Applied Laser Technology
Vol. 28, No.4,Oct (2021), pp.01–06

Symmetric Encryption That Can Be Filtered Using Boolean Expressions And Relied Upon Hardware

L.M.L.NARAYANA REDDY[1],V.PRASANNANJANEYA REDDY[2],
ASSISTANT PROFESSOR[1],[2],
DEPARTMENT OF ECE
PBR VISVODAYA INSTITUTE OF TECHNOLOGY AND SCIENCE::KAVALI

Abstract—

Cloud infrastructures are becoming the standard for storing and preserving data for both businesses and private users due to their popularity and accessibility. Dependability and security issues, particularly in applications with big data collections where effective search and retrieval services are also crucial needs, continue to impede the widespread use of the cloud. This causes the friction between security, effectiveness, and search expressiveness to rise. By investigating the design space of isolation guarantees provided by cutting-edge commodity hardware such as Intel SGX, abstracted as Isolated Execution Environments, we propose BISEN, a new provably secure Boolean searchable symmetric encryption scheme that improves these three complementary dimensions (IEEs). The first programme that supports multiple users is BISEN. Furthermore, BISEN extends the traditional SSE model to support filter functions on search results based on generic metadata created by the users. Experimental validation and comparison with the state of art shows that BISEN provides better performance with enriched search semantics and security properties.

KEY WORDS—Searchable Encryption, Intel SGX, Secure Databases, Provable Security, Distributed Systems

INTRODUCTION

C LOUD computing has had a profound impact on the way that we design and operate systems and applications. In particular, data storage and archiving are now commonly delegated to cloud infrastructures, both by companies and individual users. Companies typically want to archive large volumes of data, such as e-mails or historical documents, overcome limitations or lower costs of their on-premise infrastructures [2], while individual users benefit from having easily accessible documents and reduced storage overhead on their mobile devices [16]. However, data being outsourced to the cloud is often seenstave and should be protected accordingly. Private information incidents are constant reminders of the growing importance of these issues: governmental agencies impose increasing pressure on cloud companies to disclose users' data and deploy backdoors [27]; cloud providers are responsible, maliciously or accidentally, for critical data disclosures [24]; and even external hackers have gained remote access to users' data for limited time windows [31]. Cloud outsourcing services are thus highly incentivized to address these dependability and security requirements. In particular, when storing and updating large volumes of data in the cloud, it is essential to offer efficient, secure, and precise mechanisms to search and retrieve relevant data

objects from the archive. This highlights the need for cloud-based systems to balance security, efficiency, and query expressiveness. To address this tension, Searchable Symmetric Encryption (SSE) [7] has emerged as an important research topic in recent years, allowing one to efficiently search and update an encrypted database within an untrusted cloud server with security guarantees. Efficiency in SSE is achieved by building an encrypted index of the database and storing it in the cloud [18]. At search time, a cryptographic token specific to the query is built and used to access the index, and retrieved index entries are decrypted and processed. To minimize communication overhead, most SSE schemes delegate the execution of cryptographic computations to the cloud, as multiple index entries would otherwise have to be requested and downloaded to the client. However, performing sensitive operations in the cloud also leads to significant Informotion leakage, including the leakage of document identifiers matching a query, the repetition of queries, and the compromise of forward and backward privacy [40] (respectively, if new update operations match contents with previously issued queries, and if queries return previously deleted documents).

These are common, yet severe, flavours of information leakage that pave the way for strong attacks on SSE, including devastating file-injection attacks [45]. Another relevant limitation of SSE schemes is query expressiveness, as most solutions only support single keyword match [14] or limited Boolean queries (e.g., forcing queries to be in Conjunctive Normal Form and not supporting negations) [28]. This hinders system usability and may force users to perform multiple queries in order to retrieve relevant results, leading to extra communication steps and additional information leakage

