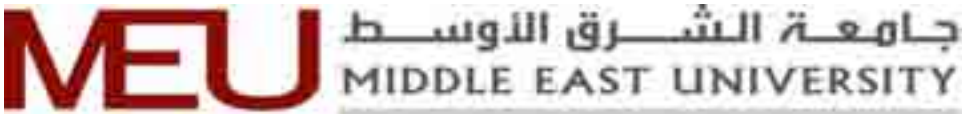


Enhancing The Bellovin and Cheswick's Algorithm by Adding Dummy Values and Reorder Process

By

Fadi Yousef Ali Eshtaiwi

Supervised by

Dr. Oleg Viktorov

Master Thesis

Submitted in Partial Fulfillment of the Requirements for the Master Degree in Computer Science

Department of Computer Science

Faculty of Information Technology

Middle East University

Amman, Jordan

January, 2012

Authorization Statement

I, Fadi Yousef Ali Eshtaiwi, authorize Middle East University to supply hardcopies and electronic copies of my thesis to libraries, establishments, or bodies and institutions concerned with research and scientific studies upon request, according to the university regulations.

Signature : 

Date : 23 / 1 /2012.

أنا الطالب ، فادي يوسف علي الشتيوي ، أفوض جامعة الشرق الأوسط لتزويد نسخ من هذا البحث الى المكتبات ، المؤسسات ، المعاهد والأشخاص وحسب الطلب .

الإسم : فادي يوسف علي الشتيوي .

التوقيع : 

التاريخ : 2012/1/ 23

Examination Committee Decision

This is to certify that the thesis entitled "Enhancing the Bellovin and Cheswick's Algorithm by Adding Dummy Values and Reorder Process" was successfully defended and approved on January 23rd 2012.

Examination Committee		Signature
Dr. Oleg Vikronov Department of Computer Science Middle East University	Chairperson& Supervisor	
Dr. Hussein Hadi Owaied Department of Computer Science Middle East University	Member	
Dr. Hussein Ismail Al-Bahadili Faculty ofInformation Technology Petra University	Member	

Declaration

I do my hereby declare the present research work has been carried out by me under the supervision of Dr. Oleg Viktorov and this work has not been submitted elsewhere for any other degree, fellowship or any other similar title.

Name : Fadi Yousef Ali Eshtaiwi.

Signature : 

Date : 23/1 /2012.

Dedication

To my father, Yousef Eshtaiwi, for being my idol in the life , my mother, for being my sunshine, my sisters , and to my wife, Natasha , for her support, and for her great patience.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Oleg Viktorov, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate his vast knowledge and skill, and his patience.

I would also like to thank my wife Natasha for the support she provided me through my entire life and in particular, I must acknowledge her, without whose love, encouragement and editing assistance, I would not have finished this thesis.

Table of contents

List of figures

Figure **Page**

List of Tables

List of Abbreviations

AES Advanced Encryption Standard

DES Data Encryption Standard

FTP File Transfer Protocol

MAC Message Authentication Code

MD5 Message Digest Algorithm

TCP Transmission Control Protocol

IP Internet Protocol

UDP User Datagram Protocol

NACKs Negative acknowledges

RTP Real-Time Transport Protocol

SRM Scalable Reliable Multicast

URGC Uniform Reliable Group Communication Protocol

MFTP The Multicast File Transfer Protocol

LBRM Log-Based Receiver-reliable Multicast

STORM Structure-Oriented Resilient Multicast

MIT Massachusetts Institute of Technology

KDC key distribution center

FTP File Transfer Protocol

POP Post Office Protocol

NFS Network File System

MD5 Message-Digest Algorithm

SHA Secure Hash Algorithm

NIST National Institutes of Standards and Technologies

DSA Digital Signature Algorithm

PKE Public Key Encryption

PEKS Public key encryption with keyword search

IBE Identity Based Encryption

SUNDR Secure Un-trusted Data Repository

BFS Byzantine fault tolerant file system

BFT Bloom Finger Table

SNS Social Networking Sites

$\mathbf{E_{KUB}}$ Encryption query by using public key of Receiver

$\mathbf{E_{KRA}}$ Encryption query by using private key of Sender

$\mathbf{D_{KRB}}$ Decryption query by using private key of Receiver

$\mathbf{D_{KUA}}$ Decryption query by using public key of Sender

D_{KRA} Decryption query by using private key of Sender

D_{KUB} Decryption query by using public key of Receiver

Terminologies

Security: The security of a system is the ability of the system to support the system availability, data integrity and confidentiality. So if the system fails to support these three characteristics or protect them, then the system amounts to a security violation or weakness.

Network Security: The network security is the protection of a computer network and its services from unauthorized modification, destruction, or disclosure. In other words, the network security is the process to make sure that the data or services will reach the target workstation and data, services will be protected from hackers.

Cryptography: it is the science and study of hiding information and secretes writings.

Digital signatures: a property that used to signing the messages.

Authentication: Only eligible and authorized voters can vote and each voter can vote only once.

Privacy: all votes must be secret. No participant other than a voter should be able to determine the value of the vote cast by that voter. In other words, neither election authorities nor anyone else can link any ballot to the voter who cast it.

Protocols: a set of rules governing communication within and between computing endpoints or entities.

Reliability: all possible steps shall be taken to avoid the possibility of fraud or unauthorized intervention affecting the system during the whole voting process.

Verification: is the act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics.

Database: is a structured collection of records or data that is stored in a computer system. A database relies upon software to organize the storage of data.

Flexibility: a system is flexible and simple, not complex.

Eligibility: is a decision making process where a population chooses an individual to hold official offices.

Enhancing The Bellovin and Cheswick's Algorithm by Adding Dummy Values and Reorder Process

Prepared by
Fadi Yousef Ali Eshtaiwi

Supervised by

Dr. Oleg Viktorov

Abstract:

Bellovin and Cheswick published a paper titled "Privacy-Enhanced Searches Using Encrypted Bloom Filters" and focused on the third party problem. It is often necessary for two or more parties that do not fully trust each other to share data selectively. They propose a search scheme based on Bloom Filters and group ciphers encryption. A semi-trusted third party can transform one party's search queries to another party's database. Third party problem shows in the privacy and it's always a sensitive position because it's the controller and it has all the secret of the agency or company and no one can send any data without its permission because it has all the keys between the sender and recipient. Third party is used in many applications nowadays except any program authored by Microsoft is a first party application.

Enhancing the Bellovin and Cheswick's algorithm by Adding Dummy Values and Reorder Process will help us to increase the privacy of search process and eliminate the "Third Party" by adding new features to Bellovin and Cheswick's algorithm.

The new algorithm will eliminate the "Third Party" and it will be appropriate for two or more parties that do not fully trust each other to selectively share data. So, two intelligence agencies may wish to let each other query their databases, while only disclosing clearly relevant documents to the other party. Even then, there may be restrictions that must be observed. So, the first part will enter the shared database and looking for any data it wants and no one can figure out what the information you look for.

تحسين خوارزمية بيلوفين وشيسويك بإضافة قيم وهمية وإعادة الترتيب

إعداد

فادي يوسف علي إشتيوي

إشراف

اوليج فيتوروف

الملخص

قام كلا من Bellovin و Cheswick بإصدار مقالة بعنوان "محركات البحث ذات الخصوصية المتطورة باستخدام معايير بلوم (Bloom) المشفرة" مع التركيز على مشكلة الجهة الوسيطة. من الضروري لأي طرفين غير موثوقين بشكل مطلق أن يقوما باختيار المعلومات المرغوب بمشاركتها فيما بينهما بحرص. حيث يقوم كل طرف بتقديم طريقة للبحث تعتمد على معايير بلوم و المجموعات المشفرة. تستطيع أي جهة غير موثوقة كلياً بنقل استعلامات جهة ما إلى قاعدة بيانات جهة أخرى، حيث تظهر مشكلة الجهة الوسيطة في الخصوصية و موقعها الحساس إذ أنها الجهة التي تتحكم بالبيانات و التي تمتلك المعلومات السرية الخاصة بالشركة و التي تقوم أيضاً بمنع تبادل البيانات دون إذن منها باعتبارها الجهة المالكة للمفاتيح بين المرسل و المستقبل. لقد بات من الشائع حالياً استخدام الجهة الوسيطة في عدة تطبيقات باستثناء تلك المصدرة من قبل مايكروسوفت على أنها طرف أساسي .

تكمن أهمية تطوير خورزمية Bellovin و Cheswick عن طريق إضافة قيم ثانوية و طريقة إعادة الترتيب في أنها ستساعد على تحسين خصوصية عمليات البحث و الحد من استخدام الجهة الوسيطة.

ستعمل الخوارزمية الجديدة على الحد من استخدام الجهة الوسيطة و سيتمكن أي طرفين غير موثوقين أو أكثر من اختيار البيانات المرغوب بمشاركتها. على سبيل المثال، قد ترغب جهتان بمنح بعضهما صلاحية الاستعلام على قاعدتي بياناتهما مع القدرة على الحد من المستندات المتوفرة للجهة الأخرى. بناء على ذلك، من الجدير بالذكر أنه لابد من وضع بعض المحددات لهذه العملية، فقد تقوم الجهة الأولى بالدخول إلى قاعدة البيانات المشتركة والبحث عن المعلومات المعنية بدون إظهار المعلومات التي يتم الاستعلام عنها.

Chapter 1

Introduction

1.1 Overview

The basic reasons the companies and partners care about information systems security are that the information of customers and clients needs to be protected against unauthorized access for legal and competitive reasons; all of the information that are stored and referred to must be protected against accidental or deliberate modification and must be available in a timely fashion. Finally, the poor security practices allow damage to systems, it may be subject to criminal or civil legal proceedings; if the negligence allows third parties to be harmed via compromised systems, there may be even more severe legal problems.

