資訊安全 Final Project

5105056013 吳嘉偉、5105056029 江青霞、5105056019 廖健智

2017/06/25

1 Introduction

Many online services let users query large public datasets: some examples include restaurant sites, product catalogs, stock quotes, and searching for directions on maps. In these services, any user can query the data, and the datasets themselves are not sensitive. However, web services can infer a great deal of identifiable and sensitive user information from these queries, such as her current location, political affiliation, sexual orientation, income, etc.

2 Proposed Method

2.1 Splinter

2.1.1 Architecture

There are two main principals in Splinter: the user and the providers. Each provider hosts a copy of the data. Providers can retrieve this data from a public repository or mirror site. For a given user query, all the providers have to run it on the same view of the data. A user splits her query into shares, using the Splinter client, and submits each share to a different provider. The user can select any providers of her choice that host the dataset. The providers use their shares to execute the user's query over the cleartext public data, using the Splinter provider library. As long as one provider is honest (does not collude with others), the user's sensitive information in the original query remains private. When the user receives the responses from the providers, she combines them to obtain the final answer to her original query.

2.1.2 Security Goals

The goal of Splinter is to hide sensitive parameters in a user's query. Specifically, Splinter lets users run parametrized queries, where both the parameters and query results are hidden from providers.

Splinter supports a subset of the SQL language, hides the information represented by the questions marks and the query's results, but the column names being selected and filtered are not hidden.

2.1.3 Threat Model

Splinter keeps the parameters in the user' s query hidden as long as at least one of the user-chosen providers does not collude with others. Splinter also assumes these providers are honest but curious: a provider can observe the inter- actions between itself and the client, but Splinter does not protect against providers returning incorrect results or maliciously modifying the dataset.

It assume that the user communicates with each provider through a secure channel (e.g., using SSL), and that the user' s Splinter client is uncompromised. The cryptographic assumptions are standard. They only assume the existence of one-way functions in our two-provider implementation. In our implementation for multiple providers, the security of Paillier encryption is also assumed.

3 Case Studies

4 Related Work

4.1 PIR(Private Information Retrieval) systems

Splinter is most closely related to systems that use Private Information Retrieval(PIR) to query a database privately. In PIR, a user queries for the ith record in the database, and the database does not learn the queried index i or the result. Much work has been done on improving PIR protocols. Work has also been done to extend PIR to return multiple records, but it is computationally expensive. Our work is most closely related to the system in, which implements a parametrized SQL-like query model similar to Splinter using PIR. However, because this system uses PIR, it has up to 10□ more round trips and much higher response times for similar queries.

Popcorn[4] is a media delivery service that uses PIR to hide user consumption habits from the provider and content distributor. However, Popcorn is optimized for streaming media databases, like Netflix, which have a small number (about 8000) of large records. The systems above have a weaker security model: all the providers need to be honest. Splinter only requires one honest provider, and it is more practical because it extends Function Secret

Sharing (FSS) [5, 6], which lets it execute complex operations such as sums in one round trip instead of only extracting one data record at a time.

4.2 Garbled circuits

Systems such as Embark [7], BlindBox [8], and private shortest path computation systems [9] use garbled circuits [10, 11] to perform private computation on a single untrusted server. Even with improvements in practicality [12], these techniques still have high computation and bandwidth costs for queries on large datasets because a new garbled circuit has to be generated for each query. (Reusable garbled circuits [13] are not yet practical.) For example, the recent map routing system by Wu et al. [9] uses garbled circuits and has 100□ higher response time and 10□ higher bandwidth cost than Splinter.

4.3 Encrypted data systems

Systems that compute on encrypted data, such as CryptDB [14], Mylar [15], SPORC [16], Depot [17], and SUNDR [18], all try to protect private data against a server compromise, which is a different problem than what Splinter tries to solve. CryptDB is most similar to Splinter because it allows for SQL-like queries over encrypted data. However, all these systems protect against a single, potentially compromised server where the user is storing data privately, but they do not hide data access patterns. In contrast, Splinter hides data access patterns and a user’ s query parameters but is only designed to operate on a public dataset that is hosted at multiple providers.

4.4 ORAM(Oblivious RAM) systems

Splinter is also related to systems that use Oblivious RAM [19, 20]. ORAM allows a user to read and write data on an untrusted server without revealing her data access patterns to the server. However, ORAM cannot be easily applied into the Splinter setting. One main requirement of ORAM is that the user can only read data that she has written. In Splinter, the provider hosts a public dataset, not created by any specific user, and many users need to access the same dataset.

