Capstone Project: Proposal for an Open Source Implementation of V-OXT, a Scalable Verifiable Encrypted Search Protocol

Daniel Intskirveli and Johannes Christ

November 2015

Abstract

In recent decades a number of factors have made the outsourcing of the storage of digital information to “the cloud” indispensable. The growth of the personal computer industry, the rise of the smart-phone, the advent of “big data”, the need to persist data to avoid its loss in the event of a hardware failure, and the large scale use of the Internet have all contributed to the trend—increasingly, data is stored not on the hard drives of personal computers but in specialized data warehouses (the cloud) which offer global access via the Internet, resiliency to hardware failure, and cheap storage. The use of the cloud brings with it special security concerns. Most notably, user data should not, under any circumstances, become accessible to others. One solution for this problem is the encryption of all data in the cloud and, while this is an effective way to keep data safe from malicious data warehouse operator or a security breach, it does not allow for partial access of data, i.e. data owners must download entire files to search through them. Experimental implementations for searchable symmetric encryption (SSE) that are both secure and scalable exist, but they do not offer protection from malicious servers. Our proposed project is an improvement on the OXT protocol developed by Cash, et al.[1] that does not rely on the assumption that the server storing the data is “honest”. Clients using the proposed system will be able to store and search through vast collections of their documents without risking the compromise of their data in the event of security breach.

1 Motivation and Benefits to the Community

The primary benefit of this project is that it makes the outsourcing of data storage (a growing industry trend) secure and easy. Data storage providers are a dime a dozen on the open Internet today. On-line cloud services such as Dropbox and Google Drive are explicitly used by millions of people to store every day data on the cloud. Furthermore, services like email and Facebook (used by billions) provide implicit data storage, with files such as photos stored as part
of users’ profiles or inside the messages they send. It is not uncommon also for organizations that require the storage of large quantities of data but do not have the resources to maintain their own hardware to subscribe to cloud data warehousing solutions like Amazon Web Services or Microsoft Azure. While some of these servers encrypt the user data that they store, none of the services mentioned about encrypt data with a client’s secret key. In other words, even if the data is encrypted, the key used for encryption is on the server itself, which makes it somewhat less effective in the event of a security breach. In addition, each of these services require a user to trust them—once the data is in the cloud, we do not truly know what is happening to it. A malicious server could be analyzing and redistributing user data. The V-OXT protocol that we aim to implement removes the requirement for trusting the server and all data is encrypted by the user. The adoption of such a protocol by the industry will give regular consumers and organizations alike an alternative that provides total safety for their data.

The community will benefit also because we aim to open source the implementation itself and use open source (and horizontally scalable) frameworks. This will allow engineers to contribute to our solution to provide further improvements or patch problems that may arise. We hope also that the open source and free nature of the software will drive faster adoption of the the protocol to provide safe on-line data storage faster.

Finally, perhaps the most important beneficiaries of this work are governments and international organizations. Because of the nature of their data storage needs, these organizations typically opt for internal hardware that is tightly secured with network boundaries in spite of the cost of the infrastructure. The existence of an implementation of a protocol that allows for safe on-line data storage opens the possibility for more data intensive work on a smaller budget.

2 System

We will develop an implementation of a verifiable encrypted search protocol proposed in a recent paper by Dr. Rosario Gennaro from the Center for Algorithms and Interactive Scientific Software (CAISS) at CCNY and his team. Professor Gennaro and the team improved on a protocol called OXT, the result being a new protocol they called V-OXT. The current solutions for Scalable Searchable Encrypted (SSE), as far as we are aware, do not consider the implications of a malicious data server, and do not have security measures in place to defend against this situation. A non-production-ready implementation of the OXT protocol has been developed in a joint project by IBM and U.C Irvine under a grant from IARPA. This implementation is scalable (has sub-linear performance) and is the only known working proof-of-concept of OXT. We managed to get hold of the OXT project’s code through Professor Gennaro to test it and explore the code-base. There are unfortunately several issues that we encountered with the
IBM/U.C. Irvine implementation of the protocol:

- “dirty”, unmaintained code-base
- closed-source
- non-flexible paradigm for data storage
- requires the use of the SQL query language
- difficult to scale

We therefore aim to build an implementation of V-OXT from scratch that is open source, uses modern software frameworks, does not restrict users to SQL, and uses Professor Gennaro’s modified version of the OXT protocol in order to protect the integrity of the data when the server is malicious or dishonest.