When information is read or copied by someone not authorized to access, the result is known as loss of confidentiality. For some types of information, confidentiality is a very important attribute. Examples include research data, medical and insurance records, new product specifications and corporate investment strategies. In some locations, there may be a legal obligation to protect the privacy of individuals. This is particularly true for banks and loan companies; debt collectors; businesses that extend credit to their customers or issue credit cards; hospitals, doctors' offices, and medical testing laboratories; individuals or agencies that offer services such as psychological counseling or drug treatment; and agencies that collect taxes (Pesante, 2008) .

Information is a fundamental human right and a cornerstone of a democratic society. It lays at the foundation of the rule of law, the secret ballot, doctor-patient confidentiality, lawyer-client privilege, the notion of private property, and the value our society places on the autonomy of the individual. With the development of new information and communication technologies, the ability of the state and the private sector to collect, record and "mine" personal information has grown exponentially. As early as 1996, Bruce Phillips, the Privacy Commissioner of Canada, warned, "We are in fact buying and

selling large elements of our human personae. The traffic in human information now is immense. There is almost nothing the commercial and governmental world is not anxious to find out about us as individuals."

Security is a very big issue in networking and it was only interests the military, but the Internet changed all that everything can be done in the real word it can be done on the internet: conduct a private conversation , keep personal papers , sign letter and contracts , vote , publish personal documents, payment and bank transactions, but all of these required a security. Computer security is a fundamental to enabling the technology of the internet. The limits of security are the limit of Internet. The lake of security can produce: loss of customers, damage to brand and loss of goodwill (Schneier, 2005).

People have attempted to hide certain data/ information that should be kept private by substitution of the information parts with numbers, pictures and symbols; this introduction on the encryption history highlights the chronology of Cryptography throughout the previous centuries. Humans were interested in encryption or protecting of their private messages for different reasons. For example, the Assyrians have been interested in protection of their pottery manufacturing trade secrets. Also, Chinese have been interested in protection of their silk manufacturing trade secrets. Moreover, Germans have been interested in protection of military secrets (using their Enigma machine which was famous). (SANS,2001).

Information security and privacy is a very important issue for any organizations and companies especially when there are sharing the data between them. It is necessary when there are two or more parties that they do not trust each other and they share data between them. For example if there are two intelligence agencies that share the data and they wish to let each party to query the other databases without disclosing the query and know what the other is searching for specific data (Bellovin and Rescorla , 2005).

Sometimes it is needed to store critical data on such untrusted server of database. Song, Wagner and Perrig have obtained a way that searches for a word that is existed in

an Encrypted textual document. The search speed has a linear property in the size of documents (Doumen et al, 2004).

Cryptography is considered as the fundamental tool of the information security. There were many techniques and algorithm were had been used in the past in order to make their message an ambiguous (meaningless) in order if their messages went to the wrong hands, they will not understand anything. There are many types of cryptography whereas if they are classical types or modern types (Stamp, 2006).

1.2 Problem Statement

Bellovin and Cheswick published a paper titled" Privacy-Enhanced Searches Using Encrypted Bloom Filters" and focused on the third party problem. They proposed an algorithm to enhance the privacy and eliminate the third party. They used the hash functions to hash the queries and used the bloom filters. The type of encryption that used in their research is Pohlig-Hellman encryption. A Bloom filter is also used in their research and shows how it's a very efficient way to store information about the existence of a record in a database. They measured the performance of their suggested system by two factors:

1- The speed of Pohlig-Hellman encryption
2- The ability of a site to rapidly search many Bloom filters.

Third party problem shows in the privacy and it's always a sensitive position because it's the controller and it has all the secret of the agency or company and no one can send any data without its permission because it has all the keys between the sender and recipient. Third party is used in many applications nowadays except any program authored by Microsoft is a first party application.

This research will develop the Bellovin and Cheswick's algorithm in-order to enhance the privacy of search process by adding new features. The reason behind this developing is making the query ambiguous and never can see what the original query by the owners of the database and without using the third party by using two key in the encryption process. At the end of this research, the new algorithm will be implemented and check the query at both side and calculate the spent time of the search process in the shared database.

1.3 Objectives

1.3.1 Study Aim

The main aim of this project is eliminating the third party in the middle. The client has no knowledge about database collection stored on server side and the server has no knowledge about the query words.

1.3.2 Study Objectives

Eliminate the third party.

- Implement the RSA algorithm.
- Calculate the collision rate.
- Calculate the search time query.

1.4 Study Significance

The significance of this study is to establish a model for how to search on encrypted data without including any third party, the main aim from this project is to query server database without disclosing the query and without knowing server key. RSA algorithm will be used depending on Private and Public key in order to encrypt and decrypt the query. There are also three different hash functions which will be used to represent all of the words that are used. The reason behind use three different hash functions is building a bloom filter on the server side. In this project there will be a generator which is used to generate random numbers for dummy words on the client side. Implementation this project will show the collision rate and the search time query. The work of this project starts by allowing both the client and the server to exchange their public keys and transferring of the query from the client to the server will be encrypted so that no third party is involved in this operation, to prevent the server from knowing the client query we use a technique that is built upon dummy values and reordering of the generated hashes.

Chapter 2 Literature Review

Kammuller, F. and Kammuller, R. (2009) discussed an approach for enhancing the privacy of a database enquiry; the approach solves the problem of the privacy by performing the search instead by not disclosing the search key. They had implemented a demonstration of the concept in Erlang (programming language) where the feasibility of the concept was achieved through Erlang's high scale parallelism. With the implemented method, the security goal of keeping the data private was achieved. The achieved results each time were evaluated for improvements. This is a simple enough extension that merely needs to identify new file names, or more generally Internet sites, to continue the database enquiry. The problem with the implemented method was that an observer could infer some information about the key from the way they continued their search because they selected the file names from the matched search results. To overcome this, they had to cover up their search and load all possible files referenced in the previous round. However, for the sake of efficiency, they would only really analyze those that they know to be interesting. For the future work, they planned to exploit a rigorous translation from Erlang to the calculus to represent their application in a calculus that is more easily accessible to a formal analysis, and then they suggested the use of existing formalizations of non-interference for the calculus to demonstrate information flow security.

Shiraki, T. et al. (2009) proposed a method based on P2P (Peer to Peer) user search based on movement records which are obtained automatically by locating detection devices. In their research, they assumed the movement records to be treated as a sequence of pairs of spot-ID and time and they are stored in a peer for each user. A Bloom Filter was applied to combine all movement records for one user as a fixed length bit array. To search a user who followed specified course, they proposed an OR/AND search method which is based on (BFT) Bloom Finger Table which extends a routing table of a Chord DHT system to retrieve elements using Bloom Filter. Using this method, user searches based on a sequence of locations with or without time can be realized efficiently. Moreover, in order to reduce the number of messages for a user search, they proposed a peer-ID assignment for BFT based on user's geographical foothold. The number of messages for a user

search can be reduced by this peer-ID assignment since users who visit same places are located closer to each other on the routing table. Evaluation results of simulations showed reduction in the number of messages compared to a naive implementation using existing P2P retrieval method. For the future work, they suggested the overhead of constructing Bloom Finger Table to be reduced. In addition, another experiment performed by real users should be evaluated for verifying the effectiveness of geographical peer-ID assignment.

Gou, C. et al. (2010) discussed the traditional intrusion detection equipments that satisfy the application requirements hardly as the data rates of modern networks rise. Adaptive load balancing algorithm may make attacks undetected due to flow remapping. In their research, they proposed an algorithm that is load-balancing based on the bloom filter. When a packet arrives load balancing module, they first determine whether the packet belongs to a new flow. If it is, they calculate the corresponding processing unit through the HRW algorithm with current weights and otherwise calculate the corresponding processing unit with the weights before adjustment. To determine whether the arriving packets belong to the old flow or not, it needs store the identifier of processed packets in a collection for query. Tens of thousands of flows will be generated per second in high-speed network, so the elements preserved in the collection will be very large. The retrieval speed of linked lists and other data structures such as self balancing binary search trees, tries hash tables, or simple arrays is getting slower and slower as the elements in the collection increase. Compared with the above data structures, Bloom Filters in space and time has a huge advantage. Bloom Filters storage space and insertion query time are constant and can be well positioned to meet the application. By analyzing the flows whose size are within the duration of 2 Δt can avoid flow remapping because of weights adjustment. Experimental data they had done shows that the algorithm has the similar load balancing effect, but with a lower rate of flow remapping.

Aïmeur, E. et al. (2010) discussed the different privacy issues raised by the current SNS (Social Networking Sites); the problem of the simple website that allowed users to create profiles, list friends and browse through their friends list, SNS are the place for keeping

in contact with old friends and meeting new familiarities. A user leaves a big trail of personal information about him and his friends on the SNS, sometimes even without being aware of it. This information can lead to privacy drifts such as damaging his reputation and credibility, security risks and profiling risks. This research paper highlighted some privacy issues raised by the growing development of SNS and identifies clearly three privacy risks. While it may seem apriori that privacy and SNS are two antagonist concepts, they identified some privacy criteria that SNS could fulfill also in order to be more respectful of the privacy of their users. Moreover, they introduced the concept of a Privacy-enhanced Social Networking Site (PSNS) and described Privacy Watch, their first implementation of a PSNS.

