Outsourced Symmetric Private Information Retrieval

Stanislaw Jarecki* Charanjit Jutla† Hugo Krawczyk‡ Marcel Rosu§ Michael Steiner¶

Abstract

In the setting of searchable symmetric encryption (SSE), a data owner $\mathcal{D}$ outsources a database (or document/file collection) to a remote server $\mathcal{E}$ in encrypted form such that $\mathcal{D}$ can later search the collection at $\mathcal{E}$ while hiding information about the database and queries from $\mathcal{E}$. Leakage to $\mathcal{E}$ is to be confined to well-defined forms of data-access and query patterns while preventing disclosure of explicit data and query plaintext values. Recently, Cash et al. presented a protocol, OXT, to run arbitrary boolean queries in the SSE setting and which is remarkably efficient even for very large databases.

In this paper we investigate a richer setting in which the data owner $\mathcal{D}$ outsources its data to a server $\mathcal{E}$ but $\mathcal{D}$ is now interested to allow clients (third parties) to search the database such that clients learn the information $\mathcal{D}$ authorizes them to learn but nothing else while $\mathcal{E}$ still does not learn about the data or queried values as in the basic SSE setting. Furthermore, motivated by a wide range of applications, we extend this model and requirements to a setting where, similarly to private information retrieval, the client's queried values need to be hidden also from the data owner $\mathcal{D}$ even though the latter still needs to authorize the query. Finally, we consider the scenario in which authorization can be enforced by the data owner $\mathcal{D}$ without $\mathcal{D}$ learning the policy, a setting that arises in court-issued search warrants.

We extend the OXT protocol of Cash et al. to support arbitrary boolean queries in all of the above models while withstanding adversarial non-colluding servers ($\mathcal{D}$ and $\mathcal{E}$) and arbitrarily malicious clients, and preserving the remarkable performance of the protocol.

1 Introduction

Consider a database DB composed of collection of documents or records and an application that needs to search DB based on the keywords contained in these records. For example, DB can be a medical relational database with records indexed by a set of attributes (e.g., name, zipcode, medical condition, etc.), an email repository indexed by English words and/or envelope information (date, sender, receivers, etc.), a collection of webpages indexed by text and metadata, etc. A search query consists of a boolean expression on keywords that returns all documents whose associated keywords satisfy that expression. In this paper we are concerned with applications where the database is outsourced to an external server $\mathcal{E}$ and search is performed at $\mathcal{E}$ *privately.* That is, $\mathcal{E}$ stores an encrypted version of the original

*U. California Irvine. Email: stasio@ics.uci.edu.
†IBM Research. Email: csjutla@us.ibm.com
‡IBM Research. Email: hugo@ee.technion.ac.il
§IBM Research. Email: rosu@us.ibm.com
¶IBM Research. Email: msteiner@us.ibm.com

database DB (plus some metadata) and answers encrypted queries from clients such that the client obtains the documents matching his query without $\mathcal{E}$ learning plaintext information about the data and queries.

The most basic setting of private data outsourcing as described above is where the owner of the data itself, $\mathcal{D}$, is the party performing the search at $\mathcal{E}$. In this setting, $\mathcal{D}$ initially processes DB into an encrypted database EDB and sends it to $\mathcal{E}$. $\mathcal{D}$ only keeps a set of cryptographic keys that allows her to later run encrypted searches on $\mathcal{E}$ and decrypt the matching documents returned by $\mathcal{E}$. This setting is known as *searchable symmetric encryption (SSE)* and has been studied extensively [10, 6, 7, 3, 5, 4, 9]. While most of the research has focused on single-keyword searches (i.e., return all documents associated with a given keyword), recently Cash et al. [2], Crypto'2013, provided the first SSE solution, the OXT protocol, that cae support in a practical and private way arbitrary boolean queries on sets of keywords and in very large DBs. The leakage to $\mathcal{E}$, which is formally specified and proven in [2], is in the form of data-access and query patterns, never as direct exposure of plaintext data or searched values.

In this work we are concerned with richer outsourcing scenarios where multiple third parties (clients) access the data at $\mathcal{E}$ but only through queries authorized by the data owner $\mathcal{D}$. For example, consider a hospital outsourcing a database to an external service $\mathcal{E}$ such that doctors and other parties can access data but only via authorized queries and without leaking information on non-matching documents. As before, $\mathcal{E}$ should learn as little as possible about data and queries.