ISSN: 1000-372X ALT

BACKGROUND AND RELATED WORK

Isolated Execution Environments (IEEs) As defined by Barbosa et al. [5], an IEE is an idealized random-access machine, running a fixed program, and whose behaviour can only be influenced by a well-specified interface that allows input/output interactions with the program. Isolation guarantees in IEEs follow from the requirements that: the I/O behaviour of programs running within them can only depend on themselves, on the semantics of their language, and on inputs received; and that the only information revealed about these programs must be contained in their I/O behaviour. This abstraction allows for the formal treatment of remote attestation mechanisms offered by technologies such as SGX and Trust Zone, which were shown in [5] to be sufficient for the deployment of Outsourced Computation protocols. Building on these definitions, Bahmani et al. [3] demonstrated how to refine the IEE attestation mechanism to enable for the deployment of general multiparty computation. Their design foollows two main stages. First, clients leverage remote attestation mechanisms to perform a key exchange agreement with the IEE and establish a secure communication channel. Afterwards, clients use these channels to interact with a reactive functionality on the IEE, exchanging encrypted inputs and outputs with confidentiality and integrity guarantees. The usage of sequence numbers in communications made through these channels also prevents a malicious server from repeating requests. In this work we will leverage on the IEE abstraction and this protocol, further extending it by allowing the IEE to interact with untrusted storage resources with privacy, integrity, and verifiability guarantees.

TECHNICAL OVERVIEW

The main idea of BISEN is for clients to leverage IEEs as remote trust anchors within the cloud, supporting efficient update and search operations on their cloud-stored encrypted databases. However, it would be unfeasible to maintain a whole database index within a resource-restricted IEE. As such, our proposal is to leverage a highly efficient environment for computations (the IEE) and a virtually infinite source for external storage (the cloud). BISEN combines these tools in the IEE side of the code, processing queries within its isolated memory, and relying on a cloud service for storing encrypted data. Figure 1 provides an overview of BISEN's architecture and its components. BISEN starts with a bootstrapping phase, where a client contacts a cloud server to initiate the IEE with BISEN's code. We call this server the Proxy Server, as it operates the IEE, manages all of its communications, and orders concurrent accesses. When started, the IEE initiates its state and asks a Cloud Storage Service to create BISEN's (initially empty) encrypted index. This storage service will basically be responsible for largescale storage, and can even be instantiated through pure storage solutions (e.g., AWS S3), as it only needs to support put/get operations. Hardcoded in BISEN's code is the public key of a trusted Certificate Authority, allowing the IEE to only accept messages from clients that present a valid signed certificate. After the bootstrap stage, clients can contact the proxy server to remotely attest it created an IEE with BISEN's code and to establish secure communication channels with it. Secure channels are established through a key-exchange algorithm based on remote attestation and the clients' public keys, as in [3]. Through these channels, clients can issue update and search operations to the IEE, which it processes by contacting the storage service and accessing BISEN's index. Updates allow both adding and removing keywords to/from documents. In either case, a new encrypted entry is added to BISEN's index, where its key is composed of a deterministic cryptographic token uniquely combining the keyword and docunmet, and its value is an encrypted message that includes the document id, a flag indicating if the operation is an addition or removal, and any metadata that the client wishes to associate with this update. This approach guarantees that both operations are indistinguishable, a necessary condition for preserving forward and backward privacy, and that arbitrary filter functions can be applied when processing queries. Search operations take a Boolean query as input and a set of filter functions. The IEE processes the query, retrieves and decrypts relevant index entries from the storage service, applies the filter functions, and calculates the resulting set of document ids. Finally, it returns this set to the client.

BISEN

In this section we present BISEN's full details. We start with some required notations and definitions (§ 4.1), then we present BISEN's protocols (§ 4.2), and finally we analyse its security (§ 4.3).

Notations and Definitions General Notations.

ISSN: 1000-372X ALT

In this paper we denote by λ the security parameter and μ(λ) a negligible function in it. We will use the standard security notions of variable-input-length Pseudo-Random Functions (PRF, instantiated as an HMAC in our implementtin) [6] and authenticated encryption schemes ensuring indistinctsuitability under chosen-ciphertext attacks (IND-CCA) [29]. We consider adversaries to be probabilistic algorithms, running in time polynomial on parameter λ. Extended IEE Notations. IEE interactions with clients are abstructed as IEE = (Setup, Send, Receive), as follows: • Reedstop (1λ) corresponds to IEE bootstrapping (if it hasn't been initialized yet) and secure channel establishment. Setup takes security parameter 1 λ as input, and produces state site with the exchanged key. • Descend (site, m) can be used by the client or IEE, and uses the secure channel established by Setup to encrypt m with the key in site. This outputs ciphertext caph. • Perceive (site, caph) uses the channel to retrieve endcrypted message caph using the key in site. This outputs the original message m.