Song, et al. (2002) discussed the need of data storage on the servers in an encrypted format in order to reduce the privacy and security risks. They described their cryptographic schemes regarding the searching problem on the encrypted data. They presented new techniques in the field of remote searching on the encrypted data; searching is done using untrusted server. Moreover, they provided proofs for the resulting crypto systems. They showed the ways of supporting functionality of searching on the encrypted data without losses in the data confidentiality. The discussed techniques/ approaches have different crucial advantages; some of the advantages are the provable security of the techniques; techniques provide secrecy for encryption that is provable, any untrusted server can't learn/reach anything regarding the plaintext only and only when the ciphertext is given. Another advantage is providing query separation for the searches and hence no untrusted server can learn anything about the plaintext more than the results of the searching process. Another important advantage is providing control in searching; hence without authorization from the user, the untrusted server will not be able to search for any arbitrary word. Note that with the discussed techniques, hidden queries are also supported and hence the user can ask the server (untrusted) to search for a word without any revealing of the word to the untrusted server. The presented techniques/ algorithms are fast, simple and introduce often no communication and space overhead, all these advantages and properties of the presented techniques are hence practical to be used today. Their scheme of remote searching is very flexible also and can be extended easily

for supporting more advanced search queries. It can be concluded that the presented scheme is very powerful for constructing secure services/ searching in the untrusted servers/ infrastructure.

Li et al (2004) discussed the ways in which well defined consistency semantics can be provided for an untrusted server. For data security and integrity,many non-networked file systems used cryptographic storage. They presented the first system of its type to provide well defined consistency semantics for an untrusted server, called as SUNDR (Secure Untrusted Data Repository). A protocol named SUNDR was published previously but not implemented because it didn't address the write-after-write conflicts (had no groups). Their presented system doesn't require replication (place of trust in machines/ servers other than the user's client). Replication is used in BFS (Byzantine fault tolerant file system) in order to insure the required integrity of the network file system. SUNDR however provides freshness guarantees that are weaker than BFS. SUNDR system uses hash trees to verify the integrity of a file block without any touch for the entire file system. SUNDR uses also version vectors for detection of consistency violations. These version vectors can detect update conflicts found between the replicas of a file system and have been also used for securing partial orderings. Their presented file system resembles timeline entanglement and reasons about the system states' temporal ordering using hash chains. The system presented guarantees provably fork consistency and ensures the behavior of the server whether it behaves in the correct way or failure will be detected upon communication between users. Measurements of their implemented system show the performance which is usually close or even better than BFS file system. They also gave a recommendation that with the reduction of the amount of trust in the server, the presented system will increase both the people's options for data managing and will improve the security of the files as well.

Bellare et al. (2005) presented a strong definition of data privacy, and the constructions for achieving them, experiments showed that for encryption schemes of public-key where the encryption algorithm is deterministic; they obtained as a consequence type of database encryption methods which allows fast search while providing provably privacy

which is strong. They explained one of their constructs, called as RSA-DOAEP, has an added feature for length preserving, hence it is an example of a public-key cipher. This was generalized to obtain a notion of searchable encryption schemes in an efficient manner which permits more privacy flexibility to search the time trade-offs through a technique that is called as "bucketization". Obtained results can answer several questions that were asked in the database community and can provide the foundations for the work done over there. These are the schemes that permit fast search. Note that encryption can be randomized, but there is a collision-resistant, deterministic function of the plaintext which can be also computed from the cipher-text and serves as a tag, hence permitting for the fast search based on comparison. Schemes that are based on deterministic encryption are a special case where the security's notion remains the same. The generalization's benefit is to permit the schemes with more privacy flexibility to search for time trade-offs. They analyzed a scheme from the literature of the database which they called as 'Hash-and-Encrypt'. It performs encryption of the plaintext with a scheme which is randomized but includes a deterministic, collision resistant hash in the ciphertext. With the presented scheme, there are some losses of privacy because of the lack of entropy in the message space which can be compensated by increasing the probability (δ) of the hash collisions; this can be done so by truncating the hash function's output for example. The trade-off can be explained that the receiver gets the false positives in response to a search query required and should spend some time to shift through them to obtain the required true answer. This technique is called 'bucketization' in the literature of the database, but the security of this technique was not rigorously analyzed previously. There implemented scheme provides privacy only for the plaintext which have high min-entropy. This cannot be considered as a weakness of the technique but is inherent in being efficiently searchable or deterministic. Their claim was to provide the best privacy subject that is possible to allow for the fast search. This may refer to no privacy in some cases but they commented that bucketization may increase the privacy when the fields of the database being encrypted don't have high min-entropy.

Jonker et al, (2004) sometimes it is needed to store critical data on such untrusted server of database. Song, Wagner and Perrig have obtained a way that searches for a word that

exists in an Encrypted textual document. The search speed has a linear property in the size of documents. The paper related with expansion search algorithm of the tree type algorithm based on the algorithm of the linear searches that are proper for XML databases. This new approach is more efficient where it exploits the structure of XML. And also building prototype implementations for both the tree search and linear case. Experiments show a main development in the time of search. Nowadays the need grows to keep stored data secure on an untrusted system. Think, for instance, of a remote database server administered by somebody else. Suppose you need the data to be secret, then it should be encrypted. The problem then arises how to get a response for database query. The most obvious solution is to download the whole database locally and then perform the query. This as known is completely inefficient. Song, Wagner and Perrig have introduced a protocol to search for word or letter in some encrypted text. The paper will propose a new protocol that is more suitable for handling the large scale (semi-structured) XML data. The new protocol exploits the XML tree structure. XPath queries can be answered secure and fast. The implementations of the prototype have been developed during the paper for both the linear and the protocol of tree search. Those prototypes have been used for finding optimal settings for the parameters used within the protocols and showing the increasing in the search speed by using the tree structure.

Curtmola et al (2006) published a paper in the searchable symmetric encryption (SSE) which it aims to allow the party outsourcing the storage of the party's data to another party in a private field, while it maintains the ability of the party to search over it. This problem is discussed in this search. It has been the focus of active several security constructions and definitions have been proposed. The authors reviewed the existing security definitions, pointing out their shortcomings, and then they will propose two new stronger definitions. Curtmola et al, present two constructions that can show secure under their new definitions. In addition to satisfy a stronger security guarantees, the authors' constructions are more efficient than the all previous of the existing constructions. Previous work on the SSE considered the setting where only the owner of the data is capable of submitting the search queries. The authors defined SSE in the

multi-user setting and they present an efficient construction. The results of this paper are summarized as the following:

1- Review the existing security definitions that are used in the searchable encryption. It includes the simulation-based definition in CM05 and the IND2-CKA Goh03.

2- Authors introduced new adversarial models for searchable symmetric encryption (SSE). They refer to as non-adaptive; they consider the adversaries make their search queries without considering the trapdoors and taking into accounts.

3- They present two constructions which the authors prove secure under the new definitions. The first scheme will be secured only in the non-adaptive setting. On another level, it is the most efficient SSE in the construction to date in order to be achieved. The searches can be achieved in one of each communication round which requires an amount of any work on the server proportional to the specific number of the documents. These documents contain the queried word that requires constant storage on the client and linear storage on the server.

Boneh et al, (2004) studied the problem of searching process on data which is encrypted by using a public key technique. Consider user B who will send an email to user A an encrypted under A's public key. An email gateway will test whereas the email contains the keyword "urgent". So it will route the email accordingly. On the other hand it will not give the gateway the ability to decrypt all messages. Boneh et al defined and then construct a mechanism that gives enable user A to provide a key to the gateway of the email that will enable the gateway to be tested if the word "urgent" is a keyword in the email, this will be done without learning anything else about the email. The authors refer to this suggested mechanism as the Public Key Encryption with keyword Search. They defined the concept of encryption using public key techniques with keyword search and then give several constructions.

Bellovin and Rescorla, (2005) in this study cryptographic protocol was analyzed where this protocol depends directly on the hash function. Also this study focused on SHA-1 [Nat02] and MD5 [Riv92] which are widely used. These hash functions are usually derived by using MD4 [Riv90]. This was known to be slight [Dob96, Dob98] for a long

period and thus will lead to concerns that the authors might have common weaknesses. It is clear which it will be necessary to do this in the not-too-distant. This will result a number of challenges for certificate-based protocols in a specific way. Bellovin and Rescorla, analyze a number of protocols which include TLS and S/MIME that will result change in the way of implementation the change. They explain the necessary changes and show how the conversion that will be done, and then list what the measures that should be taken immediately.

Chapter 3

Methodology

3.1 Background

This project is designed to remove the third party and enhance the privacy in the search process by using the bloom filters. This project to be implemented will require to design and write many functions starting with how to make a connection between client and server. In addition, hash functions will be used in both server and client. Bloom filter will be used to build the server's database. The query will be encrypted and decrypted based on RSA technique (private and public). This project will need to read many books, journal, websites and conferences. First, the data for this thesis will be obtained by reviewing the previous study in cryptography, bloom filters, hash functions and other materials. The suggested algorithm required using RSA algorithm (Public and Private Keys), bloom filters and hash functions (present the words into database). Implementation this project will be done by using Visual basic.Net and SQL Server 2005. Vb.net will write an appropriate to code to program all of RSA algorithm, hash functions, adding dummy values and reorder the query. SQL server 2005 will build the server's database within the VB.net. The code will contain the three techniques and then the results will be seen as the spent time of the query and the collision rate.

