International Journal of Advanced Technology in Engineering and Science
Vol. No.4, Issue No. 04, April 2016
www.ijates.com
ijates
ISSN 2348 - 7550

SEARCHABLE SYMMETRIC ENCRYPTION

Er. Hariom Rathore[1] Dr. Amit Sharma[2]

[1]M.Tech Scholar,[2]Associate Professor, Department of Computer Science & Engineering ,
Vedant College of Engineering & Technology ,Bundi ,Rajasthan,(India)

ABSTRACT

Searchable symmetric encryption (SSE) allows a client to encrypt its data in such a way that this data can still be searched. The most immediate application of SSE is to cloud storage, where it enables a client to securely outsource its data to an untrusted cloud provider without sacrificing the ability to search over it.

SSE has been the focus of active research and a multitude of schemes that achieve various levels of security and efficiency have been proposed. Any practical SSE scheme, however, should (at a minimum) satisfy the following properties: sub linear search time, security against adaptive chosen keyword attacks, compact indexes and the ability to add and delete efficiently. Unfortunately, none of the previously-known SSE constructions achieve all these properties at the same time. This severely limits the practical value of SSE and decreases its chance of deployment in real-world cloud storage systems.

To address this, we propose the first SSE scheme to satisfy all the properties outlined above. Our construction extends the inverted index approach (Curtmola et al., CCS 2016) in several non-trivial ways and introduces new techniques for the design of SSE. In addition, we implement our scheme and conduct a performance evaluation, showing that our approach is highly efficient and ready for deployment.

Keywords: Component: Searchable symmetric encryption, multi-user searchable encryption, security definitions.

I. INTRODUCTION

1.1 Symmetric Key Encryption

Symmetric-key algorithms are algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. The keys may be identical or there may be a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in comparison to public-key encryption. Examples of Symmetric Key Encryption algorithms are IDEA and DES.

- ***Key Generation***

  When used with asymmetric ciphers for key transfer, pseudorandom key generators are nearly always used to generate the symmetric cipher session keys. However, lack of randomness in those generators or in their initialization vectors is disastrous and has led to cryptanalytic breaks in the past. Therefore, it is essential that an implementation uses a source of high entropy for its initialization

- ***Security***

Symmetric ciphers have historically been susceptible to known-plaintext attacks, chosen-plaintext attacks, differential cryptanalysis and linear cryptanalysis. Careful construction of the functions for each round can greatly reduce the chances of a successful attack.

1.2 Asymmetric/ Public Key Encryption

Asymmetric/Public-key cryptography refers to a set of cryptographic algorithms that are based on mathematical problems that currently admit no efficient solution -- particularly those inherent in certain integer factorization, discrete logarithm, and elliptic curve relationships. It is computationally easy for a user to generate a public and private key-pair and to use it for encryption and decryption. The strength lies in the "impossibility" (computational impracticality) for a properly generated private key to be determined from its corresponding public key. Thus the public key may be published without compromising security. Security depends only on keeping the private key private. Public key algorithms, unlike symmetric key algorithms, do not require a secure channel for the initial exchange of one (or more) secret keys between the parties.

Because of the computational complexity of asymmetric encryption, it is typically only used for short messages, typically the transfer of a symmetric encryption key. This symmetric key is then used to encrypt the rest of the potentially long conversation. The symmetric encryption/decryption is based on simpler algorithms and is much faster.

Message authentication involves hashing the message to produce a "digest," and encrypting the digest with the private key to produce a digital signature. Thereafter anyone can verify this signature by (1) computing the hash of the message, (2) decrypting the signature with the signer's public key, and (3) comparing the computed digest with the decrypted digest. Equality between the digests confirms the message is unmodified since it was signed, and that the signer, and no one else, intentionally performed the signature operation — presuming the signer's private key has remained secret. The security of such procedure depends on a hash algorithm of such quality that it is computationally impossible to alter or find a substitute message that produces the same digest - but studies have shown that even with the MD5 and SHA-1 algorithms, producing an altered or substitute message is not impossible. The current hashing standard for encryption is SHA-2. The message itself can also be used in place of the digest.