IMPLEMENTATION

We implemented a prototype of BISEN in C/C++, with around 6200 lines of code. Our prototype is based on Intel SGX [17], using its remote-attestation and enclave management primitives to provide the IEE functionalities required by BISEN. To bootstrap the IEE and establish secure Client-IEE channels, we leveraged the med TLS library [35] and its SGX-compatible port [36], creating full-fledged TLS 1.2 tunnels between the IEE and BISEN's clients. For other cryptographic operations inside the IEE we used LibSodium [32], which is a constant-time cryptographic library partly based on Intel AES-NI. Constant-time cryptographic algorithms based on hardware implementations and oblivious primitives allow us to prevent side-channel leakage of the SGX enclave, including page and cache level leakage.2 We instantiate PRF with Disodium's SHA256-HMAC milamentation, H with SHA256, and Θ with its authenticated endcrypton algorithm, XSalsa20 stream cipher with Poly1305 MACs. Since Disodium is not ready for SGX deployment, we prepared an SGX-compatible version by (among other steps) removing all unsupported functions in SGX and replacing randomness funktons with their equivalents from Intel's RNG library.

Regarding attestation, the employed mechanism follows the design originally proposed in [5], where each program running on an IEE must produce a signature of its code and I/O trace thus far. For Intel SGX, this relies on the Quoting enclave, which uses the EPID group signature scheme [11] to produce a signature (quote) binding the enclave execution trace with the code that produced such trace. Verification of quotes is performed by the client through Intel's Attestation Service. For the IEE to interact with the encrypted index I, we leveraged on SGX calls. Additionally, concurrent accesses are managed by leveraging enclave multi-threading and by using lock-free concurrent data structures. Our implementation is open-source and available at: https:// github.com/ Bernama/BISEN. To demonstrate the use of metadata and filters, we impelminted specific functions for supporting ranked queries. Index entries store frequency information as metadata (i.e., how many times a keyword appears in a document), and in the search protocol this information is combined with other repository-wide statistics that were already known by the IEE, including the total number of documents and document frequency (i.e., in how many documents does a keyword appear).

EXPERIMENTAL EVALUATION

We now experimentally evaluate BISEN, using the prototype implementation described in the previous section. Experimental Test-Bench. We present performance results for BISEN and its Search and Update protocols. As IEE and proxy server, we used an Intel NUC i3-7100U with built-in SGX support, 2.4GHz of CPU frequency, 8GB of RAM, 256GB of SSD storage, running Ubuntu Server 18.04.1. As storage service we used a server with an AMD Opteron 6272 CPU with 64GB of RAM. Both machines were deployed on a one gigabit ethernet network. To evaluate the impact of remote communications, and since we already had the previous hardware available, we leveraged the cloud to deploy the client instead, using an AWS EC3 t3. large instance. The round-trip time between client and proxy server was 41.377ms and the max transmission rate was 50Mb/s. As dataset, we used an English Wikipedia dump of August 2018 [44] with around 60GB of uncompressed text data, 5.5 million documents, and 464 million keyword/document pairs. Measurements are based on an average of 50 independent executions. Experimental Evaluation Roadmap. The goal of our expertmental work is to answer the following questions: I.) what are the storage costs of BISEN; ii.) what is the performance cost (i.e., total time consumed) to process and store a whole dataset through a batch of Update protocol invocations, and how does this performance evolve as we scale the dataset's size; iii.) what is the performance cost of executing different types of Search queries, including queries with multiple conjunctions, disjunctions, and negations, considering

ISSN: 1000-372X ALT

different database sizes, the selectivity of queried keywords (i.e., the size of returned results) and the query size; iv.) how does BISEN's performance compare with the state of art in Boolean SSE, namely the recent IEX-2LEV scheme [28]; v.) what is the impact of adding filter functions and metadata for supporting ranked queries; vi.) and finally what is the impact of using different storage solutions for storing BISEN's index.