Figure 1 shows Bellovin and Cheswick's Algorithm **(Before Development).**

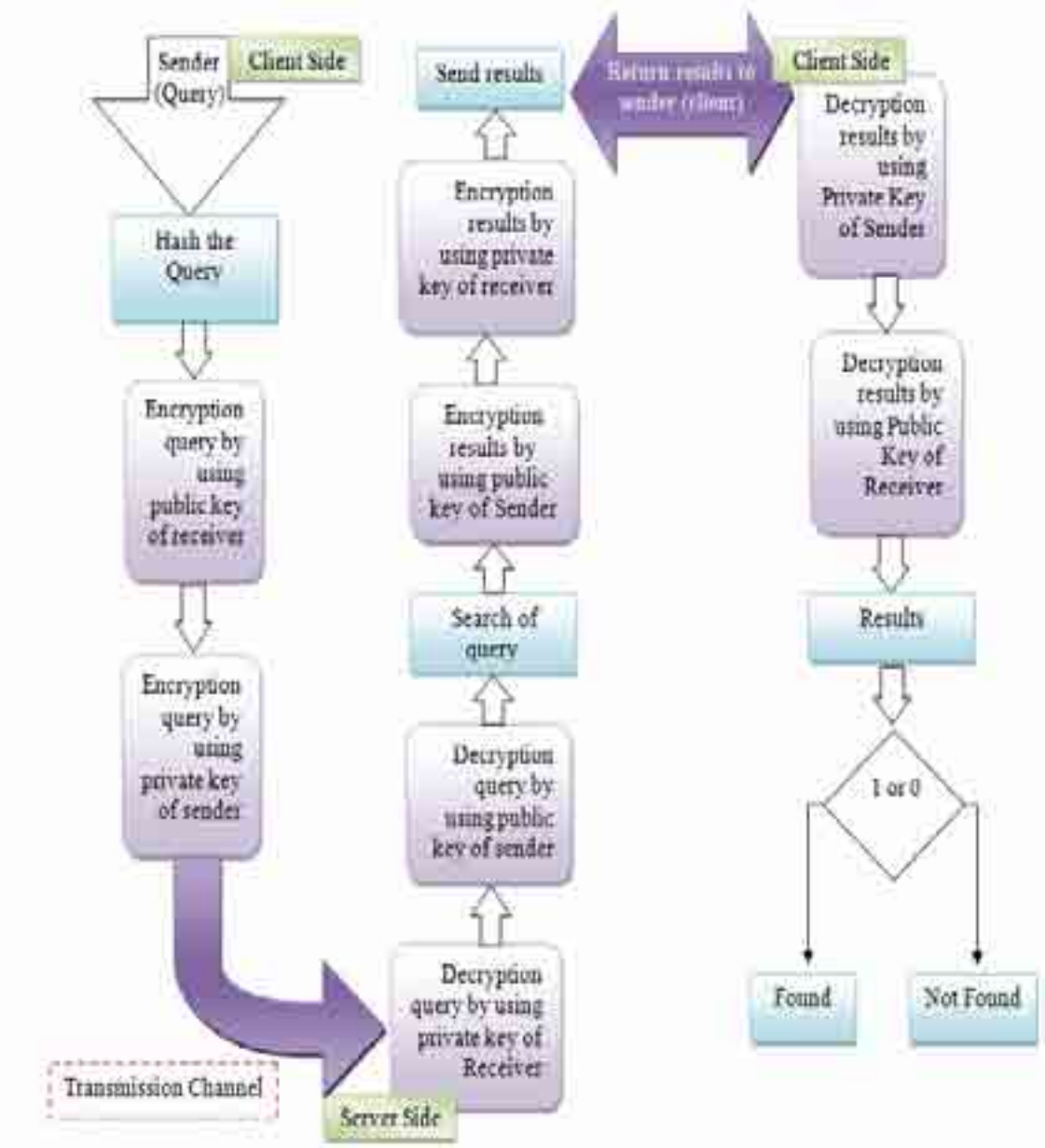


Figure 1: Bellovin and Cheswick's Algorithm.

Figure 2 shows Bellovin and Cheswick's Algorithm **(After Development).**

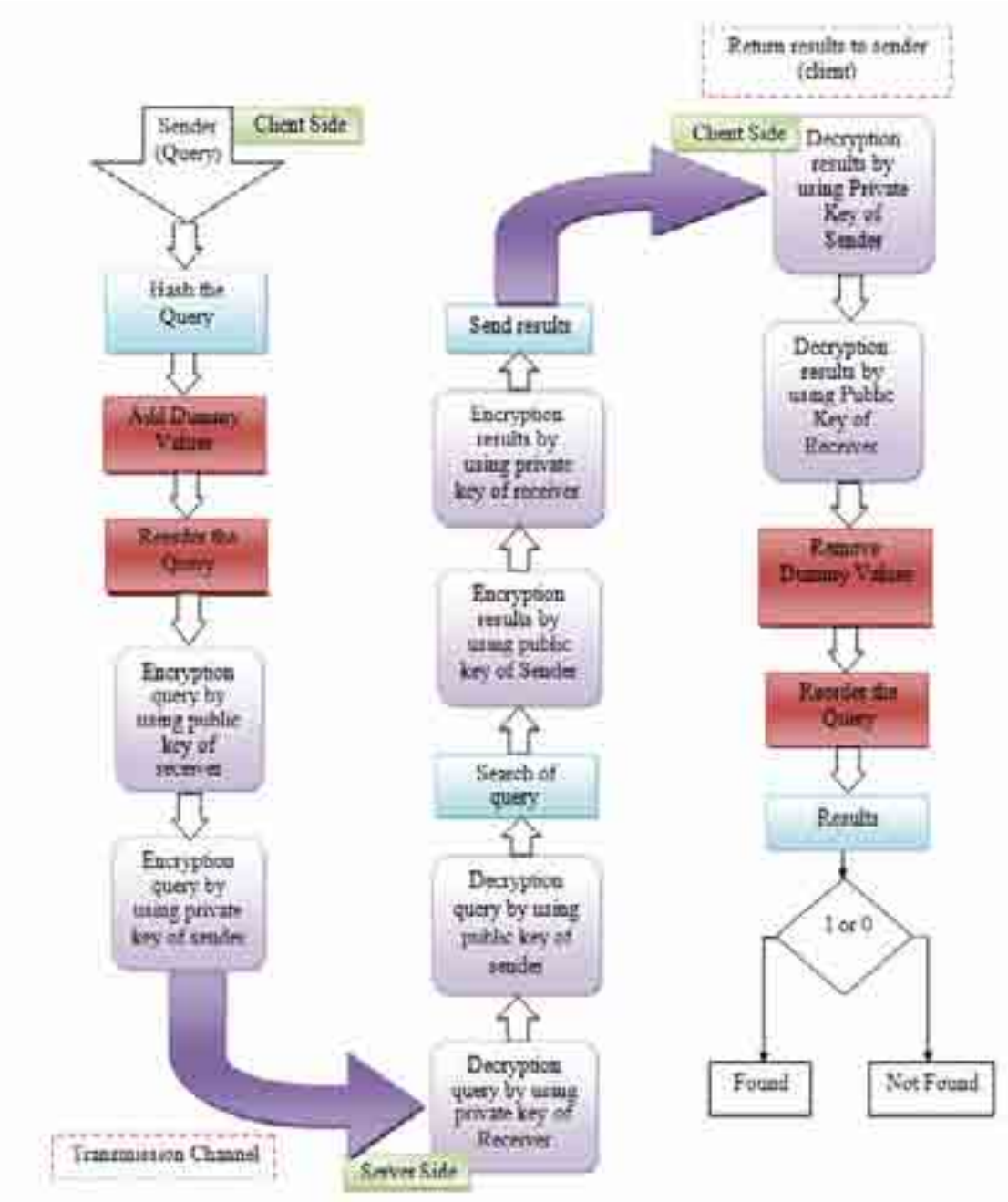


Figure 2: Bellovin and Cheswick's Algorithm (After Development).

There are many steps will be included in order to achieve this project. See Figure 3.

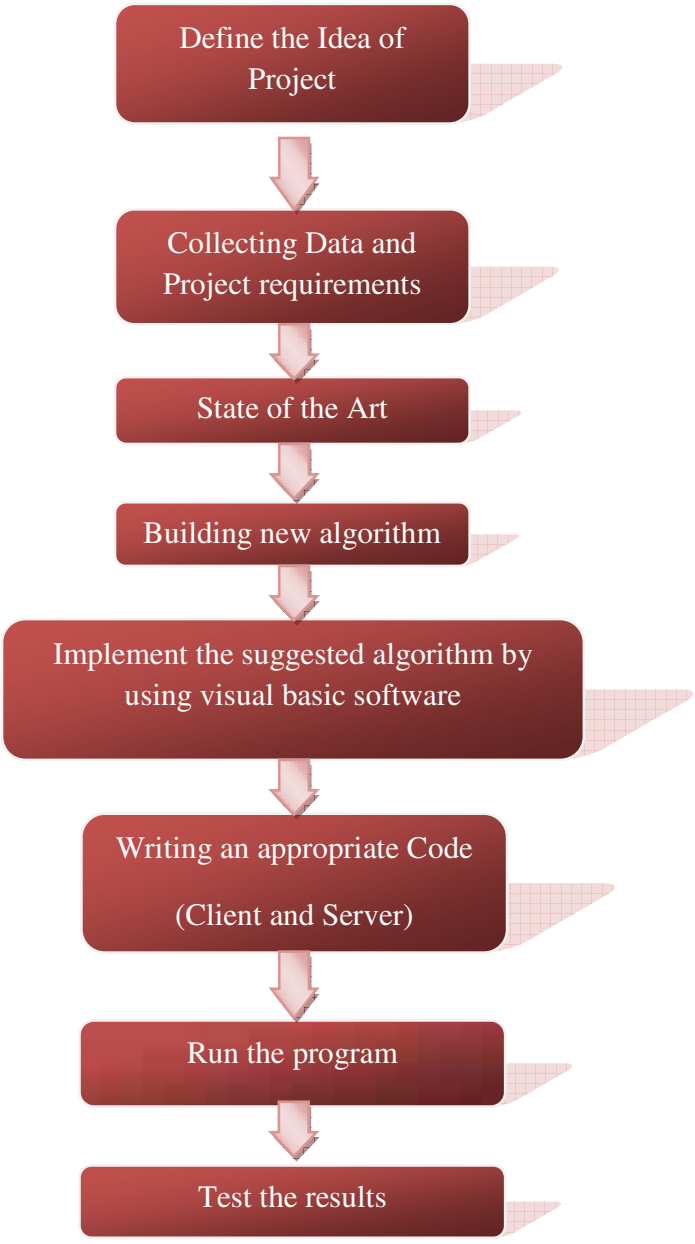


Figure 3: Design Methodology.

