2009-08-19

# KiwiVault: Encryption Software for Portable Storage Devices

Trevor Bradshaw Florence
*Brigham Young University - Provo*

# KIWIVAULT - ENCRYPTION SOFTWARE FOR PORTABLE STORAGE DEVICES

by

Trevor B. Florence

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

December 2009

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Trevor B. Florence

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

| | |
|---|---|
| Date | Kent E. Seamons, Chair |
| Date | Dan R. Olsen Jr. |
| Date | Charles D. Knutson |

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Trevor B. Florence in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

| | |
|---|---|
| Date | Kent E. Seamons<br>Chair, Graduate Committee |

Accepted for the Department

| | |
|---|---|
| Date | Kent E. Seamons<br>Graduate Coordinator |

Accepted for the College

| | |
|---|---|
| Date | Thomas W. Sederberg<br>Associate Dean, College of Physical and Mathematical Sciences |

ABSTRACT

KIWIVAULT - ENCRYPTION SOFTWARE FOR PORTABLE STORAGE DEVICES

Trevor B. Florence

Department of Computer Science

Master of Science

While many people use USB flash drives, most do not protect their stored documents. Solutions for protecting flash drives exist but inherently limit functionality found in unprotected drives such as portability, usability, and the ability to share documents between multiple people. In addition, other drawbacks are introduced such as the possibility of losing access to protected documents if a password is lost. Assuming protecting portable documents is important, in order for people to be willing to protect their documents they should be required to make as few sacrifices in functionality as possible. We introduce KiwiVault, a USB flash drive encryption solution that retains more of the functionality found in unprotected storage devices than preceding solutions. In addition, this thesis reviews encryption solutions appropriate for portable data storage, reviews security components used by KiwiVault, discusses the design and implementation of KiwiVault, discusses a user study and threat analysis conducted to validate KiwiVault as a solution, and proposes future work.

# ACKNOWLEDGMENTS

I would like to thank my wife Lindsay and son Calvin for their support and love, Dr. Seamons for his encouragement and guidance, Tim for his advice and inventing the security components KiwiVault is built on, the Internet Security Research Lab for their feedback in writing this thesis, and my committee for providing feedback throughout the thesis process.

# Contents

# Chapter 1

## Introduction

Computer hardware becomes less expensive and more powerful year after year. This trend can be seen in portable storage which has progressed from magnetic tape, to floppy disks, to Zip disks, and more recently to USB flash drives. It is now very common for people to carry around several gigabytes of data with them. With so much portable information, protecting data from loss and theft is more important than ever. Several security solutions for protecting flash drives exist that add protection through either added software or specialized hardware. All security solutions inherently limit functionality found in unprotected drives such as portability, usability, and the ability to share documents between multiple people. In addition, other drawbacks are introduced such as the possibility of losing access to protected documents if a password is lost. In order for people to be willing to protect their flash drives they should be required to make as few sacrifices in functionality as possible.

We introduce KiwiVault, a USB flash drive encryption solution that retains more of the functionality found in unprotected storage devices than preceding solutions: it is portable across any machine with Java installed, it is usable to USB flash drive users because its interface is familiar and functions like an explorer window, it provides encrypted document recovery when users forget their password, and it allows users to share their encrypted documents with others using their email addresses. Encrypted document sharing is possible thanks to the system KiwiVault is built on: Kiwi, an identity-based symmetric key encryption system which allows for a user to

encrypt documents for themselves and for others using symmetric keys derived by and delivered from a trusted key server.

## 1.1 Motivation

The following motivating scenarios help establish use cases for KiwiVault:

- Alice is a student who stores her personal information such as scholastic records, bank statements, and a personal journal on her USB flash drive. Because she carries her flash drive everywhere, she is concerned that it might be stolen or lost. Alice decides she wants to encrypt her sensitive documents.
- Bob is traveling to San Francisco, California for business with his coworkers Carol and Dan. They are finishing up a sensitive presentation stored on Bob's flash drive. To protect the presentation in case of theft, Bob encrypted it so that only he and his coworkers can decrypt and edit it.
- Ethan is a student who stores his personal information on his USB flash drive and also uses it to transfer documents between computers. Fiona asks Ethan if she can borrow his flash drive to copy some documents from one computer to another. Ethan consents but decides to encrypt his personal documents first so that Fiona will not be able to access them accidentally or otherwise.
- Genevieve does not own a computer and uses public computers to do her work. She stores all of her documents and work on her USB flash drive. Genevieve would like to encrypt her documents but is concerned that she will not be able to install the required software on public computers.
- Harold has a patchy long-term memory and so he stores his banking and credit card information on his USB flash drive. He wants to encrypt these sensitive documents but does not want to lose access to them if he forgets his password.

## 1.2 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 reviews existing encryption solutions for protecting computer documents. Chapter 3 reviews security components used by KiwiVault. Chapter 4 discusses the design of KiwiVault. Chapter 5 describes the first implementation of KiwiVault. Chapter 6 discusses a user study and threat analysis conducted to validate KiwiVault. Chapter 7 summarizes the conclusions and discusses potential future work.

# Chapter 2

## Related Work

This chapter discusses several portable security solutions for protecting computer documents comprising Sections 2.1 through 2.6. These solutions range from software and hardware to online solutions. Afterwards, a comparison of the solutions to KiwiVault is presented in Section 2.7.

### 2.1 Voltage SecureFile

Voltage SecureFile [7] has many characteristics similar to KiwiVault. Voltage sells security products which use Identity-Based Encryption (see Section 2.1.1). SecureFile allows for Windows users to encrypt and share documents using email addresses as identifiers. The program is integrated into Windows Explorer and is operated through the right-click file context menu. It also provides advanced features for Microsoft Office 2007 documents such as permissions for editing.

One major difference is that this system is not specifically designed to protect USB flash drives. It is primarily designed to encrypt documents on a user's local computer or on a shared network. SecureFile requires administrative privileges to install which limits its use on public computers. In contrast, KiwiVault runs as an executable Jar file from the USB flash drive which does not require administrative privileges when Java is already installed on the system.

### 2.1.1 Identity-Based Encryption

Identity-Based Encryption (IBE) was originally proposed by Adi Shamir [14]. IBE provides similar features as public-key cryptography (encrypting and signing files) but instead of using a random key as a public key, it utilizes user identifiers to derive public keys and delivers private keys to users from a Private Key Generator (PKG) server. At the time, Shamir did not know how to implement such a system but provided the requirements for creating one. He believed it would only be a matter of time before a suitable solution was discovered. One of the key advantages of IBE is that public keys can be generated for any user off-line using information published by the PKG.

It took seventeen years for the first practical implementation of IBE to be discovered by Dan Boneh and Matt Franklin [1]. Their solution is based on the Weil pairing whose specifics are beyond the scope of this thesis. This solution allows for all public keys to be generated by any user and is the IBE system used by Voltage SecureFile.

## 2.2 SecuTok

SecuTok [4] is an encryption suite designed for businesses who wish to allow their employees to transfer sensitive documents in a portable, cost efficient, and auditable manner. SecuTok provides secure document storage on USB flash drives. Encrypted files on SecuTok are either audited or unaudited. Audited encrypted files store the initialization vector (IV) used for encryption on a server whereas unaudited files store the IV on the device itself. To decrypt an audited file, the IV must be requested from the server. Every time the IV is sent to or requested from the server, information about the machine making the request is stored on the server, this way the server knows about all encryption and decryption events associated with a specific audited

file. The goal of file auditing is to have documentation of when the company's files are encrypted and decrypted. If the company becomes aware of a data breach or other problem, these records can serve as a starting point for investigation. The suite includes several different applications for performing its tasks: an administration application for setting up protected devices, a client to encrypt and decrypt files on the protected device, a server to store audited encrypted file information as well as a log of encryption and decryption events, and a web application for viewing information about active devices along with their logs.

## 2.3 TrueCrypt

TrueCrypt [5] is an open source cross-platform solution which allows its users to create encrypted virtual drives. Because TrueCrypt is installed on the user's computer, it allows for seamless "on-the-fly-encryption," which means users can edit, copy, and move files within, to, and from the encrypted volume as if it were a regular drive. TrueCrypt encrypts the volume based on a user-supplied strong password. It has many advanced features such as only storing decrypted data in RAM, hidden volumes, and even hidden operating systems. TrueCrypt works very well as a local encryption system, but as a portable encryption system it has some limitations. There is no built-in support for sharing encrypted files with other users, and it is complicated to set up an encrypted volume on a flash drive that works with multiple operating systems.

## 2.4 Folder and File Locking Programs

Most USB flash drive users are not targets for data theft via physical theft. If they are robbed or lose their drive, they most likely only need to protect their data from non-technical thieves. With this in mind, folder and file locking programs have been

developed such as Folder Lock [11] which provides locking or encryption. These systems essentially make files or folders temporarily unreadable, unwritable, and hidden. For the user who is only concerned about protecting their documents from naive computer users, this may be sufficient. Positive aspects of file locking systems are that they are fast (because they do not perform encryption) and they are portable. Negative aspects include that they generally protect locked files using a password which limits sharing and data recovery and that locking only works on the operating system it was performed on.

## 2.5 Hardware-Encrypted USB Flash Drives

Several hardware-based encryption systems exist (e.g., IronKey [8], DataTraveler Vault [2], and SanDisk Cruzer Professional [3]). These devices are essentially USB flash drives with added hardware for performing file encryption and storing encryption keys. Generally, they require a master password (strong or weak) for accessing the drive. Because these systems are hardware based, they can prevent brute force password guessing attacks by self-destructing the device when too many bad passwords are entered. In addition, these systems are portable and accessible from any operating system as the encryption is hardware-based. Currently these devices are significantly more expensive than unencrypted drives (roughly four times the price) and do not allow sharing encrypted files without sharing the drive password.

## 2.6 iDataGuard

iDataGuard [9] allows its users to store encrypted files online. This system, in the form of client-side software, interacts with various online data storage providers and creates a unified interface for interacting with each service. All of the stored files are encrypted on the user's computer before they are stored in remote storage. iData-

Guard interacts with multiple storage providers by utilizing an abstract service model for communication which can be extended to support new providers.

To use the program, users either have to download a copy and install it on each computer they want to use it on or install it on a USB flash drive which they travel with. Also, because all of the files are stored remotely, latency can be an issue. Lastly, iDataGuard has no mechanism for sharing encrypted documents with other users.

## 2.7 Comparison

Table 2.1 shows a comparison of the security programs related to KiwiVault in terms of their features. The features compared are:

**Runs on Windows** Whether the program runs on Microsoft Windows. In its collective versions, Windows controls roughly 90% of the computer market share.

**Runs on Non-Windows OS** Whether the program runs on any operating systems apart from Windows such as Mac OS X and Linux.

**Runs from flash drive** Whether the program runs directly from the flash drive. Running from the flash drive implies that a user can run the program on different machines without requiring them to install the program again.

**Provides Encryption** Whether the program encrypts the files it is protecting.

**Provides Sharing** Whether the program allows a user to share their files with others without divulging their password or other secret.

**Provides Document Recovery** Whether the program provides a mechanism for recovering ones files after the password used to protect them has been lost.

| | Secure File | Secu-Tok | True-Crypt | File Lock | Hardware Encrypted | iData-Guard | Kiwi-Vault |
|---|---|---|---|---|---|---|---|
| Runs on Windows | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Runs on Non-Windows OS | | | ✓ | | ✓ | ✓ | ✓ |
| Runs from Flash Drive | | ✓ | | | ✓ | | ✓ |
| Provides Encryption | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ |
| Provides Sharing | ✓ | | | | | | ✓ |
| Provides Document Recovery | ✓ | | | ✓ | | | ✓ |
| Installs w/o Admin Rights | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Integrates with Explorer | ✓ | ✓ | ✓ | | ✓ | | ✓* |

Table 2.1: A comparison of security programs related to KiwiVault in terms of their features. The systems are as follows: SecureFile, SecuTok, TrueCrypt, Folder and File Locking Programs, Hardware-Encrypted USB Flash Drives, iDataGuard, and KiwiVault. *Closely approximates the experience of a file explorer.

**Installs w/o Admin Rights** Whether the program can be installed if the user does not have administrative privileges.

**Integrates with Explorer** Whether the program is managed in the file explorer or through a different interface. While KiwiVault does not run from the file explorer, its interface is designed to look and behave like it.

# Chapter 3

## Foundations

This chapter discusses the technical and cryptographic foundations KiwiVault is built upon. The four main foundations discussed are: Source-Specific Symmetric Key Exchange (3SKE), Simple Authentication for the Web (SAW), and Password-Based Key Derivation Function 2 (PBKDF2).

### 3.1 3SKE

Source-Specific Symmetric Key Exchange[1] (3SKE) is a key distribution solution for symmetric keys. The key distribution problem of symmetric keys has to do with the requirement for users to establish a shared secret key before communicating. 3SKE addresses this problem by having users obtain secret keys from a third-party key server and, in some cases, derive them off-line. This allows for users to establish a shared secret without having to first contact the other user. 3SKE is made up of three components:

1. **Authentication server**: Authenticates clients and issues authorization tokens that enable them to request keys from the key server.
2. **Key server**: Derives and delivers creator and viewer keys to authorized clients.
3. **Client**: Automates the required interactions with the authentication and key servers and uses the creator and viewer keys to create and view encrypted files.

[1] 3SKE is unpublished work by Timothy W. van der Horst.

### 3.1.1 Authentication Server

Authentication servers authenticate users and provide them with an authorization token AT for use with the key server. There are no specific requirements for how authentication servers should validate their users; they can be implemented to work with any form of authentication desired. AT is calculated by the authentication server using the following function:

$$\text{AT} \leftarrow f(\kappa_{\text{AS}}, id_u, \tau) = \text{KDF}_{\kappa_{\text{AS}}}(id_u \| \tau), \tag{3.1}$$

where KDF is a key deriving function, $\kappa_{\text{AS}}$ is the server's unique master key which is also registered with a key server, $id_u$ is the identity of the client's user, and $\tau$ is the time period this authorization token is valid for.

### 3.1.2 Key Server

Each key server has a unique and secret master key $\kappa_{\text{KS}}$. The key server uses $\kappa_{\text{KS}}$ to derive creator keys. If $\kappa_{\text{KS}}$ is stolen, the security of the entire system (including all encrypted messages) is compromised. For this reason, $\kappa_{\text{KS}}$ must be protected on a secure server (e.g., on secure cryptographic hardware).

Key servers derive creator keys and viewer keys for authenticated users using their master key. Key servers are stateless in that they do not need to keep track of the requests made to them. Their storage requirements are small; they store their key server master key and an index of authentication servers they honor containing its identifier (e.g., URL) and authentication server master key. They inherently provide key escrow services because all creator and viewer keys can be generated by the key server.

When a client makes a request to the key server, they send AT along with the time period $\tau$, $id_u$, and the identifier of the authentication server who issued the token.

The key server looks up $\kappa_{\text{AS}}$ using the authentication server URL and calculates AT using Equation 3.1. If the calculated AT and the client-supplied AT match, the key server knows the user has been authenticated and performs the request. Otherwise, an invalid authorization token error is returned to the client. Clients can request any number of creator or viewer keys specific to a time period $\tau$.

