Scientific Journal of Impact Factor (SJIF): 4.14

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 3, Issue 7, July -2016

A Searchable and secured data storage without encryption

[1]Yathish D P, [2]A H Shanthakumara

[1]*M.Tech, Dept. of CSE, Siddaganga institute of technology, Tumakuru*
[2]*Asst. Prof, Dept. of CSE, Siddaganga institute of technology, Tumakuru*

Abstract--*Users are storing ever-increasing amounts of information digitally, driven by many factors including government regulations and the public's desire to digitally record their personal histories. Maintaining information privacy is challenging when sharing data across a distributed long-term datastore. In such applications, secret splitting the data across independent sites has been shown to be a superior alternative to fixed-key encryption.As the amount of digital assets increase, systems that ensure the durability, integrity, and accessibility of digital data become increasingly important. Distributed on-line archival storage systems are designed for this very purpose. Fixed-key encryption has also been used to securely pre-index the data, but in addition to key management issues, it is not well suited for long term applications. The exposure of a key is bounded by computation effort and management of encryption keys become as much of a problem as the management of the data the key is protecting. The data structures used are organized in such a way that an unauthorized user is very difficult to launch targeted attack on the system. The result is a flexible design that can be applied to both new and existing secret-split data stores.*

Keywords--*cloud computing, Secret splitting, Secret split datastore, and Long term datastore.*

I. INTRODUCTION

In today's computing environment, more and more data is being migrated from hard copy to digital form. This trend is catalysed by a number of motivations revolving around economic efficiency. Digital data offers many economic advantages compared to traditional, tangible publishing methods such as books, magazines, and films. Digital data is often much easier and cheaper to transport and store in comparison to traditional mediums such as paper and ink based printing. Security is often a critical issue for long-term storage, particularly given recent incidents involving insiders releasing large amounts of private or classified information much of this risk is due to traditional storage systems having a single point of compromise: the data server. If that one point is compromised at any time during the datastore's lifespan, information can be leaked. This threat is obviously magnified in a distributed environment since, by its nature, the data is stored in multiple locations.

The access patterns of archival storage are distinctly different from general purpose storage. Archival storage is heavily written centric information is written to an archive and it may be a long time, if ever, before that data is accessed from the archive. An example of this would be a collection of business documents that must be pre-served for legal reasons even though they are rarely if ever requested.

To address the need to maintain information privacy while searching a secret-split datastore, we developed a novel system that accomplishes these tasks without relying on fixed-key encryption. Furthermore, this is completely agnostic with regards to the datastore's implementation since whether the datastore is based on potshards or Clever-safe, the user is left with some kind of identifier that can be used to retrieve the user's data. Protection over Time, Securely Harbouring and Reliably Distributing Stuff, an archival storage system designed for a computing environment where relatively static data must be preserved for an indefinite period of time. Potshards separates data redundancy and data secrecy and utilizes a geographically distributed array of network attached storage devices, called *archives*. This avoids the problem introduced by keyed cryptography where the key represents a single point of failure which could render data recovery infeasible. This also helps avoid the problem of preserving historic keys associated with an archive of encrypted files. The set of reverse indexes is generated, each individual index is secret split; these resulting shares are each sent to a different query server in the distributed environment. We define a query server as a security module backedby one or more machines working together as a single, logical key-value store.

Now clients can be viewed as a secure key-value store whose job it is to service search request from authorized clients, and respond with the share of the correct reverse index. Once a client has retrieved the shares to a single reverse index, it is able to reconstruct that index, thereby obtaining the set of data object identifiers that can then be used to retrieve the desired data from the underlying datastore.

II. RELATED WORK

This section provides the background of the proposed scheme, the methodology used for performing the study, and any reference materials used in conducting the study for the scheme.

The design concepts and motivation of the project borrow from various research projects. These projects range from general purpose distributed storage systems, to distributed content delivery systems, to archival systems designed for very specific uses.