3.2 Project Implementation Requirement

Figure 4 shows an example of the suggested system.

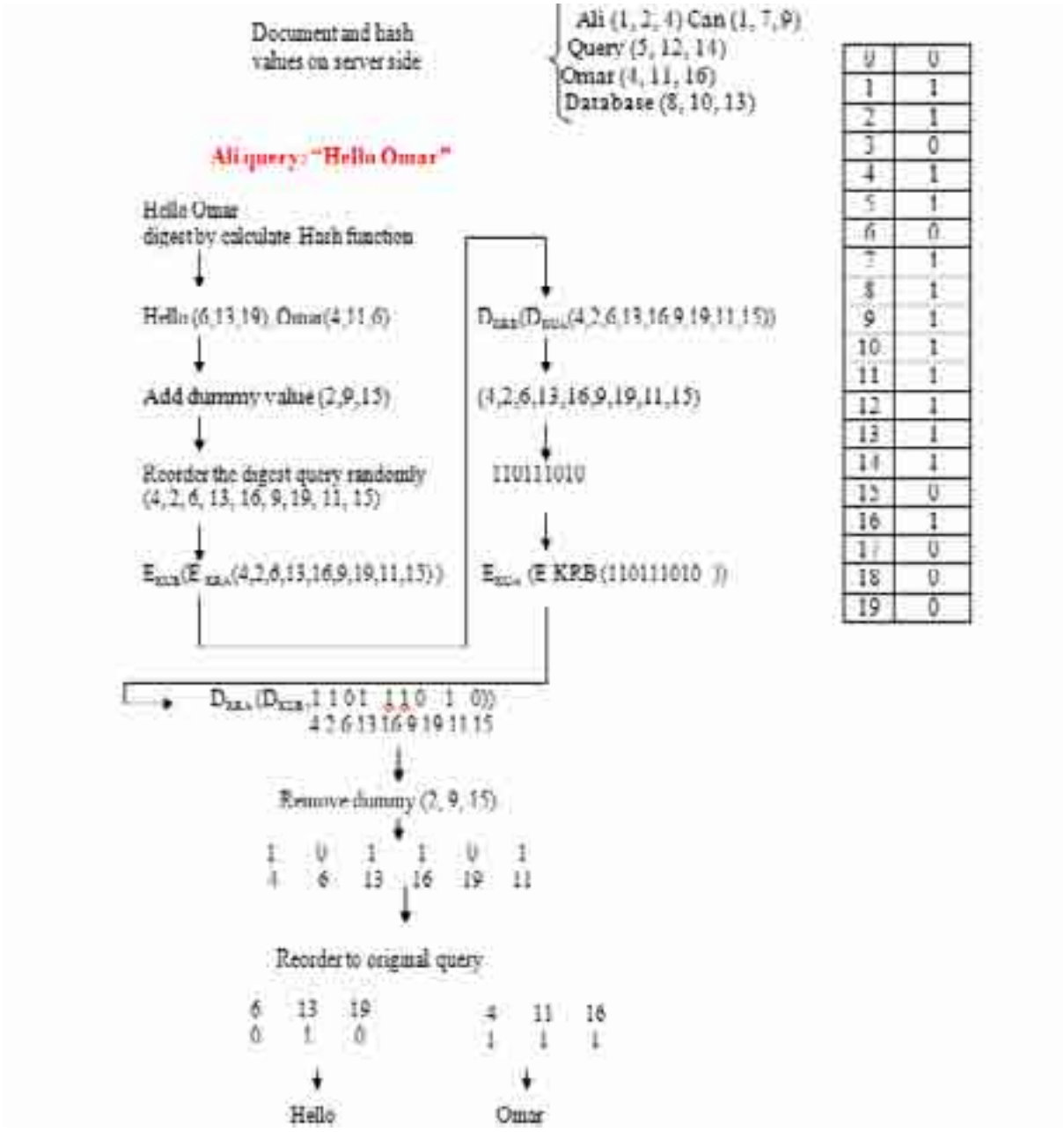


Figure 4: An Example on Suggested System.

Before implementing this project, there are many criteria which will be defined such as Bloom filters, Hash Function and RSA algorithm in order to complete the project and suggest an appropriate algorithm to be adopted in software field.

3.2.1 Bloom Filters

Bloom filters can be defined as a structure of compact data for the probabilistic representation to support the membership queries (i.e. queries that ask: "Is the first element X in set Y?"). This representation in compact form is the payoff to be used in allowing the small rate of false positives in the membership queries, which is, queries could be incorrectly recognized an element as member of the set such as X set.

Consider that there is a set $A = \{a_1, a_2, \ldots, a_n\}$ where n is number of elements. Bloom filter describes the membership information of A using a bit vector V of length m. For this, there will be k hash functions, $h_1, h_2, \ldots, h_k$. Figure 5 shows the bloom filter:

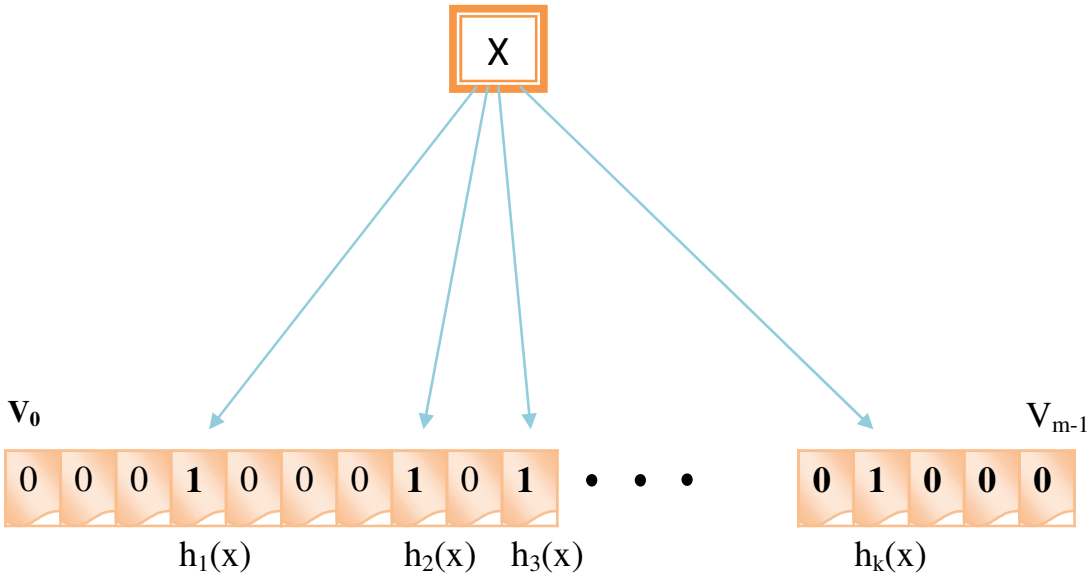


Figure 5: Bloom Filter.

The Implementation of Bloom Filters in this Project

Bloom filters represent the database of the server. The bloom filter will contain the data that the client will search in . So the bloom filter can be considered as table with three columns: *id*, *enc_id* and *enc_num*. *id* can be considered as pointer for each raw in the database and each data to be inserted by the administrator will take a raw in the table. So, if the administrator inserts data to the bloom filter (server's data base), the program will hash the inserted data first and then the result of hash function will be considered as encrypted ID. The program was designed to insert the hashed data in sequence. Figure 6 shows the bloom filter in this project as example. For example: if the administrator inserts "HEY" to database and the result of hash function was 94754, the program will insert new raw to database and set the value (*enc_num*) equal 1. See Figure 6.

	id	enc_id	enc_num
	190	94429	1
	191	94754	1

Figure 6: An Example of Bloom Filter.

Collision Rate

Collision rate can be shown in bloom filters and these criteria can be calculated based on the following equations, (Ripeanu and Iamnitchi, 2002).

Calculating the collision rate depends on the following equations

$$P_0 = \left(1 - \frac{1}{m}\right)^{kn} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Where:

n = number of primary Keys (the number of keys in this project are two).

m = size of document (number of words).

k = number of hash functions (number of used hash function are three).

And then calculate the error (P_{err}) which equal

$$P_{err=}(1-P_0)^k \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Database Dictionary Feature

Database will contain three columns: *id*, *enc_id* and *en_num*.

The following points will show the common SQL terms and phrases

- **CREATE DATABASE**: this order creates a new blank database by using SQL 2005.
- **CREATE TABLE**: creates a new table to store database within this table (this table is the bloom filter).
- **SELECT**: this command to call the database to extract that data that meet with the required data.
- **IDENTITY**: to count the ID within the table that was created (steps of counting can be specified when the bloom filter created.
- **PRIMARY KEY:** is a unique member which avoids the Repetition in one column (any table created should be have a primary key).
- **INSERT:** to add data for the table that was created, during this order one can determine the number of columns required to add data (such as add data for column 2).
- **DELETE:** to delete the data from the database when a new data will be inputted.
- **FROM:** specify the table that will be used to be (edit data, insert data, select data and delete data).
- **WHERE:** to specify the column that will be used in search process.

