rQkzVuMiMFI=
```

```
github.com/kr/pretty v0.1.0/go.mod h1:dAy3ld7l9f0ibDNOQOHHMYYIIbhfbHSm3C4ZsoJORNo=
github.com/kr/pty v1.1.1/go.mod h1:pFQYn66WHrOpPYNljwOMqo10TkYh1fy3cYio2l3bCsQ=
github.com/kr/text v0.1.0 h1:45sCR5RtlFHMR4UwH9sdQ5TC8v0qDQCHnXt+kaKSTVE=
github.com/kr/text v0.1.0/go.mod h1:4Jbv+DJW3UT/LiOwJeYQe1efqtUx/iVham/4vfdArNI=
github.com/mattn/go-colorable v0.1.4 h1:snbPLB8fVfU9iwbbo30TPtbLRzwWu6aJS6Xh4eaaviA=
github.com/mattn/go-colorable v0.1.4/go.mod h1:
    U0ppj6V5qS13XJ6of8GYAs25YV2eR4EVcfRqFIhoBtE=
github.com/mattn/go-isatty v0.0.8/go.mod h1:Iq45c/XA43vh69/j3iqttzPXn0bhXyGjM0Hdxcsrc5s=
github.com/mattn/go-isatty v0.0.11 h1:FxPOTFNqGkuDUGi3H/qkUbQO4ZiBa2brKq5r0l8TGeM=
github.com/mattn/go-isatty v0.0.11/go.mod h1:PhnuNfih5lzO57/f3n+odYbM4JtupLOxQOAqxQCu2WE=
github.com/minio/minio-go v6.0.14+incompatible h1:fnV+GD28LeqdN6vT2XdGKW8Qe/
    IfjJDswNVuni6km9o=
github.com/minio/minio-go v6.0.14+incompatible/go.mod h1:7
    guKYtitv8dktvNUGrhzmNlA5wrAABTQXCoesZdFQO8=
github.com/minio/sio v0.2.0 h1:NCRCFLx0r5pRbXf65LVNjxbCGZgNQvNFQkgX3XF4BoA=
github.com/minio/sio v0.2.0/go.mod h1:nKM5GIWSrqbOZp0uhyj6M1iA0X6xQzSGtYSaTKSCut0=
github.com/mitchellh/go-homedir v1.1.0 h1:lukF9ziXFxDFPkA1vsr5zpc1XuPDn/wFntq5mG+4E0Y=
github.com/mitchellh/go-homedir v1.1.0/go.mod h1:
    SfyaCUpYCn1Vlf4IUYiD9fPX4A5wJrkLzIz1N1q0pr0=
github.com/mtibben/percent v0.2.1 h1:5gssi8Nqo8QU/r2pynCm+hBQHpkB/uNK7BJCFogWdzs=
github.com/mtibben/percent v0.2.1/go.mod h1:KG9uO+SZkUp+VkRHsCdYQV3XSZrrSpR3O9ibNBTZrns=
github.com/niemeyer/pretty v0.0.0-20200227124842-a10e7caefd8e h1:
    fD57ERR4JtEqsWbfPhv4DMiApHyliiK5xCTNVSPiaAs=
github.com/niemeyer/pretty v0.0.0-20200227124842-a10e7caefd8e/go.mod h1:
    zD1mROLANZcx1PVRCS0qkT7pwLkGfwJo4zjcN/Tysno=
github.com/pmezard/go-difflib v1.0.0 h1:4DBwDE0NGyQoBHbLQYPwSUPoCMWR5BEzIk/f1lZbAQM=
github.com/pmezard/go-difflib v1.0.0/go.mod h1:iKH77koFhYxTK1pcRnkKkqfTogsbg7gZNVY4sRDYZ
    /4=
github.com/smartystreets/assertions v0.0.0-20180927180507-b2de0cb4f26d h1:zE9ykElWQ6/
    NYmHa3jpm/yHnI4xSofP+UP6SpjHcSeM=
github.com/smartystreets/assertions v0.0.0-20180927180507-b2de0cb4f26d/go.mod h1:
    OnSkiWE9lh6wB0YB77sQom3nweQdgAjqCqsofrRNTgc=
github.com/smartystreets/goconvey v1.6.4 h1:fv0U8FUIMPNf1L9lnHLvLhgicrIVChEkdzIKYqbNC9s=
github.com/smartystreets/goconvey v1.6.4/go.mod h1:syvi0/a8iFYH4r/RixwvyeAJjdLS9QV7WQ/
    tjFTllLA=
github.com/stretchr/objx v0.1.0/go.mod h1:HFkY916IF+rwdDfMAkV7OtwuqBVzrE8GR6GFx+wExME=
github.com/stretchr/objx v0.2.0 h1:Hbg2NidpLE8veEBkEZTL3CvlkUIVzuU9jDplZO54c48=
github.com/stretchr/objx v0.2.0/go.mod h1:qt09Ya8vawLte6SNmTgCsAVtYtaKzEcn8ATUoHMkEqE=
github.com/stretchr/testify v1.3.0 h1:TivCn/peBQ7UY8ooIcPgZFpTNSz0Q2U6UrFlUfqbe0Q=
github.com/stretchr/testify v1.3.0/go.mod h1:M5WIy9Dh21IEIfnGCwXGc5bZfKNJtfHm1UVUgZn+9EI=
github.com/stretchr/testify v1.5.1 h1:nOGnQDM7FYENwehXlg/kFVnos3rEvtKTjRvOWSzb6H4=
github.com/stretchr/testify v1.5.1/go.mod h1:5W2xD1RspED5o8YsWQXVCued0rvSQ+mT+I5cxcmMvtA=
go.etcd.io/bbolt v1.3.4 h1:hi1bXHMVrlQh6WwxAy+qZCV/SYIlqo+Ushwdpa4tAKg=
go.etcd.io/bbolt v1.3.4/go.mod h1:G5EMThwa9y8QZGBClrRx5EY+Yw9kAhnjy3bSjsnlVTQ=
golang.org/x/crypto v0.0.0-20181106171534-e4dc69e5b2fd/go.mod h1:6
    SG95UA2DQfeDnfUPMdvaQW0Q7yPrPDi9nlGo2tz2b4=
golang.org/x/crypto v0.0.0-20190308221718-c2843e01d9a2/go.mod h1:djNgcEr1/
    C05ACkg1iLfiJU5Ep61QUkGW8qpdssI0+w=
golang.org/x/crypto v0.0.0-20190701094942-4def268fd1a4 h1:
    HuIa8hRrWRSrqYzx1qI49NNxhdi2PrY7gxVSq1JjLDc=
golang.org/x/crypto v0.0.0-20190701094942-4def268fd1a4/go.mod h1:
    yigFU9vqHzYiE8UmvKecakEJjdnWj3jj499lnFckfCI=
golang.org/x/crypto v0.0.0-20200510223506-06a226fb4e37 h1:cg5LA/
    zNPRzIXIWSCxQW10Rvpy94aQh3LT/ShoCpkHw=
```