**Creator Keys** Clients and key servers use creator keys $\kappa_c$ to derive viewer keys. Each $\kappa_c$ is specific to a creator denoted by an identifier $id_c$. A creator is someone encrypting a message or document for themselves or to share with other viewers. $\kappa_c$ is calculated by the key server using the following function:

$$\kappa_c \leftarrow f(\kappa_{\text{KS}}, id_c, \tau) = \text{KDF}_{\kappa_{\text{KS}}}(id_c \| \tau), \tag{3.2}$$

where KDF is a key deriving function, $\kappa_{\text{KS}}$ is the key server's master key, $id_c$ is the identity of the creator, and $\tau$ is the time period $\kappa_c$ is valid for. The key period limits the damage of losing $\kappa_c$ or the viewer keys derived by $\kappa_c$. Because Equation 3.2 requires $\kappa_{\text{KS}}$, only the key server can derive creator keys.

**Viewer Keys** A viewer key $\kappa_{c,v}$ is the shared secret between creator and viewer users. $\kappa_{c,v}$ is "source-specific" which means that each viewer has a different $\kappa_{c,v}$ for each creator. $\kappa_{c,v}$ is calculated by the key server as well as creators using the following function:

$$\kappa_{c,v} \leftarrow f(\kappa_c, id_v) = \text{KDF}_{\kappa_c}(id_v) \tag{3.3}$$

where KDF is a key derivation function, $\kappa_c$ is the creator key, and $id_v$ is the identity of the viewer.

### 3.1.3 Client

Clients allow users to use the viewer keys from the 3SKE system to encrypt and decrypt messages. Clients allow users to participate as creators and viewers.

Creators share encrypted messages with other viewers. They can request their creator key from the key server, generate viewer keys for any viewer, encrypt messages using a randomly generated message key, encrypt the message key for each viewer using their respective viewer key, and share the encrypted message along with the encrypted message key to the viewers.

Viewers decrypt messages they receive from creators. They can request their viewer key (specific to a creator) from the key server, decrypt the message key using their viewer key, and decrypt the message using the message key.

Practically, users are both creators and viewers.

## 3.2 SAW

Simple Authentication for the Web (SAW) [15] is a system for authenticating users via email. In traditional email based authentication, after a user indicates their email address, a service sends an activation link to the user's email to prove they have access to the indicated email account. SAW builds on this idea by creating an additional required component to authenticate the user. In SAW, when a service wishes to authenticate a user to an email address, the service first creates a random value which is split into two tokens; one token is set as a cookie in the user's browser while the other token is sent to the user's email account. To authenticate, the user goes to their email account and clicks on an authentication link in their email which contains the token as a GET parameter in the URL or the link. Both tokens are then sent to the authentication server (one as a cookie and the other as a GET parameter). The

server combines the tokens and verifies that it matches the original token to authorize the account.

## 3.3 PBKDF2

Systems where user-entered passwords need to be used as cryptographic keys require a key derivation function. The RSA specification PKCS #5 v2.1 [10] describes two such functions: PBKDF1 and PBKDF2 (Password-Based Key Derivation Function), the latter of which is recommended. PBKDF2 creates keys using a pseudorandom number generator and takes as input the user-entered password, a salt, an iteration count, and an output key length. The salt helps prevent cached password attacks where all possible passwords and their corresponding keys are cached and the iteration count makes brute-force attacks more expensive.

# Chapter 4

## Design

## 4.1 Design Goals

KiwiVault is designed with the following design traits in mind: usability, portability, and security.

### 4.1.1 Usability

KiwiVault is designed for people who use portable storage devices such as USB flash drives. Because it cannot be assumed that all users in this group understand computer security, KiwiVault does not use encryption terms in its interface. For example, users are not exposed to encryption keys and they do not have to manually choose which documents within KiwiVault should be encrypted. Instead, all files within KiwiVault are encrypted and decrypted transparently to the user.

To further help make the program usable, KiwiVault is designed to look and feel like a drive window which users can interact with as they would other desktop windows. Features like drag and drop, keyboard shortcuts, and copy and paste are supported in KiwiVault, just as they are in regular drives. It is designed to feel familiar to its users.

### 4.1.2 Portability

For KiwiVault, being portable comprises the following design goals:

- KiwiVault should not require administrative privileges to run,
- KiwiVault should be contained within the device it is installed on; it should not store preferences or other persistent files on computers it runs on,
- KiwiVault should be able to be moved between devices by moving all of the files it stores onto another drive.

Essentially, KiwiVault should exist solely on the device it is installed on.

### 4.1.3 Security

KiwiVault is designed to be a usable security program. Usability and security are frequently inversely related—improving one area often diminishes the other area. KiwiVault attempts to find an appropriate balance between these two design considerations. KiwiVault keeps the following confidential:

- **Stored documents**: Filename, file types, file contents, viewer email addresses, and creator email address.
- **Accounts**: Email addresses used to create KiwiVault accounts along with their account passwords.
- **Keys**: 3SKE creator and viewer keys.

To allow for viewing and editing encrypted documents, KiwiVault will decrypt a document and store it in a temporary directory in the user's computer. KiwiVault deletes all temporary files upon exiting.

For a detailed analysis of KiwiVault's security, see Section 6.2.

## 4.2 Stored Components

An installation of KiwiVault consists of an executable and a hidden folder which contains key managers to allow off-line access, identity managers to cache computa-

| File | Description |
|---|---|
| /KiwiVault.jar | KiwiVault executable jar. |
| /.KiwiVaultResources/ | Hidden resource directory. |
| KeyManagers/ | All user account key managers. |
| salt | Installation salt copy 1. |
| BASE64($\kappa_{\text{UID+PWD}}^{\text{“kmFilename”}}$) | Encrypted key manager.<br>One for each user. |
| IdentityManagers/ | All user account identity managers. |
| salt | Installation salt copy 2. |
| BASE64($\kappa_{\text{UID+PWD}}^{\text{“imFilename”}}$) | Encrypted identity manager.<br>One for each user. |
| Files/ | All encrypted documents. |
| salt | Installation salt copy 3. |
| BASE64($DocumentID$)/ | Unique $DocumentID$.<br>One for each encrypted document. |
| lookup.xml | Used to determine the CreatorID and if a user can access the document. |
| package.xml | Kiwi package. |
| metadata.secure | Encrypted metadata file. |
| file.secure | The encrypted document itself. |
| preview.secure | An encrypted document preview. |

Table 4.1: KiwiVault internal file structure.

tionally expensive email address hashes, encrypted documents stored by users, and three copies of the drive salt. Each installation is stored on a separate drive and multiple instances of KiwiVault can be invoked at a time. The contents of a KiwiVault installation are summarized in Table 4.1.

**Notation** This section contains various equations which use the following notation:

- $\text{ENC}_{key}(value)$: Encrypts the $value$ with the given $key$.
- $\text{MAC}_{key}(value)$: Computes a message authentication code (MAC) of the $value$ using the given $key$. The MAC can also be used as a key.
- $\text{PBKDF2}(value, salt, iterations)$: Computes a key from the $value$ using the Password-based Key Derivation Function 2 (discussed in Section 3.3) which uses the given $salt$ and is run for the specified number of $iterations$.

- $\kappa_{value}$: Is a key computed using PBKDF2($string, salt, iterations$). In this section, $2^{16}$ *iterations* are used to mitigate brute-force password guessing attacks.
- $\kappa_{value}^{purpose}$: Is a purpose specific key computed using $\text{MAC}_{\kappa_{value}}(purpose)$.

### 4.2.1 Key Manager

A key manager is an encrypted cache of 3SKE creator and viewer keys (see Section 3.1.2). It reduces the number of requests to the key server and allows the client to run off-line. Between sessions of the KiwiVault installation, the key manager is stored encrypted on the drive as an XML document. While the program is running, the keys are stored unencrypted in memory. The key manager's encryption key and filename are derived from $\kappa_{\text{UID+PWD}}$ which is computed using the following function:

$$\kappa_{\text{UID+PWD}} \leftarrow f(\text{UID}, \text{PWD}, iSalt) = \text{PBKDF2}(\text{UID} \| @ \| \text{PWD}, iSalt, 2^{16}), \tag{4.1}$$

where UID is the user's email address, PWD is the user's KiwiVault account password, and $iSalt$ is the installation salt. The '@' symbol is used to delimit UID and PWD because it cannot be found at the end of an email address; this distinguishes between email/password combinations such as "columbia@mail.co"/"marinero" and "columbia@mail.com"/"arinero". The encryption key used to encrypt the key manager is $\kappa_{\text{UID+PWD}}^{\text{"kmEncrypt"}}$ and the filename is $\text{BASE64}(\kappa_{\text{UID+PWD}}^{\text{"kmFilename"}})$.

### 4.2.2 Identity Manager

In KiwiVault, email addresses denote which users have access to encrypted documents (see Section 4.2.3). Because Equation 4.1 uses email addresses (UID), knowledge of the email addresses of accounts on the drive makes brute forcing KiwiVault account passwords much faster. For this reason, email addresses in lookup files are also put through a key deriving function to produce $\kappa_{\text{UID}}$. This computation takes on average

237 milliseconds on a 2.67 GHz Intel Core 2 CPU running Java 6 on Windows. This delay is tolerable for one email address at a time but is prohibitively slow when multiple addresses must be processed. To deal with this issue, email addresses and their corresponding $\kappa_{\text{UID}}$ values are cached in an identity manager. $\kappa_{\text{UID}}$ is calculated using the following equation:

$$\kappa_{\text{UID}} \leftarrow f(\text{UID}, iSalt) = \text{PBKDF2}(\text{UID}, iSalt, 2^{16}), \tag{4.2}$$

where UID is an email address and $iSalt$ is the installation salt. The $\kappa_{\text{UID}}$ value for an email address is calculated and stored when the user adds it as viewer to a document and it is not already stored in the identity manager.

The identity manager is encrypted and stored on the drive using the same approach as key managers; that is, $\kappa_{\text{UID+PWD}}$ is used to derive an encryption key and a filename. The encryption key used to encrypt the identity manager is $\kappa_{\text{UID+PWD}}^{\text{“imEncrypt”}}$ and the filename is BASE64($\kappa_{\text{UID+PWD}}^{\text{“imFilename”}}$).

### 4.2.3 Encrypted Documents

Each encrypted document consists of five files which allow KiwiVault to retrieve information about the encrypted document without having to decrypt the entire document. The files are labeled as lookup, package, metadata, file, and preview.

**Lookup** The “lookup” file allows users to determine if they have access to an encrypted document and what the email address of the document creator is. It is a plaintext XML document which consists of a nonce ($fileNonce$), an obfuscated list of users with access to the document, and encrypted copies of the email address of the document creator, one for each user (see Appendix A.1 for a sample lookup document). To protect privacy, the email addresses are not visible anywhere in the system. Instead, email addresses are encrypted or obfuscated. $\kappa_{\text{UID}}$ is used to obfuscate the

email addresses of the document viewers, to encrypt the email address of the creator, and is calculated as described in Section 4.2.2 and shown in Equation 4.2.

For each viewer, *lookup* and $encCreatorID$ values are stored in the lookup document. The obfuscated user email address *lookup* is calculated using the function:

$$lookup \leftarrow f(\kappa_{\text{UID}}^{\text{“lookup”}}, fileNonce) = \text{BASE64}(\text{MAC}_{\kappa_{\text{UID}}^{\text{“lookup”}}}(fileNonce)), \tag{4.3}$$

and the value *encrypt*:

$$encrypt \leftarrow f(\kappa_{\text{UID}}^{\text{“encrypt”}}, fileNonce) = \text{MAC}_{\kappa_{\text{UID}}^{\text{“encrypt”}}}(fileNonce), \tag{4.4}$$

is used to calculate the encrypted creator email address $encCreatorID$:

$$encCreatorID \leftarrow f(encrypt, creatorId) = \text{BASE64}(\text{ENC}_{encrypt}(creatorID)). \tag{4.5}$$

If a user wants to determine whether they have access to an encrypted document, they first compute *lookup* and check if it is contained in the lookup file. If *lookup* is found, they have access to the file. The user can then determine the creator by decrypting $encCreatorID$ using the value *encrypt*. If the lookup file is munged or removed, the encrypted document associated with it becomes inaccessible as the creator identifier is no longer obtainable.

**Package** The “package” file is a Kiwi package file (see Section 5.1) which gives access to the “metadata”, “file”, and “preview” files. It contains the symmetric key required to decrypt these files as well as integrity information for each file. Kiwi uses the viewer key, specific to the creator, stored in the user’s key manager in order to access the package file. See Appendix A.2 for a sample package document.

**Metadata** The "metadata" file is an encrypted XML document with the following information: file name, creation date, modified date, whether there is a thumbnail, who created the file, and the user access list specifying the access level (full or read-only) of this document's users. See Appendix A.3 for a sample metadata document.

**File** The "file" file is the actual encrypted document.

**Preview** The "preview" file is an encrypted image preview of documents that are images. The preview image is generated from the original image before it is encrypted by scaling it down so that its maximum dimension is 256. This allows for the interface to show a preview of the image without having to first decrypt the larger original image.

### 4.2.4 Installation Salt

Each installation of KiwiVault contains an installation salt *iSalt* for use with PBKDF2. Different salts are used for each installation to prevent attacks involving caches of email addresses and $\kappa_{\text{UID}}$values. Using different salts also makes users' key manager filenames different when the same email/password combinations are used on multiple installations. A consequence of using an installation salt is that, if it is completely removed from a drive, users will not be able to login and all encrypted documents on the installation become inaccessible. However, it is possible that the documents could be recovered if the user provides their own email address and those of the encrypted document creators.

Three copies of the installation salt are stored in case one copy is accidentally deleted or becomes corrupted. When KiwiVault launches, it reads the salt values from all three copies and checks to see they match. If one copy is different, the salt value of the copies that agree is determined to be the installation salt.

## 4.3 KiwiVault Functions

### 4.3.1 Sign In

Users sign into KiwiVault using their email address and password which the program combines using Equation 4.1 and checks to see if a key manager is stored on the device with the filename BASE64($\kappa_{\text{UID+PWD}}^{\text{“kmFilename”}}$) as discussed in Section 4.2.1. If the key manager is found and can be decrypted, KiwiVault begins to identify and display the documents the user has access to. If not, the user can create a new account associated with their email address or, in the case of a mistyped password, attempt to sign in again.

### 4.3.2 Create an Account

New users, as well as users who have forgotten their password, can create a new account for KiwiVault by specifying their email address as well as a password. The program contacts the authentication server to authenticate the user's email address using SAW (see Sections 3.1.1 and 3.2) and to establish an authorization token used when requesting keys from the key server (see Section 3.1.2). The authorization token is composed from two parts: one part directly from the authentication server, and the other from an email sent to the user's email account. After KiwiVault combines the two parts of the token, it requests the user's current creator key from the key server. If the transaction is successful, the user's account is created and KiwiVault begins to identify and display the user's documents.