3.2.2 Hash Functions

A hash function can be defined as a reproducible method of turning various kinds of data into a small number (relatively) which it could serve as a digital "fingerprint" of the data used in this project. Cryptographic hash functions can be used for many purposes in the applications connected with information security, the hash function will be in this project used to build the bloom filters. This can be done by converting all words on a collection of documents into digital format (digital numbers).

Hash functions are designed to be fast and to yield few hash collisions in expected input domains. In hash tables and data processing, collisions inhibit the distinguishing of data, making records more costly to find.

A hash function must be deterministic. For example, if there are two hashes generated by the same hash function they will be different and then the two inputs were different in the same way.

Using a hash function will be useful to detect errors in transmission of the straightforward. The hash functions are computed for the data at the sender side and then the value of this hash is sent with the data. The hash functions are performed again the receiving end and if the hash values do not match, this means there is an error occurred at point during the transmission. This is process called a "redundancy check". Cryptographic grade of hash functions is used in common as integrity a check values to identify files and verify their integrity.

The Implement of Hash Function in this Project

Hash function had been used in both client and server in this project. Hash function had been used to provide the security by presenting all data in digital format neither than string format.

Hash function code is built function in VB.net software and the following code is applied in this project in order to hash the data in both client and server:

```
numHashedTxt1 = objHash.Hash1(arr(i))
If numHashedTxt1 < 0 Then
numHashedTxt1 = numHashedTxt1 * -1
numHashedTxt1 =
Strings.Right(Convert.ToString(numHashedTxt1),
Convert.ToString(numHashedTxt1).Length - 5)
Else
numHashedTxt1 =
Strings.Right(Convert.ToString(numHashedTxt1),
Convert.ToString(numHashedTxt1).Length - 5)

End If
```

The technique that is used in this project is converting the inserted text to numbers by built functions in the VB.net environment. The length of each hash value is five digits. So, if the result of two hash value exists the program will not be added to the data base in the server side, the program will not add a new line (new ID).

3.2.3 RSA Algorithm

This algorithm was developed in 1977 by three students Rivest, Shamir, and Adleman. This algorithm is the most commonly used authentication algorithm and encryption, the mathematical details of the RSA algorithm used in obtaining public and private keys. (Rivest et al, 1987) The algorithm will involve multiplying two large prime numbers and then through the additional operations deriving a set of two numbers which constitutes the public key and the private key. Both the public and the private keys will be needed to be used in the encryption and decryption but the private key is known by the owner that ever needs to know it. While using the RSA algorithm, the private key will never need to be sent across the Internet. Table 1 shows the functions of public and private key (Davis, 2003).

Table 1: Public and Private Keys Functions.

To do this	Use whose	Kind of key
Send an encrypted signature	Use the sender's	Private key
Send an encrypted message	Use the receiver's	Public key
Decrypt an encrypted signature (and authenticate the sender)	Use the sender's	Public key
Decrypt an encrypted message	Use the receiver's	Private key

The basic idea of this encryption is multiplying two prime numbers together. So, it is simple to perform a multiplication process for two numbers together and it will be very simple within computers. But it will be very difficult factoring the numbers.

Example:

If someone is asked to multiply two number together (85614 and 34987), he can use the calculator to multiply and find 2995377018. But factoring number (reverse problem) is much harder.

If the given number was, it's very difficult to factor the number and find they are 85614 and 34987. The computer can perform the factoring process quickly by trying the most of the possible combinations. The computer firstly, has to check something that is of the order of the size of the square-root of the number to be factored. So the square root of 2995377018 is 54730. Computer will not take a long time to try out 54730 possibilities but this for ten digits. So, if the result of multiplied two numbers together is 400 digits and the square-root will be 200 digits and it's needed for a very long time (lifetime of the universe enough for 18 digits). For example if a computer can perform a one million factorizations per second, in the universe's lifetime it could check 10^{24} possibilities. But for a 400 digit, there are 10^{200} possibilities.

RSA encryption works as finding two huge prime numbers, p and (100 or maybe 200 digits each). P and q have to be secret because they will be the private key. P and q will be multiplied (N=p*q).

The Cracking of RSA Algorithm

RSA algorithms varied based on the size of keys (number of bits). Many of RSA had been cracked such as 512bit and 768-bit RSA (Kleinjung, 2010).

In March 2, 2010, The Kaspersky Lab Security New Services published an article titled "RSA 2010: Experts Expect Several Ciphers to Be Cracked Soon". This article discussed that the cryptographers are expecting several cryptographic systems that

are in use today will be broken in the near future. Rivest (one of the inventors of the RSA algorithm) said in the Cryptographers Panel session at the RSA Conference, he expected

that RSA 1024 will be broken within a decade. Rivest mentioned that the people should start moving to 2048 soon (Fisher, 2010).

Table 2: Advantages and Disadvantages of Using RSA.

<table>
<tr><td rowspan="2">Advantages</td><td>1- The primary advantage is increased security and convenience while the private keys will never need to be revealed or transmitted to anyone.</td><td rowspan="2">Disadvantages</td><td rowspan="2">Speed is the most disadvantage of using RSA. Speed of RSA algorithm depends on many criteria:
1- The size of Public and Private Key.
2- Multiplication Techniques: Fast techniques such as FFT (Fast Fourier Transform) can perform the multiplication process fast.</td></tr>
<tr><td>2- RSA systems can provide a digital signature which means the message cannot be repudiated.</td></tr>
</table>

Table 3: Bloom Filters VS RSA Applications

Bloom Filters	**RSA**
Counting filters: it's used to provide a way to implement the *delete* operation on a Bloom filter without involving any recreating process for the filter afresh.	Send an encrypted signature
Data synchronization: Bloom filters that could associate a value with each element that had been inserted, implementing an associative array	Send an encrypted message
Bloomier filters: used in the association a value in the each element which had been inserted.	Decrypt an encrypted signature (and authenticate the sender)
Compact approximators: it is used in the lattice-based of Bloom filters in general.	Decrypt an encrypted message
Stable Bloom Filters: used as a variant of Bloom filters in the streaming data. The main idea is	

giving the ability to the bloom continuously information to make room for the recent elements when there is no way to store the entire history of a stream.	

- **Dummy Values**

The point of adding dummy values in the client's query is making the query ambiguous and the server has no idea what the client are looking for.

For example if the client query the server by "hello", the program will have this word and the server can do this too. So, where is the privacy here??!! .

Dummy values plays a very important role in this project. The client will add a dummy value in specific order to be removed correctly after the server sends the results. The following example shows how the dummy value will be added to the query.

Client query: "Hello"

Hello
digest by calculate Hash function

↓

Hello (6, 13,19),

↓

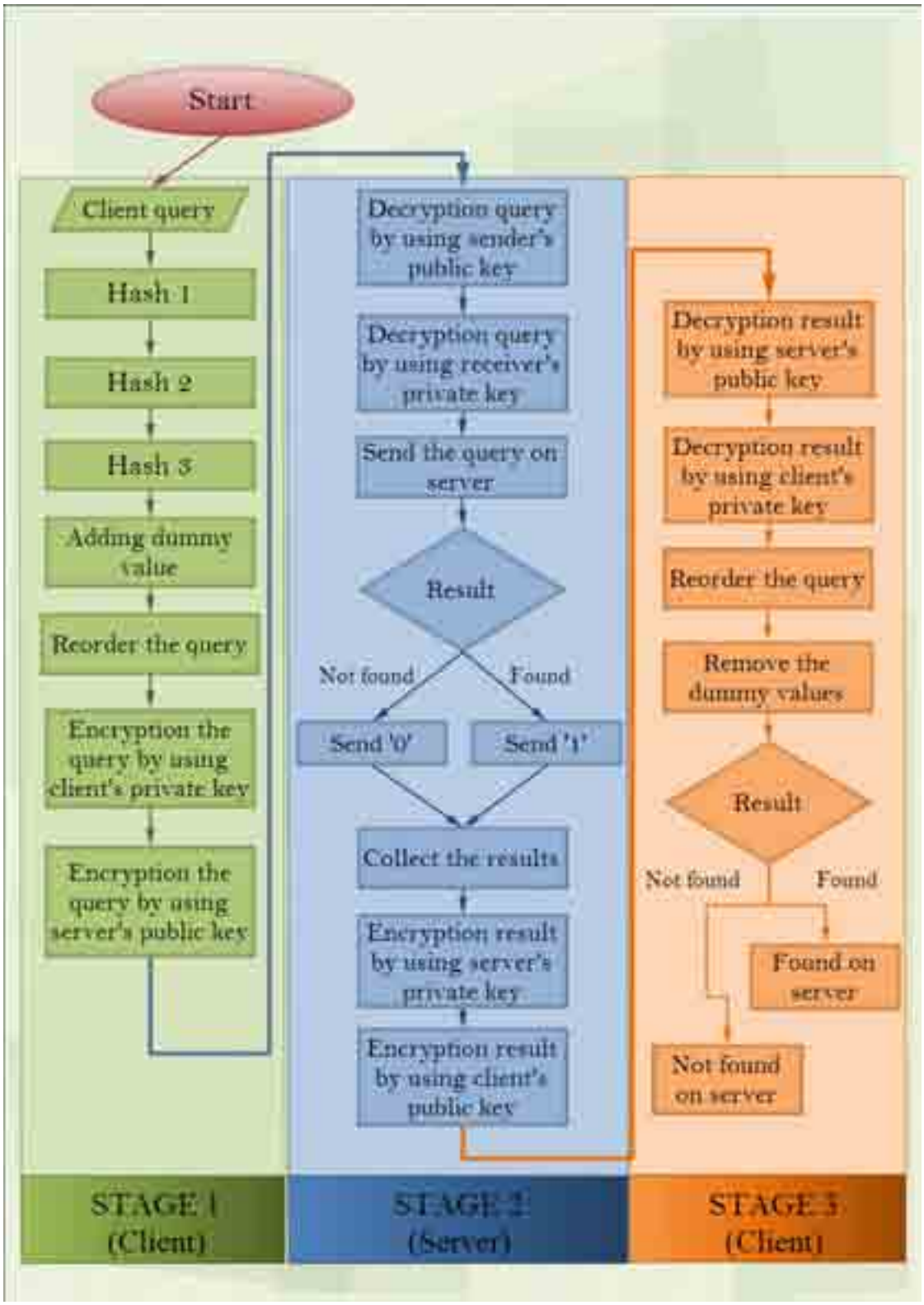
Add dummy value (2, 9, 15)

↓

Reorder the digest query randomly
(2, 6, 13, 9, 19, 15)

From the example it can be seen the query became two words rather than one, so the server will think that the client searches for two words not one.