5 Experimental Result

6 Conclusion

7 Reference

[1] Splinter: Practical Private Queries on Public Data Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan,

Matei Zaharia† MIT CSAIL, †Stanford InfoLab

[2] E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. In Proceedings of the 34th Annual International Confer- ence on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), pages 337-367. Sofia, Bul- garia, Apr. 2015.

[3] N. Gilboa and Y. Ishai. Distributed point functions and their applications. In Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), pages 640 - 658. Copenhagen, Denmark, May 2014.

[4] T.Gupta, N.Crooks, S.T.Setty, L.Alvisi, and M.Walfish. Scalable and private media consumption with Popcorn. In Proceedings of the 13th Symposium on Networked Systems Design and Implementation (NSDI), pages 91 - 107, Santa Clara, CA, Mar. 2016.

[5] E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. In Proceedings of the 34th Annual International Confer- ence on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), pages 337-367. Sofia, Bul- garia, Apr. 2015.

[6] N. Gilboa and Y. Ishai. Distributed point functions and their applications. In Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), pages 640 - 658. Copenhagen, Denmark, May 2014.

[7] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, and Z. Liu. Embark: Securely outsourcing middleboxes to the cloud. In Proceedings of the 13th Symposium on Networked Sys- tems Design and Implementation (NSDI), pages 255 - 273, Santa Clara, CA, Mar. 2016.

[8] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy. Blind- box: Deep packet inspection over encrypted traffic. In Proceedings of the 2015 ACM SIGCOMM, pages 213-226, London, United Kingdom, Aug. 2015.

[9] D. J. Wu, J. Zimmerman, J. Planul, and J. C. Mitchell. Privacy-preserving shortest path computation. In Proceed- ings of the 2016 Annual Network and Distributed System Security Symposium, San Diego, CA, Feb. 2016.

[10] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In Proceedings of the 19th ACM Confer- ence on Computer and Communications Security (CCS), pages 784 - 796, Raleigh, NC, Oct. 2012.

[11] S.Goldwasser.Multipartycomputations:pastandpres In Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing (PDC), pages 1-6, 1997.

[12] M.Bellare,V.T.Hoang,S.Keelveedhi,andP.Rogaway. Efficient garbling from a fixed-key blockcipher. In Pro- ceedings of the 34th

IEEE Symposium on Security and Privacy, pages 478 - 492, San Francisco, CA, May 2013.

[13] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC), pages 555-564, Palo Alto, CA, June 2013.

[14] R.A.Popa,C.M.S.Redfield,N.Zeldovich,and H. Balakrishnan. CryptDB: Protecting confidentiality with en- crypted query processing. In Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP), pages 85 - 100, Cascais, Portugal, Oct. 2011.

[15] R. A. Popa, E. Stark, J. Helfer, S. Valdez, N. Zeldovich, M. F. Kaashoek, and H. Balakrishnan. Building web applications on top of encrypted data using Mylar. In Proceed- ings of the 11th Symposium on Networked Systems Design and Implementation (NSDI), pages 157 - 172, Seattle, WA, Apr. 2014.

[16]A.J.Feldman,W.P.Zeller,M.J.Freedman,and E. W. Felten. SPORC: Group collaboration using untrusted cloud resources. In Proceedings of the 9th Symposium on Oper- ating Systems Design and Implementation (OSDI), Van- couver, Canada, Oct. 2010.

[17] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish. Depot: Cloud storage with minimal trust. In Proceedings of the 9th Symposium on Operating Systems Design and Implementation (OSDI), Vancouver, Canada, Oct. 2010.

[18] J. Li, M. Krohn, D. Mazières, and D. Shasha. Secure untrusted data repository (SUNDR). In Proceedings of the 6th Symposium on Operating Systems Design and Imple- mentation (OSDI), pages 91 - 106, San Francisco, CA, Dec. 2004.

[19] J. R. Lorch, B. Parno, J. Mickens, M. Raykova, and J. Schiffman. Shroud: Ensuring private access to large- scale data in the data center. In Proceedings of the 11th USENIX Conference on File and Storage Technolo- gies (FAST), pages 199-213, San Jose, CA, Feb. 2013.

[20] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path ORAM: An extremely sim- ple oblivious RAM protocol. In Proceedings of the 20th ACM Conference on Computer and Communications Se- curity (CCS), Berlin, Germany, Nov. 2013.