SecureCSearch: Secure Searching in PDF over Untrusted Cloud Servers

Meet D. Shah
GlobalFoundries
Malta, NY, USA
Email: meetshah51245@gmail.com

Manoranjan Mohanty
University of Auckland
Auckland, New Zealand
Email: m.mohanty@auckland.ac.nz

Pradeep K. Atrey
University at Albany, SUNY
Albany, NY, USA
Email: patrey@albany.edu

***Abstract*—The usage of cloud for data storage has become ubiquitous. To prevent data leakage and hacks, it is common to encrypt the data (e.g. PDF files) before sending it to a cloud. However, this limits the search for specific files containing certain keywords over an encrypted cloud data. The traditional method is to take down all files from a cloud, store them locally, decrypt and then search over them, defeating the purpose of using a cloud. In this paper, we propose a method, called SecureCSearch, to perform keyword search operations on the encrypted PDF files over cloud in an efficient manner. The proposed method makes use of Shamir's Secret Sharing scheme in a novel way to create encrypted shares of the PDF file and the keyword to search. We show that the proposed method maintains the security of the data and incurs minimal computation cost.**

***Keywords*-PDF; Security; Searching; Cloud; Shamir's Secret Sharing; Encrypted Domain**

I. Introduction

Portable Document Format (PDF) files are very commonly stored and accessed over the cloud. On the PDF documents, certain operations, such as text searching and highlighting, are very common. Often, PDF files contain confidential information. One way to protect such information is to encrypt the PDF files before sending them to the cloud. However, a major challenge here is performing text searching and highlighting operations on the encrypted PDF files in a secure and efficient manner. This paper deals with this challenge.

In order to solve this problem, this paper presents a method called SecureCSearch that uses the (k, n) Shamir's Secret Sharing (SSS) scheme [7] for encrypting PDF files. A significant advantage of the SSS scheme is that it possesses information theoretic security, meaning that an adversary cannot get any information about the secret PDF file until all the k shares are known. However, when applied on text, the SSS scheme is prone to frequency analysis attack [1]. To counter this attack, we propose necessary changes in the deployment of the SSS scheme as follows. In the traditional (k, n) SSS scheme, $k-1$ coefficients from a finite field defined over Z_q (where q is a prime) are required. In the modified scheme, we build a set of $l = (k-1) \times r$ coefficients (r being a parameter that can be tuned as per requirement), from which different $k-1$ coefficients are randomly chosen for different occurrences of a word in a PDF file. This way we get different share values (or ciphertext) for different occurrences of a word in a PDF, which effectively reduces the Index of Coincidence (IC) and hence counters the frequency analysis attack.

We chose the SSS scheme over the conventional symmetric encryption schemes, such as the Advanced Encryption Standard (AES), because AES (similar argument for other schemes) requires a block of 128 bits to be encrypted at a time, which may usually contain more than one word, nullifying the possibility of searching individual words. Alternately, if we use entire 128 bits for each word (by adding a fixed pad if the length of the word is less than 128 bits, which is mostly the case), it would significantly increase the data overhead. On the contrary, the SSS scheme does not result in such data overhead.

There have been various attempts for searching keywords on encrypted data, database or files, on a server. Song et al. [9] was one of the first group of researchers to work on searchable encryption. Using two layered encryption, they provided a practical solution for searching functionality without losing data confidentiality. Wang et al. [12] introduced secure keyword search on encrypted cloud data. The authors focused on the usability of the privacy-preserving data hosting services and introduced a ranked searchable symmetric encryption method using Order-Preserving Symmetric Encryption (OPSE). In another work, Curtmola et al. [3] considered a multi-user setting in the searchable symmetric encryption where more than one user can send search queries. Furthermore, using public key encryption, Boneh et al. [2] provided a method to search on encrypted data. Later, Tseng et al. [10] introduced iPEKS, which is an interactive keyword search system over the cloud.