3.3 Flowchart shows Bellovin and Cheswick's Algorithm (After Development).



3.4 Design and Implementation Screens

This project will require using two types of software to be implemented: VB.net and SQL server 2005. Vb.net will be used to program the client and server screens and SQL server will be used to create the server's database.

The program has two main sides: server and client. See Figure 7.

Figure 7: Client and Server Options.

3.4.1 Server Screen

The server page has many functions and the data base can be built through it. At the beginning the server will be empty until the server uploads file in text format or type the text that will present the server. This page contains a textbox and three buttons. Textbox allows the user to insert a text to the server's database. Figure 8 shows the server's screen.

Save Button: this button will add the test to the server's database.

Clear Database: this button will clear all the data in the database.

Close Button: this button will close the program.

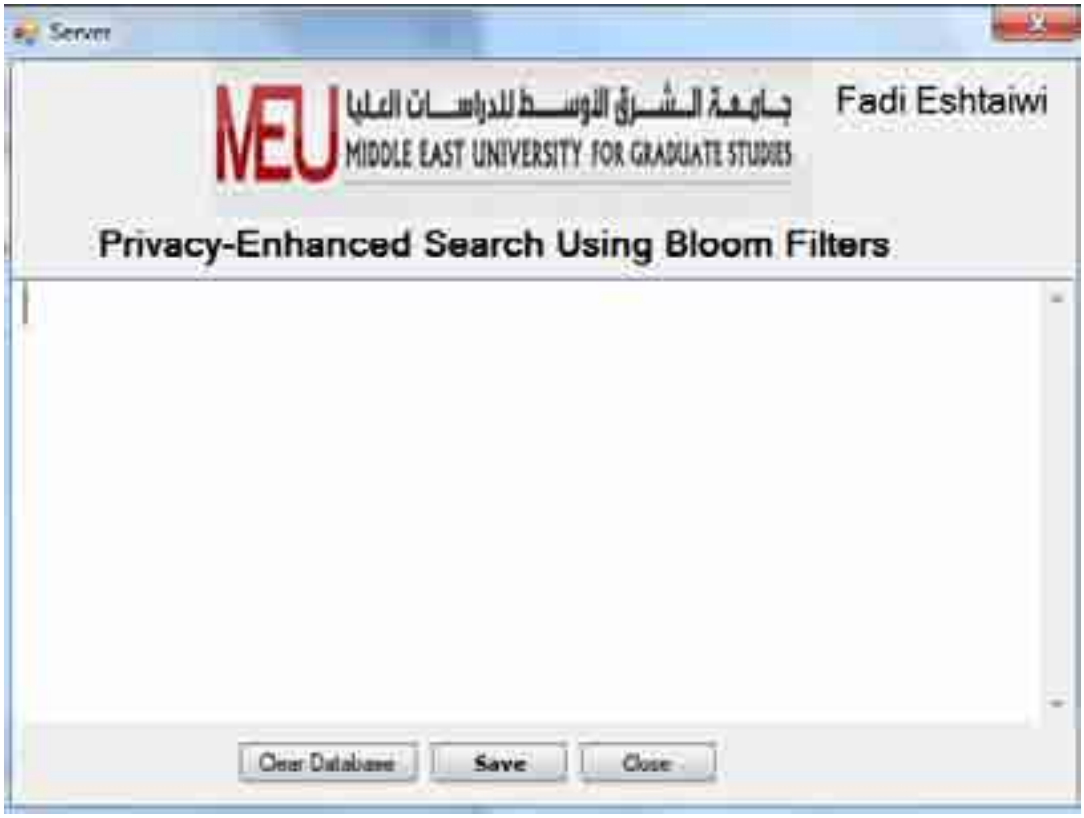


Figure 8: Server Screen.

If the administrator (person who is responsible to add data to the database) clicked the save button and the textbox is empty, the program will present a message box that tells the administrator there is no data can be added to the database. See Figure 9.

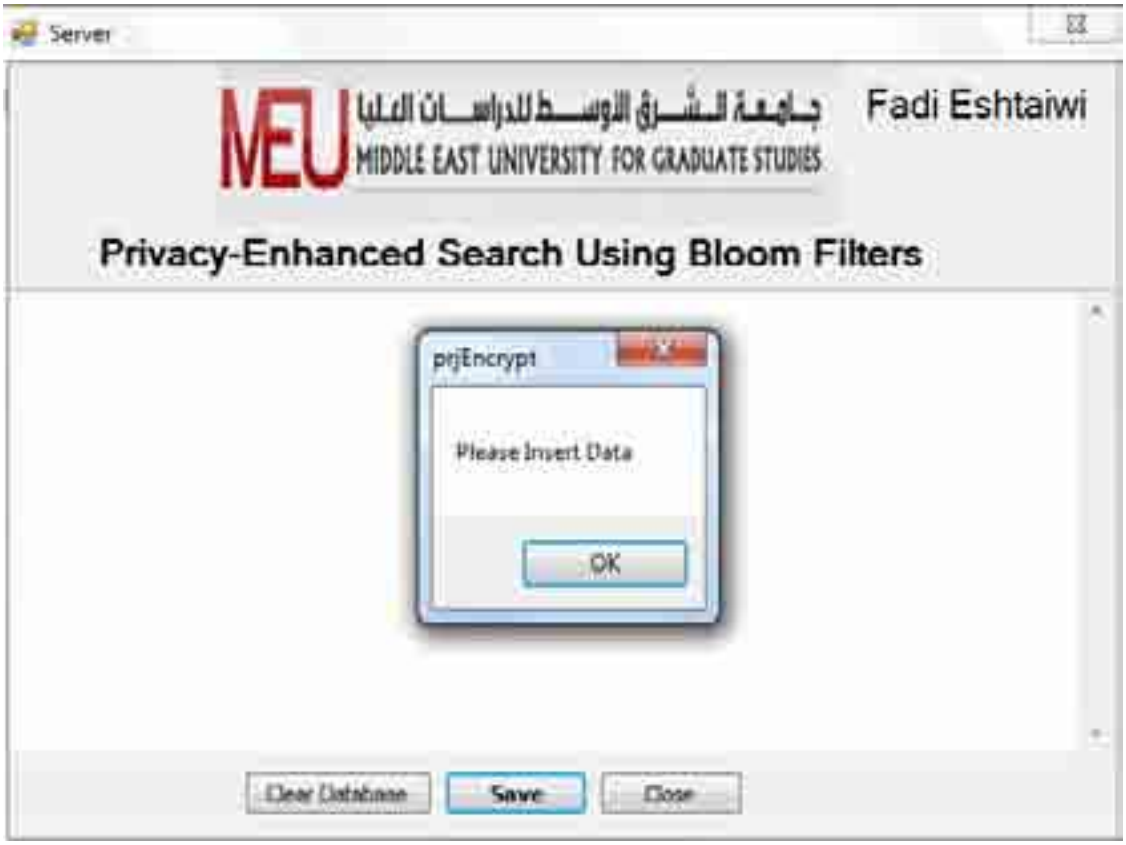


Figure 9: Insert Data Message Box.

3.4.2 Client Design

The client can query the data base through writing the words in TextBox and clicks Search Button. The program will perform six steps beginning with hashing the query, add dummy values, and reorder the dummy values plus the hash values of the query and then two encryption processes with different keys (Public and Private). Figure 10 shows the client interface. It also contains Result box. This box contains two rows: Result and status. It will show the result if the query is found or not.

