Performance Evaluation of Encrypted Data Inquiries using Chained Perfect Hashing (CPH)

Dr. Nada M. Al-Hakkak
Baghdad College of Economic Sciences University
Computer Science Department
Abstract

In this paper, a new scenario has been proposed to retrieve encrypted data on remote servers. Chained Perfect Hashing (CPH) technique will be described for search in encrypted files depending on public key encryption stored on untrusted servers. And retrieve the files without giving any information about the stored files. CPH is secure and support hidden queries. CPH has constant search time with fix rate of collisions for any file size.

Key words: Chained Perfect Hashing, Public Key, Untrusted Server, Retrieve, Hidden Queries.

1. Introduction

It is preferred to store data on remote trusted servers in an encrypted way to reduce security risks. But if a client needed to retrieve certain documents containing some desired words, how this could be done, letting the server answer the query without having knowledge about the document's contents!

Many researches and solutions have been proposed, like making a hash table for the content of each email and encrypt the email, then send all to the server side, or splitting the content of the email in some deterministic parts and save one part on the client then send the rest to the server, and many other solutions, but they all suffer from the communication overhead and long search time that results the disaster of collision.

This paper presents a new solution of the searching on remote encrypted emails in a trusted way that take into account the search time and collision rate, it's Chained Perfect Hashing (CPH), where it will use the idea of Perfect Hashing which will be modified in a chained way, for describing the search time and avoiding any collisions. The expected results should be very fast search time without any collisions.

This paper is structured as follows: first there will be a review of the related works that have some relation with the search in an encrypted documents, then there will be some discussion for the meaning of the perfect hashing, because there will be a use of cryptography algorithm then, Secure Hashing Algorithm (SHA) will be taken as an example, finally (CPH).
2. Related Work

In [1] the author used a new idea, called HSED, Heuristic Search on Encrypted Data, which reduces communication overhead on the email server, it require no additional computation except for simply calculating a hash function that serves as the address for an entry in a Hash Table (HT), but it suffers from collision and construction time that is related to the file size, and it deals with each document alone for searching on specific keyword, which will need a lot of search time.

In [2] the author modified the above idea and called it, GHSED, Applying Advanced Search Technique to Encrypted data, to solve the problem handling each document alone by making a global table contain all documents, also it solved the problem of repeated words in each document, but the time needed to embed heuristic table into GHT could be increased among file size, also it can deal with compressed files, and have high collision rate which will affect the increasing of search time.

In [3] the author used the idea of extendible hashing EH, to search for encrypted keyword over a remote encrypted e-mails in other untrusted servers, its fast and secure with isolation query search but it did not solve the extra time if there was two operations at the same time, for example insert and search, also there is the problem of skewed data, multi records have the same search key so multi search key might been assigned the same bucket, which will cause overflow problems, also it cause overhead.

In [4] the author solved the problem of trying the user to retrieve some encrypted files from server by using encrypted keywords, in a way that ensures security but it will increase the overhead in terms of bandwidth and storage, because we are dealing with user that have mobile cell phones.

In [4] which study the problem of privacy-preserving access to the database, by suggestion of general solutions rely on a new connection between keyword search and oblivious pseudorandom functions, but in the whole paper the author have not show any results for the author have not show any results for the implementation, so there is no idea how it will affect the search time and the level of security.

In [5] it uses the idea of bloom filter. If the query contain the word (X), then the search time $O(1)$ only if the index contain X. It's security depends on the right use of trap door, for generating the secret key. The security model known as semantic security against adaptive chosen keyword attack (IND-CKA), they also develop an efficient IND-CKA secure index construction called Z-IDX using random function and bloom filters. The computational cost was low, but their indexes can securely handle updates.

In [6] it used when two parities share data but do not trust each others, it propose a search scheme based on bloom filter and pohling-hellman encryption. There will be a need of a trusted third party that can transform one party's database, but in a way that neither the third party nor the database owner can see the original query. Also the encryption keys are hidden from the third party, which are used to construct the bloom filter, but now much does the third part trusted, they did not measure that.