A number of systems such as Ocean Store [5], Far-Site [1], and PAST [9] rely on the explicit use of keyed encryption to provide file secrecy. While this may work reasonably well for short-term file secrecy it is less than ideal for the very long-term storage problem that this concept is addressing. Further evidence that concept is designed for a different application can be found in the design choices made by the authors of the systems mentioned previously. For example, in OceanStore straight replication was chosen in favour of erasure coding in order to provide for better read performance. In contrast, the design emphasis on this is reliability for very long-term storage.

Another class of storage projects that use distributed storage techniques but rely on keyed encryption for file secrecy do not provide any method for insuring long-term file persistence. These systems, such as Glacier [4] and Freenet [3] are designed to deal with the specific needs of content delivery as opposed to the requirements of long-term storage. An archival storage system must explicitly address the problem of insuring the persistence of the system's contents.
Another class of systems is aimed at long-term storage but with the explicit goal of open content. Systems such as LOCKSS [7] and Intermemory [2] are designed around preserving digital data for libraries and archive where file consistency and accessibility are paramount. These systems are developed around the central idea of very long-term access for public information and thus file secrecy is explicitly not part of the design.

The PASIS architecture [13] and the work of SubbiahandBlough [11] avoid the use of keyed encryption by using secret sharing threshold schemes. While this prevents the introduction of the singular point of failure that keyed encryption introduces to a system, the design of these system only use one level of secret sharing. In effect this combines the secrecy and redundancy aspects of the systems. While related, these two elements of security are, in many respects, orthogonal to one another. Combining the secrecy and redundancy aspects of the system also has the possible effect of introducing compromises into the system by restricting the choices of secret sharing schemes. By separating secrecy and redundancy, an implementation of this is able to utilize a security mechanism optimized for redundancy or secrecy.

III. THE PROPOSED SCHEME

A. Overview

The design of our "Searchable and secured data storage without encryption" scheme is applicable for long term storage systems without encryption. The authorized user has to enter the keyword that he wants to search from the file when the keyword matches the files containing that keyword will be listed to the user. The user can also download the file that he wants.

A. SECRECY TECHNIQUES

This utilizes three primary techniques in the creation and long-term storage of shards. First, secret splitting algorithms provide file secrecy without the need to periodically update the algorithm. This is due to the fact that perfect secret splitting is information theoretically secure as opposed to only computationally secure. Second, approximate pointers between shards allow objects to be recovered from only the shards themselves. Thus, even if all indices over a user's shards are lost, their data can be recovered in a reasonable amount of time. Third, secure, distributed RAID techniquesacross multiple independent archives allow data to be recovered in the event of an archive failure without exposing the original data during archive reconstruction.

This provides data secrecy through the use of secret splitting algorithms [17]; thus, there is no need tomaintain key history because this does not use traditional encryption keys. Additionally, this utilizes secret splitting in a way that does not combine the secrecy and redundancy parameters. Storage of the secret shares is also handled in a manner that dramatically reduces the effectiveness of insider attacks. By using secret splitting techniques, the secrecy in this has a degree of future proofing built into it.