Other works specifically focusing on search operations on an encrypted relational database include CryptoDB [6] and MONOMI [11]. In these works, SQL operations are performed over the encrypted data using the OPSE scheme. The limitation of these methods is that data owners must store the ordering information with the encrypted data. Also, in recent years, Li et al. [5] and Demertzis et al. [4] proposed techniques to perform the range queries over untrusted encrypted database servers using an indexing approach.

Compared to the existing work, there are three key advantages of the proposed SSS-based approach. Firstly, the

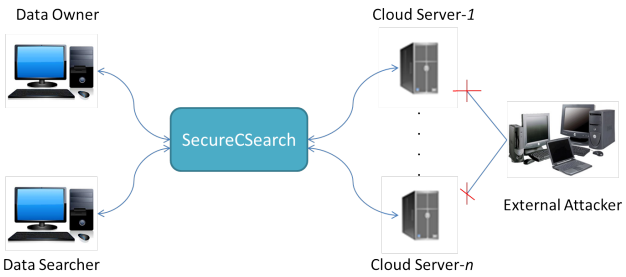

Figure 1: System model

proposed scheme has significantly less storage overhead than the searchable block-based symmetric encryption schemes, such as AES, and less computation overhead than the public-key searchable encryption methods. Secondly, it provides information theoretic security. Finally, it offers fault-tolerance in the sense that even if a few (i.e. up to $n - k$) servers are non-functional or unavailable, the search functionality can still be supported. The only limitation in using the SSS scheme is the assumption that multiple (i.e. k or more) cloud servers do not collude; however, a recent work [13] shows that this limitation can be overcome.

The SSS-based approach has been used by Sharma et al. [8] to merge two or more PDF files in encrypted form over third party web servers. To the best of our knowledge, this is the first work in which the SSS-based approach with necessary adaption is used for searching in the encrypted domain.

We divide the rest of the paper as follows. In Section II, we discuss the system model under which the proposed approach works. Section III, presents the approach. In Section IV, we provide a security analysis and in Section V, we provide experiments and analysis. Finally, we conclude the paper in Section VI.

II. System Model

In this work, we consider that an individual or an organization stores encrypted PDF files on a cloud. The cloud is responsible for storing the encrypted files, searching a keyword in the stored file, and returning a file with all occurrences of the keyword highlighted, as needed. In our system, we have the following entities (Figure 1):

- **Data Owner**: This entity is an individual, an organization or a computer application that uploads PDF files to *cloud servers*. This entity is an authorized entity, and when uploading, it must ensure that the file is not disclosed to adversaries. To hide the content of the files from the *cloud servers*, the *data owner* must encrypt the files before sending them to the servers. The *data owner* can search keywords in the encrypted files over the cloud and download them as needed.
- **Cloud Server**: A *cloud server* stores the encrypted PDF files and provides to the *data searcher* an option of keyword searching in the stored files. The *cloud servers* must have enough storage and processing power to store and process a large number of files. These entities must ensure that the keyword searcher (i.e., the *data owner* or the *data searcher*), if authorized, can download only those PDF files which contain the keyword. All the occurrences of the keyword must have been highlighted (i.e. by changing the background color) on the downloaded files. Our approach requires n *cloud servers* to store n shares of the files.
- **Data Searcher**: This entity is an individual or a computer application that searches a keyword on the encrypted PDF files stored over the *cloud servers*. This entity can or cannot download the PDF files containing the searched keywords (where the keywords are highlighted), depending upon whether it has required authorization. For example, an organization can store PDF files (e.g., technical specifications of a product) which will later be fetched by its clients where they could be authorized to download the files. In an alternate scenario, a security agency may be allowed to screen through the encrypted PDF documents for certain keywords, but they may not be allowed to download and decrypt until they receive proper authorization.
- **External Attacker**: This entity is an individual or a computer application that attempts to access (upload, search, or download) the encrypted PDF files without authorization. For example, an *external attacker* can try to hack the *cloud servers* to obtain the files.