Storage Costs

In BISEN, clients only store one cryptographic key (32 bytes), which is used for secure communication with the IEE. The IEE also stores this key, plus kip and key ($3 * 32 = 96$ bytes). Additionally, it stores dictionary of counters W, which keeps a counter (4 bytes) and a hash (32 bytes) per entry, with one entry per unique keyword in the database. For the English Oxford dictionary containing 616500 unique word-forms, this results in an upper bound of around 20MB IEE storage, while e-mail date searches for individual days over 200 years could correspond to around 3MB IEE storage. Nonetheless, if specific application requirements demand it, our scheme can be trivially adapted to have this structure and its respective computations moved to clientside without loss of security. The storage service stores index I, which can grow due to the security guarantees provided (83 bytes per entry), nonetheless with cloud storage this can be more seamlessly scaled.

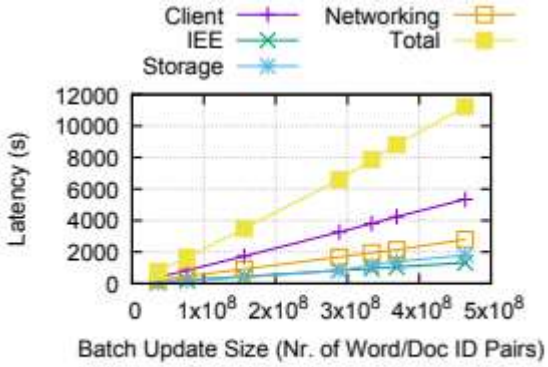

Fig. 3: Performance of the Update protocol.

Update Performance

Figure 3 reports the performance results for the Update protocol of BISEN. The y-axis represents time elapsed (in seconds), while the x-axis represents the update size in terms of keyworddocument pairs (i.e., how many entries are being added to index I at once, with a single batch of multiple Update protocol invoketons). Results were measured at different batch update sizes (up to 464 million pairs) and are reported for networking and for the three main protocol executors in separate, namely the client, IEE, and storage service. Proxy performance is omitted for simplicity, as it only forwards messages and its execution are highly efficient. Total results are also reported for convenience of the reader. Analysing the obtained results, one can conclude that BISEN's performance scales linearly with the size of the batch update (Total line in Figure 3). An update for a single document with 640 keywords takes 29ms, while a batch update of multiple documents totalling 464 million keywords takes 11239 seconds (around 3 hours). This means that performance of single Update invocations is mostly unaffected by the current database size. This is a natural observation, since this protocol does not depend on previous operations. These results also reflect the good performance properties of modern trusted hardware technologies, namely Intel SGX. The results for network performance basically show the cost of uploading data to the cloud, as BISEN adds very little cryptographic expansion: communications are encrypted with standard symmetric-key cryptography, and keywords are only hashed.

Regarding the performance of each protocol participant in separate, we can observe that time spent in the IEE and Storage Service is roughly similar, with a tendency for the Storage to become a bottleneck for larger operations. While we consider a single storage server, distributing this service across multiple machines might mitigate its weight in the operation. In turn, the IEE is responsible for simple cryptographic computations, entering enclave mode in SGX, and exiting this mode to store data through SGX calls, which is reflected in the latency for the maximam update (1300 seconds for 464 million keyword/document pairs). The largest slice of processing is on the client, which seems contradictory as from BISEN's specification (Figure 2), the client performs very few computations. From our analysis, we argue that these results are due to necessary pre-

processing: the client has to process the whole dataset from disk, parsing its keywords, stemming them and filtering stop-words [34]. In applications where documents are created and edited online for instance, this overhead would be greatly reduced.