Public-key algorithms are fundamental security ingredients in cryptosystems, applications and protocols. They underpin various Internet standards, such as Transport Layer Security (TLS), S/MIME, PGP, and GPG. Some public key algorithms provide key distribution and secrecy, some provide digital signatures and some both.

1.3 Symmetric Searchable Encryption

Searchable Encryption schemes encode a database of documents into a data structure that can be outsourced to a remote server, in order to allow keyword-based search while preserving the privacy of both the database and the queries. Cloud-based storage is a direct application of these schemes: cloud providers supply data storage with high-availability guarantees and strong security protection. While availability failures can be immediately detected by users, security breaches can be known long after their use by malicious people. Even if the cloud providers should do their best to guarantee both availability and security, external or even internal people may want to (illegally) get access to the data or to alter computations performed on the data. As a consequence, data

privacy and correctness of the computations should be guaranteed by design. Since external storage is shared between many users with huge amounts of data to be processed, these security guarantees should be at a low cost for both the server and the client. The goal of searchable encryption schemes is to address this problem: the server itself should not learn anything about the search queries (which keywords the user is looking for), nor about the database updates triggered by the client.

Desirable properties of a practical SSE scheme are as follows:

***Dynamism*:** It should permit adding new files or deleting existing files from the encrypted file collection securely after the system set-up.

***Computational and Parallelization*:** It should offer fast search/updates Moreover, which are parallelizable across multiple processors.

***Storage Efficiency*:** The SEE storage overhead of the server depends on the encrypted data structure (i.e. encrypted index) that enables keyword searches. The number of bits required to represent a file-keyword pair in the encrypted index should be small. The size of encrypted index should not grow with the number of update operations (eventually results in re-encrypting the entire index). The persistent storage at the client should be minimum.

***Communication Efficiency*:** Non-interactive update/search operations should be possible to avoid the delays, with a minimum amount of data transmission.

***Security*:** The information leakage due to search/update operations must be precisely quantified based on formal SSE security notions.

II. HISTORY OF SEARCHABLE SYMMETRIC ENCRYPTION

The work by Song et al [1] was the first one in the direction of achieving Symmetric Searchable Encryption (SSE). In their paper, authors describe their cryptographic schemes for the problem of searching on encrypted data and provide proofs of security for the resulting crypto systems. The proposed techniques have a number of crucial advantages.

- They are provably secure, i.e. they provide provable secrecy for encryption, in the sense that the untrusted server cannot learn anything about the plaintext when only given the cipher-text;
- They provide query isolation for searches, meaning that the untrusted server cannot learn anything more about the plaintext than the search result; they provide controlled searching, so that the untrusted server cannot search for anarbitrary word without the user's authorization;
- They also support hidden queries, so that the user may ask the untrusted server to search for a secret word without revealing the word to the server.

The proposed algorithms are simple, fast (for a document of length n , the encryption and search algorithms only need $O(n)$ stream cipher and block cipher operations), and introduce almost no space and communication overhead, and hence are practical to use today.

Curtmola et al. [2] gave the first efficient SSE construction, achieving sub linear search time: the previous schemes, like the seminal work of Song et al. [1], required search complexity linear in the number of documents

stored in the database. They also improved the previous security models and introduced the notion of adaptive security for SSE.

Kamara and Papamanthou pointed out the sequential nature of the previous SSE schemes. Motivated by advances in multi-core architectures, the authors present a new method for constructing sub-linear SSE schemes. Their proposed approach is highly parallelizable and dynamic. With roughly a logarithmic number of cores in place, searches for a keyword w in the scheme execute in o(r) parallel time, where r is the number of documents containing keyword w (with more cores, this bound can go down to O(log n) , i.e., independent of the result size r). Such time complexity outperforms the optimal O(r) sequential search time—a similar bound holds for the updates. The proposed scheme also achieves the following important properties:

- It enjoys a strong notion of security, namely security against adaptive chosen-keyword attacks;
- Compared to existing sub-linear dynamic SSE schemes [], updates in our scheme do not leak any information, apart from information that can be inferred from previous search tokens;
- It can be implemented efficiently in external memory (with logarithmic I/O overhead). Our technique is simple and uses a red-black tree data structure; its security is proven in the random oracle model.

