CS CAPSTONE REQUIREMENTS DOCUMENT

NOVEMBER 14, 2017

PRIVACY PRESERVING CLOUD, EMAIL, AND PASSWORD SYSTEMS

PREPARED FOR

OSU

ATTILA YAVUZ

PREPARED BY

GROUP 38

THE SECRET BUNNY TEAM

ANDREW EKSTEDT

SCOTT MERRILL

SCOTT RUSSELL

Abstract

This document describes the requirements for the Privacy Preserving Cloud, Email, and Password Systems Senior Capstone project. This will be accomplished by explaining in detail the benchmarks for how our group will implement David Cash's DSSE algorithm and integrate that onto an email and cloud system.

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 is a research-oriented project that aims to find a way to implement the DSSE scheme proposed in the paper ”Dynamic Searchable Encryption in Very-Large Databases” [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.
SSE (Searchable Symmetric Encryption)	Allows a client to encrypt its data in such a way that this data can still be searched.
DSSE (Dynamic Searchable Symmetric Encryption)	A SSE scheme where documents and keywords can be incrementally added and deleted after the fact, without completely rebuilding the encrypted search index
Cash-DSSE	A DSSE scheme described by Cash et al. in [3]
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

1.4 References

[1] Attila Yavuz. ”Privacy-Preserving Cloud, Email and Password Systems.” *CS Senior Capstone* http://eecs.oregonstate.edu/capstone/cs/capstone.cgi?project=334

[2] Attila Yavuz and Jorge Guajardo. ”Dynamic Searchable Symmetric Encryption with Minimal Leakage and Efficient Updates on Commodity Hardware” *ACM SAC 2015* http://web.engr.oregonstate.edu/ yavuza/Yavuz_DSSE_SAC2015.pdf

[3] David Cash et al. ”Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation” *Cryptology ePrint Archive, Report 2014/853* 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 [3]. 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

Our client has requested that we implement the software in C++, which limits the choice of libraries we can use.

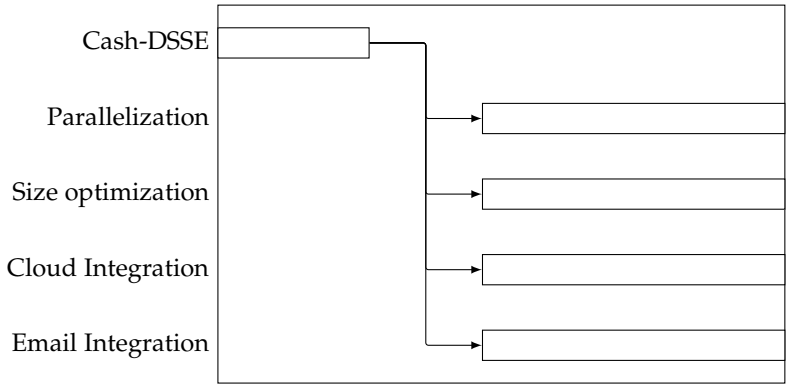

Figure 1. Gantt chart of the project dependencies

2.5 Assumptions and dependencies

We assume that we can find a cloud provider which offers a sufficiently open API for accessing documents, and that there is a good C++ library available.

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.

See 1 for a Gantt chart showing the dependencies.

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.
- The user will be able to query the server using the client from the command line.

From the basic client program, a user will be able to:

- search the document store by a single keyword, and receive a list of a all documents containing that keyword.
- add a file to the search database.
- delete a document from the search database.

This is the main functionality of an SSE system.

The cloud user interface will have the same functionality requirements as the basic client. Additionally, the user will be able to:

- tell the client to download documents from a previously-configured cloud service, to be added to the search database.

The email user interface will have the same functionality requirements as basic implementation. Additionally, a user will be able to:

- tell the client to check for new emails, which will be downloaded and added to the search database.
- launch a daemon which automatically downloads emails every 2 minutes.

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. Our client has provided a software implementation of [2], which we will use as a base for our work.

- Atilla's implementation will be used as a template for many of the server/client connection calls.
- Since the algorithm itself is completely different it will 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 runtime complexity. 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. This along with optimization of the performance will be worked on if time permits.

Once we have a basic software package with the ability to run a Cash-DSSE we will than move from a Client Server connection to communication with a cloud database.

- Cloud storage module will communicate with a cloud service using a service such as Dropbox or Google Drive in order to download files and add them to the search index.

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.
- An automatically updating background process will run while the connection to Gmail is open every 2 minutes to update new incoming emails.

Below are listed the three main goals of this project. In addition to these we will also optimization and parallelism of Cash-DSSE with time allowing.

1) An implementation of Cash-DSSE
2) A cloud based Client-Server where data can be updated and stored.
3) An email Client Server integration.