```
golang.org/x/crypto v0.0.0-20200510223506-06a226fb4e37/go.mod h1:
    LzIPMQfyMNhhGPhUkYOs5KpL4U8rLKemX1yGLhDgUto=
golang.org/x/net v0.0.0-20190311183353-d8887717615a/go.mod h1:
    t9HGtf8HONx5eT2rtn7q6eTqICYqUVnKs3thJo3Qplg=
golang.org/x/net v0.0.0-20190404232315-eb5bcb51f2a3/go.mod h1:
    t9HGtf8HONx5eT2rtn7q6eTqICYqUVnKs3thJo3Qplg=
golang.org/x/net v0.0.0-20200513185701-a91f0712d120 h1:EZ3cVSzKOlJxAd8e8YAJ7no8nNypTxexh/
    YE/xW3ZEY=
golang.org/x/net v0.0.0-20200513185701-a91f0712d120/go.mod h1:qpuaurCH72eLCgpAm/
    N6yyVIVM9cpaDIP3A8BGJEC5A=
golang.org/x/net v0.0.0-20200520004742-59133d7f0dd7 h1:AeiKBIuRw3UomYXSbLy0Mc2dDLfdtbT/
    IVn4keq83P0=
golang.org/x/net v0.0.0-20200520004742-59133d7f0dd7/go.mod h1:qpuaurCH72eLCgpAm/
    N6yyVIVM9cpaDIP3A8BGJEC5A=
golang.org/x/sys v0.0.0-20181107165924-66b7b1311ac8/go.mod h1:STP8DvDyc/dI5b8T5hshtkjS+
    E42TnysNCUPdjciGhY=
golang.org/x/sys v0.0.0-20190215142949-d0b11bdaac8a/go.mod h1:STP8DvDyc/dI5b8T5hshtkjS+
    E42TnysNCUPdjciGhY=
golang.org/x/sys v0.0.0-20190222072716-a9d3bda3a223/go.mod h1:STP8DvDyc/dI5b8T5hshtkjS+
    E42TnysNCUPdjciGhY=
golang.org/x/sys v0.0.0-20190412213103-97732733099d/go.mod h1:
    h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/sys v0.0.0-20190712062909-fae7ac547cb7/go.mod h1:
    h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/sys v0.0.0-20191026070338-33540a1f6037/go.mod h1:
    h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/sys v0.0.0-20200202164722-d101bd2416d5 h1:
    LfCXLvNmTYH9kEmVgqbnsWfruoXZIrh4YBgqVHtDvw0=
golang.org/x/sys v0.0.0-20200202164722-d101bd2416d5/go.mod h1:
    h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/sys v0.0.0-20200323222414-85ca7c5b95cd h1:xhmwyvizuTgC2qz7ZlMluP20uW+C3Rm0FD
    /WLDX8884=
golang.org/x/sys v0.0.0-20200323222414-85ca7c5b95cd/go.mod h1:
    h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/text v0.3.0 h1:g61tztE5qeGQ89tm6NTjjM9VPIm088od1l6aSorWRWg=
golang.org/x/text v0.3.0/go.mod h1:NqM8EUOU14njkJ3fqMW+pc6Ldnwhi/IjpwHt7yyuwOQ=
golang.org/x/tools v0.0.0-20190328211700-ab21143f2384 h1:TFlARGu6Czu1z7q93HTxcP1P+/ZFC/
    IKythI5RzrnRg=
golang.org/x/tools v0.0.0-20190328211700-ab21143f2384/go.mod h1:
    LCzVGOaR6xXOjkQ3onu1FJEFr0SW1gC7cKk1uF8kGRs=
gopkg.in/check.v1 v0.0.0-20161208181325-20d25e280405/go.mod h1:
    Co6ibVJAznAaIkqp8huTwlJQCZ016jof/cbN4VW5Yz0=
gopkg.in/check.v1 v1.0.0-20180628173108-788fd7840127 h1:qIbj1fsPNlZgppZ+VLlY7N33q108Sa+
    fhmuc+sWQYwY=
gopkg.in/check.v1 v1.0.0-20180628173108-788fd7840127/go.mod h1:
    Co6ibVJAznAaIkqp8huTwlJQCZ016jof/cbN4VW5Yz0=
gopkg.in/check.v1 v1.0.0-20200227125254-8fa46927fb4f h1:BLraFXnmrev5lT+xlilqcH8XK9/
    i0At2xKjWk4p6zsU=
gopkg.in/check.v1 v1.0.0-20200227125254-8fa46927fb4f/go.mod h1:
    Co6ibVJAznAaIkqp8huTwlJQCZ016jof/cbN4VW5Yz0=
gopkg.in/ini.v1 v1.56.0 h1:DPMeDvGTM54DXbPkVIZsp19fp/I2K7zwA/itHYHKo8Y=
gopkg.in/ini.v1 v1.56.0/go.mod h1:pNLf8WUiyNEtQjuu5G5vTm06TEv9tsIgeAvK8hOrP4k=
gopkg.in/yaml.v2 v2.2.2/go.mod h1:hI93XBmqTisBFMUTm0b8Fm+jr3Dg1NNxqwp+5A1VGuI=
gopkg.in/yaml.v2 v2.3.0 h1:clyUAQHOM3G0M3f5vQj7LuJrETvjVot3Z5el9nffUtU=
```