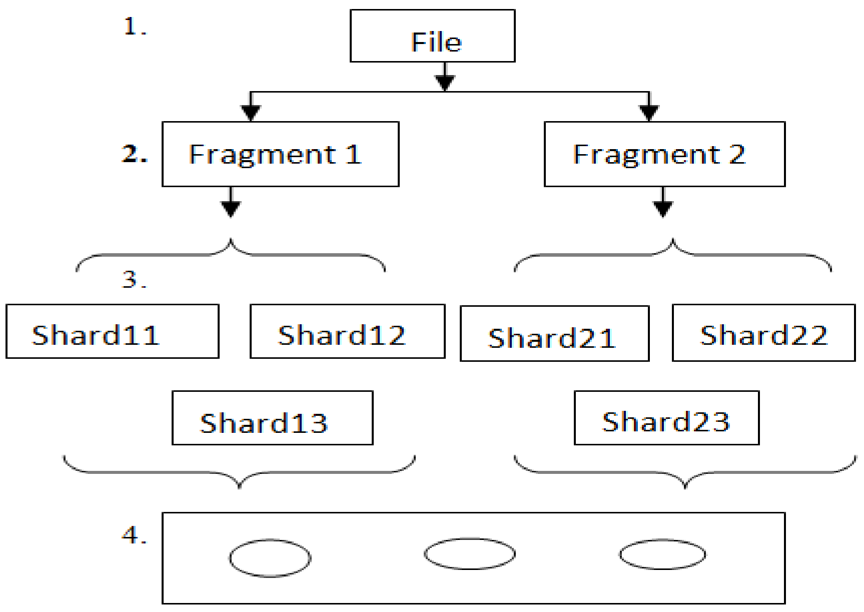

Figure 1.Secrecy techniques

Four data transformation layers:
1.Pre-processing- file into object
2.Secrecysplit- object into set of fragments
3.Availability split- fragment into set of shards
4.Placement- set of shards into messages for archives

B. INGESTION

The first step during ingestion is to identify the keywords for each data object. For our purposes, keywords are synonymous with search terms. The set of keywords for a data object define how that data object can be retrieved as a result of a search operation, i.e. how clients will search for their data using search terms. Once the set of keywords has been identified for each data object to be ingested, the reverse indexes for the corpus are generated by turning the mapping of data object to keywords to become a mapping of keyword to data object(s). Each of these reverse indexes can be viewed as a successful result to a search operation.

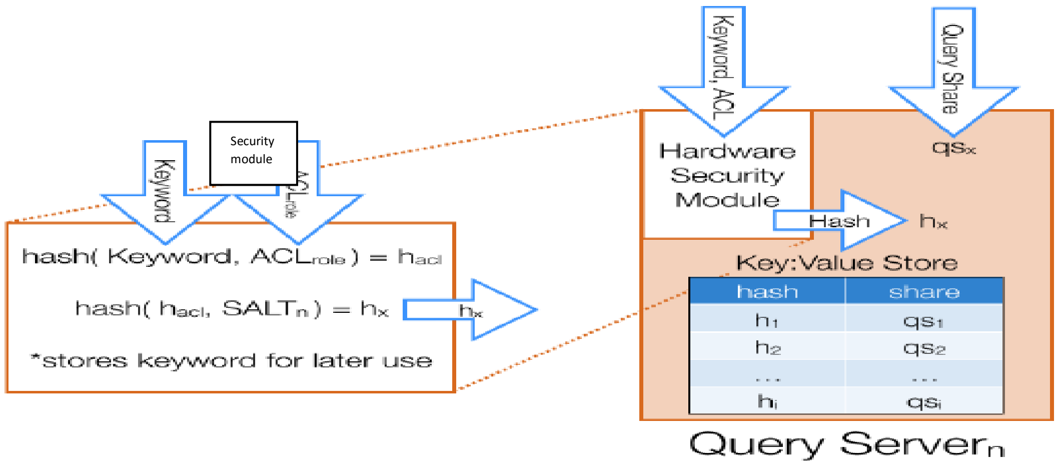

Figure 2.Overview of ingesting a query

The credential pair is sent directly to the security module over its own network interface, while the share is sent directly to the key-value store backend of the query server via its separate network interface. This is to ensure that the actual keyword is never exposed to the query server, and is instead pre-processed by the security module. During this pre-processing, the security moduleperforms a hash of the keyword and the provided credential, which is the same credential required by an authorized client in order to retrieve this query share in the future. The security modulethen performs one more salted hash using the query server's unique salt to ensure security.

C. SEARCHING

To conduct a query, or search request, the client first hashes the search term with the user's RBAC credential. This hash string is then sent to the query servers, via their security module, which hashes the user's input with the query server's salt.