Threat Model: We assume that the *data owner* is a trusted entity. The trusted entity is authorized to access only those files which contain the searched keyword. On the other hand, the *data searcher* can be a trusted or an honest-but-curious entity. The honest-but-curious entity is allowed to perform keyword searches on the encrypted files, but it is not allowed to download and decrypt any file until they get special permission to do so and their status is changed to trusted entity. A *cloud server* is assumed to be a honest-but-curious entity. In other words, it is assumed that the *cloud server* performs the assigned tasks honestly (e.g., storing, searching, etc.) but is curious to know the content of the file (for which it has no authorization). The *external attacker* is a non-trusted entity.

Further, we assume that k or more *cloud servers* (where $k \leq n$) never collude. The Internet connections between *data owner* and *cloud server* and between *cloud server* and *data searcher* are assumed to be secure (e.g., using network security measures such as SSL and IPSec).

III. The SecureCSearch Method

A. Preliminaries: SSS Scheme

Initially presented by Adi Shamir in 1979, the SSS algorithm [7] grew to importance in the cryptography field

soon after. The main idea behind the algorithm is to compute n shares from a secret S. The shares are arbitrary data and no information about the secret is revealed until and unless k or more shares are brought together to reconstruct the secret, where $2 \leq k \leq n$. That means, no information is leaked with $k-1$ or fewer shares contributing to the secret. This algorithm is also referred as (k, n)-SSS or (k, n) threshold scheme.

When dividing the shares of secret S between n participants, a polynomial function $f(x)$ of degree $k-1$ is employed. This requires the use of $k-1$ random coefficients a_1, a_2, ..., a_{k-1} from a finite field $GF(q)$. The function $f(x)$ is given by:

$$f(x) = (a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1}) \mod q \quad (1)$$

Here, a_0 is the secret S and q is a prime number greater than a_0.

The secret is reconstructed by inputting any k shares to the Lagrange equation and by calculating $f(0)$. The popularity of this algorithm arises from the fact that when encrypting or decrypting a secret, no key is required.

B. Proposed Approach

In our approach, the *data owner* creates n shares of a given PDF file using the SSS scheme. We call these shares as file-shares here on. Note that the n file-shares are encrypted files and stored on n different *cloud servers* (with each *cloud server* storing one file-share). For each PDF file, the share creation and distribution are one-time operations, which can be performed off-line. Later (which can be run-time), the *data searcher* searches for a keyword over the file-shares stored on the *cloud servers* by creating n shares of the keyword; we call the shares of the keyword as keyword-shares. Note that the keyword sharing must be done with the same coefficients that are used in the file sharing. Also, if the i^{th} *cloud server* has the i^{th} file-share, it must be provided the i^{th} keyword-share. During a particular search operation, a *cloud server* fetches only those file-shares which contain the keyword-shares. The locations of the keyword-shares on the fetched file-shares are also noted (for highlighting purpose). The fetched file-shares and the locations of the keyword-shares are then sent to the *data searcher*. By receiving at least k file-shares, the *data searcher*, having necessary authorization, can reconstruct the PDF file. In the reconstructed file, the searched keywords are marked using the PDF's highlighting function.

The following sections explain the process of share creation, keyword searching, and secret reconstruction.

1) Creating file-shares: A PDF file is shared using the (k, n)-SSS scheme. The shares are created for each page of the file independently by sharing the PDF objects in the page (similar to Sharma et al. [8]). The objects are then identified as text objects and image objects. Each type of object contains meta information. For example, the text objects contain font size, text layout and text location, and image objects contain image width, image height, and the location. The extracted text objects are then processed to get words using delimiters. The shares of a word (called word-shares) are created by sharing each character in the word. Here, we assume that the characters are ASCII characters having values less than 128. Thus, in the share creation process, the value of q (in Equation (1)) is taken as 127. The shares of an image object (called image-shares) are created by sharing the color components of each pixel. Each word-share and image-share are then drawn to the PDF file-share using their location and layout information.

