CS CAPSTONE REQUIREMENTS DOCUMENT

OCTOBER 28, 2017

PRIVACY PRESERVING CLOUD, EMAIL, AND PASSWORD SYSTEMS

PREPARED FOR

OSU

ATTILA YAVUZ _____ _____
Signature *Date*

PREPARED BY

GROUP 38

THE CLEVERLY NAMED BUNNY TEAM

ANDREW EKSTEDT _____ _____
Signature *Date*

SCOTT MERRILL _____ _____
Signature *Date*

SCOTT RUSSELL _____ _____
Signature *Date*

Abstract

This document is written using one sentence per line. This allows you to have sensible diffs when you use LaTeXwith version control, as well as giving a quick visual test to see if sentences are too short/long. If you have questions, “The Not So Short Guide to LaTeX” is a great resource (https://tobi.oetiker.ch/lshort/lshort.pdf)

CONTENTS

1 INTRODUCTION

1.1 Purpose

The purpose of this document is to outline the project requirements for the ”Privacy Preserving Cloud, Email and Password Manager” Capstone project. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be presented at OSU’s Spring 2018 Engineering Expo as a proof of concept and a reference for developing the first version of this specific DSSE scheme.

1.2 Scope

The ”Privacy Preserving Cloud, Email and Password Manager” Capstone project at it’s roots a research oriented project that aims to find a way to implement the DSSE scheme proposed in the research paper ”Practical Techniques for Searches on Encrypted Data” [3].

This implementation will be executed through command line prompts and hosted on OSU’s engineering servers. A user can use this system with a client - server model to perform actions, such as search or update, on a ”cloud-based” database. User interface is not considered a priority as this project is not intended to be used in any commercial capacity.

1.3 Definitions, acronyms, and abbreviations

Term	**Definition**
User	Someone who interacts with the system
Encryption	The process of encoding a message or information in such a way that only authorized parties can access it.
Searchable Symmetric Encryption (SSE)	Allows a client to encrypt its data in such a way that this data can still be searched.
Client	A computer application, such as a web browser, that runs on a user’s local computer or workstation and connects to a server as necessary
Server	A software program, such as a web server, that runs on a remote server, reachable from a user’s local computer or workstation.
SFTP	Secure File Transfer Protocol
insert term	insert definition
insert term	insert definition
insert term	insert definition
insert term	insert definition

1.4 References

[1] http://eecs.oregonstate.edu/capstone/cs/capstone.cgi?project=334

[2] http://web.engr.oregonstate.edu/ yavuza/Yavuz_DSSE_SAC2015.pdf

[3] https://dl.acm.org/citation.cfm?id=884426

[4] Cash, David et al. ”Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation” https://eprint.iacr.org/2014/853

1.5 Overview

The rest of the document will include two more section. Section 2 will be overview of the project system, giving a product perspective, defining the functions of the system, listing user characteristics, describing system constraints, explaining what assumptions and dependencies we are making in regards to the implementation of this system.
Section 3 will include the requirements for the project. This will be the most detailed section describing the specific objectives and required functionality of the system.

2 Overall description

2.1 Product perspective

The main purpose of the project is to investigate ways to integrate SSE with common internet applications. Specifically, we will investigate ways to apply SSE to cloud storage and email.

The system will have three main components: basic SSE implementation, cloud storage integration, and email integration. The basic SSE implementation will be split into client and server programs. The server will provide access to the encrypted search index, and the client will allow the user to perform searches on the server. Cloud and email integration will take the form of additional client programs which download documents or emails from a remote server and upload them to the SSE server.

2.2 Product functions

The central SSE module will be an implementation of [4]. It will provide functions to create an encrypted search index, perform search queries, add documents, and delete documents.

An intended research topic is to attempt to speed up the SSE by parallelizing certain operations. Another intended research topic is to find ways to reduce the storage space used by the encrypted index. We intend ultimately to compare the performance with [2].

The cloud storage module will download files from a cloud storage provider like Dropbox and add them to the encrypted search index.

The email module will download emails from a mail provider like Google Mail and add them to the encrypted search index. We will first implement a batch mode program and then investigate building a daemon which downloads emails in the background.

2.3 User characteristics

The primary audience for this software system is technical users who are interested in a proof-of-concept implementation of SSE. It is not a goal of this project to target non-technical users.

2.4 Constraints

2.5 Assumptions and dependencies

Completing the cloud storage and email modules will depend on having a basic SSE module in place first. Once the basic SSE functionality is built, we can work on optimizing it in parallel with the work on cloud integration and email integration.

3 SPECIFIC REQUIREMENTS

3.1 External interfaces

3.1.1 User interfaces

The primary user interface to the SSE system will be a pair of client-server programs. A user will be able to launch the server from the command line, and to query the server with a command line client program. This initially will be done in an offline self contained client on a single computer but will be adapted to a Client Server system and finally an email system. These are the main user functionalities of the SSE: Keyword Search, Add Files, Delete Files, Build Data Structure, and Save and Exit. The Keyword Search asks for a specific keyword input and than searches the entire data base for any files that contain the said keyword. The Delete Files call will delete a filename from the system if it exists. Add file will add a file to the database, if it already exists it will prompt for an overwrite on the said file. Successful implementation of these functionalities is a goal of this project.

The Gmail user interface will have the same functionality requirements as the Online Client-Server. However, now we are working with cloud storage data. The data is not stored on a local computer but now on Gmail's cloud system. As a user you will have the ability to manually force an email download as well as automatically updating the system every 2 minutes while the connection is open through a Gmail Daemon. The Client will still be able to Search, Update, and Delete data on the cloud just as the offline client does.

3.1.2 Software interfaces

The implementation of SSE will be take the form of a C++ API which can be used by the rest of the system. Software of previous work from Atilla's implementation of the bit matrix provided by Thang will be used as a main software example and can be used as a template for many of the server/client connection calls. However since the algorithm itself is completely different it will mostly be used to allow the team to focus more coding on the process of research and implementation of David Cash's algorithm. Asymptotically David Cash's algorithm has a faster Big O complexity speed. This will be tested by running 50 calls to Search, Update, and Delete on both Atilla's Bit Matrix and our David Cash implementation. Parallelism should also be possible with David Cash's algorithm and optimization of the performance will be worked on if time permits.

Once we have a basic software package with the ability to run a David Cash SSE we will than move from a Client Server connection to communication with a cloud database. The cloud storage module will communicate with a cloud service using Dropbox or Google Drive in order to download files and add them to the search index. This will be a research area before implementation.

Next will we focus on Email Client Server integration. We will research the ability to connect our system with the Gmail interface. This is of key importance to the project as it provides a real life uses for David Cash's SSE. The email module will communicate with remote email servers via POP3 or IMAP in order to download email messages. This will focus on researching Secure File Transfer Protocols (SFTP) as well as general knowledge of Email-Client servers. An automatically updating background process will run while the Client-Gmail connection is open every 2 minutes to update new incoming emails.

In conclusion the three main software products that we will be delivering through this project are: an implementation of David Cash's search algorithm, a cloud based Client-Server where data can be updated and stored and finally a Email Client Server integration.