III. OBJECTIVE AND PROBLEM DEFINITION

Objectives

"Designing a masking system to secure Searchable symmetric encryption systems against leakage and disclosure of access patterns"

Methodology

- Implementing existing scheme in a programming language.
- Collaborating the proposed objective in the masking scheme of the base algorithm.
- Constructing theoretical proves and recording empirical results from the program.

IV. EFFICIENT AND SECURE SEARCHABLE SYMMETRIC ENCRYPTION

In this section we present our efficient SSE constructions, and state their security in terms of the definitions presented in Section 3. We start by introducing some additional notation and the data structures used by the constructions. Let Δ', $\Delta' \subseteq \Delta$, be the set of distinct words that exist in the document collection D. We assume that words in _ can be represented using at most p bits. Also, recall that D(w) is the set of identifiers of documents in D that contain word w ordered in lexicographic order.

We use several data structures, including arrays, linked lists and look-up tables. Given an array A, we refer to the element at address i in A as A[i], and to the address of element x relative to A as addr(A(x)). So if A[i] = x, then addr(A(x)) = i. In addition, a linked list L, stored in an array A, is a set of nodes Ni = hvi; addr(A(Ni+1))i, where $1 \leq i \leq |L|$, vi is an arbitrary string and addr(A(Ni+1)) is the memory address of the next node in the list.

Keygen($1^k, 1^\ell$): Generate random keys $s, y, z \xleftarrow{R} \{0,1\}^k$ and output $K = (s, y, z, 1^\ell)$.

BuildIndex($K, \mathcal{D}$):

1. Initialization:
   a) scan $\mathcal{D}$ and build Δ', the set of distinct words in $\mathcal{D}$. For each word $w \in \Delta'$, build $\mathcal{D}(w)$;
   b) initialize a global counter $\mathsf{ctr} = 1$.
2. Build array A:
   a) for each $w_i \in \Delta'$: // (build a linked list $\mathtt{L}_i$ with nodes $\mathtt{N}_{i,j}$ and store it in array A)
   - generate $\kappa_{i,0} \xleftarrow{R} \{0,1\}^\ell$
   - for $1 \le j \le |\mathcal{D}(w_i)|$:
     - generate $\kappa_{i,j} \xleftarrow{R} \{0,1\}^\ell$ and set node $\mathtt{N}_{i,j} = \langle \mathsf{id}(D_{i,j}) \| \kappa_{i,j} \| \psi_s(\mathsf{ctr}+1) \rangle$, where $\mathsf{id}(D_{i,j})$ is the j^{th} identifier in $\mathcal{D}(w_i)$;
     - compute $\mathcal{E}_{\kappa_{i,j-1}}(\mathtt{N}_{i,j})$, and store it in $\mathtt{A}[\psi_s(\mathsf{ctr})]$;
     - $\mathsf{ctr} = \mathsf{ctr} + 1$
   - for the last node of $\mathtt{L}_i$ (i.e., $\mathtt{N}_{i,|\mathcal{D}(w_i)|}$), before encryption, set the address of the next node to NULL;

   b) let $m' = \sum_{w_i \in \Delta'} |\mathcal{D}(w_i)|$. If $m' < m$, then set remaining $(m - m')$ entries of A to random values of the same size as the existing m' entries of A.
3. Build look-up table T:
   a) for each $w_i \in \Delta'$:
   - $\mathtt{value} = \langle \mathsf{addr}(\mathtt{A}(\mathtt{N}_{i,1})) \| \kappa_{i,0} \rangle \oplus f_y(w_i)$;
   - set $\mathtt{T}[\pi_z(w_i)] = \mathtt{value}$.

   b) if $|\Delta'| < |\Delta|$, then set the remaining $(|\Delta| - |\Delta'|)$ entries of T to random values.
4. Output $\mathcal{I} = (\mathtt{A}, \mathtt{T})$.