2) Selecting coefficients for SSS polynomial: The selection of polynomial coefficients in Equation 1 is one of the most important aspects of the SSS scheme. The coefficients should be selected randomly and must be kept secret (or destroyed). This requirement, however, is an issue for encrypted-domain keyword searching since the text in the file and the searched keyword must have been shared using the same polynomials. This is because the i^{th} shares of a word that have been created using two different set of polynomials will be different. In the context of sharing (or encrypting) the text, the frequency analysis attack is another issue. For example, if a file is shared using only one set of coefficients, then it can be exposed to the frequency analysis attack (as the occurrence of a secret letter, say A, will be known from its shares). We address the frequency analysis attack by using a set of $l = r \times (k-1)$ random coefficients. For sharing a word, $k-1$ coefficients are randomly chosen from the l coefficients, and then, each character of the keyword is shared using these $k-1$ coefficients. For sharing another word, another set of coefficients is chosen. This way, different words are shared using different sets of coefficients, so that the frequency analysis attack is countered (security proof is provided in Section IV). Also, different characters of a word are shared using a single set of coefficients, so that efficient keyword searching can be done in the encrypted domain.

The value of r determines the security against the frequency analysis attack. In the ideal case, r must be greater than or equal to the occurrences of the highest-frequency character. However, obtaining the occurrences of the highest-frequency character from all the pages can be computationally expensive and impractical, as the whole document must be processed to find it. Therefore, we experimentally choose the value of r such that the IC is below the acceptable threshold.

As explained above, all the l coefficients, however, must be securely sent to the *data searcher*. Using these coefficients, the *data searcher* creates shares of the keyword. We fulfill this requirement by creating shares of the coefficients using a different set of coefficients, and then sending shares of the coefficients to the *cloud servers* (along with the file-shares), as depicted in Figure 2. Essentially, the i^{th} share of

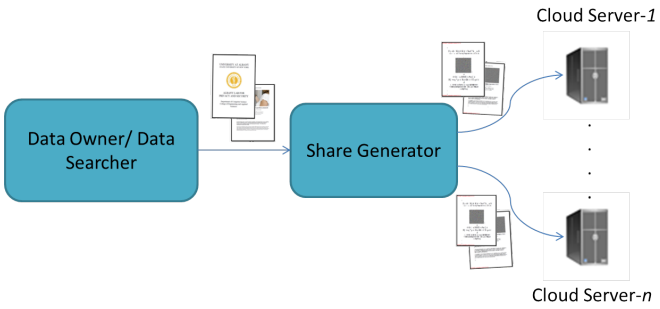

Figure 2: Share creation procedure

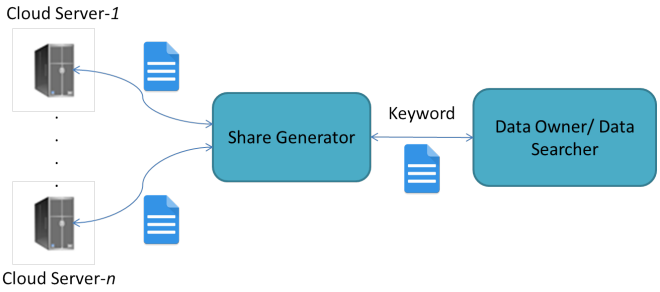

Figure 3: Keyword searching procedure

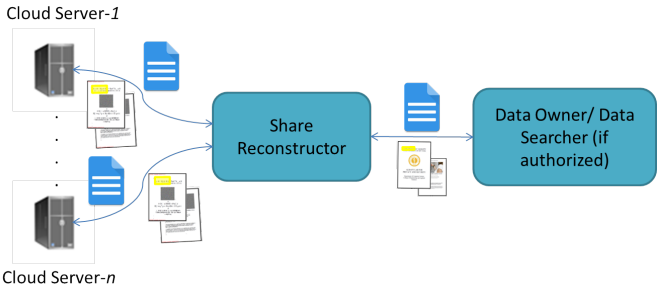

Figure 4: Secret reconstruction procedure

the coefficients are sent to the i^{th} *cloud server* as header information of the i^{th} file-share. Note that the new set of coefficients that is used to create the shares of these coefficients is destroyed. Therefore, no information about the coefficients and the file are known from less than k file-shares.