In [7] the content of the audit log are very sensitive information, so if an organization whishes to search for certain information, it might need to search all the entries to match some keywords, so it need to be done in a secure way. The server generating and it logs entries encrypt entries with public key corresponding to the keywords that are derived from those entries. The escrow agent, a third trust party, will
construct a search capability for a given keyword as the private key corresponding to the given keyword, and the adversary as the private key corresponding to the given keyword, and the adversary cannot tell which public key was used to create the cipher text, so when an encrypted audit log entry is created, even if it's search keyword are hidden but it introduces considerable overhead.

In [8] which provide more that one technique that have feature of high speed, where the time needed to encryption and search is $O(n)$ for document of size $n$. all these methods or scheme take the form of probabilistic searching, to control the number of errors for some parameters in the encryption algorithm, but their index cannot securely handle updates.

In [9] it focus on the idea of the user was interested in documents containing the same searched keywords, the cost of the proposed model depend on the number of searched keywords, the server should learn nothing other than the results of the conjunctive query, they depend on the feature of Decisional Diffie-Hellman (DDH) and Bilinear Decisional Deffie-Hellman (BDDH) to proof the security of their model, the model contain two scenarios which depend on the type of bandwidth connection, if it was high then the user pre-compute a lot of proto-capabilities and send them to the server which saves it, beside it's belong documents until they are used, which can only used once. If the user has only access to a low-bandwidth connection, he will combine the two parts and access it, but it all depend on the reliability and availability of the connected network, their proposed idea suffer from the high communication cost in order of the number of the keywords, and it's security relies a new hardness assumption.

3. Produced Scheme

3.1. Prefect Hashing (PH)

The most common use of hashing is to organize the database files. A sequential search of 100,000 records for customer will require too much disk I/O. An indexed file scheme (like B-tree) would probably make the search faster, but not faster enough to the responsiveness of customer's requests. Hashing is an efficient solution to this problem; it's a mathematical function that transforms the key to an integer value, the key here is the customer's record, the hashing function uses the record key to calculate a location of the record. An advantage of hashing is speed, and the most obvious disadvantage is collisions, for example we have two customers, two records, first one is 192 and second one is 195, but the two of them have the hash value 105, means for the same position. A simple solution to collision is to store the record to the next empty position. Hash function have the most important feature; speed, and B-tree have the unlimited storage ability, these can be combined together to have Extendable Hashing, which views the hash keys as an index into a page of pointer. A hash function that produces no collision for a given set of keys called perfect hashing function. If the hash function also maps those keys onto consecutive numbers with no gaps, then it is called a minimal perfect hashing function. [11]

Hashing used to solve the problem of large number of elements that no table could handle, uses a hash function $h(k)$ that maps $k$ randomly into slots of a hash-table $T$,
when two keys hash to the same slot, a collision will be accord, this can be solved by chaining. In chaining, we put all elements that hash to the same slot in a linked list.

But this weakness could lead to denial of serves on the application using hashing, that's why it's important to choose a hash function at random, this is called universal hashing. If we do not want to use the chaining method to solve the collision problem, we could go back to the universal hashing family and choose the hash function twice in two levels, the first level is essential the same for hashing with chaining: the n keys are hashed into m slots using a hash function h carefully selected from a family of universal hash functions. Instead of making a list of the key hashing to slot j, we use a small secondary hash table Sj with an associated hash function hj. In order to guarantee that there are no collisions at the secondary level we need to let the size of mj of hash table Sj be the square of the number nj, (mj= nj²).