Trapdoor(w): Output $T_w = (\pi_z(w), f_y(w))$.

Search($\mathcal{I}, T_w$):

1. Let $(\gamma, \eta) = T_w$. Retrieve $\theta = \mathtt{T}[\gamma]$. Let $\langle \alpha \| \kappa \rangle = \theta \oplus \eta$.
2. Decrypt the list L starting with the node at address α encrypted under key κ.
3. Output the list of document identifiers contained in L.

Figure 1: Efficient SSE construction (SSE-1)

V. CONCLUSION

Searchable encryption is an important cryptographic primitive that is well motivated by the popularity of cloud storage services like Drop box, Microsoft SkyDrive and Apple iCloud and public cloud storage infrastructures like Amazon S3 and Microsoft Azure Storage. Any practical SSE scheme, however, should satisfy certain properties such as sub linear (and preferably optimal) search, adaptive security, compactness and the ability to support addition and deletion of _les. In this work, we gave the first SSE construction to achieve all these properties. In addition, we implemented our scheme and evaluated its performance. Our experiments show that our construction is highly efficient and ready for deployment.

REFERENCES

[1] Dawn Xiaodong Song, David Wagner, and Adrian 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.

[2] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, ACM CCS 06, pages 79–88. ACM Press, October / November 2006.

[3] Seny Kamara and Charalampos Papamanthou, "Parallel and Dynamic Searchable Symmetric Encryption". Available at https://eprint.iacr.org/2013/335.pdf

[4] Raluca Ada Popa and Nickolai Zeldovich, "Multi-Key Searchable Encryption", Cryptology ePrint Archive: Report 2013/508.

[5] David Cash and Joseph Jaeger and Stanislaw Jarecki and Charanjit Jutla and Hugo Krawczyk and Marcel-Cătălin Roşu and Michael Steiner, " Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation", Cryptology ePrint Archive: Report 2014/853

[6] Bram Leenders, "Simple and Efficient Searchable Symmetric Encryption with Application", Proceedings of the 20th Twente Student Conference on IT, January, 2014.

[7] Yuval Ishai, Eyal Kushilevitz, Steve Lu 3 and Rafail Ostrovsky, "Private Large-Scale Databases with Distributed Searchable Symmetric Encryption". Available at https://eprint.iacr.org/2015/1190.pdf

[8] Melissa Chase and Emily Shen, "Substring-Searchable Symmetric Encryption", Proceedings on Privacy Enhancing Technologies 2015; 2015 (2), pp. 263–281.

[9] Attila A. Yavuz and Jorge Guajardo, "Dynamic Searchable Symmetric Encryption with Minimal Leakage and Efficient Updates on Commodity Hardware". Available at https://eprint.iacr.org/2015/107.pdf

[10] Raphael Bost and Pierre-Alain Fouque and David Pointcheval, "Verifiable Dynamic Symmetric Searchable Encryption: Optimality and Forward Security", Cryptology ePrint Archive: Report 2016/062, 2016.

[11] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and e_cient constructions. In Proc. ACM Conference on Computer and Communica- tions Security (CCS), pages 79{88, 2006.

[12] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved de_nitions and e_cient constructions. Journal of Computer Security, 19(5):895{934, 2011.

[13] Y. Dodis, T. Ristenpart, J. Steinberger, and S. Tessaro. To hash or not to hash again? (In)di_erentiability results for H2 and HMAC. Proc. Int. Cryptology Conference (CRYPTO), 2012.

[14] Enron email dataset. http://www.cs.cmu.edu/_enron/, 2009.

[15] FIPS 180-3. Secure Hash Standard (SHS). Federal Information Processing Standard (FIPS), Publication 180-3, National Institute of Standards and Technology, Washington, DC, October 2008.

[16] [13] FIPS 197. Advanced Encryption Standard (AES). Federal Information Processing Standard (FIPS), Publication 197, National Institute of Standards and Technology, Washington, DC, November 2001