3) Searching keywords: The keyword searching process is shown in Figure 3. As mentioned before, the keyword searching is done in encrypted domain over the *cloud servers*, i.e, each i^{th} *cloud server* searches the i^{th} share of the keyword in the i^{th} file-share.

The *data searcher* creates the shares of the keyword and sends them to the respective *cloud servers*. However, the shares of the keywords must be created using the coefficients which were used to create the shares of the corresponding word in the file. As discussed in previous sections, the shares of the coefficients that are used to create the file-shares are stored over the *cloud servers* as metadata of the file-shares. Using this metadata, the *data searcher* first reconstructs the coefficients from at least k shares. Then, the *data searcher* generates the shares of the keyword for all possible combinations of the coefficients. With that, $\binom{l}{k-1}$ shares of the keyword are created. Each keyword-share is then sent to its corresponding *cloud server*.

The *cloud servers* perform the keyword searching operation by matching every possible word in the file-share with all keyword-shares. Once the *cloud server* has found a match in the PDF, it returns the file name, location of the keyword (i.e. the page number and the coordinates of the keyword on the page), and the total number of occurrences of the keyword in the PDF.

4) Reconstructing files containing the keyword: The PDF files are reconstructed using the file-shares (which contain the keyword) obtained from at least k *cloud servers*, as shown in Figure 4. Only the *data owner* (who can be the *data searcher*) is allowed to download the PDF files from the *cloud servers*. In the file, each text object and image object is reconstructed by using the reconstruction module of the SSS scheme.

IV. Security Analysis

The security of the proposed method is leveraged by the information theoretic security of the SSS scheme. The only security concern with the proposed method is its resistance against frequency analysis attack. Below, we provide a lemma to prove the security of the proposed method against frequency analysis attack.

Lemma 1. *The SecureCSearch method is secure against frequency analysis attack with a probability* $(1 - \frac{1}{r})$*, where* $r \geq 2$*.*

Proof: In the SecureCSearch method, for the SSS scheme we choose the $l = (k-1) \times r$ coefficients which are used for creating the shares of each character in the PDF. Clearly, if $r = 1$, the SSS scheme works like a monoalphabetic cipher, which is known to be prone to frequency analysis attack. With the (k, n)-SSS scheme, let us assume that for any given character m in the document, the corresponding shares are c_i, $1 \leq i \leq n$. For $r = 1$, m always maps to c_i, $1 \leq i \leq n$, with a fixed mapping, resulting in a high chance of frequency analysis attack.

For $r \geq 2$, the given character m maps to r different sets of ciphertext, i.e. $c_{i,1}, c_{i,2}, \ldots, c_{i,r}$, $1 \leq i \leq n$. This implies that any given share of m can assume a particular value from the set of r values, with a probability of $\frac{1}{r}$. Therefore, the probability with which the character m would have a different value of ciphertext is $(1 - \frac{1}{r})$; making it a polyalphabetic cipher and reducing the probability of a frequency analysis attack to $(1 - \frac{1}{r})$. ■

Table I: Data set containing 6 PDF files with varying number of pages and of different size (in KB)

Total Pages	1	5	10	20	50	100
Size (in KB)	115	450	1138	2260	4510	12560

V. Experiments and Analysis

In order to test the feasibility of the proposed method, we implemented it using C# on a 2.40GHz i7 CPU with 8GB RAM. The developed application can be run on cross-platform such as Windows, Unix and Mac OS. To represent a realistic environment, we simulated the production environment of the server on our local windows machine. In practical scenarios, real servers can be used and the choice of the number of servers to store shares of PDFs is subjective to user preference. While performing tests, for sake of simplicity, we have chosen $n = 3$ and $k = 2$. Also, we varied value of r between 1 and 10.