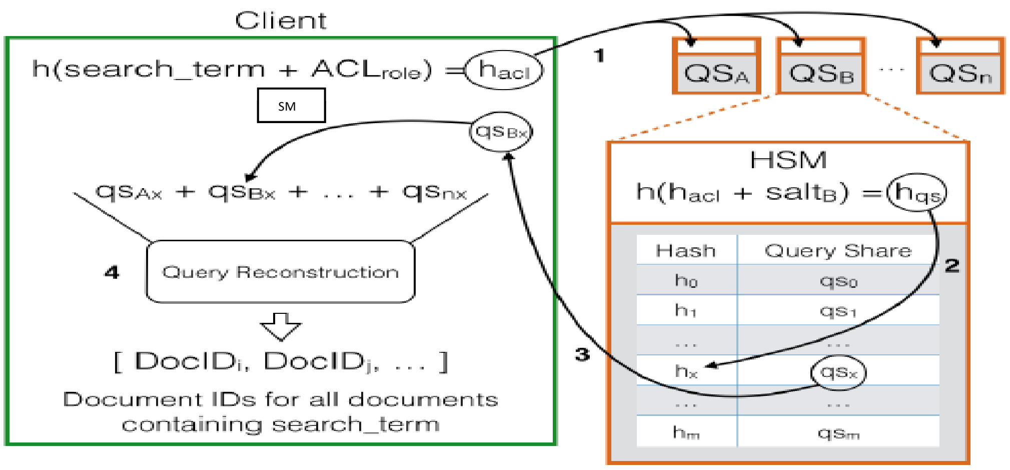

Figure 3.Simple single search term

The final hash string is then passed to the key-value store, who finds the query share, if it exists, and sends it back to the client. If no entry is found for the given hash string, the query server responds with a non-empty response signifying to the client that there is no data that satisfies the search request. The size of this 'not found' response is the same size as a reverse index share; this makes all responses, regardless of search outcome, the same size, thus preventing an attacker from differentiating between response types.

IV. IMPLEMENTATION RESULTS

In order to analyze the effect varying the number of key-words per data object has upon both the size and quantity of reverse indexes; we performed the steps choosing 10, 20, 30, 50 and 100 keywords per data object. Figure depicts a cumulative density graph for each experiment. As expected, it can be seen that as the number of keywords per data object increases, the total number of keywords present in the corpus also increases. It is also worth noting that 2=3 of the reverse indexes were found to contain a single data object identifier, and almost all reverse indexes were found to contain fewer than ten identifiers.

While these results are very specific, they illustrate the potential space overhead required to secret split the reverse indexes since all shares must be the same size, i.e. the majority of shares must be padded in order to be the size of the largest shares in the corpus. If they are not, then their size becomes a distinguishing characteristic, which would enable targeted theft. For example, if shares are not padded to be of equal size, and there exists only a few shares that are X bytes, then an attacker would be able to identify and steal those shares, simply based upon their size, from non-compromised key-value stores.

With regards to search performance, search requests were found to complete in less than one second; the only variance in performance being due to random network issues causing a small number of resend operations. We measure search performance using the completion time since standard metrics, i.e. precision and recall, are dependent solely upon the number and accuracy of the keywords chosen during ingestion, which are at the implementer's discretion. Search performance is based on both the number of keywords and the size of the corpus. The number of keywords can potentially impact the hash table look up time, i.e. from the query server's key-value store, and the size of the corpus.

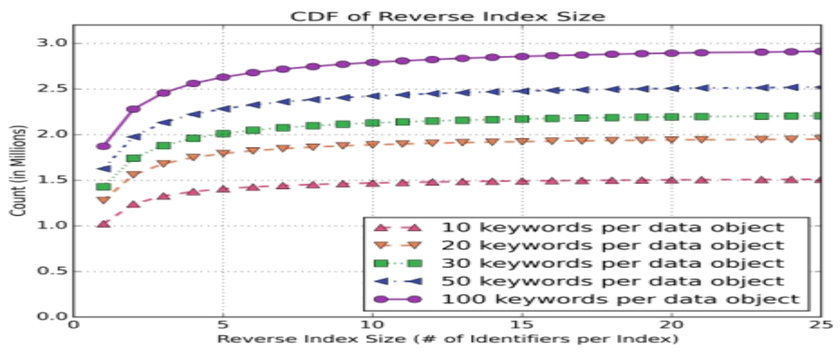

Figure 4.Cumulative distribution function of reverse index size

V. CONCLUSION

This paper introduced Searchable and secured data storage without encryption, a system designed to provide secure long term archival storage to address the new challenges and new security threats posed by archives that must securely preserve data for decades or longer.
In developing this, we made several key contributions to secure long-term data archival. First, we use multiple layers of secret splitting, approximate pointers, and archives located in independent authorization domains to ensure secrecy, shifting security of long-lived data away from a reliance on encryption. The combination of secret splitting and approximate pointers forces an attacker to steal an exponential number of shares in order to reconstitute a single fragment of user data; because he does not know which particular shares are needed, he must obtain all of the possibly required shares. Second, we demonstrated that a user’s data can be rebuilt in a relatively short time from the stored shares only if sufficiently many pieces can be acquired.