Example:
K = {10, 22, 37, 40, 60, 70, 75}
Outer : h(k)=((ak+b)mod p) mode m where a=3, b=42, p=101, m=9
Inner : hj(k)=((ajk+bj)mod p)mod mj
If h(75)=2 then key-75 will be hashed to slot 2 of table T, and if h2(75)=1, then the key-75 will be hashed to slot-1 of table S2

There are no collisions in any secondary hash tables, so searching take constant time in the worst case. [12]
3.2. Secure Hashing Algorithm (SHA)

Hash functions also called "digest" or "one way" functions, it has input as string of any length, and its output is a short, fixed length string called a "message digest" or "finger print"; should be collision-resistant. Their advantages are fast, keyless operation and keyed operations, while its disadvantage is applicable to a limited number of situations. The uses of hash functions are in password storage; message and file integrity and commitment scheme. \[17\]

Secure Hash Algorithm SHA, also known as Secure Hash Standards SHS this hash algorithm was published by the United States government. This algorithm can produce an output of 160-bit hash value. \[14\]

Hashing algorithms must be used to ensure the integrity of the message \( M \) which required that the SHA be used. \[15\]

SHA is the third form of cryptology, where the two other forms are (enciphering and deciphering), its one way encryption. A hash is a cryptographic algorithm that takes a data input of any length and produces an output of a fixed length. The hash output is called a digital signature which used for data integrity. The larger the signature the more secure the hash. \[16\]

It takes an input from the large domain and return output in a smaller range. Its easy to compute. \[17\]

![Hash Function Diagram](image)

Figure (3) range values of I/O SHA values

Hash functions must have the following properties: can be applied to any size of data block, produce fixed length output, easy to compute, not feasible to reverse and not feasible to find two messages that give the same length. \[17\]

3.3. Chained Perfect Hashing (CPH)

This section will explain the proposed algorithm in detail, but before that there will be comparison between the proposed algorithm and three of the most related ideas (HSED, CHSED, EH).
Form the above; it's obvious that the proposed idea, CPH have better performance from the others, both in the search time and the collision rates.
3.4 CPH-steps

**Sender-Side:**
1. Scan the document and divide it into n blocks of size m byte, each block will contain number of words separated by blank.
2. For each word, calculate: \(x = \text{ascii (1st char)}, y = \text{sum of ascii (all char)}, h_1(x) \) and \(h_2(y)\).
3. Build the list: the parent node will contain \(h_1(x)\) and the child node will contain \(h_2(y) || E(\text{Keyword} || \text{doc#})\).
4. Encrypt the document using SHA-1024.
5. Send all the above to the server.

**Email Server-Side:**
1. The server receives the encrypted e-mail with its list.
2. The server updates the doc# according to the last number it has (for the client node).
3. The server scans its list for the parent nodes, if exist then just add the child, or add the new parent with its child.
4. The server saves the encrypted documents without index according to its doc# that have been given to it.

**Searcher-Side:**
1. The person who wants to search will send a trapdoor.
2. Enc(trapdoor(keyword_{searcher public key} , E(H_2(y)_{searcher public key})).
3. The server will receive the trapdoor and do the following:
   a. The server checks the signature of the searcher.
   b. The server decrypts(keyword_{searcher public key}, and calc h_1(x).
4. If the server finds a parent = h_1(x) then send doc.s to the searcher
   Else (not found)

**3.5 Complexity time (CPH)**
1. Construction time : O(n^2)  
   Loop: read from file
   \(H_1(k) = \text{parent}\)
   \(H_2(K) = \text{child}\)
   Loop: insert into linked list
   If empty list then add parent & its child
   Else if parent exit then add its child
   Else add parent & its child
2. Search time : O(1)  
   Read word
   \(H_1(k)\)
   \(H_2(K)\)
   Loop: scan the list for existence parent
   If found then scan for existence child.
4. Conclusion and Future Work
Searching in encrypted is an important topic in security area. In this paper, a new scenario has been proposed to retrieve encrypted data on remote servers. Chained Perfect Hashing CPH technique will be described for search in encrypted files depending on public key encryption stored on untrusted servers. And retrieve the files without giving any information about the stored files. CPH is secure and support hidden queries. The proposed technique has been tested on number of files range in size from 250 K.B. up to 6000 K.B. and proven to have minimal collision rate and search time.

5. References