The proposed method is tested on six PDF files of different sizes in terms of KBs and number of pages. The data set is provided in Table I. Experiments are performed with the following two objectives: 1) analyzing the overhead (i.e. computation and storage) associated with the proposed method (Section V-A), and 2) examining the IC value based on varying r indicating the resistance of the proposed method against the frequency analysis attack (Section V-B).

A. Performance Analysis

1) Share creation and secret reconstruction time: In order to measure the computation time, we analyzed the runtime of the whole process. The computation time can be affected by any task in the process, i.e. share creation time (or encryption time), searching keywords or secret reconstruction time (or decryption time) and thus, we did analysis on all three components.

In terms of the variables affecting the computation cost, following are the two major factors to consider: 1) the number of pages, and 2) the size of the file. As shown in Figure 5, when we upload a one-page PDF file of size 115 KB, the share creation time is 1.123 seconds and the secret reconstruction time is 0.823 seconds. Also, the share creation and secret reconstruction time are 113.890 seconds and 91.158 seconds respectively for a PDF file of 100 pages (of 12560 KB size). The data clearly suggests when we gradually increase the file size, it also increases the share creation and secret reconstruction time. Also, as shown in Figure 5, the secret reconstruction time is less than the share creation time because the latter includes the time to generate a set of l coefficients, pick $k-1$ coefficients randomly from the this set, create their shares and store the shares of coefficients in the header of the PDF as meta information.

2) Search time: The search time is also directly proportional to the content length of the PDF (i.e. the number of pages and the size of the PDF), since the proposed method

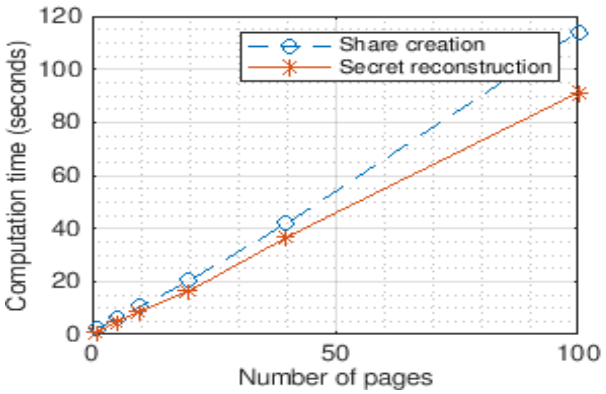

Figure 5: Computation time for share creation and secret reconstruction for PDF files with varying number of pages, with $n = 3$, $k = 2$ and $r = 5$.

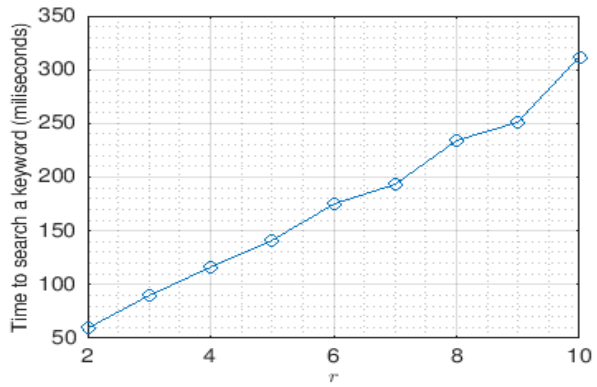

Figure 6: Time to search a keyword in a one-page PDF file for varying value of r.

is applied at the content level of the PDF. As can be seen in Figure 6, the search time is 59.8 ms to find and highlight all the occurrences of a keyword in the one page PDF, when the value of r is 2. On the other hand, it takes 311 ms to perform the same operation when the value of r is 10. The reason behind the increase in search time is that with the higher the value of r, the value of l gets higher as l is directly proportional to r. Another factor that affects the search time is the value of k as we search $\binom{l}{k-1}$ encrypted keywords to find all the occurrences of a keyword in the encrypted file. So, as the value of k grows, the number of encrypted keywords to search for decreases. That clearly states that the search time is directly proportional to $\frac{r \times n}{k}$.