### 4.3.3 Identifying Viewable Encrypted Documents

After a user has signed in or created an account, KiwiVault scans all of the encrypted documents stored on the drive and determines which documents the user has permission to access by checking the encrypted document's lookup file as described in

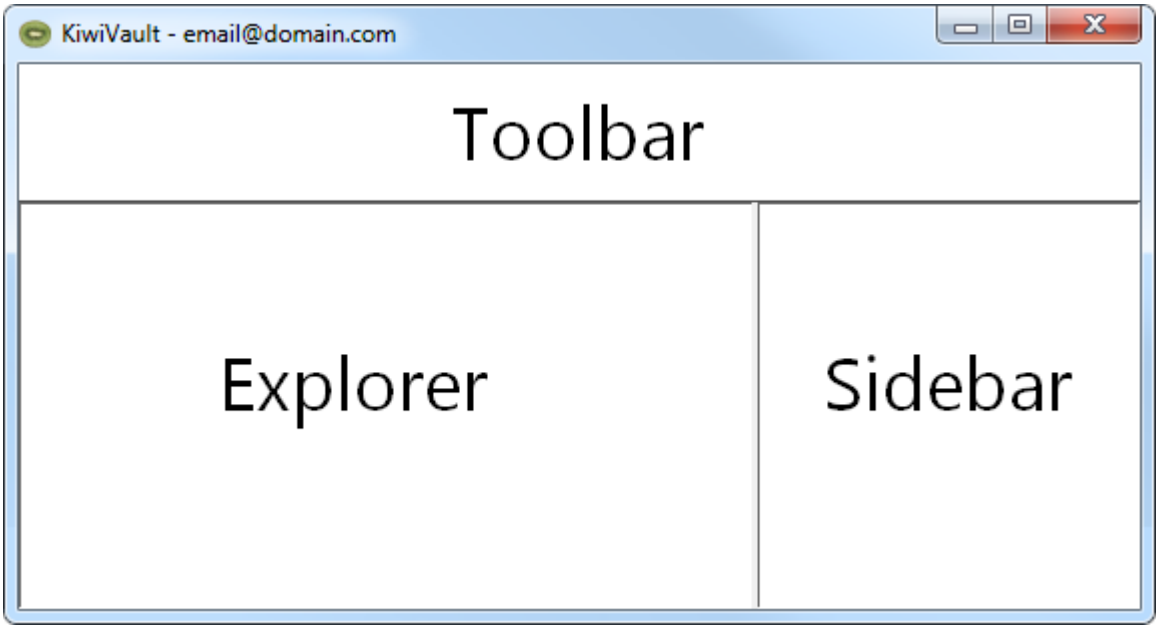



Figure 4.1: The three areas of KiwiVault: Toolbar, Explorer, and Sidebar.

Section 4.2.3. Documents that the user can view or edit are displayed in KiwiVault's file explorer.

### 4.3.4 Viewing or Editing Encrypted Documents

When users open a document within KiwiVault, a decrypted copy is created in a temporary directory on the user's computer and is automatically opened. For read-only users, the copy is marked as read-only and the user is forced to save a copy elsewhere if they want to make changes. When the user is done editing the document, KiwiVault updates the encrypted copy. When KiwiVault closes, the decrypted files are deleted from the temporary directory. If any files cannot be deleted, the user is informed that the file must be closed before they can exit.

## 4.4 User Interface

The KiwiVault user interface is a window which is divided into three areas as shown in Figure 4.1. They are:

- **Toolbar**: The toolbar provides controls to change locations in the explorer and to interact with documents.
- **Explorer**: The explorer contains a grid of icons representing the file structure of the encrypted documents on the drive.
- **Sidebar**: The sidebar contains dialogs used to manage accounts on KiwiVault as well as see information and settings of selected documents

The explorer and sidebar can be resized by dragging the divider between them. In addition to the three areas of the window, the titlebar shows the email address of the user who is currently logged into the drive.

### 4.4.1 Account Management in the Sidebar

Before a user signs into KiwiVault, the sidebar is the only active area of the program. At this stage, the sidebar displays different dialogs which provides the current user with account management operations. The user can sign into KiwiVault, create a new account in the KiwiVault installation, recover access to KiwiVault, and verify that they own their email addresses.

#### 4.4.1.1 Sign In

The *Sign In* dialog shown in Figure 4.2 is displayed to the user when KiwiVault opens. It allows users to perform one of three possible operations:

1. Sign into the drive by typing their email address and KiwiVault account password and pressing the "Sign in" button,
2. Recover access to the drive by clicking the "Forgot your password?" link, or
3. Create an account by pressing the "Create account" button.

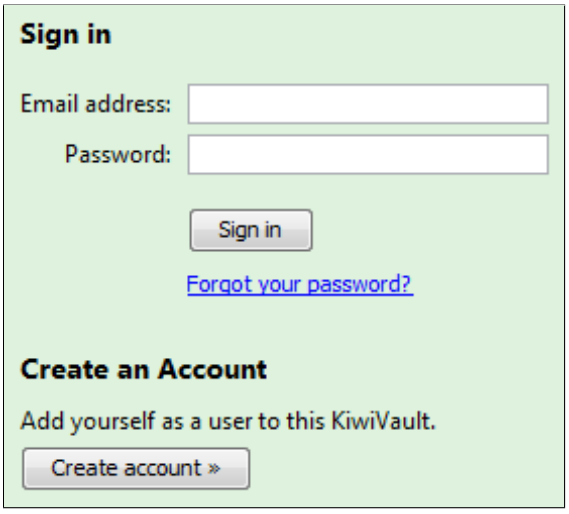



Figure 4.2: The *Sign In* dialog is the starting point of KiwiVault and allows users to sign in, recover access, and create an account.

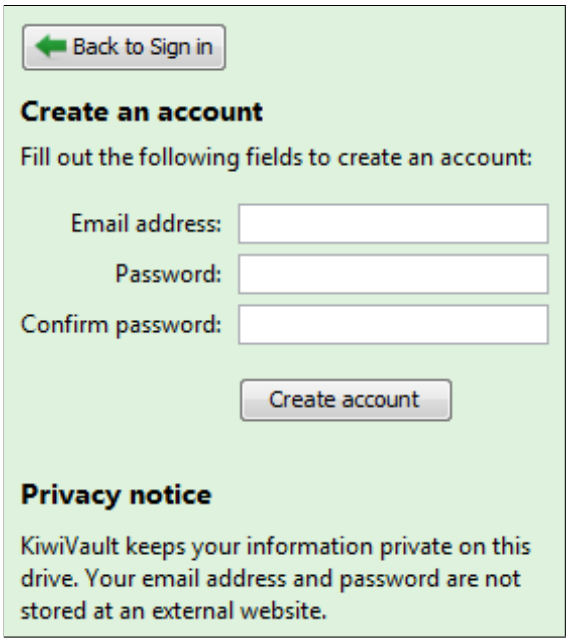



Figure 4.3: The *Create an Account* dialog allows users to create accounts in KiwiVault using an email address they own and a new password.

#### 4.4.1.2 Create an Account

The *Create an Account* dialog, shown in Figure 4.3, is accessible from the sign in dialog. To create an account, the user must specify an email address they own and a password for their KiwiVault account. After submitting these fields, KiwiVault contacts the authentication server to request an authorization token and the user is shown the verify email address dialog.

#### 4.4.1.3 Recover Access

The *Recover Access* dialog, shown in Figure 4.4, allows users to regain access to all of their encrypted documents in the event of a lost password. Recovering access to the drive is exactly the same process as creating an account on the drive: the user specifies an email address they own and a new password. Also like creating an account, after

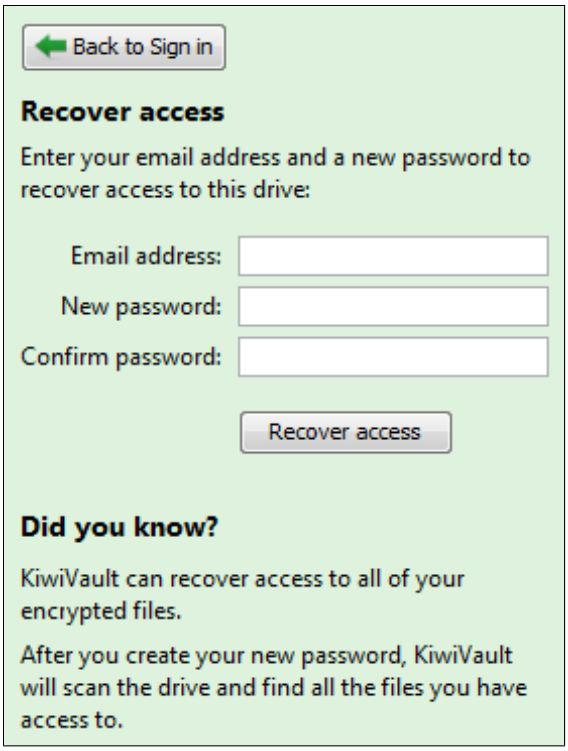



Figure 4.4: The *Recover Access* dialog allows users to regain access to their encrypted documents if they forget their password.

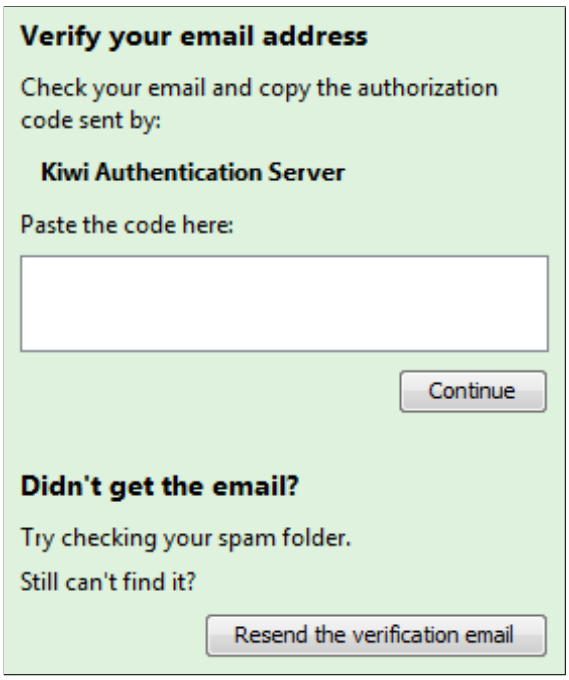



Figure 4.5: Users use the *Verify Email Address* dialog by pasting an authorization code into the text box and pressing the "Continue" button.

submitting their email and password, KiwiVault request an authorization token from the authentication server and they are shown the verify email address dialog.

#### 4.4.1.4 Verify Email Address

After creating an account or recovering access to the drive, users need to verify that they own their email address by copying a code sent to their account by the authentication server and pasting it into the *Verify Email Address* dialog shown in Figure 4.5. Because the KiwiVault window has no menu bar, instead of using the paste command from the Edit menu, users paste the code by typing the paste command or right clicking the text box and selecting Paste from the context menu. After pasting the code, users press the "Continue" button. In the event that they did not receive

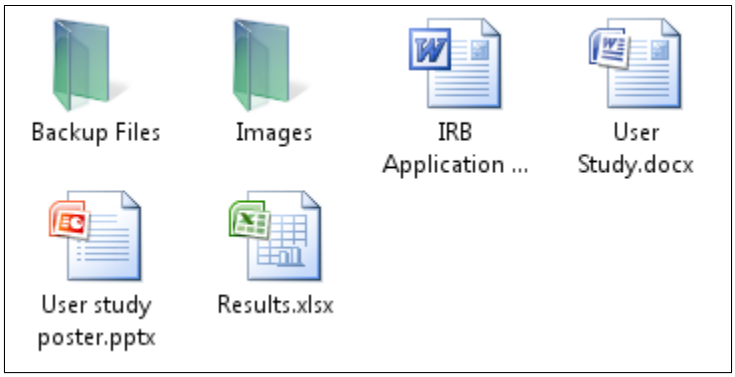



Figure 4.6: The file explorer allows users to see and interact with their encrypted documents.

the verification email (e.g., due to server latency), they can press the "Resend the verification email" button.

In addition to verifying their email address when creating an account or recovering access, users use this dialog to update their authorization token for the current key period once a month (see Section 3.1.1).

### 4.4.2 File Explorer

The file explorer is displayed to the user after they have successfully signed in, created an account, or recovered access to their drive. As seen in Figure 4.6, the file explorer provides the user navigation to all of the documents they have stored on the drive. The explorer is hierarchical and allows the user to navigate through directories and interact with the drive using the icons or other controls in the toolbar (see Section 4.4.4).

To perform operations on the documents, the user can use the mouse, keyboard, or context buttons in the toolbar. With the mouse, users can select multiple documents and folders, move documents between directories, copy documents to the desktop, copy documents into KiwiVault from the desktop, or right click on documents to access a context menu with more options. An example of a context menu on a document is shown in Figure 4.7.

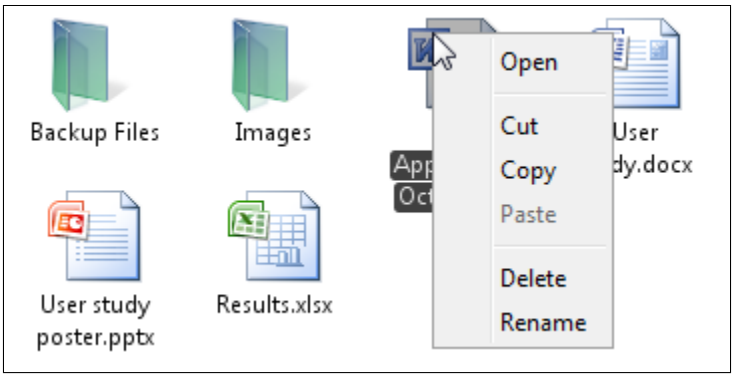



Figure 4.7: Right clicking on a document displays a context menu to perform operations on the document.

The documents and folders in the explorer are displayed in a grid and their order is configurable in the toolbar.

### 4.4.3 Preview Area

After the user signs into KiwiVault, the sidebar is used as a preview area to show information about the currently selected documents and folders. If there are no documents selected, the preview area shows information about the current directory as shown in Figure 4.8.

If a document is selected, information about that document is displayed. If the document is an image, a preview of the image is displayed as shown in Figure 4.9. Documents without previews show a larger version of their icon. When one or more documents are selected, the "Sharing and Permissions" section is displayed. In this area, users can add and remove viewers of the document as well as specify which viewers have read-only access.

If multiple documents are selected, information about those documents are displayed in addition to the common sharing properties of those documents. An example of multiple documents selected is shown in Figure 4.10.