In this multi-client scenario (to which we refer as *MC-SSE*), $\mathcal{D}$ provides search tokens to clients based on their queries and according to a given authorization policy. Security considers multiple clients acting maliciously and possibly colluding with each other (trying to gain information beyond what they are authorized for) and a semi-trusted server $\mathcal{E}$ which acts as "honest-but-curious" but does not collude with clients. Extending SSE solutions to the multi-client scenario is straightforward when (a) search tokens are fully determined by the query and (b) the SSE protocol does not return false positives (returning false positives, i.e. documents that do not match a query, is allowed in SSE since the recipient in that case is the owner of the data but not in the multi-client setting where clients are not allowed to learn data they were not authorized for). In such cases, $\mathcal{D}$ would receive the client's query, generate the corresponding SSE search tokens as if $\mathcal{D}$ herself was searching the database, and provide the tokens to the client together with a signature that $\mathcal{E}$ can check before processing the search. However, for enabling general boolean queries, the SSE OXT protocol of [2] requires a number of tokens that is not known a-priori (it depends on the searched data, not only on the query) and therefore the above immediate adaptation does not work.

Our first contribution is in extending the OXT protocol from [2] to the MC-SSE setting while preserving its full boolean-query capabilities and performance. In this extension, $\mathcal{D}$ provides the client $\mathcal{C}$ with a set of n query-specific trapdoors, where n is the number of keywords in the query, which the client can then transform into search tokens as required by OXT. The set of n trapdoors given to $\mathcal{C}$ is fully determined by the query and independent of the searched data. An additional subtle technical challenge posed by OXT is how to allow $\mathcal{E}$ to verify that the search tokens presented by $\mathcal{C}$ are authorized by $\mathcal{D}$. The simple solution is for $\mathcal{D}$ to sign the n trapdoors, however in OXT these trapdoors need to be hidden from $\mathcal{E}$ (otherwise $\mathcal{E}$ can learn information about unauthorized searches) so a simple signature on them cannot be verified by $\mathcal{E}$. Our solution uses a *homomorphic signature* by $\mathcal{D}$ on the n

trapdoors that $\mathcal{C}$ can then transform homomorphically into signatures on the search tokens. We show that forging the tokens or their signatures is infeasible even by fully malicious clients.

The resulting MC-OXT protocol preserves the full functional properties of OXT, namely support for arbitrary boolean queries, the same level of privacy (i.e., same leakage profile) with respect to $\mathcal{E}$, and the same remarkable performance. Privacy with respect to clients is near-optimal with leakage confined only to information on the number of documents matching one of the query terms (typically, the least-frequent term).

Next, we extend the MC-OXT protocol to an even more challenging setting we call Outsourced Symmetric Private Information Retrieval (OSPIR), where on top of the MC-SSE requirements, one asks that *client queries be hidden from* $\mathcal{D}$ - similarly to the Private Information Retrieval (PIR) primitive. This requirement arises in important outsourcing scenarios. In the medical database example mentioned above, the hospital authorizes doctors or other parties to search the medical database according to certain policy; however, in some cases the actual query values are to be kept secret from the hospital itself (due to privacy, liability and regulatory requirements). Only the minimal information for determining the compliance of a query to the policy should be disclosed to the hospital. For example, the policy may indicate that only conjunctions with a minimal number of terms are allowed or that the query needs to include at least three of a set of attributes, etc. In such a case, there is no need for the hospital to learn the particular values being searched (such as a specific last name or medical condition). In other cases, as in outsourced patent or financial information search, the provider $\mathcal{D}$ may want to enforce that a client $\mathcal{C}$ pays for the type of query it is interested in but $\mathcal{C}$ wants to keep his query hidden from both $\mathcal{D}$ and $\mathcal{E}$. Relevant intelligence scenarios are presented in [8, 11].

Thus, we relax the query privacy requirement with respect to $\mathcal{D}$ to allow for minimal information needed for $\mathcal{D}$ to determine policy compliance. Specifically, we consider the case where keywords are formed by pairs of attribute-values. For example, in a relational database, attributes are defined by the database columns (e.g., SSN, name, citizenship, etc.), while in an email repository attributes can refer to envelope information such as sender and receivers or to the message body (in which case the values are, say, English words). In this case, a policy defines the class of boolean expressions allowed for a given client and the attributes that may be used as inputs into these expressions. In order to enforce the policy, $\mathcal{D}$ learns the boolean expression and attributes but nothing about the searched values. For policies defined via classes of attributes (e.g. allowing any attribute from the set of attributes {name, city, zipcode}) leakage to $\mathcal{D}$ can be further reduced by revealing the class and not the specific attributes in the query.