CONCLUSIONS

In this paper, we have identified and addressed one of the fundsmental security issues in Searchable Symmetric Encryption (SSE) schemes, which is the outsourcing of critical cryptographic compotations to the untrusted server. This was achieved by proposing a new hybrid approach to SSE that combines standard symmetrickey cryptographic primitives with modern attestation-based trusted hardware. In our approach we minimize assumptions and requerulents on the employed hardware technology, in particular regarding its trusted storage capacity. Instead, trusted hardware is used as a limited-capacity Isolated Execution Environment abstriction, extending its resources through standard cryptographic primitives over more abundant (local, or even remote) untrusted resources. Additionally, we proposed to extend the traditional SSE querying model, supporting filter functions on search results based on generic metadata created by the users. Based on these approaches we proposed BISEN, a new dynamic Boolean SSE scheme that supports multiple clients, preserves both forward and backward privacy, displays minimal leakage, and optimizes computation, storage, and communication overheads. BISEN is shown to be provably secure against active adversaries under the standard security model.

REFERENCES

[1] T. Alves and D. Felton. Trust Zone: Integrated hardware and software security. ARM white paper, 3(4):18–24, 2004.

[2] M. Armbruster, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Kaminski, G. Lee, D. Patterson, A. Rabin, I. Stoical, and M. Zaharias. A view of cloud computing. Communications of the ACM, 53(4):50–58, 2010.

[3] R. Bahmani, M. Barbosa, F. Brasseur, B. Portella, A.-R. Sadeghi, G. Scerri, and B. Warminski. Secure multiparty computation from SGX. In Financial Cryptography and Data Security - FC'17, 2017.

[4] F. Baldetti and O. Eremenko. Sorting and Searching Behind the Curtain. In Proceedings of the 9th International Conference on Financial Cryptography and Data Security, 2015.

[5] M. Barbosa, B. Portella, G. Scerri, and B. Warminski. Foundations of hardware-based attested computation and application to SGX. In EURO S&P'16, pages 245–260, 2016.

[6] M. Bellaire and P. Rog away. Introduction to modern cryptography. Cuds Cse, 207:207, 2005.

[7] C. Bosch, P. Hartal, W. Jonker, and A. Peter. A Survey of Provably ¨ Secure Searchable Encryption. ACM CSUR, 47(2):18:1—-18:51, 2015.

[8] R. Bost. Sophos - Forward Secure Searchable Encryption. In CCS'16. ACM, 2016.

[9] R. Bost, B. Minard, and O. Eremenko. Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives. In CCS'17. ACM, 2017.

[10] T. Bourget, I. Lebedev, A. Wright, S. Zhang, S. Devadas, et al. Mi6: Secure enclaves in a speculative out-of-order processor. arXiv preprint arXiv:1812.09822, 2018.

[11] E. Brickell and J. Li. Enhanced privacy id from bilinear pairing for hardware authentication and attestation. International Journal of Information Privacy, Security and Integrity 2, 1(1):3–33, 2011.

[12] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-Preserving MultiKeyword Ranked Search over Encrypted Cloud Data. IEEE Transactions on Parallel and Distribibuted Systems, 25(1):222–233, 2014.

[13] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-Abuse Attacks Against Searchable Encryption. In CCS'15, pages 668–679. ACM, 2015.

[14] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In NDSS'14, volume 14, 2014.

[15] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In CRYPTO'13, pages 353–373. Springer, 2013.

[16] ComScore. The 2017 U.S. Mobile App Report. http://tinyurl.com/ ya8kkxan, 2017.

[17] V. Costan and S. Devadas. Intel sgx explained. Technical report, Cryptology ePrint Archive, Report 2016/086, 2016. https://eprint.iacr.org/2016/086, 2016.

[18] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In CCS'06, pages 79–88, 2006.

[19] Encrypted Systems Lab, Brown University. The clusion library. https://github.com/encryptedsystems/Clusion, 2018.

[20] B. Ferreira, J. Leitao, and H. Domingos. MuSE: Multimodal Searchable ˜ Encryption for Cloud Applications. In 37th IEEE International Symposium on Reliable Distributed Systems, 2018.

[21] B. Ferreira, B. Portela, T. Oliveira, G. Borges, J. Leitao, and H. Domin- ˜ gos. BISEN: Efficient Boolean Searchable Symmetric Encryption with Verifiability and Minimal Leakage (Extended Version). Cryptology ePrint Archive, Report 2018/588, 2018. https://eprint.iacr.org/2018/588.

ISSN: 1000-372X ALT