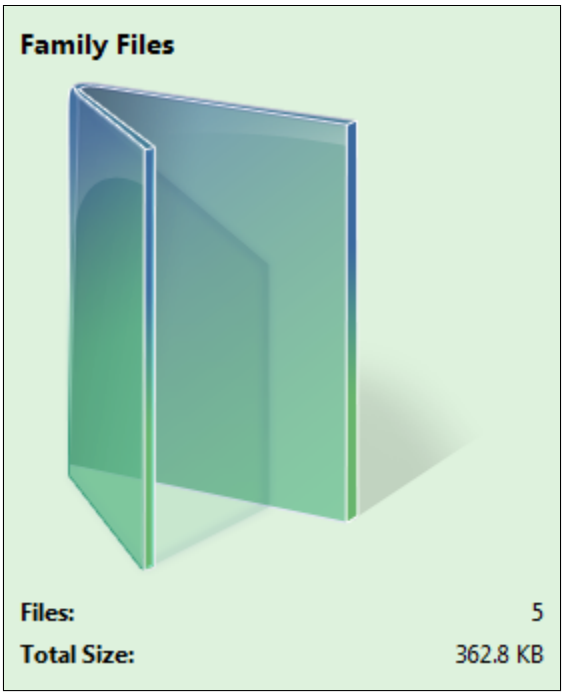



Figure 4.8: The preview area displaying information about the current directory.

### 4.4.4 Toolbar

The toolbar, shown in Figure 4.11, is used to navigate the explorer as well as perform operations on the file system. There are three parts to the toolbar: navigation controls, a bread crumb bar, and file control buttons.

**Navigation Controls** The navigation controls feature a back, forward, and up button. The back and forward buttons allow the user to browse their navigation history and the up button changes the current directory to its parent directory.

**Bread Crumb Bar** The bread crumb bar is another tool for navigating the directories of KiwiVault. A bread crumb bar is a visual method of displaying the current directory path with some added features. Clicking on any of the entries in the path will make that directory the current displayed directory. Each bread crumb directory in the path also features an arrow next to it which, when clicked, shows a drop down

menu of all the directories available from that point in the path. An example of this feature is displayed in Figure 4.12.

**Control Buttons** The bottom portion of the toolbar displays the current actions that can be performed in the file system. Depending on the current selection of documents and folders in the file explorer, different buttons are available. For example, when a single document is selected, the buttons shown in Figure 4.13 are shown. In total, the available buttons are:

- **Order by**: This button allows the user to change the order that the documents in the file explorer are displayed. Available orders are: Name, Creator, Last Modified, Size, and Type.
- **New directory**: Adds a new directory inside the current directory.
- **Add files**: Allows the user to add documents from the computer into KiwiVault.
- **Save a copy**: Allows the user to save a copy of the selected document or documents to the computer.
- **Delete**: Deletes the currently selected documents from KiwiVault.

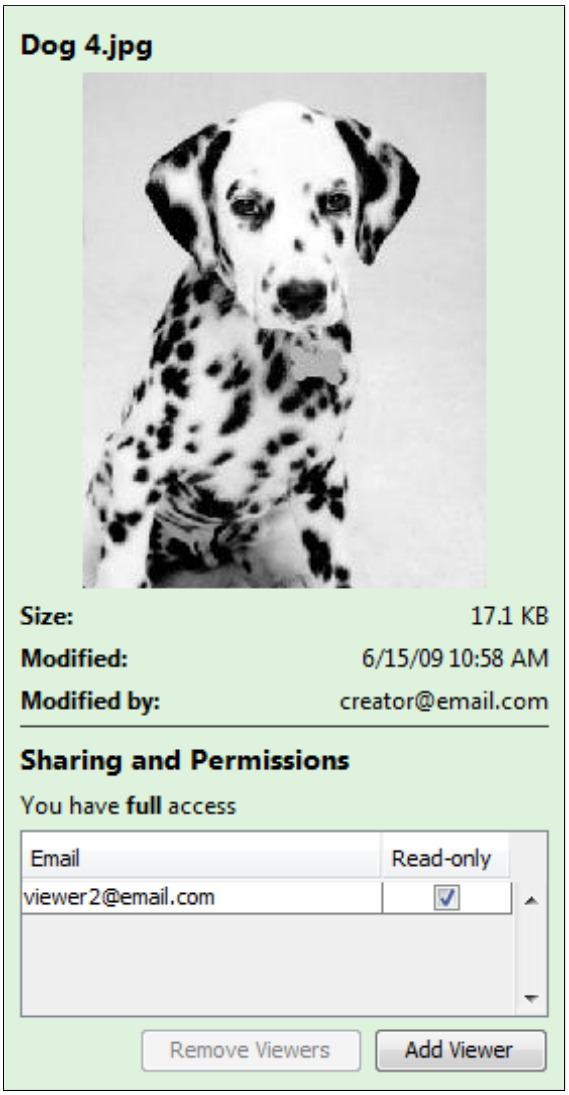



Figure 4.9: The preview area displaying information about the selected document. In this case, the document is an image so a preview of the image is displayed.

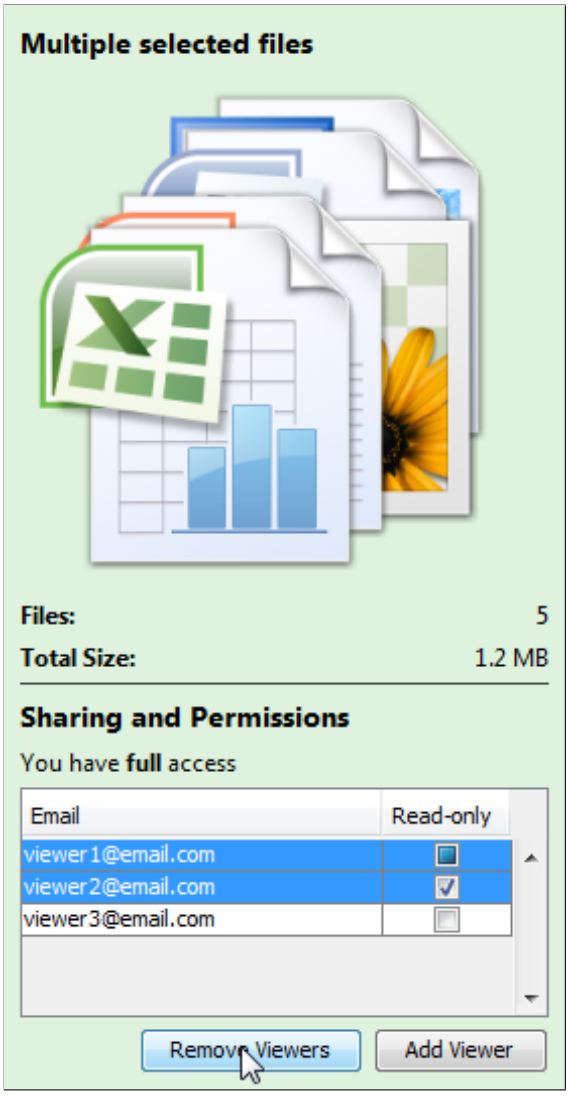



Figure 4.10: The preview area displaying information about multiple selected documents.

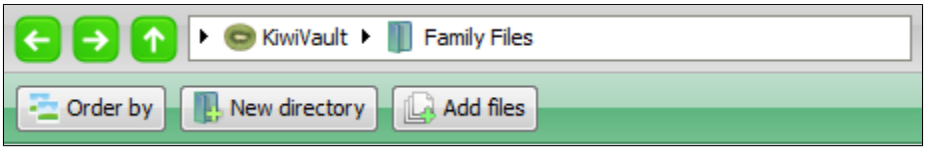



Figure 4.11: The toolbar as displayed when no documents are currently selected.

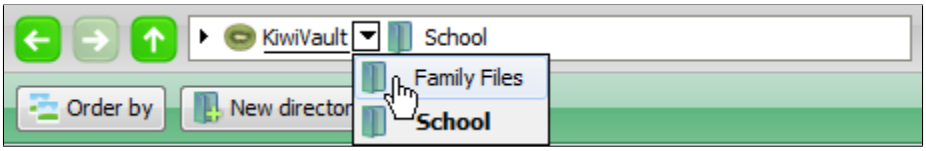



Figure 4.12: The toolbar features a bread crumb bar which allows the user to quickly change between the folders of the drive.

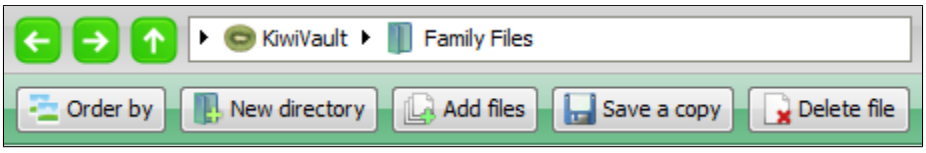



Figure 4.13: The toolbar as displayed when a document is selected.

# Chapter 5

## Implementation

This chapter discusses the implementation of Kiwi and KiwiVault. Kiwi is a suite of libraries and applications used by KiwiVault.

## 5.1 Kiwi

Kiwi is a framework in development by the BYU Internet Security Research Lab which uses 3SKE (see Section 3.1) to encrypt and package a document or message so that one or more users can view it. Kiwi has the following components:

1. **3SKE Implementation**: Key server and authentication server implementations.
2. **libkiwi**: A library for shared code between the key server, authentication server, and clients.
3. **Clients**: Programs and extensions that utilize libkiwi and 3SKE. These include KiwiVault and KiwiMail, a client for sending encrypted email.

### 5.1.1 3SKE Implementation

As discussed in Section 3.1, 3SKE is a generic key distribution protocol that leaves room for design decisions in its implementation. Kiwi's implementation of 3SKE uses email addresses as user identifiers and SAW (see Section 3.2) for user authentication on the authentication server. Kiwi implements the authentication and key servers

as Java Servlets which use XML request and response documents for all communication. The equations discussed in Section 3.1 for deriving creator and viewer keys are implemented in *libkiwi* in a variety of languages.

### 5.1.2 libkiwi

*libkiwi* consists of three libraries: one for cryptographic operations, another for XML documents used when encrypting documents or messages and communicating between the clients and servers, and a third for packaging messages for one or more viewers.

**Cryptographic Library** The cryptographic library in *libkiwi* provides API's for:

- Combining and splitting tokens (for use with SAW),
- Computing HMACs on data,
- Encrypting and decrypting data using a symmetric key,
- Generating random symmetric keys,
- Encrypting symmetric keys (for use with sharing a message key with multiple viewers),
- Computing the authorization token according to Equation 3.1,
- Deriving the creator key according to Equation 3.2,
- Deriving the viewer key according to Equation 3.3, and
- Deriving purpose specific keys from other keys (e.g., using the viewer key to derive a key specific to encryption).

**XML Library** Kiwi uses XML as the standard way of encoding data for storage and network communication. The *libkiwi* XML library contains classes which can generate and parse XML documents. These classes allow for accessing the stored data

instead of interacting with the XML directly. The following XML document classes are provided:

- **Authentication request and response documents**: Used in communicating with the authentication server. See Appendix A.4 and A.5 for sample authentication request and response documents.
- **Key request and response documents**: Used in communicating with the key server. See Appendix A.6 and A.7 for sample key request and response documents.
- **Key manager document**: Used for storing creator and viewer keys for a client along with data needed to request more keys. See Appendix A.8 for a sample key manager document.
- **Error documents**: The key and authentication servers return error XML documents when the request is invalid or some other error occurs.
- **Package document**: The package library of *libkiwi* uses this document to store encrypted items and to track external encrypted documents. See Appendix A.2 for a sample package document.

**Package Library** The package library utilizes viewer keys in order to encrypt data for a user in the form of a package. A package is a collection of encrypted files that can be stored in the package itself or external to the package. In addition, the package stores information about who can view the package as well as the key server used to retrieve the required viewer key to decrypt the contents of the package. Each package also contains a MAC of the files so that a client can detect if the attachments have been modified or corrupted. The package itself is an XML document which stores the following pieces or information:

- The location of the key server used to obtain the viewer key needed to decrypt the files referenced by the package,
- The key period of the viewer key,
- The email address of the creator of this package,
- A nonce used by the package in its encryption operations,
- The message authentication code for the contents of the package,
- The encrypted contents of the package which is a Zip file containing MACs for each referenced file and the plain text files of each internal file. This Zip is referred to as a "message bundle" because it bundles together several files,
- For each viewer there is an encrypted copy of the key used to encrypt the message bundle as well as an encrypted copy of the key used to compute the MAC of the encrypted message bundle.

The package library contains code for creating packages, adding files to the package, adding viewers to the package, and for decrypting the files referenced by the package.

### 5.1.3 Clients

Kiwi contains several clients for performing encryption for different applications. Currently, the available clients include KiwiVault and KiwiMail. KiwiMail is a system for sending secure email messages using existing email providers and programs. It is being developed as a Firefox extension, Thunderbird extension, Internet Explorer plugin, and Microsoft Outlook plugin. Future applications of Kiwi will be developed and included as clients.

## 5.2 KiwiVault Implementation

KiwiVault was implemented in Java according to its design discussed in Chapter 4. Java was chosen because of its portability: one application can run on multiple computers and operating systems. KiwiVault uses the Kiwi libraries for handling encryption and key management, Swing for the interface, and Flamingo [6] for the bread crumb bar used in the toolbar.

### 5.2.1 Java

Java [12] was designed to be portable and compiles binaries that can be run on many different platforms. Its ease of use and built-in libraries for user interface development made it a good choice for KiwiVault. The most recent version, Java 6, was used for its improved user interface libraries.

Unfortunately, choosing Java resulted in a few problems. At the time of development, we incorrectly thought that Java 6 worked on every major operating system. Sun develops JDK's and JRE's for Windows, Solaris, and Linux but not for Mac OS X. Apple manages the creation and distribution of the JDK and JRE compatible with their operating system. Instead of making Java versions work on any version of Mac OS X, Apple generally waits to include the newest version of Java with a new release of Mac OS X. Consequently, Mac OS X is generally a few years behind the rest of the Java community in terms of its version. Java 6 will not be standard across Apple computers until Snow Leopard is released and adopted by users. An attempt to backport KiwiVault to Java 5 was unsuccessful because of KiwiVault's dependency on Swing features introduced with Java 6. After Snow Leopard is released, KiwiVault will work on Apple computers as well.

Another problem with Java is that, although you can create JAR files (portable executables that run anywhere Java is installed), you can't create icons for these JAR files. Because KiwiVault was implemented to run on any Operating System,

this means that you either must create a JAR file without a custom icon, or create different executables for each operating system that launches the JAR file indirectly. KiwiVault is compiled into a single JAR file which simplifies installation.

### 5.2.2 Swing

Swing was used in order to create the content found in the toolbar, explorer, and file explorer. Because the main window of KiwiVault is resizable, the Spring Layout Manager was used in order to dynamically adjust the layout. The file explorer was implemented from scratch and was designed to feel similar to the file explorers used in Windows Vista and Mac OS X. While Java does provide libraries for retrieving icons for any file, these icons are limited at $16 \times 16$ pixels, far smaller than the maximum sizes allowed on modern operating systems. Instead of using this interface, KiwiVault stores its icons internally as PNGs for use with the file explorer and the preview area. All icons used by KiwiVault are cached in memory after they are first used so that multiple copies are not loaded.

# Chapter 6

## Validation

To validate whether KiwiVault is an effective and simple system for storing and managing encrypted documents on USB flash drives, a user study was conducted. To demonstrate security properties it provides to its users, a threat analysis was also conducted.