```
gopkg.in/yaml.v2 v2.3.0/go.mod h1:hI93XBmqTisBFMUTm0b8Fm+jr3Dg1NNxqwp+5A1VGuI=
```

B.2.37 README.md

```
# multi-lock

## Testing
‘go test ./...‘

## Tags (Notes)
   go build ./...
   go build -tags debug,pedantic ./...
   go run -tags debug,pedantic cmd/escrow.go

- **debug** (enables debug commands)
- **pedantic** (enables time logging and detailed debug)
```

B.2.38 LICENSE

```
                GNU GENERAL PUBLIC LICENSE
                   Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                        Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works. By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users. We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors. You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price. Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights. Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received. You must make sure that they, too, receive
```

or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those

subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met. This License explicitly affirms your unlimited
permission to run the unmodified Program. The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work. This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force. You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright. Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below. Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and

appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7. This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy. This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged. This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit. Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product

(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium
customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a
written offer, valid for at least three years and valid for as
long as you offer spare parts or customer support for that product
model, to give anyone who possesses the object code either (1) a
copy of the Corresponding Source for all the software in the
product that is covered by this License, on a durable physical
medium customarily used for software interchange, for a price no
more than your reasonable cost of physically performing this
conveying of source, or (2) access to copy the
Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the
written offer to provide the Corresponding Source. This
alternative is allowed only occasionally and noncommercially, and
only if you received the object code with such an offer, in accord
with subsection 6b.

d) Convey the object code by offering access from a designated
place (gratis or for a charge), and offer equivalent access to the
Corresponding Source in the same way through the same place at no
further charge. You need not require recipients to copy the
Corresponding Source along with the object code. If the place to
copy the object code is a network server, the Corresponding Source
may be on a different server (operated by you or a third party)
that supports equivalent copying facilities, provided you maintain
clear directions next to the object code saying where to find the
Corresponding Source. Regardless of what server hosts the
Corresponding Source, you remain obligated to ensure that it is
available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided
you inform other peers where the object code and Corresponding
Source of the work are being offered to the general public at no
charge under subsection 6d.

A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling. In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage. For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product. A product
is a consumer product regardless of whether the product has substantial

commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for

sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

  If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

  A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is

conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published

by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively

state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year> <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program> Copyright (C) <year> <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate
parts of the General Public License. Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<https://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs. If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library. If this is what you want to do, use the GNU Lesser General
Public License instead of this License. But first, please read
<https://www.gnu.org/licenses/why-not-lgpl.html>.