Our most advanced result is extending the OXT protocol to the OSPIR setting where in addition to all the MC-SSE functional, performance and privacy requirements, queries are to be hidden from $\mathcal{D}$ while allowing $\mathcal{D}$ to determine policy compliance. The resultant protocol, OSPIR-OXT, adds some crucial new ingredients to OXT: It uses *oblivious PRFs (OPRF)* for hiding the query values from $\mathcal{D}$, uses attribute-specific keys for enforcing policy compliance, and uses homomorphic signatures (or the more general abstraction of *shared OPRFs*) for query verification by $\mathcal{E}$. A further extension of the protocol accommodates an external policy manager, e.g., a judge in a warrant-based search, who checks policy compliance and allows server $\mathcal{D}$ to enforce the policy without learning the attributes being searched.

We achieve provable security against honest-but-curious but adaptive server $\mathcal{E}$, against arbitrarily malicious (but non-colluding with $\mathcal{E}$) server $\mathcal{D}$, and against arbitrarily malicious clients. Our security models extend those for SSE [5, 4, 2] to the more complex settings of MC-SSE and OSPIR. In all cases security is defined in the real-vs-ideal model and is parametrized by a specified leakage function $\mathcal{L}(\mathsf{DB}, \mathbf{q})$. A protocol is said to be secure with leakage profile $\mathcal{L}(\mathsf{DB}, \mathbf{q})$ against adversary $\mathcal{A}$ if the actions of $\mathcal{A}$ on adversarially-chosen input DB and queries set $\mathbf{q}$ can be simulated with access to the leakage information $\mathcal{L}(\mathsf{DB}, \mathbf{q})$ only (and not to DB or $\mathbf{q}$). This allows modeling and bounding partial leakage allowed by SSE protocols.

Implementation and performance. Performance-wise our extensions to OXT preserve the protocol's performance in both pre-processing (creating EDB) and search phases. $\mathsf{OSPIR\text{-}OXT}$ adds to the computational cost by adding a few exponentiations but these are generally inexpensive relative to the I/O cost (especially thanks to common-base optimizations). The protocols we provide for MC-SSE and OSPIR models support encrypted search over database containing tens of billions record-keyword pairs. Specifically, the practicality of the proposed schemes was validated by experiments on DBs which included e.g. English-language Wikipedia (13,284,801 records / 2,732,311,945 indexed tuples), and a synthetic US census database (100 million records / 22,525,274,592 index tuples, resulting in EDB with 1.7 TB TSet and 0.4 TB XSet). To illustrate search efficiency, in the census DB case we executed complex queries like SELECT ID WHERE FNAME='CHARLIE' AND SEX='FEMALE' AND NOT (STATE='NY' OR STATE='MA' OR STATE='PA' OR STATE='NJ) in about 4 seconds on an IBM Blade dual Intel 4-core Xeon processor and storage provided by a (low-end) 6.2TB RAID-5 storage system. Preprocessing of such large DBs (TSet and XSet creation) has been feasible by, among other things, optimization of common-base exponentiations, achieving aggregated (parallel) rate of about 500,000 exp.'s/sec. for NIST 224p. Details are provided in [1] which also presents mechanisms to support dynamic (i.e., updatable) DBs that readily apply to our extensions.

Full paper. The full paper presenting these results is available from the authors. It will be presented in ACM CCS 2013 in November.

References

[1] D. Cash, J. Jagger, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Dynamic Searchable Encryption in Very Large Databases: Data Structures and Implementation. manuscript.

[2] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. Crypto'2013. Cryptology ePrint Archive, Report 2013/169, Mar. 2013. http://eprint.iacr.org/2013/169.

[3] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 442–455. Springer, June 2005.

[4] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *ASIACRYPT 2010*, LNCS, pages 577–594. Springer, Dec. 2010.

[5] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 79–88. ACM Press, Oct. / Nov. 2006.

[6] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. http://eprint.iacr.org/.

[7] P. Golle, J. Staddon, and B. R. Waters. Secure conjunctive keyword search over encrypted data. In M. Jakobsson, M. Yung, and J. Zhou, editors, *ACNS 04*, volume 3089 of *LNCS*, pages 31–45. Springer, June 2004.

[8] IARPA. Security and Privacy Assurance Research (SPAR) Program - BAA, 2011. http://www.iarpa.gov/solicitations_spar.html/.

[9] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In *Proc. of CCS'2012*, 2012.

[10] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society Press, May 2000.

[11] WSJ. U.S. Terrorism Agency to Tap a Vast Database of Citizens. Wall Street Journal 12/13/12. http://alturl.com/ot72x.