## 6.1 User Study

After KiwiVault was implemented, a user study was designed and conducted with 30 participants. The user study was designed with the following goals:

- Validate that USB flash drive users can successfully complete KiwiVault tasks,
- Determine if KiwiVault's design helps or impedes participants in completing these tasks, and
- Identify design areas of KiwiVault that could be improved.

Although there are many operations KiwiVault can perform, this user study focuses on features specific to KiwiVault and favors security tasks over simple file management tasks. The tasks which participants were required to perform are as follows:

- Installing KiwiVault on a USB flash drive.
- Creating an account on a KiwiVault installation.

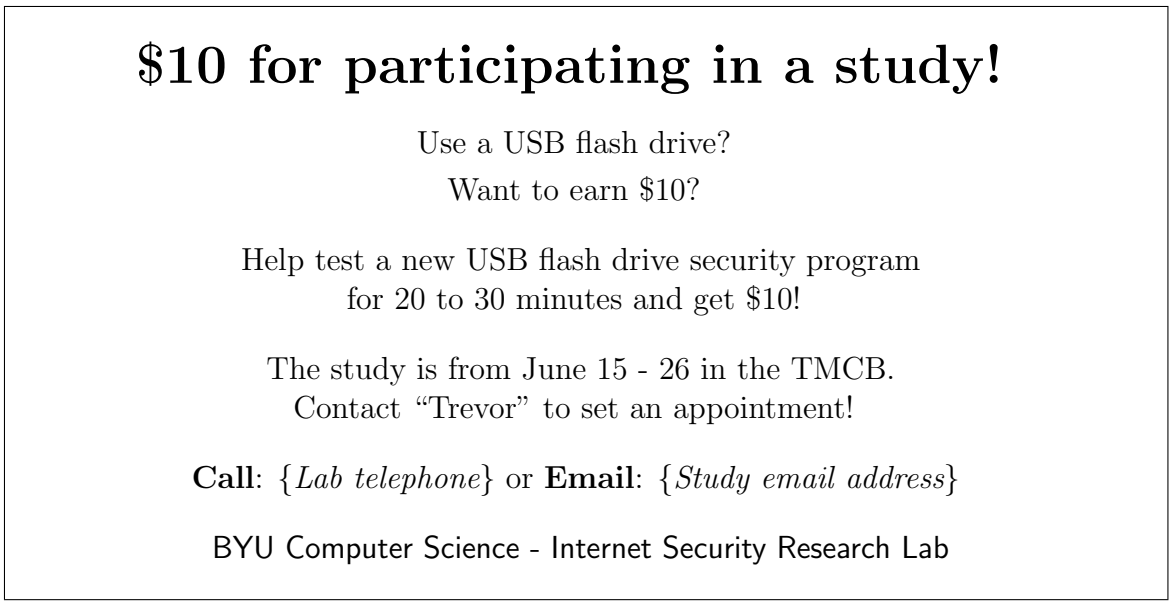



Figure 6.1: Content of the poster used to recruit participants for the user study.

- Storing documents from the desktop onto KiwiVault.
- Copying documents from KiwiVault to the desktop.
- Sharing encrypted documents.
- Editing encrypted documents.

### 6.1.1 Participants

Participants were college-age students recruited from BYU campus and were recruited through posters that were posted on bulletin boards in buildings on BYU campus. The contents of the poster are shown in Figure 6.1. The main incentive of participation was a $10 payment while minor incentives included enjoyment in participating in studies and testing new software products. While the poster did not specify that participants must be students, college attendees or recent graduates replied.

Approximately 50 people responded to the posters through email and phone of which the first 30 to confirm an appointment were included. All scheduled participants

completed the study although three rescheduled. All participants were paid regardless of their performance in the study.

### 6.1.2 Procedure of the Study

Each participant was scheduled for a 30 minute time period to complete the study. The study was conducted in TMCB 1060: a spare office on the ground floor of the Talmage building. The office was cleared and outfitted with a computer, monitor, keyboard, mouse, speakers, USB hub, and flash drives to use in the study. Windows XP Professional was installed on the computer because most students have used this operating system on BYU's Open Access computers.

Upon arriving, participants filled out a form which the proctor used to obtain a $10 BYU Cashier's Office voucher for the participant. Next they were instructed on the contents of the user study via a video which consisted of the following:

**KiwiVault User Study** Instructional Video

**Purpose** In this study, we are assessing the usability of a new program, KiwiVault, which protects documents on USB flash drives.

**What to expect** In this study you will answer several questions and perform common tasks in KiwiVault. The study will be administered using an online survey which includes instructions and questions. This study will take about 20 minutes to complete. Try to perform each task as quickly and accurately as you can.

**Materials** On the desktop are two USB flash drives which are used at different parts of the study. The online survey will instruct you when to use each flash drive.

**To Start** After this video ends, start your web browser using either Internet Explorer or Mozilla Firefox. The browser will open to the online survey as well as an

email account you will use in the study.

At this point, the proctor left the room and the participant completed the study using the online survey. After completing, the proctor gave the participant their voucher which they then took to the BYU Cashier's Office to obtain their $10.

### 6.1.3 Survey Contents

A survey was used to conduct the study because it allowed for gathering the participants' computer experience and feedback as well as measure which steps they were able to complete. The survey is divided into four sections: Computer Experience, Scenario 1, Scenario 2, and Feedback. The Computer Experience and Feedback sections have all their questions on single pages while the Scenario sections shows one question at a time. In the Scenario 1 and 2 sections, if participants fail to complete a step, they are skipped ahead to the next step they can complete. For example, if the user fails to install KiwiDrive on the flash drive, they cannot encrypt documents and so they are skipped ahead.

The actual contents of the survey used in the study are as follows:

#### 6.1.3.1 Introduction

KiwiVault is a program that protects documents on a USB flash drive.

In this study you will answer several questions and perform common tasks in KiwiVault.

This study will take about 20 minutes. Try to perform each task as quickly and accurately as you can.

Thank you for participating!

#### 6.1.3.2 Computer Experience

How often do you use a computer?

- ◯ Daily
- ◯ Weekly
- ◯ Monthly
- ◯ Hardly Ever

How often do you use a USB flash drive?

- ◯ Daily
- ◯ Weekly
- ◯ Monthly
- ◯ Hardly Ever

Have you lost or had stolen a USB flash drive or other portable storage device?

- ◯ Yes
- ◯ No

Where do you store your computer documents? (Check all that apply)

- ☐ On my computer
- ☐ On my USB flash drive
- ☐ On the Internet (E.g., Mozy, Sky Drive, Google Docs, Gmail, etc.)
- ☐ On a backup hard drive

How important is maintaining the privacy of your computer documents?

◯ Very Important

◯ Important

◯ Neither Important nor Unimportant

◯ Unimportant

◯ Very Unimportant

Have you ever encrypted a computer document?

◯ Yes

◯ No

#### 6.1.3.3 Scenario 1 - Install KiwiVault on a flash drive

In this part of the test, you will install and use KiwiVault on a USB flash drive.

Please try to complete each step perform each task as quickly and accurately as you can.

**Step 1**: Install KiwiVault on Flash Drive A. Visit http://haley.cs.byu.edu/kiwi/ and follow the instructions on the page.

◯ Step completed

◯ Step not completed

**Step 2**: Create an account on KiwiVault using the email address "vault.study@gmail.com." Follow the instructions in the KiwiVault program. The "vault.study@gmail.com" Gmail account is open in the next tab.

◯ Step completed

◯ Step not completed

**Step 3**: Add these items from the desktop to KiwiVault: The file "Expense report.xlsx," and the folder "Group Project."

◯ Step completed

◯ Step not completed

**Step 4**: Share files with "{*Proctor's Email Address*}." Share the file "Expense report.xlsx." Share the folder "Group project."

◯ Step completed

◯ Step not completed

**Step 5**: Give "{*Proctor's Email Address*}" read-only access to "Expense report.xlsx."

◯ Step completed

◯ Step not completed

**Step 6**: Create a Word document and save it into KiwiVault. Open Microsoft Word from the Windows start menu. After typing some text in a new document, store it in KiwiVault.

◯ Step completed

◯ Step not completed

**Step 7**: Close KiwiVault and remove Flash Drive A from the USB hub.

◯ Step completed

◯ Step not completed

#### 6.1.3.4 Scenario 2 - Use KiwiVault on Trevor's flash drive

In this part of the test, you will use the KiwiVault program on Trevor Florence's flash drive.

Trevor uses this flash drive to store his personal files and has shared some of the files with you.

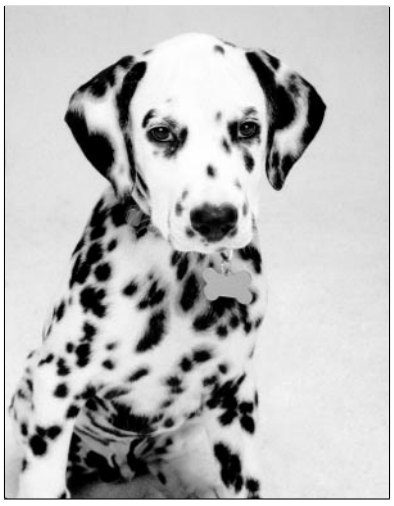
Insert Flash Drive B into the USB hub, open the flash drive from the desktop, and run the KiwiVault.jar file.

**Step 1**: Sign in to Trevor's KiwiVault using the email address "vault.study@gmail.com" and password "cougars."

◯ Step completed

◯ Step not completed

**Step 2**: Find this picture in the "Dogs" folder:



◯ Step completed

◯ Step not completed

**Step 3**: In the "Security Guidelines" folder, identify if you have full or read-only access to each file.

| | Full access | Read-only access | Not sure |
|---|---|---|---|
| Disclosure guidelines.docx | ◯ | ◯ | ◯ |
| Operating procedures.docx | ◯ | ◯ | ◯ |
| Security briefing.pptx | ◯ | ◯ | ◯ |
| Personnel directory.xlsx | ◯ | ◯ | ◯ |

**Step 4**: Make a copy of the file "Schedule.docx" to the desktop.

◯ Step completed

◯ Step not completed

**Step 5**: Edit the "Member list.docx" document. Add your favorite cartoon character or movie star to the list. Save and close the document.

◯ Step completed

◯ Step not completed

Were your changes in "Member list.docx" saved in KiwiVault?

◯ Yes

◯ No

◯ Not sure

**Step 6**: Close KiwiVault and remove Flash Drive B from the USB hub.

◯ Step completed

◯ Step not completed

#### 6.1.3.5 Feedback

This is the last part of the test where you can share your feelings about the experience.

How would you rate your experience with KiwiVault?

- ◯ Very Good
- ◯ Good
- ◯ Neutral
- ◯ Bad
- ◯ Very Bad

Would you use KiwiVault to protect your personal documents on a USB flash drive?

- ◯ Very Likely
- ◯ Likely
- ◯ Undecided
- ◯ Unlikely
- ◯ Very Unlikely

How usable was KiwiVault?

- ◯ Very Usable
- ◯ Usable
- ◯ Somewhat Usable
- ◯ Unusable
- ◯ Very Unusable

What didn't you like about KiwiVault?

What did you like about KiwiVault?

### 6.1.4 Results

The Computer Experience section of the survey shows that 97% of participants use a computer daily (see Figure 6.2) and that 73% use a USB flash drive at least weekly (see Figure 6.3). This shows that most participants are from KiwiVault's target audience: computer users who use USB flash drives. In addition, 37% of participants had lost or have had their portable storage device stolen (see Figure 6.4), 77% store their documents on flash drives (see Figure 6.5), and 93% of the participants feel that maintaining the privacy of their documents is Important or Very Important (see Figure 6.6). These results show that a majority of the participants do store documents on their flash drives and would benefit from KiwiVault as it would protect their device in case of loss or theft. Lastly, only 17% of the participants have encrypted a document before (see Figure 6.7). This shows that whether or not people understand encryption they do not make use of it.

The results from the first scenario displayed in Figure 6.8 show that users, for the most part, are able to perform the tasks in KiwiVault. The one participant who was not able to create an account in Step 2 could not create an account due to network connection issues with the key server. This person was the second to participate in the study and the error was avoided afterwards by running the key server directly on the testing machine. The participant who was not able to complete Step 3 attempted to add the documents directly to the USB flash drive within Windows Explorer instead of

Figures 6.2 through 6.12: Charts of the user study results. All X-Axes represent the number of survey participants who selected a particular question choice or performed a step correctly.

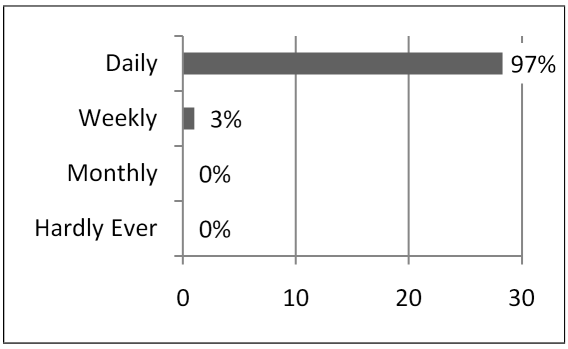



Figure 6.2: Results for "How often do you use a computer?"

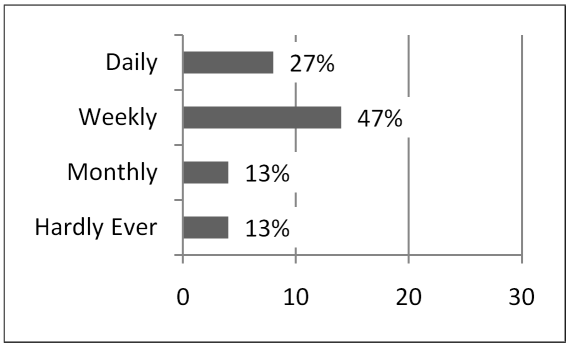



Figure 6.3: Results for "How often do you use a USB flash drive?"

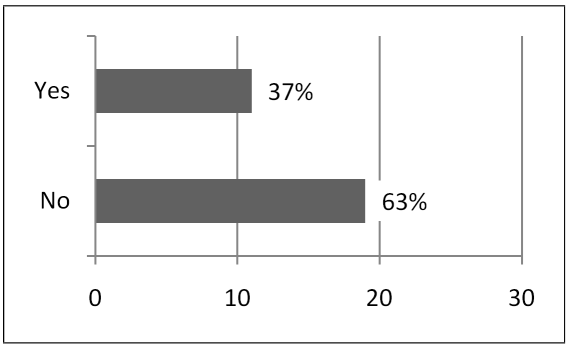



Figure 6.4: Results for "Have you lost or had stolen a USB flash drive or other portable storage device?"