Our implementation of V-OXT will use newer technologies that make use of parallel processing paradigms and will provide the flexibility that is necessary to store and retrieve data quickly in a generic way and to generate the encrypted database in the first place. We will take advantage of the open source MapReduce framework as implemented in Apache Hadoop for parallel processing of data and we’ll use Apache Cassandra as a distributed data store for high-performing horizontally-scalable storage.

A prerequisite for a working V-OXT solution is preprocessing, wherein the client generates indexes necessary for the function of the encrypted database. This process is computationally expensive and in IBM’s implementation it is static—it must be recreated every time a new file is added to the system, and the data as well as its schemas are closely coupled to the code. We will use the MapReduce as it is provided by Apache Hadoop for distributed pre-processing of the encrypted database. The data structures generated in the pre-processing stage make SSE possible—they allow the user to search through her encrypted documents stored on the server (the cloud) using encrypted key words.

One of the purposes of this project is to build the first working implementation of V-OXT, Professor Gennaro’s modification of the OXT protocol that ensures search result integrity in the case of a dishonest server. This essentially involves two changes to the OXT protocol. One change is the addition of a cryptographically secure signature to one of the indexing data structures that prevents the server from maliciously returning incorrect search results. In this scheme, however, a malicious server may still return an empty set of results because it does not have to provide a signature in this case. This leads us to the second change, the use of a data structure called an authenticated set, which uses cryptographic accumulators to verify that an element exists in a set[3]. An authenticated set will be constructed from the set of words that exist in the documents, and the server will have to prove that a word does not exist in that set when sending an empty response.
3 Background and Literature

There are two published papers, [1] and [2], that are historically relevant for the work outlined in the paper. The main concepts developed in these papers have to do with the implementation of high performance search on encrypted data and the retrieval of the encrypted information. In short, the papers outline the development and inner workings of the OXT protocol.

Techniques other than those used in the OXT protocol exist to implement SSE and some are even used in production. However, OXT provides the most attractive trade-offs between performance and security. For example, a well-known implementation of SSE built at MIT CSAIL called CryptDB offers little protection from a malicious server, since querying the encrypted data reveals the entire plain-text contents of entire relevant columns to the server. In addition, conjunctive queries perform with linear time complexity[4] (this is considered not scalable).

The first paper explains the implementation of a sub-linear algorithm to search on encrypted data. The basics of this algorithm is the implementation of highly efficient paradigms to create sets for the keywords that can be searched on later. For example, when a single keyword is searched against this set (called the T-set) then it will create a response consisting of the data that comes associated with this keyword. In specific this data consists of an encrypted pointer to a file. We will build upon this response in the sense that we will add an authentication method to it. For the case when multiple keywords are searched (conjunctive search) this search will be performed on another set called the X-set, this set is optimized for creating the intersection between the individual responses of each keyword. The main performance gain comes from the usage of Bloom filters to speed up the searches on multiple keywords. This filter consists of probabilistic analysis that can be run efficiently before venturing to perform a full search.

The second paper deals with the main concern of this system: security. When an index is created then it leaks some information than can be susceptible for usage with malicious intentions. For example, let’s say that the server that contains the index has been taken over by a hacker, then the question is: What this hacker can learn from the index? What can he infer if there is a memory of the most frequently searched keywords? Up to what point can he emulate the behavior of this server in order to trick the client to get more information about him? This paper has investigated the data leakage of the implementation of the first paper and provides a better leakage profile by extending an implementation of arbitrary Boolean queries to the SSE provided in the first paper.

The last paper used in this project has the verifiable feature added to the before mentioned paradigms. This paper hasn’t been published yet by Professor Rosario so we will be in contact with him in order to provide certain references.
4 High Level Overview

The OXT system (and, by extension, our implementation of V-OXT) essentially has two parts to it: the pre-processing stage, which entails the encryption of documents and then generation of indexes that make SSE possible, and data access, which happens at any time after the encrypted database is generated. There are two models for access data in any SSE protocol. The first model is the simple single-client case, in which a data-owner accesses her own data. The second model is the multi-client model wherein the data-owner grants access to one more more third parties to search the data-owner’s data for specific words. The multi-client model is made non-trivial by the cryptography involved in making the model secure.