Further, when generating the number of coefficients, l, the value of r affects the share creation time, the secret reconstruction time and the search time. The benefit of choosing coefficients randomly is to cut the computational cost while searching for a keyword in encrypted domain. For example, if coefficients are chosen randomly for each word, it generates $\binom{N}{k-1}$ encrypted keywords to perform the search

Table II: IC of the shares for different values of r

r	IC (Share 1)	IC (Share 2)	IC (Share 3)
1	0.11362	0.15194	0.12148
2	0.05093	0.05078	0.05117
3	0.03344	0.03231	0.03098
4	0.02654	0.02963	0.02698
5	0.02592	0.02537	0.02269
6	0.02365	0.02237	0.02339
7	0.02247	0.02104	0.02197
8	0.02006	0.01945	0.02037
9	0.01935	0.02004	0.01933
10	0.01958	0.02027	0.01912

operation, N being the total number of words in the PDF file. Whereas, if a pre-defined set of randomly chosen coefficients is created, it only generates $\binom{l}{k-1}$ encrypted keywords to search for, which is significantly lower than the set of encrypted keywords created using different coefficients for all words assuming that $l < N$.

In a nutshell, the greater the value of r, the more computation time it takes to create shares, reconstruct the secret and search the keywords. Furthermore, it is important to note that the share creation is only a one-time process. The search and secret reconstruction time can be reiterated as much as needed for different keywords.

B. IC Analysis

Given a text string, the IC is the probability of two randomly selected letters being the same. In the context of the proposed method, the IC is considered as an indicator of how the share of a character is different for its different occurrences. The lower the value of IC, the more different the share is for the different occurrences of characters. As shown in Table II, the IC value of three shares of a given PDF file is in the range of 0.11 to 0.15. It reduces significantly as the value of r increases (e.g. 0.50 for $r = 2$ and 0.33 for $r = 3$, and so on), reducing the probability of frequency analysis attack.

VI. Conclusion

The proposed SecureCSearch method, which uses a modified SSS-based approach, is able to search the keywords in the encrypted PDF documents over *cloud servers*. The SecureCSearch method is not only secure against the frequency analysis attack, but also efficient in terms of computation time as well as storage overhead in comparison to the key-based encryption schemes. Future work will be to extend the method to facilitate additional functions provided by PDF such as searching for a sequence of keywords and image components, and scaling and cropping in multiple PDF documents.

References

[1] P. K. Atrey, K. Hildebrand, and S. Ramanna. An efficient method for protection of text documents using secret sharing. In *Proceedings of the International Conference on Frontiers of Computer Science (ICFocs)*, Bangalore, India.

[2] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Proceedings of the Springer International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 506–522, Interlaken, Switzerland, 2004.

[3] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[4] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, M. Garofalakis, and C. Papamanthou. Practical private range search in depth. *ACM Transactions on Database Systems*, 43(1):2, 2018.

[5] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar. Fast range query processing with strong privacy protection for cloud computing. *Proceedings of the VLDB Endowment (VLDB)*, 7(14):1953–1964, 2014.

[6] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*, pages 85–100, Cascais, Portugal, 2011.

[7] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[8] N. Sharma, P. Singh, and P. K. Atrey. SecureCMerge: Secure PDF merging over untrusted servers. In *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval (MIPR)*, Miami, FL, USA.

[9] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 44–55, Berkeley, CA, USA, 2000.

[10] F.-K. Tseng, R.-J. Chen, and B.-S. P. Lin. ipeks: Fast and secure cloud data retrieval from the public-key encryption with keyword search. In *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 452–458, Melbourne, Australia, 2013.

[11] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. In *Proceedings of the VLDB Endowment (VLDB)*, volume 6, pages 289–300, Riva del Garda, Italy, 2013.

[12] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou. Secure ranked keyword search over encrypted cloud data. In *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 253–262, Genoa, Italy, 2010.

[13] Z. Wang, Y. Luo, and S. Cheung. Efficient multi-party computation with collusion-deterred secret sharing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7401–7405, Florence, Italy, 2014.