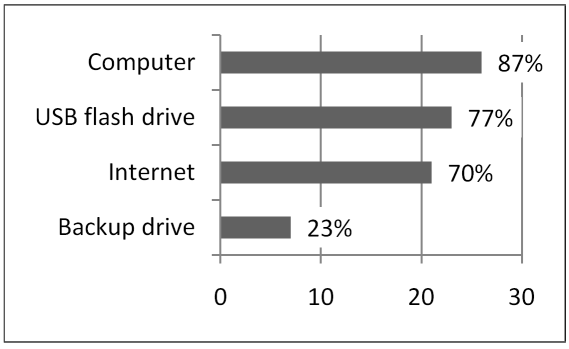



Figure 6.5: Results for "Where do you store your computer documents? (Check all that apply)"

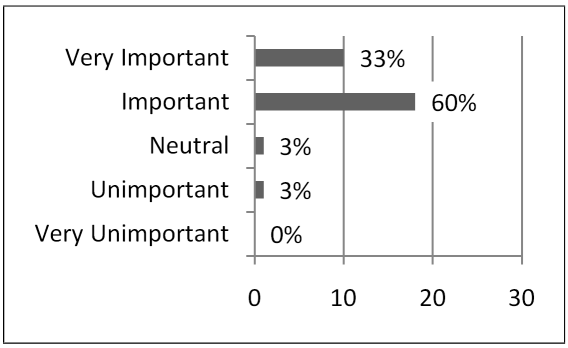



Figure 6.6: Results for "How important is maintaining the privacy of your computer documents?"

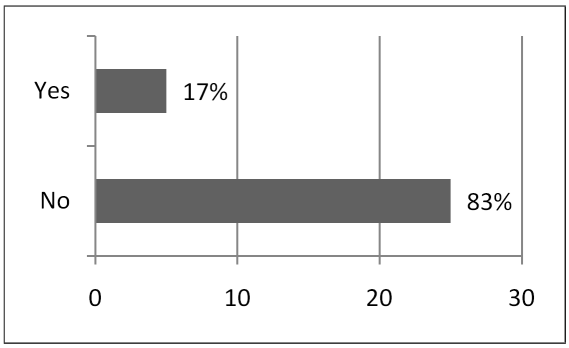



Figure 6.7: Results for "Have you ever encrypted a computer document?"

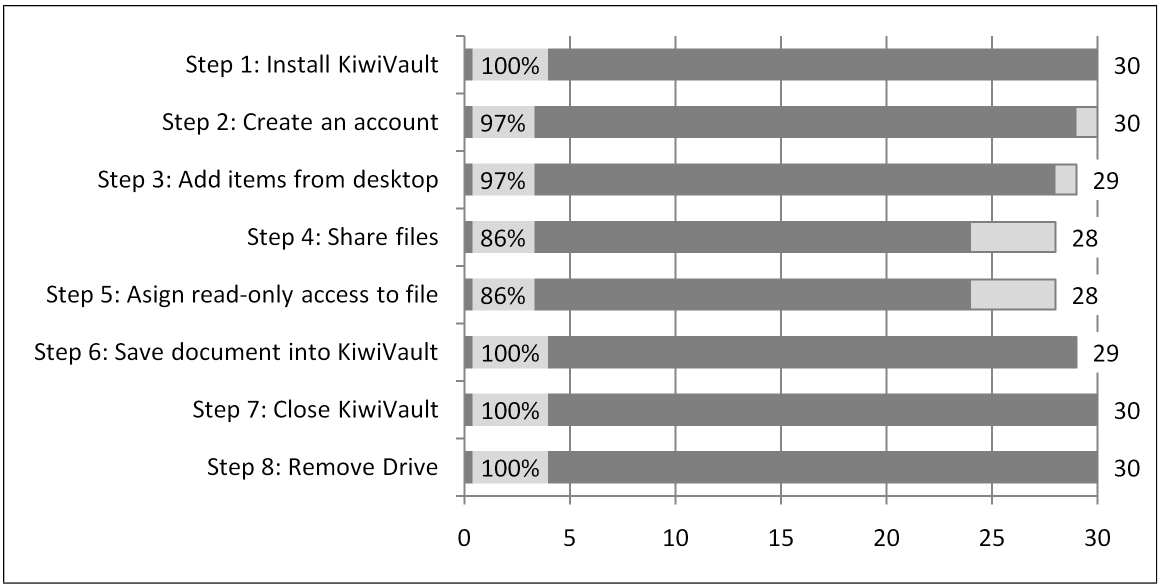



Figure 6.8: Successful completion rates for Scenario 1. The light grey bars and numbers on the right show the number of participants who reached the step. The dark grey bars show how many completed the step correctly. The percents show how many completed it correctly out of how many reached the step.

into the KiwiVault window. Four participants were unsuccessful in sharing encrypted documents in Steps 4 and 5, of these, three attempted to email the documents directly to the email address specified.

The results from the second scenario seen in Figure 6.9 were generally positive as well. The one participant who could not sign into KiwiVault in Step 1 mistakenly inserted both flash drives into the computer and could not figure out how to access the second drive. The participants in Step 3 that did not correctly identify which documents were read-only show that improvement can be made in the sharing aspect of the program. The participant who tried to copy a document to the desktop in Step 4 tried to copy the document using the system copy and paste which is not implemented in KiwiVault. Two participants did not successfully close the program due to a bug where KiwiVault thinks that a document that was being edited is still open for edits.

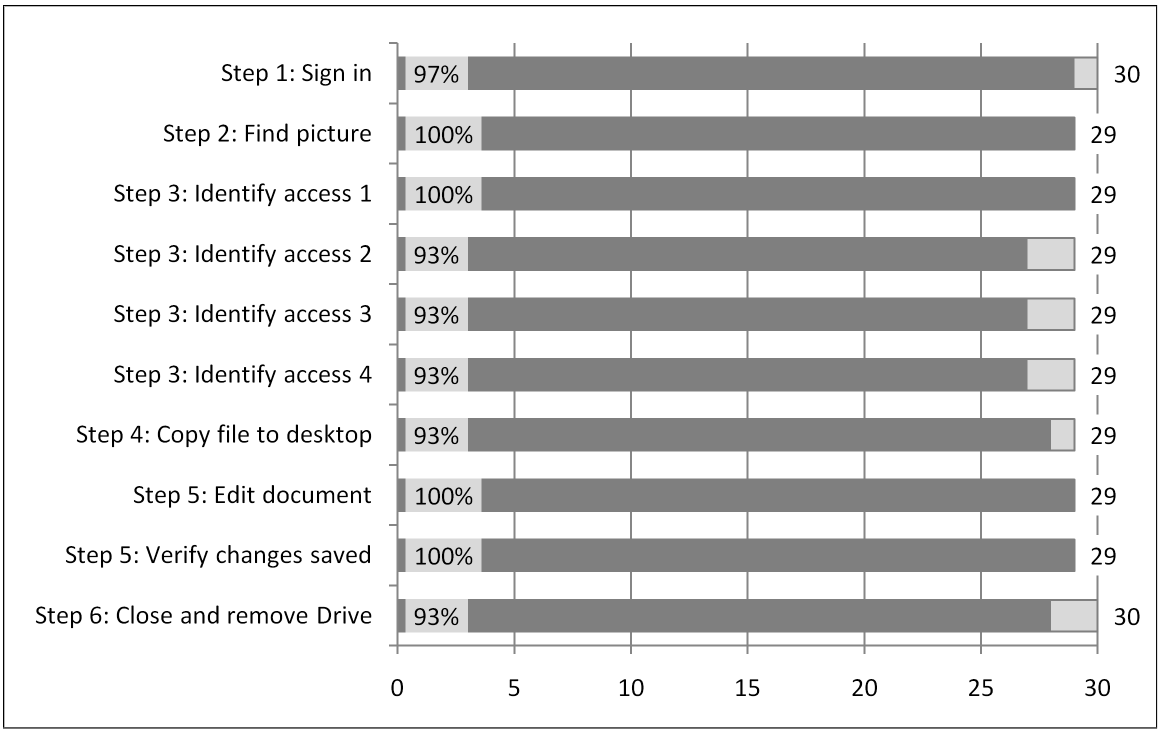



Figure 6.9: Successful completion rates for Scenario 2. This figure is organized in the same manner as Figure 6.8.

The feedback section showed that 97% of participants had a good (57%) or very good (40%) experience with KiwiVault (see Figure 6.10) and 97% found KiwiVault usable (23%) or very usable (73%) (see Figure 6.11). When asked if they would use KiwiVault to protect their documents, 23% were undecided, 47% responded that they would likely use it, and 30% reported very likely. While 77% reporting that they would likely or very likely use this program is not a negative result, it shows that the participants didn't appreciate or understand the program very well. This point is stated by one participant: "Didn't really quite get the point of it, except I guess it somehow makes things more secure? Unless I understood the point, I probably wouldn't use it." See Appendix Section B.1 for all the textual negative feedback and Section B.2 for the positive feedback.

The user study was designed to take twenty minutes to complete and, in the end, the average completion time was 17.6 minutes. A histogram displaying the

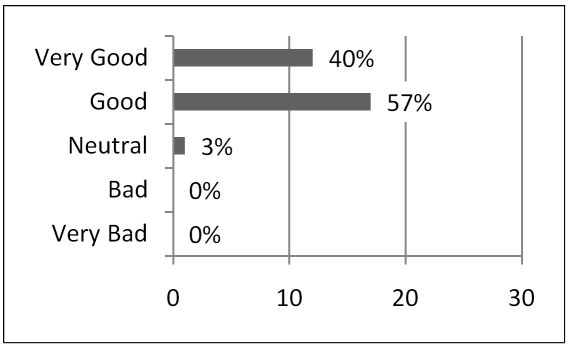



Figure 6.10: Results for "How would you rate your experience with KiwiVault?"

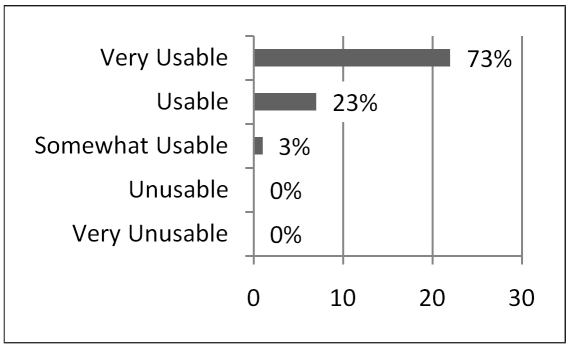



Figure 6.11: Results for "How usable was KiwiVault?"

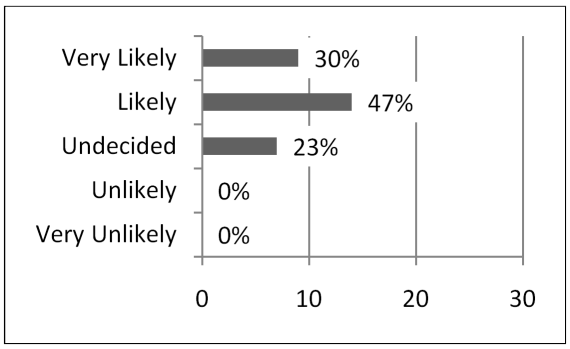



Figure 6.12: Results for "Would you use KiwiVault to protect your personal documents on a USB flash drive?"

participant completion time is shown in Figure 6.13. The histogram rounds the time to the nearest minute and then totals each time into a separate bin ranging from 1 to 33. 80% of participants took 20 minutes or less to complete the study.

### 6.1.5 Conclusions

As stated in Section 6.1, the user study was designed with three goals in mind. This section will discuss each goal and whether or not it was met.

**Validate that USB flash drive users can successfully complete KiwiVault tasks** The results in Section 6.1.4 clearly show that this goal was met. The lowest completion rate for any step of the two scenarios is 86% and the average completion rate of all steps is 96%.

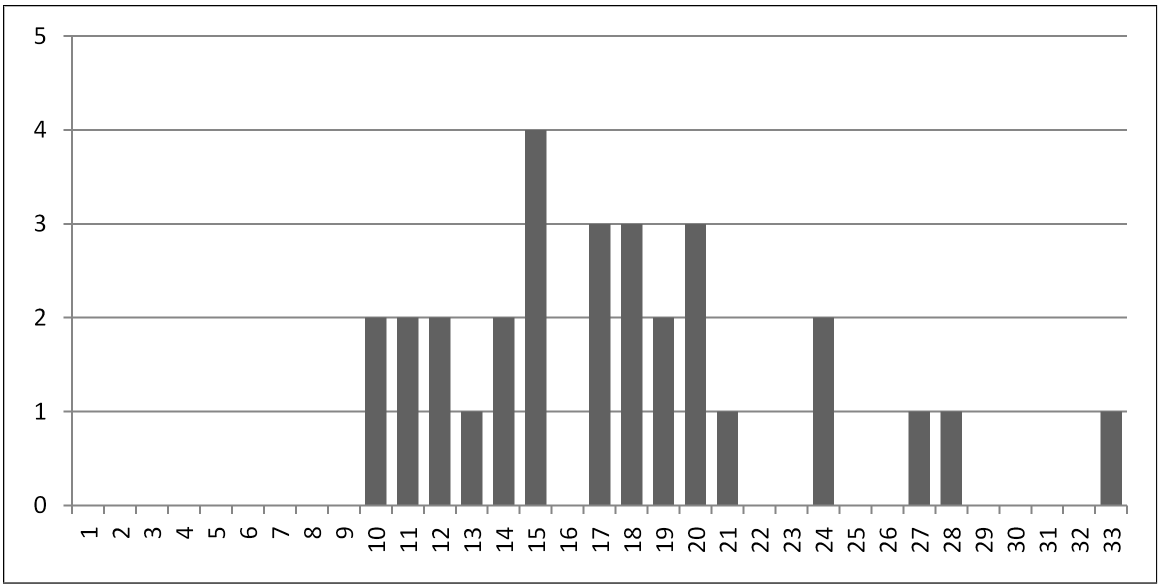



Figure 6.13: Participant time to complete the user study rounded to the nearest minute and totaled.

**Determine if KiwiVault's design helps or impedes participants in completing these tasks** Based on user feedback and the results of the study itself, the design appeared to help users complete tasks. Most users commented that the interface was easy to understand, nevertheless, several struggled with completing the sharing and permissions tasks, which are not generally found in traditional file explorer interfaces. While participants recognized how to complete tasks found in traditional file explorers because of the familiar interface, they did not complete the tasks related to sharing and permissions as successfully because the interface was not sufficient to help aid them in completing an unfamiliar task.

**Identify design areas of KiwiVault that could be improved** The main areas where KiwiVault could be improved are related to file sharing and system compatibility.

Currently, KiwiVault requires its users to select one or more documents in order to perform any tasks related to sharing. During the study, when it came to sharing documents, several participants did not think to click on the document they

needed to share. One possible way to improve sharing visibility and ease of use would be to provide a "Share files" button in the toolbar which would always be present and would prompt the user to select which documents they would like to share and then show them the sharing interface in its regular sidebar location.

As shown by the study results, KiwiVault needs to interoperate with the operating system it is running on. All expected commands such as right clicking contextual menus as well as copy and paste support for documents and text should be supported. While the current design of KiwiVault provides many of these interoperabilities, there are still some lacking. Examples of features that could be added are copy and paste support for documents between the desktop and KiwiVault along with the ability to paste using a context menu in all text fields.