4.1 Pre-processing

In the MIT/U.C. Irvine implementation of SSE, the preprocessing stage is done on a single client computer, with some parallelism achieved through the use of multiple threads. This is only scalable to a certain point, and may take a prohibitively long amount of time for larger amounts of data (in our tests, it took around an hour to generate the encrypted database for 6 gigabytes of data). We propose using the MapReduce paradigm for the preprocessing stage (encrypting documents, uploading them, generating indexes that enable searching the encrypted data set). This will give clients the option to initially generate the database on a cluster of computers that they own. They can later update the database on a single client machine, as this process is far less expensive. We think this is a good solution because the requirement for a Hadoop cluster is temporary—several departments within an organization can share one and still outsource their data storage, or an organization can rent a cluster for a few hours.
4.2 Single-Client Mode

After the generation of the encrypted database, a client can perform a query to search through her documents. This is the simplest case for the use of the V-OXT protocol:
1. Client encrypts search terms with her secret key
2. Client sends query including encrypted search terms to server.
3. Server uses V-OXT indexes to search and perform Boolean operations.
4. Server responds with encrypted indexes and signatures (or authenticated set verification in the absence of response).
5. Client can decrypt indexes and download documents if necessary.

4.3 Multi-Client Mode

Another model for accessing a client’s data is the multi-client mode, which V-OXT also supports. In this model, a third-party requests access for searching the client’s data for some fixed set of key words. The client can grant access for some words and reject access for others, sending tokens back to the third-party for the search terms that the third-party is allowed to search for. The third-party can then communicate directly with the server using the tokens provided by the client:
1. Third-party contacts data owner, asking for permission to query for some search terms

2. Data owning client responds with cryptographically secure tokens for searching the terms for which it is granting access.

3. Third-party uses tokens provided by data owner to search through data owner’s documents.

5 Collaboration

We are Daniel Intskirveli and Johannes Christ, a team of two working on an open source implementation of the V-OXT protocol designed to be flexible and horizontally scalable. Although this is a fairly complex project, we believe that our collective skill set is sufficient for the work outlined here. Daniel has considerable industry experience working with “big data” and is familiar with the Hadoop framework. Johannes’ expert knowledge of the Java language as well as his experience with front-end software will help us implement protocol specific features and user-friendly applications quickly. In addition, we will have the opportunity to consult with and learn from Professor Gennaro himself. This will help us build an optimal implementation of the V-OXT protocol. It is possible that during the course of the project we will come up with optimizations not mentioned in any of the papers cited here. In this case we aim to write a report outlining our ideas for the benefit of the community.

5.1 Roadmap

1. Proof-of-concept Java application. **2-3 weeks.** A simple Java application that emulates the protocol without the complexities added by parallel encrypted database generation. It should use real cryptography libraries and should have accurate algorithm implementations. This will allow us to learn about the protocol and sort out any issues early in the process.

2. MapReduce code for V-OXT index generation. **1-2 weeks.** Write the MapReduce code for the generation of V-OXT indexes using Apache Hadoop APIs. A lot of the code will be shared with the proof-of-concept. Test it on a single node.

3. Server application. **1-2 weeks.** Write the code for the server. It should support uploading documents, uploading V-OXT index data structures, and searching. It needs to interface with Apache Cassandra as well as some file store.

4. Integrate MapReduce code with server. **1-2 weeks.** Modify the MapReduce code to integrate correctly with the server, uploading documents and V-OCT index data structures as needed.
5. Full-featured test on a single node with small data set. 1 week. Perform whatever work is remaining to get the entire system running on a single computer.

6. Scalability test using AWS (or other) with large data set. 2-3 weeks. Make the configuration of the system abstract and run it on AWS, or another cloud provider that supports Hadoop. Test it with a larger dataset (such as a subset of Wikipedia).

7. Client software. 1-2 weeks. Write an application that makes using the system user-friendly.

6 Conclusion

We hope the proposed open source implementation of the V-OXT protocol has the dual benefits of helping the community and teaching us about cryptography. We think the work involved is both difficult in nature and reasonable provided the time frame given. We look forward to providing a design document that explores deeper into the V-OXT algorithm and our implementation of it.

References