REFERENCES

[1] A. Adya, W. J. Bolosky, M. Castro, R. Chaiken, G. Cer-mak, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In Proceedings of the 5th Symposium on Operating Sys-tems Design and Implementation (OSDI), Boston, MA, Dec. 2002.USENIX.

[2] Y. Chen, J. Edler, A. Goldberg, A. Gottlieb, S. Sobti, and P. Yianilos. A prototype implementation of archival inter-memory.InProceedings of the Fourth ACM International Conference on Digital Libraries, 1999.

[3] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. Lecture Notes in Computer Science, 2009:46+, 2001.

[4] A. Haeberlen, A. Mislove, and P. Druschel. Glacier: Highly durable, decentralized storage despite massive cor-related failures. In2nd Symposium on Networked Systems Design and Implementation (NSDI ’05), Boston, MA, USA, May 2005.

[5] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weather-spoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. InPro-ceedings of the Ninth International Conference on Archi-tectural Support for Programming Languages and Oper-ating Systems, pages 190–201, Nov 2000.

[6] W. Litwin and T. Schwarz.Algebraic signatures for scal-able distributed data structures. Technical Report CE-RIA Technical Report, Universit´e Paris 9 Dauphine, Sept. 2002.

[7] P. Maniatis, M. Roussopoulos, T. Giuli, D. S. H. Rosen-thal, M. Baker, and Y. Muliandi. Preserving peer replicas by rate-limited sampled voting. InProceedings of the Sym-posium on Operating Systems Principles, Bolton Landing, NY, Oct 2003.ACM.

[8] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. Handbook of Applied Cryptography.CRC Press, 2001.

[9] A. Rowstron and P. Druschel.Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In Proceedings of the 18th ACM Sym-posium on Operating Systems Principles, Chateau Lake Louise, Banff, Alberta, Canada, Oct 2001.

[10] D. R. Stinson. Cryptography Theory and Practice. Chap-man & Hall/CRC, Boca Raton, FL, second edition, 2002.

[11] A. Subbiah and D. M. Blough.An approach for fault tol-erant and secure data storage in collaborative work envi-ronements.InProceedings of the 2005 ACM Workshop on Storage Security and Survivabiilty, pages 84–93, Fairfax, VA, Nov. 2005.

[12] T. M. Wong, C. Wang, and J. M. Wing.Verifiable secret redistribution for threshold sharing schemes. Technical Report CMU-CS-02-114-R, Carnegie Mellon University, Oct. 2002.

[13] J. J. Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Kilic¸c ¸¨ote, and P. K. Khosla. Survivable storage sys-tems. IEEE Computer, pages 61–68, Aug. 2000.

[14] “Digital Corpora – computer forensics research,” 2015, [Online; accessed 1-Feb-2015]. [Online].Available http://digitalcorpora.org

[15] A. Rajaraman and J. D. Ullman, Mining of Massive Datasets.Cam-bridge University Press; 1 edition (December 30, 2011), 2011.

[16] M. A. Olson, K. Bostic, and M. Seltzer, “Berkeley db,” in Proceedings of the Annual Conference on USENIX Annual Technical Conference, ser. ATEC ’99. Berkeley, CA, USA: USENIX Association, 1999, pp. 43–43.

[17] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti, “POTSHARDS: secure long-term storage without encryption,” in Proceedings of the 2007 USENIX Annual Technical Conference, Jun. 2007, pp. 143–156. [Online]. Available: http://www.ssrc.ucsc.edu/Papers/storer-usenix07.pdf.