Preventing Memory Access Pattern Leakage in Searchable Encryption

Shauna Michelle Policicchio, University of Pittsburgh
Attila A. Yavuz, Oregon State University

Abstract
With the recent popularity of outsourcing private data to the cloud, there is an increased interest in privacy-enhancing technologies. These technologies were developed to maintain the privacy of a user's identity and have evolved alongside the Internet. The current, most popular solution to maintaining the privacy of this data is with encryption. Searchable encryption was developed to allow a user to search an encrypted data repository without decrypting the data first, but it is susceptible to information leakage through memory access patterns. To address the leakages, oblivious RAM obfuscates the memory accesses of data, so that access patterns do not leak information about the stored data. This poster will look at combining oblivious RAM with encrypted search to prevent access pattern leakage and the associated problems, as well as other proposed solutions.

Keywords: Privacy-enhancing technologies, searchable encryption, oblivious RAM
Citation: Policicchio, S.M., Yavuz, A.A. (2015). Preventing Memory Access Pattern Leakage in Searchable Encryption. In *iConference 2015 Proceedings*.

Acknowledgements: Shauna is a NSF CyberCorp SFS student pursuing graduate studies in Information Security at the University of Pittsburgh and is supported by the NSF-DGE Award #1027167.
Contact: smh137@pitt.edu, attila.yavuz@oregonstate.edu

1. Introduction

Since the late 1990s when the World Wide Web became popular, people have been worried about the privacy of information they share over the internet, leading to the development of *privacy-enhancing technologies*. These technologies were developed to maintain the privacy of a user's identity when sending email, surfing the web, et cetera, and have evolved along with the Internet. The movement to outsource data and processing to a cloud provider has increased interest in these technologies. The current, most popular solution to maintaining the privacy of this data is encryption. Traditional encryption requires an entire encrypted file to be decrypted to find anything out about the file contents. This is not always desirable, for example to search for files containing a particular subject all files would have to be downloaded and decrypted before they could be searched. Searchable encryption was developed to allow a user to search an encrypted data repository without decrypting the data first, to protect the privacy of the data. However, searchable encryption is susceptible to information leakage through memory access patterns. To address the leakages, oblivious RAM obfuscates the memory accesses of data, so that access patterns do not leak information about the stored data. This poster looks at the problem of memory access pattern leakage in searchable encryption and discusses oblivious RAM as a possible solution, but also points out some problems with that solution.

2. Searchable Encryption

When large amounts of encrypted data are stored in a cloud computing environment, it is desirable to use *searchable encryption* on the data so that only relevant data needs to be retrieved. Searchable encryption is achieved by providing a keyword to the server that returns any encrypted files containing that keyword, without decrypting the data first. Search is implemented in two ways: a sequential scan to find the keyword in each document or an index search for the keyword to discover documents (Song, Wagner, & Perrig 2000).

While searchable encryption maintains the encryption of data for keyword search, it is vulnerable to memory access pattern leakage, which can allow inferences to be made about data stored in particular locations. In other words, while the contents of the keyword search queries and files being stored are protected by encryption, the location of the file that is returned can be leaked to an adversary watching the server. In this scenario, the threat model is an honest-but-curious server administrator who can watch accesses being made on the server. Figure 1 depicts this threat model. The client, Bob, sends encrypted queries q1, q2, q3 to the server where his encrypted data is stored. The honest-but-curious server administrator, Oscar, can watch Bob's encrypted queries and see which physical memory locations are accessed to return the results. With repeated queries, Oscar can make assumptions about the data that is stored. For example, if every time these queries are performed Bob's company performs a stock exchange, Oscar can make assumptions about the contents of data at locations X, Y, Z, and also predict user actions after the queries are made (Pinkas & Reinman 2010). Oscar can also determine the importance of data stored at specific locations based on the frequency of access.

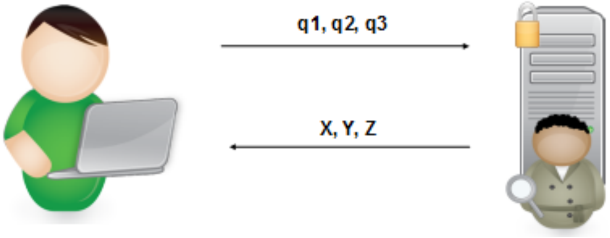

Figure 1: Oscar views access pattern leakage from Bob's queries.