One design modification that would improve the experience would be to make the interface of KiwiVault be an actual virtual drive in the operating system much like TrueCrypt described in Section 2.3. The additional interface controls needed to function could be added to windows of KiwiVault virtual drives as a side bar. This would provide a much more seamless experience but would make portability more challenging.

One last area that could be improved would be to help users visualize that their documents are protected. Visual cues such as a status badge could be placed on the document icon indicating that it is open for editing, read-only, or encrypted. This would help users appreciate the program better and make it easier to understand what is protected (e.g., documents in KiwiVault) and what is not (e.g., files stored directly on the flash drive).

## 6.2 Threat Analysis

This threat analysis looks at the types of attackers who could attack KiwiVault, what the attacks are, how they are executed, and any improvements that could be made to alleviate the threats.

### 6.2.1 Attackers

Generally, attackers will either attempt to exploit KiwiVault physically or remotely. Remote attackers can only access KiwiVault's files while the user has their flash drive inserted into their computer. Physical attackers can perform their attacks while the owner is still in possession of the drive or when the drive has been lost or stolen.

**Remote** These attackers do not have physical access to the machine but attack through networks the user's machine is connected to. They may use software exploits to gain access to the user's machine or have a remote access account on the computer.

**Physical** These attackers have physical access to the machine but do not steal the USB flash drive KiwiVault is installed on. They may gain access to a machine that is logged in when the user is away or may have an account on the computer.

**Obtainer** These attackers are technical or non-technical people who either find or steal the USB flash drive with KiwiVault installed on it. Technical obtainers are those who are familiar with performing attacks on encrypted documents or protected passwords. While they do have physical access to the drive, they do not have access to the owner's computer.

### 6.2.2 Attacks

The protected email addresses of KiwiVault accounts and the encrypted documents stored in KiwiVault can be attacked directly or remotely by attacking components of

Kiwi or SAW. Regardless of the method of attack, because KiwiVault exists only on the drive it was installed on, all attacks require access to the drive or a copy of the files on the drive.

The main attack that KiwiVault protects users against is unauthorized access to their stored documents. Decrypting a document stored in KiwiVault can only be accomplished by having the viewer key of one of the viewers or the creator key of the creator. Obtaining one of these keys requires that the attacker must first discover the creator email address or a viewer email address. Once the email address of an account is determined, the attacker needs the authorization token, creator key, or viewer key. These can be obtained by compromising the key or authentication server, using the user's email account password or their KiwiVault account password, or compromising an instance of KiwiVault the user has logged into.

Apart from gaining unauthorized access to the encrypted documents, other attacks include gaining information about KiwiVault installations and limiting availability of the drive for its users. These attacks could be serious if any information about the drive being divulged is intolerable or the user has no backups of their encrypted documents.

#### 6.2.2.1 Determining account email addresses

In order to access an encrypted document, an attacker needs to know the email address of the creator or one of the viewers. KiwiVault obfuscates the email addresses of its users by putting the email addresses through PBKDF2 for $2^{16}$ iterations to generate $\kappa_{\text{UID}}$ (see Equation 4.2). The email address of the creator of the document is also obfuscated by encrypting it with $\kappa_{\text{UID}}$ for each viewer. To determine if an email address belongs to a viewer of an encrypted document, one must calculate *lookup* using Equation 4.3 and see if it is stored in the encrypted document's lookup file (see Section 4.2.3). Once a viewer is known, the creator email address can be decrypted.

If an attacker suspects a particular person with a known email address has access to a KiwiVault installation, the attacker can check if the person has access by seeing if there exists an encrypted document on the drive that the person has access to. Checking if a single person can view any document on the drive is a fairly quick process but is not a guarantee that they have an account on the installation. It is possible that a person can create an account on KiwiVault and have no encrypted documents viewable to them or that a person can be a viewer of encrypted documents on KiwiVault and have no account on the drive. However, the creator can be determined when a viewer is known and is much more likely to have an account on the drive.

If an attacker has found or stolen the drive and is not aware of any of the viewers on the drive, they will be forced to guess email address using a brute force search. Email addresses can have 64 characters before the @ symbol (where 39 possible characters can be used), and 256 characters after [13]. Assuming the attacker was only attempting to brute force a common email domain—say "gmail.com"—using only 10 characters before the @ symbol and that they can check one hundred email addresses a second, it would take roughly 2.6 million years to try all possible email addresses. If an attacker knows any information about a user, they could guess the email address much faster or simply find it using information available on the Internet from search engines and social networking sites.

#### 6.2.2.2 Determining an existing accounts password

If an email address corresponding to a creator or viewer on the drive is determined, the attacker can attempt to brute force the password off-line by deriving $\kappa_{\text{UID+PWD}}$ using Equation 4.1 and seeing if a key manager with the title of BASE64($\kappa_{\text{UID+PWD}}^{\text{"kmFilename"}}$) exists.

#### 6.2.2.3 Creating a new account with a known email address

An alternative to brute forcing an account's password is to create a new account on the installation using the email address of the creator or a viewer of the encrypted document. To accomplish this, the attacker either needs access to the secret keys used by the key or authentication server or the creator or viewer's email account.

Ideally, the secret keys used by the Key and Authorization Servers should be stored on cryptographic hardware such that they can only be removed via physical means. If, somehow, an attacker managed to obtain the key server's master key, the security of *all* KiwiVault installations would be compromised. Although a new key server master key could be put in place, all documents encrypted before the key was changed remain vulnerable. If an attacker obtains an authentication server's master key, that attacker can generate authorization tokens for any viewer and the security of the system is compromised for all users who obtained their authorization token from the server. Once a new authentication secret key is selected, all users must re-authenticate to the authentication server. At this point, the attacker will not be able to create or use authorization tokens honored by the key server anymore.

Instead of stealing heavily protected secret server keys, an attacker can attempt to gain access to the user's email account to retrieve the authorization token. The attacker can initiate an authentication for an email address with the authentication server from KiwiVault itself and then attempt to retrieve the email from the user's email account either through monitoring email traffic, exploiting the password reset facilities offered by email providers, or some other means. If the attacker has physical or remote access to the user's machine, the user may have left their email account logged in or their credentials saved which makes accessing the account trivial.

#### 6.2.2.4 Obtaining keys and data from system memory

While KiwiVault is running, much of the data that is stored encrypted on the flash drive is stored unencrypted in memory. The following are loaded into memory and persist until KiwiVault closes: authorization tokens, creator keys, viewer keys, email addresses of people the user has shared with, and the metadata of the encrypted documents the user has access to.

Although operating systems sandbox processes into their own memory space, there are attacks for accessing memory such as cold boot attacks and analyzing the deallocated memory contents of a program after it closes or is killed. There are also methods to mitigate these attacks such as reducing the amount of time sensitive data is stored in memory and locking pages in memory to prevent them from being paged out to disk.

#### 6.2.2.5 Untrusted KiwiVault Installations

It is possible that an attacker could share a malicious untrusted copy of KiwiVault with a victim that collects the victim's private information such as their KiwiVault password or their creator or viewer keys which could then be used to decrypt documents on other KiwiVault installations obtained through physical or remote access. When running dangerous untrusted software, the victim is susceptible to an unlimited number of attacks as the program is authorized to do anything the user is authorized to do.

Currently, the KiwiVault application is stored as a Jar file on the user's USB flash drive. This is convenient for users as the entire application travels with them and can be used portably and off-line. Despite these conveniences, it is possible that an attacker could create a fake KiwiVault by adding malicious code to a valid copy or creating a program that looks like KiwiVault but only contains sufficient functionality to steal user data. The malicious program would be placed on an attacker's USB flash

drive, shared with the victim, and then would store the victim's authorization token when they create an account. With access to this token, an attacker can access any of the user's encrypted documents on any installation of KiwiVault where the same email address was used.

Unfortunately, trusting the source of the executable is a problem with nearly all software. One way to mitigate the problem is to allow users to download a copy of KiwiVault from a trusted source and install it on their own computer. This way, users could use their one trusted copy of KiwiVault on other people's flash drives as well as their own. This would be convenient for the user as they would not have to create a new account for every drive. A copy of KiwiVault could still be installed on the user's flash drive for their personal use on other machines, including public computers.

#### 6.2.2.6 Discovering information about KiwiVault installations

Besides attempting to decrypt the contents of documents stored on KiwiVault, there is other obtainable information about the drive. This section looks at several different things that can be learned about the drive.

**How many accounts exist on the drive** Because the only files associated with a user's account are their key and account managers, anyone can determine how many accounts there are on a KiwiVault installation by counting these files.

**When each account last accessed the drive** After logging out of KiwiVault, updated copies of the key and account managers are stored on KiwiVault. The last modified date of these files correspond to when the account was last accessed.

**When each encrypted document was last modified** Similarly, encrypted documents are stored as separate files on the drive and their metadata (e.g., creation

and modified dates) is publicly viewable. This gives enough information to determine when each encrypted document was last modified.

**Which documents were edited during an account's last session** By viewing the last modified dates of the key manager and encrypted files, which encrypted document was modified by which account in their last session can be determined.

**If an encrypted document has a preview** When a stored document has a preview available, it is stored along with the full-size document. This information indicates which documents have previews.

This vulnerability can be resolved if fake previews are stored for those documents that do not have previews. Because all previews have a maximum size, KiwiVault could just generate random binary files of a random size according to an appropriate size distribution. These random files would be indistinguishable from regular encrypted images. The overhead of this method, however, may outweigh the benefits. On USB flash drives, write speeds are much slower than read speeds, so storing these extra fake files could slow the program down.

**The number and size of documents** The number and size of encrypted documents is visible on the drive because they are system files.

**Unencrypted documents** If the drive is used incorrectly and documents are stored on the drive itself instead of in KiwiVault, they are publicly accessible. KiwiVault is launched from the drive window and users are free to add documents here whether or not KiwiVault is running. An attacker can simply insert the drive and check to see if any unprotected documents remain on the drive.

If KiwiVault is running, this problem could be mitigated by having KiwiVault detect unencrypted documents on startup and warn the user when they add docu-

ments directly to the drive and provide the user with the option to automatically add the documents to KiwiVault.

**Crash residue** If KiwiVault crashes, unencrypted documents could be left in the temporary directory of the user's computer. KiwiVault does not stop running if the flash drive is pulled out prematurely so should still be able to delete the temporary files, but decrypted documents could remain if KiwiVault is incorrect shutdown (e.g., through a system or crash or a user killing the program). One option would be to delete unencrypted documents whenever the user returns to KiwiVault but this has its problems as there is no way to absolutely determine if a file is being edited or not.

All of the attacks discussed here related to examining the modified date or different files could be alleviated by having KiwiVault "touch" every file on the drive before exiting thus making all the dates on the drive correspond to when the last user logged off.

#### 6.2.2.7 Limiting availability

If an attacker has physical access to a KiwiVault installation, there is nothing stopping them from deleting or changing any file on the drive (see Section 4.2 for an explanation of what files are stored on the drive). If an attacker deletes or changes key or identity managers, all of the documents can be recovered after the user creates a new account. Until that time, the user cannot use the drive off-line. If an attacker deletes or modifies two or more of the installation salts, the key and identity managers stored on the drive become permanently inaccessible and to recover access to the documents, the user would have to supply their email address and the email addresses of the document creators. While the encrypted documents can be deleted, all modifications can be detected using the MACs stored for each encrypted file.

# Chapter 7

## Conclusions and Future Work

### 7.1 Conclusions

The purpose of KiwiVault is to provide protection to documents on USB flash drives with minimal loss in functionality compared to unprotected drives. KiwiVault adds document encryption to flash drives and maintains unprotected drive functionality such as portability, usability, the ability to share documents between multiple people, and access to stored documents even if a password is forgotten.

The main contributions that KiwiVault brings to security software is a unique method for recovering encrypted files after losing a password and support for collaborating between multiple users with encrypted documents. Because KiwiVault uses 3SKE for key distribution from a trusted key server, the keys used to encrypt a document can be re-obtained in the event that a user forgets their password.

KiwiVault is successful in that it maintains much of the functionality of unprotected drives. It is as portable as Java 6 is, the results of the user study show that it is usable to new users, it allows users to share documents and collaborate, and users can regain access to their encrypted documents if they forget their password.

Weaknesses of KiwiVault include that it is not truly a seamless experience and that it could be more efficient. The interface is the same in every operating system and the main interface window, while similar to most file explorers, is not exactly the same. Whenever KiwiVault loads, it scans through all the lookup files stored on the

drive and finds the files that the user has permission to access. If the user has access to a large number of files, it could take a while for all of them to load.

## 7.2 Future Work

There are several possible improvements and additions to KiwiVault.

KiwiVault's accounts are very minimal and consist of an email address and password. KiwiVault could be improved by adding additional settings associated with an account such as which folder to show at startup, custom icons for folders, and favorite locations.

All documents are shown in an icon view laid out in a grid pattern. KiwiVault could support common OS file views such as thumbnails, list, and details. The views set for folders could be saved with the user's account settings.

Every time KiwiVault starts up, it scans for encrypted documents the user has permission to access. While this is a fairly quick process, it could take too long if the user has a lot of stored documents. One way to speed up this process would be for accounts to store a list of all the files they have access to and to notify other accounts when new files become available to them.

KiwiVault runs as a separate program from the desktop but could be modified so that it instead mounts a virtual drive with access to the encrypted documents using the actual desktop file explorer. With a virtual drive, the operating system could read and write to KiwiVault as if it were a physical drive. For example, currently in KiwiVault users edit an encrypted document by navigating to the document in KiwiVault and opening it, with a virtual drive they could open the encrypted document from an application's open file dialog. Any additional controls needed (such as setting document viewers) could be provided in a side bar added to the window.

# Bibliography


[1] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, August 2001.

[2] Kingston Technology Corporation. Datatraveler vault. http://www.kingston.com/flash/DTVault.asp.

[3] SanDisk Corporation. Sandisk cruzer professional. http://www.sandisk.com/Enterprise/.

[4] Daniel Edge, Karl Richardson, Emma Rowland, and Paul Taylor. "Secutok" - Securing Removable Media. http://www.cs.kent.ac.uk/pubs/ug/2006/co600-projects/secutok/paper.pdf, March 2006.

[5] TrueCrypt Foundation. Truecrypt. http://www.truecrypt.org.

[6] Kirill Grouchnikov. Flamingo. http://flamingo.dev.java.net/.

[7] Voltage Security Inc. Voltage secuirefile. http://www.voltage.com/products/securefile.htm.

[8] IronKey. Ironkey. http://www.ironkey.com.

[9] Ravi Chandra Jammalamadaka, Roberto Gamboni, Sharad Mehrotra, Kent Seamons, and Nalini Venkatasubramanian. iDataGuard: An Interoperable Security Middleware for Untrusted Internet Data Storage. In *Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion*, December 2008.

[10] RSA Laboratories. PKCS #5 V2.1: Password Based Cryptography Standard, 2006.

[11] NewSoftwares LLC. Folder lock. http://www.newsoftwares.net/folderlock/.

[12] Sun Microsystems. Java 6. http://java.sun.com/javase/6/.

