CS CAPSTONE PROBLEM STATEMENT

OCTOBER 16, 2017

PRIVACY PRESERVING CLOUD, EMAIL, AND PASSWORD SYSTEMS

PREPARED FOR

OSU

ATTILA YAVUZ _____ *Signature* _____ *Date*

PREPARED BY

GROUP 38

ANDREW EKSTEDT _____ *Signature* _____ *Date*

SCOTT MERRILL _____ *Signature* _____ *Date*

SCOTT RUSSELL _____ *Signature* _____ *Date*

Abstract

We want to be able to store private data on the cloud in such a way that the provider cannot decrypt it, yet we can still perform keyword searches efficiently. Our project will be to implement and build on algorithms developed by Attila Yavuz and David Cash which could potentially solve this problem. We will demonstrate practical searchable encryption of documents, email, and possibly passwords.

Contents

1 Motivation

In today's technological world privacy is of key importance. Faced with the growing public panic over mass surveillance, companies have caught onto the significance of security and are creating products with strong encryption built in. Web browser makers like Chrome and Mozilla are pushing for a secure-by-default world where every website has an SSL certificate, marking those without as untrusted. Apple is positioning themselves as a privacy-conscious company by building strong encryption features into recent iPhone models. Billions of users are flocking to secure chat apps like Signal and WhatsApp.

Despite these advances, there are many areas where security is lacking. Encrypted email is still an unsolved problem. You can use GPG to encrypt messages, but then you lose the ability to quickly search your messages. This problem can be addressed by giving your email provider a copy of your decryption keys, but now they have the ability to read all your email, defeating the purpose of encryption. A similar problem exists for cloud storage systems: Google Drive will transparently encrypt your files for you, but Google necessarily hangs onto a copy of the encryption keys, which means that Google has the ability to decrypt your files any time they want.

There has been some research into searchable encryption, but no open-source implementations of the algorithms that have been developed. The need for a fast and efficient open-source implementation of dynamic searchable encryption is the basis for our research.

2 Goals

Our primary goal is to demonstrate a practical, open-source implementation of searchable encryption on a cloud provider.

Attila's research group has developed an implementation of an algorithm for searchable encryption [1], but it only runs locally, not over the network. To convincingly demonstrate that searchable encryption is practical, it needs to work over a network. We will build a pair of client-server programs that can do searchable encryption. Ideally, we will be able to hook it up to something like Amazon S3 so that all the data is stored in the cloud.

Part of the project will be to research more efficient data structures for searchable encryption. The implementation that Attila's group did uses a fairly naïve data structure; it performs very well despite that, but there is a possibility that a more complicated data structure could provide further performance boosts. We will build on research by David Cash for this step. [2]

The second project is searchable encrypted email. Once we have built a basic working implementation of searchable encryption, we would like to hook it up to email. This will probably involve writing a daemon which downloads messages from an email provider like gmail and adds them to an encrypted database. Presumably we would want to throw some email-specific UI and search functionality on top. This part of the project is TBD and we expect to learn more about what is possible after we complete the first part of the project, and as we start working on this part.

A third, noncritical, project is to implement a password manager using ORAM (oblivious RAM), which is a random-access data structure that hides memory access patterns from observers. This is very much a "stretch goal" which we would only work on after completing the first two projects. It's fine if we don't get to it at all.

3 Metrics

There are two metrics, equally important: correctness and performance.

The first metric is correctess. The software should return the correct set of documents when searching for a keyword. It should be able to dynamically add and delete files from the search index and return correct search results using the modified index.

The second metric is performance. For the secure cloud, we want to aim for under a hundred milliseconds to search for a file. Attila has measured the previous implementation to be able to perform a search in around 10 milliseconds or less, even with an index containing millions of files, so this should be doable. Performance for the email database is TBD. The password manager (if we get to it) will be less performant due to the overheads of ORAM, but we want to aim for 100 milliseconds.

REFERENCES

[1] A. A. Yavuz and J. Guajardo, "Dynamic searchable symmetric encryption with minimal leakage and efficient updates on commodity hardware," Selected Areas in Cryptography (SAC) 2015, Sackville, New Brunswick, Canada, August 2015, http://web.engr.oregonstate.edu/~yavuza/Yavuz_DSSE_SAC2015.pdf.

[2] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rou, and M. Steiner, "Dynamic searchable encryption in very-large databases: Data structures and implementation," Cryptology ePrint Archive, Report 2014/853, 2014, http://eprint.iacr.org/2014/853.