3. Oblivious RAM

Oblivious RAM provides a solution to memory access pattern leakage by obfuscating memory accesses so an adversary cannot distinguish one request from another (Goldreich & Ostrovsky 1996). This is accomplished by issuing multiple data requests for every one actual request, thus increasing overhead and security. For a single data request, the server returns several blocks of data for every requested block of data. These blocks are stored in a cache on the client, then periodically evicted back to a random location on the server. The addition of dummy blocks, or encrypted blocks of random data, adds to the obfuscation. By adding oblivious RAM to a searchable encryption scheme, we can increase the privacy of the data.

Figure 2 depicts the same access pattern scenario as Figure 1, with the addition of oblivious RAM to obfuscate access patterns. Oscar can still view Bob's access pattern, but the oblivious RAM obfuscates the pattern by adding in extra file accesses and returning extra blocks of data along with the requested blocks. The data blocks are stored in Bob's local cache and eventually rewritten to the server in different locations. As a result, Oscar can make no inferences based on Bob's access patterns.

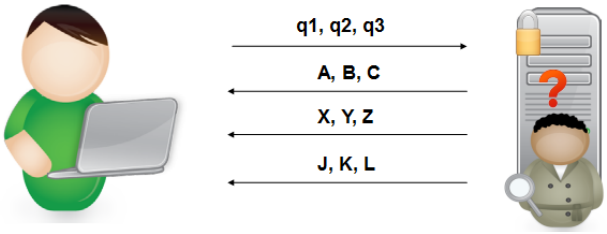

Figure 2: Due to the use of oblivious RAM, Oscar can no longer make predictions based on Bob's physical memory accesses.

4. Research Challenges

The addition of oblivious RAM would prevent memory access pattern leakage in searchable encryption. However, the high overhead costs associated with privacy-enhancing technologies such as searchable encryption and oblivious RAM play a key role in their lack of adoption individually. The combined overhead of both technologies together would be very costly.

Part of the problem is the block sizes necessary for each technology. Oblivious RAM can achieve constant client storage requirements and relatively efficient overhead with large block sizes of data, which is not compatible with small document-key pairs used in searchable encryption schemes. With a smaller block size, oblivious RAM requires higher client storage or higher bandwidth, since more blocks are used to store the same amount of data than a scheme using large block sizes. Oblivious RAM can also be added to a trusted processor (Fletcher, 2013). This increases the overhead of the data processing on the server, rather than the networking bottleneck that client-server oblivious RAMs run into. However, on top of costly overhead, cloud customer data may be stored on a server with several other customers' data, providing many challenges to the installation of a trusted oblivious processor.

5. Other Solutions

More recent solutions to access pattern leakage acknowledge oblivious RAM as a solution but discount it due to the overhead cost. Instead, these solutions sacrifice some security for the sake of better performance. Blind Storage hides file data from the server until a file is downloaded with less overhead than oblivious RAM (Naveed, Prabhakaran, & Gunter 2014). In Blind Storage, the server does not discover the length of a file, how many files are stored, file name, or contents, but can notice if the same file is downloaded twice in a row. A searchable encryption scheme is built on top of blind storage, providing a more secure solution with competitive efficiency. Another solution rebuilds levels of the data structure that is used on the server on every update, using an oblivious RAM or oblivious sort to shuffle the data blocks (Stefanov, Papamanthou, & Shi 2013). This solution considers specific leakages to be acceptable, such as the hashes of the keywords being searched. These solutions do not contain full protection from leakage, but provide added security at a reasonable cost.

6. Conclusion

The access pattern leakage found in searchable encryption can be solved using oblivious RAM, but current algorithms and implementations require a costly overhead when combined with searchable encryption. Current searchable encryption schemes acknowledge access pattern leakage but do not address it, while others trade off security for practicality.

References

Fletcher, C. (2013). Ascend: An architecture for performing secure computation on encrypted data.

Goldreich, O., & Ostrovsky, R. (1996). Software protection and simulation on oblivious RAMs. *Journal of the ACM*, *43*(3), 431–473.

Naveed, M., Prabhakaran, M., & Gunter, C. A. (2014). Dynamic Searchable Encryption via Blind Storage. *2014 IEEE Symposium on Security and Privacy*, *2014*, 639-654.

Pinkas, B., & Reinman, T. (2010). Oblivious RAM Revisited. *Advances in Cryptology–CRYPTO 2010*, 502–519.

Song, D. X., Wagner, D., & Perrig, A. (2000). Practical techniques for searches on encrypted data. *Proceedings of the 2000 IEEE Symposium onSercurity and Privacy*, 44–55.

Stefanov, E., Papamanthou, C., & Shi, E. (2013). Practical Dynamic Searchable Encryption with Small Leakage. *Network and Distributed System Security Symposium (NDSS)*, *2014*.

Table of Figures