[13] Ed. P. Resnick. Rfc 5322 - internet message format. http://tools.ietf.org/html/rfc5322, October 2008.

[14] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, August 1985.

[15] Timothy W. van der Horst and Kent E. Seamons. Simple Authentication for the Web. In *SecureComm '07: Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks*, September 2007.

# Appendix A

## Sample Kiwi XML Documents

### A.1 Lookup Document

The lookup document allows users to determine if they have access to an encrypted document.

```
<?xml version="1.0" encoding="UTF-8"?>
<Lookup>
    <Nonce value="KNcswfDxrKA=" />
    <AccessList>
        <User encryptedCreatorID="8lwGMuAu2qI=" lookupID="szF/0CJidwk=" />
        <User encryptedCreatorID="6SEHpJISnwA=" lookupID="k1qdPoZJF5o=" />
        <User encryptedCreatorID="HksHp7rpzHE=" lookupID="m0RyadForu4=" />
    </AccessList>
</Lookup>
```

### A.2 Package Document

The package document is used to store encrypted files internally, to keep track of external encrypted files, and to store MACs of both internal and external files to maintain integrity.

```
<?xml version="1.0" encoding="UTF-8"?>
<PackagedMessage verison="0">
  <AuthenticatedData>
    <KAMEKRecipients>
      <KAMEKRecipient id="YVeKPH6gR8o=">
        <EncryptedKeyAndMac algorithm="AES/CBC/PKCS5Padding"
          params="zf1JceLuPU0=">mtqxEgrQWzw=</EncryptedKeyAndMac>
      </KAMEKRecipient>
      <KAMEKRecipient id="lAuyEnFgNPk=">
        <EncryptedKeyAndMac algorithm="AES/CBC/PKCS5Padding"
          params="rLasvLKOXxs=">I3DZMc/rEhY=</EncryptedKeyAndMac>
```

```
      </KAMEKRecipient>
    </KAMEKRecipients>
    <EnvelopedData>
      <KEKRecipients>
        <KEKRecipient id="X62hLN7bXKg=">
          <EncryptedKey algorithm="AES/CBC/PKCS5Padding"
            params="EPgojR1cXpI=">klBcCdonhmc=</EncryptedKey>
        </KEKRecipient>
        <KEKRecipient id="ex/sJa5hWe4=">
          <EncryptedKey algorithm="AES/CBC/PKCS5Padding"
            params="5II/qKJZsn0=">IQPG2Nyddr4=</EncryptedKey>
        </KEKRecipient>
      </KEKRecipients>
      <EncryptedData algorithm="AES/CBC/PKCS5Padding"
        params="Oc81UK4EZw8=">ZWnNdrHcgPUIQPG2Nyddr4=</EncryptedData>
    </EnvelopedData>
    <MessageAuthenticationCode
      algorithm="HmacSHA1">/Am6ATwucrk=</MessageAuthenticationCode>
    <UnauthenticatedAttributes>
      <Attribute id="ksLocation">http://kiwi.byu.edu/as</Attribute>
      <Attribute id="keyPeriod">2009-07</Attribute>
      <Attribute id="nonce">ex/sJa5hWe4=</Attribute>
      <Attribute id="creatorID">n/a</Attribute>
    </UnauthenticatedAttributes>
  </AuthenticatedData>
</PackagedMessage>
```

## A.3 Metadata Document

The metadata document stores information about an encrypted document stored in KiwiVault.

```
<?xml version="1.0" encoding="UTF-8"?>
<Metadata>
    <AccessList>
        <Viewer email="creator@cs.byu.edu" readOnly="false" />
        <Viewer email="viewer1@cs.byu.edu" readOnly="false" />
        <Viewer email="viewer2@cs.byu.edu" readOnly="true" />
    </AccessList>
    <FilePath value="/Family␣Files/Directory.pdf" />
    <CreatedDate value="Jul␣1,␣2009␣12:00:00␣AM" />
    <ModifiedDate value="Jul␣7,␣2009␣12:00:00␣AM" />
    <PreviewAvailable value="false" />
```

```
    <CreatorID value="creator@cs.byu.edu" />
</Metadata>
```

## A.4 Authentication Request

The authentication request document is used to obtain an authorization token from the authentication server. It allows for authenticating multiple email addresses at the same time.

```
<?xml version="1.0" encoding="UTF-8"?>
<AuthenticationRequest>
  <User id="user1@cs.byu.edu" />
  <User id="user2@cs.byu.edu" />
</AuthenticationRequest>
```

## A.5 Authentication Response

The authentication response document contains a share of the authorization token for each requested email address. The other share is sent via email to each user.

```
<?xml version="1.0" encoding="UTF-8"?>
<AuthenticationResponse>
    <AuthenticationServer id="http://kiwi.byu.edu/as/" />
    <KeyServer id="http://kiwi.byu.edu/ks/" />
    <Transaction id="45128428" />
    <Token expiration="2009-08" method="1rSAW" share="true"
      userid="user1@cs.byu.edu" value="YVeKPH6gR8o=" />
    <Token expiration="2009-08" method="1rSAW" share="true"
      userid="user2@cs.byu.edu" value="lAuyEnFgNPk=" />
</AuthenticationResponse>
```

## A.6 Key Request

The key request document is used by clients to request creator or viewer keys from the key server.

```
<?xml version="1.0" encoding="UTF-8"?>
<KeyRequest>
  <User id="user@cs.byu.edu" />
  <AuthenticationServer location="http://kiwi.byu.edu/as/" />
  <AuthorizationToken expiration="2009-07"
```

```
    value="NaspWXlV9dB8nwmHJYbOdg==" />
  <CreatorKeys>
    <Key period="2009-06" />
    <Key period="2009-07" />
  </CreatorKeys>
  <ViewerKeys>
    <Creator id="creator1@cs.byu.edu">
      <Key period="2009-06" />
      <Key period="2009-07" />
    </Creator>
    <Creator id="creator2@cs.byu.edu">
      <Key period="2009-06" />
      <Key period="2009-07" />
    </Creator>
  </ViewerKeys>
</KeyRequest>
```

## A.7 Key Response

The key response document contains the keys requested from the key server by the client.

```
<?xml version="1.0" encoding="UTF-8"?>
<KeyResponse>
  <CreatorKeys>
    <Key period="2009-06" value="ItE2Gk0ajW/LKOZrGCiL2g==" />
    <Key period="2009-07" value="hiFjWv+zWUY7h9Npc+uAhw==" />
  </CreatorKeys>
  <ViewerKeys>
    <Creator id="creator1@cs.byu.edu">
      <Key period="2009-06" value="/8n9dxQoBOsTHa510Rcl3Q==" />
      <Key period="2009-07" value="QkR9/CJJSrhZrab7XLH18w==" />
    </Creator>
    <Creator id="creator2@cs.byu.edu">
      <Key period="2009-06" value="IRWWFpLvui9kFS+2Pji5FQ==" />
      <Key period="2009-07" value="rNlPHlxOwa5JxJnLF1jthw==" />
    </Creator>
  </ViewerKeys>
</KeyResponse>
```

## A.8 Key Manager

The key manager document stores all the information required to obtain viewer and creator keys in addition to a cache of all previously obtained viewer and creator keys.

```
<?xml version="1.0" encoding="UTF-8"?>
<KeyManager>
  <User id="user@cs.byu.edu" />
  <KeyServer location="http://kiwi.byu.edu/ks/" />
  <AuthenticationServer location="http://kiwi.byu.edu/as/" />
  <AuthorizationToken expiration="2009-07"
    value="NaspWXlV9dB8nwmHJYbOdg==" />
  <CreatorKeys>
    <Key period="2009-06" value="ItE2Gk0ajW/LKOZrGCiL2g==" />
    <Key period="2009-07" value="hiFjWv+zWUY7h9Npc+uAhw==" />
  </CreatorKeys>
  <ViewerKeys>
    <Creator id="creator1@cs.byu.edu">
      <Key period="2009-06" value="/8n9dxQoBOsTHa510Rcl3Q==" />
      <Key period="2009-07" value="QkR9/CJJSrhZrab7XLH18w==" />
    </Creator>
    <Creator id="creator2@cs.byu.edu">
      <Key period="2009-06" value="IRWWFpLvui9kFS+2Pji5FQ==" />
      <Key period="2009-07" value="rNlPHlxOwa5JxJnLF1jthw==" />
    </Creator>
  </ViewerKeys>
</KeyManager>
```

# Appendix B

## User Study Text Responses

### B.1 Negative Feedback

1. The little read only button should be checked if it's read only. I know it says it above it, but the check mark should show up too. Because I hadn't used it before, I wasn't always sure if what i did actually worked, or if I would have to push another button to save it in some special way or something. Didn't really quite get the point of it, except I guess it somehow makes things more secure? Unless I understood the point, I probably wouldn't use it.
2. I didn't find anything about KiwiVault that I didn't like.
3. I don't know if the error message that I got at the end was due to Kiwivault or not, but I suspect that it is. I had closed the final document and the computer gave me an error message that it was still open.
4. How do I know when a file is encrypted. I feel reassured when there is some kind of file extension that lets me know my file is encrypted. In fact, I have a flash drive that has some file encryption software on it, and I have used it quite a bit. I recently started using Linux and it included some file encryption software and so I think I will start using that instead, but I definitely appreciate some good file encryption software. One thing I liked about the previous software is that it was visually obvious which files were encrypted and which were not. Based on the short period that I used Kiwi vault I did not see (immediately at least) when it was encrypted and when it wasn't.
5. I didn't like how there wasn't a right click button in some areas of the KiwiVault. I had to use keyboard shortcuts instead (for example to paste the authorization code).
6. It is just not familiar, but after a couple uses, it would be quicker to access and use everything. I also didn't like that you couldn't view pictures as thumbnails.

7. I'm still a little unsure of how it works.
8. I couldn't find anything wrong with it. I don't use jump drives that often, so I don't know how applicable it would be to my personal situation, but for those who need this type of program, it would be great.
9. My only problem is I got confused as to how to share a document with other viewers. I wish there was a help button I could press so I could get further information on problems.
10. When I had to share the document, I didn't understand how to do it at first.
11. Not being able to right click to paste information into KiwiVault.
12. When I was making an account, it wouldn't allow me to sign up until I entered in a longer password. There was no information about password requirements.
13. The only issue was that it didn't automatically open, which isn't a huge deal, but it'd made it easier to use.
14. I didn't know what it did. Is it just a password protected USB drive? The buttons on the top of the KiwiVault browser were not the most intuitive. Some were hard to find, and I didn't even use some of them. / My biggest concern is a USB virus, not that someone will take my USB drive and take my files from it, but I guess others have different concerns.
15. KiwiVault isn't integrated fully into Windows Explorer. Additionally, visual elements do not preserve the hierarchy of the information being presented, such as finding one's permission level to a document. There are no visual indicators of whether sharing is enabled or disabled without drilling down on a granular level. Finally, there are no ways I can verify or alter encryption keys or other security settings based on whatever needs I may have for a project or client.
16. I didn't run into any problems that would make me dislike it.
17. nothing
18. *-No response-*
19. For read-only files, the read-only box wasn't checked when I looked to see if it was read-only.
20. *-No response-*
21. It took me a minute to figure out what it meant by sharing a file. But once I found the Add Viewer button it was really easy.

22. *-No response-*
23. It took awhile to figure out the interface but after a few times using it, it got easier.
24. It gave the appearance that you could enable/disable read-only access when you used flash drive b to look at documents.
25. Even though the program was very user friendly there didn't seem to be any start up instructions on how to use the program. Maybe a tutorial at the beginning would be good to have.
26. I often open and close documents and folders on my flash drive. I'm worried that accessing them through kiwivault may slow me down. Can I open multiple windows of kiwivault? Can I use MyComputer to edit files, and then secure them with kiwi afterwards?
27. Seems like all doc. can be access. Although it has classified deferent level of accessibility. How about some other files that I wound not want any one to see (but I guess the log in system prevents this from happening...)
28. I might have done something wrong but i couldn't log into it in the beginning of the second part at first, but that could have been my fault.
29. It will be a problem if I forgot my password and need to pull out a file from the USB drive immediately.
30. *-No response-*

## B.2 Positive Feedback

1. I thought it was pretty usable. It pretty much seemed to make sense and work for the most part and did what I thought it should do.
2. I wasn't familiar with KiwiVault, yet I was able to use it.
3. I liked that could you share documents with people, give them restricted access, and protect personal files. I like that a lot because I have some personal documents that are on my USB drive that I would like to be protected, like my TAX forms, but they have no security settings. Anyone who might get USB drive could easily open those and have access to my personal information. This program is pretty cool!
4. It seems to have a lot of things it can do. It also has a nice picture of a kiwi.

5. I liked how by clicking once on an item you could see information about it. That actually confused me at first as to where to find some information, because I didn't realize it would be so easy. I am more used to having to search places to find the information.
6. You can invite people to see your files. This is useful for when a usb is lost, that way if your email has access to it, you have not lost everything.
7. It is very organized and easy to find what you are looking for. It isn't too complicated...
8. Super easy to use. My mom could even use it, and that's sayin something.
9. Once I got the hang of it, I felt like KiwiVault was very user friendly, and even someone who has very little experience with computers, like me, would be able to use it! I also love that you can choose what type of access to give those who you share with.
10. It seems like a safe program because my documents would be safe even if my USB drive was stolen.
11. Easy to find and work with secure documents.
12. It was very easy to use–very self-explanatory. I had very little trouble with the tasks, and I think most people could figure it out. I also like the name. I like that I would be able to keep all the info on my flash drive safe.
13. It was quick and there wasn't any real issues with usability.
14. Easy to use. Double clicking on the icons opened things up. You could quickly move the documents to the desktop and use them.
15. KiwiVault makes a good stab at bringing document encryption to the masses, something that's desperately needed.
16. I liked the fact that it would be easy to password protect my device.
17. I liked it.
18. The fact that you were able to determine who could and could not have access to your documents.
19. Easy to use. Attractive interface.
20. *-No response-*
21. I think it's really visually appealing and uncluttered. Also, it's neat to be able to share files that quickly without emailing and uploading attachments, etc.

22. It seems very useful and accessible to the lay person. It makes sharing files easier, and I like the option of sharing either as read-only or with full access.
23. Quick and simple. Very user friendly. I would use it.
24. A great user interface. I liked that I could easily find things. It had a similar layout to other applications that I have used before.
25. It was very user friendly, not too confusing. The controls were very simple. I like that you can drag and drop into it, that is very handy.
26. When I did access through kiwi, the files were automatically updated. / The access that I can give to others is nice too, and the functions for giving and receiving access are simple. The program is fairly easy to use.
27. The restriction on accessibility makes sure the file is only available to autherised individual–keeping privacy. and it is very easy to use.
28. It was pretty easy to use and it seemed pretty secure and provided an easy way to share files with friends and colleagues
29. The layout looks good. It's secure.
30. It prevents anyone from modifying files saved on the flash drive without a password. It is very easy to use, and I think anyone could figure out how to use it very quickly. I also like the visual design, it looks simple and there aren't a lot of extra buttons or menus.