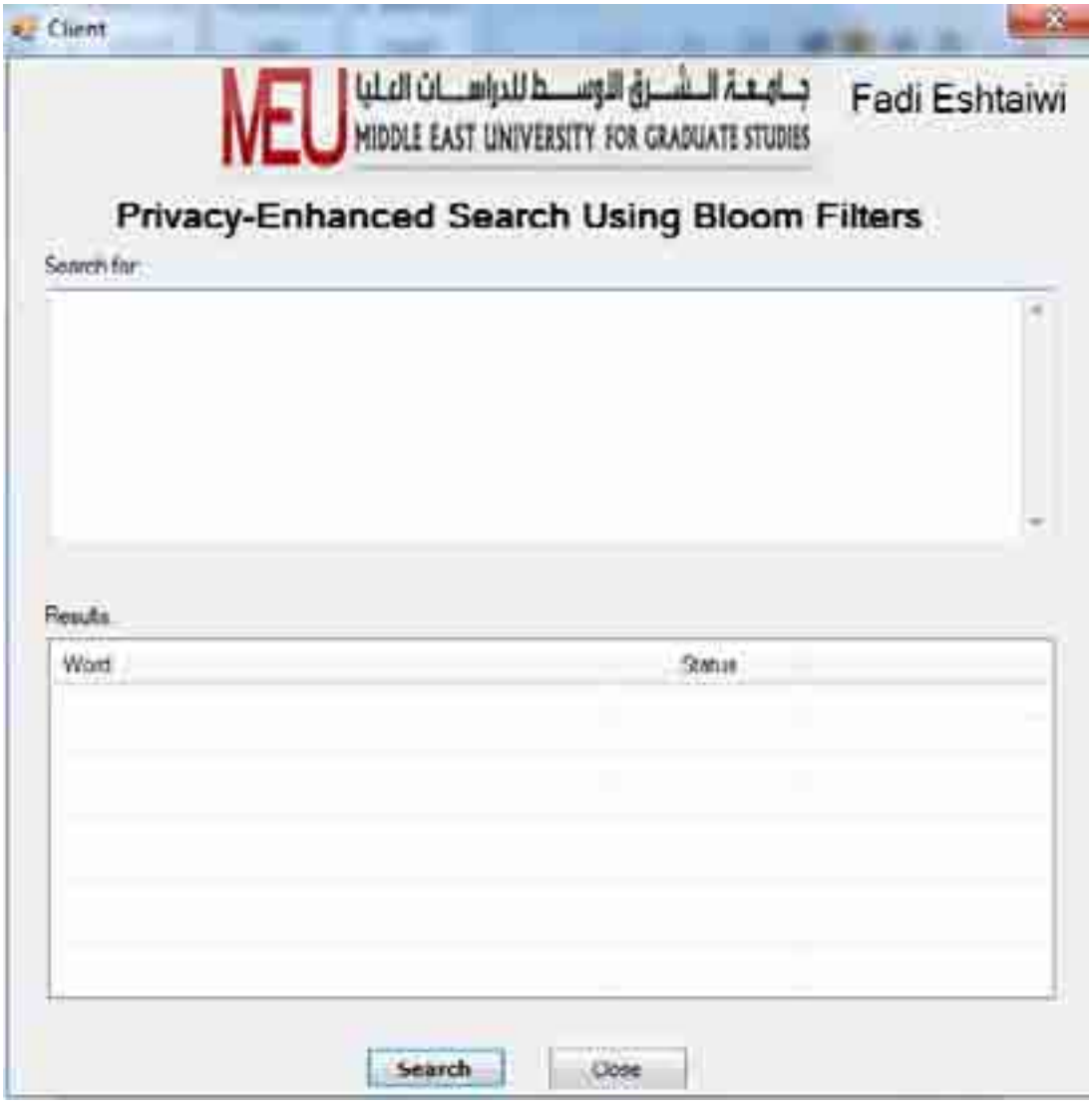


Figure 10: Client's Interface.

If the client queries the server's database, the program will show message box telling the client there is no query which will perform on the server's database. See Figure 11.

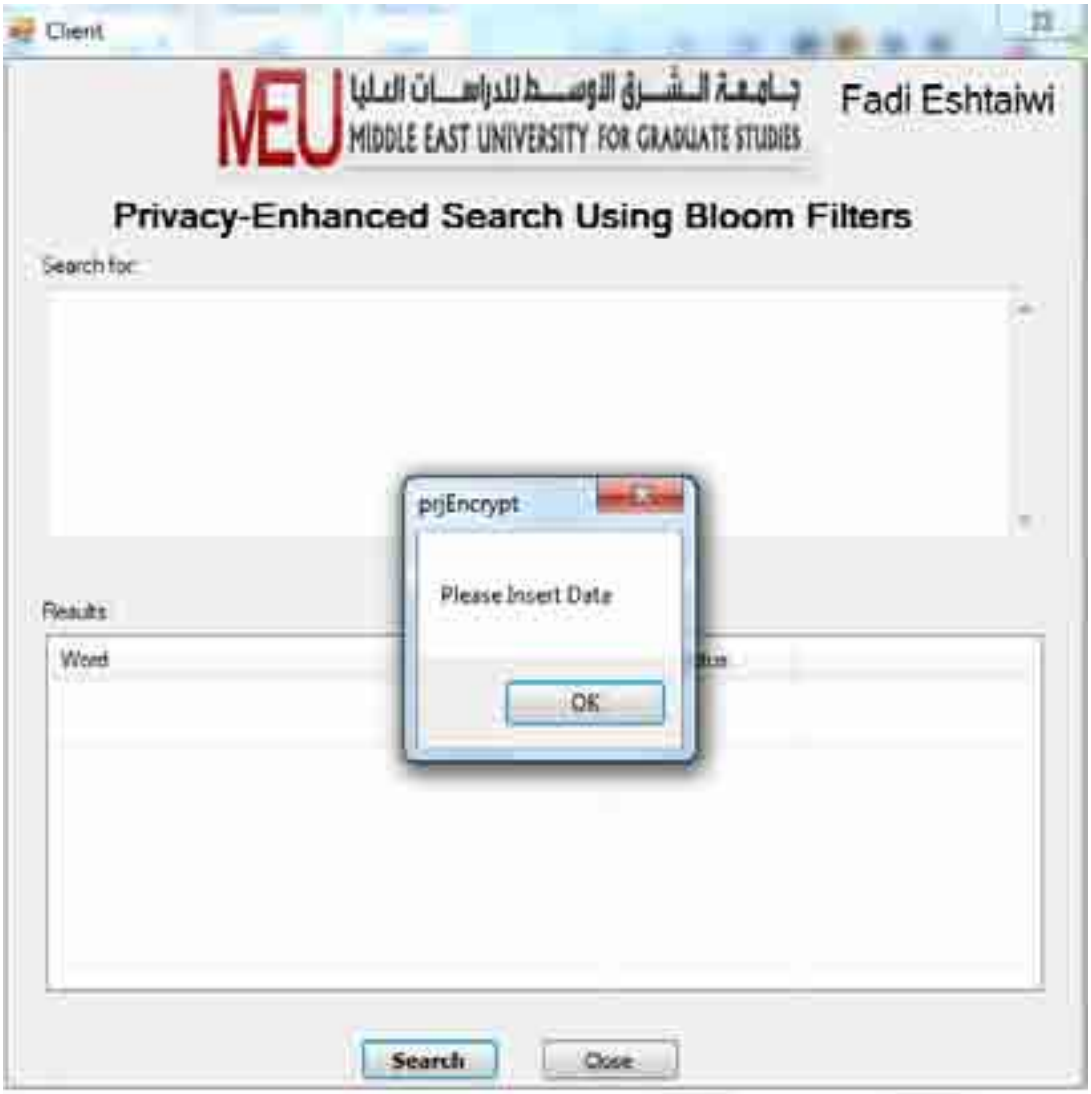


Figure 11: Message Box tells the Clients there is no Query.

Chapter 4

Test and Evaluations of Application and Examples

The program will be tested in order to check if the program is correctly designed and programmed. The test process will require creating database in order to allow the client query the server's database for specific words. While the client queries the server's database, it must type the word that is looking for in the textbox and then click search.

Search button presents the whole program. If the client clicked in the search button the program will act like the following:

1- Hashing the client query.
2- Add dummy values.
3- Reorder the query.
4- Encryption the query (public key)
5- Encryption the query (private key).

The test process will include each main process in the project.

4.1 Test the Hash Function Process

At the beginning, the program will hash the query. Figure 12 shows how the program will hash the "Fadi" word for one time. The type of the hash function that is used is *Object.GetHash()* and this is a built function in Microsoft visual basic.net (VB.net).

Figure 12: The First Hash Value of Fadi word.

It also will hash “Fadi” for second time and the result shown in figure 13.

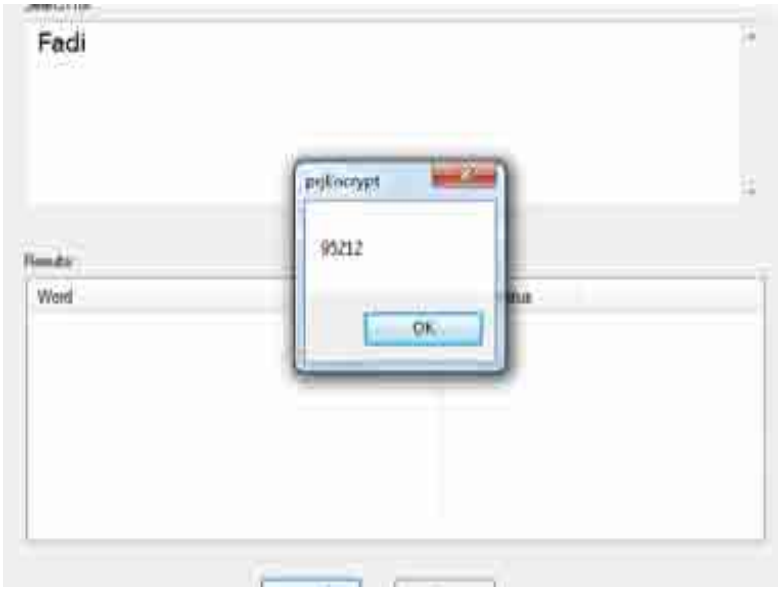


Figure 13: The Second Hash Value of Fadi word.

The program will hash “Fadi” for the third time as shown in the Figure 14.

Figure 14: The Third Hash Value of Fadi word.

Table 4 shows the result of hash functions for three times with “Fadi” word.

Hash functions	**Results**
1st Hash function of “Fadi”	95584
2nd Hash process of “Fadi”	95212
3rd Hash process of “Fadi”	94884

Table 4: Hash Function Results.

4.2 RSA Test (public and private).

Before the client sends the query to the server, this query must be encrypted by two different keys (public and private). The input of RSA function will be as the following:

Hash values + dummy values= input of RSA function. RSA function consists of formula which applies many procedures to get the result. Figure 15 shows the result of encrypting process of “Fadi” by using server’s public key.

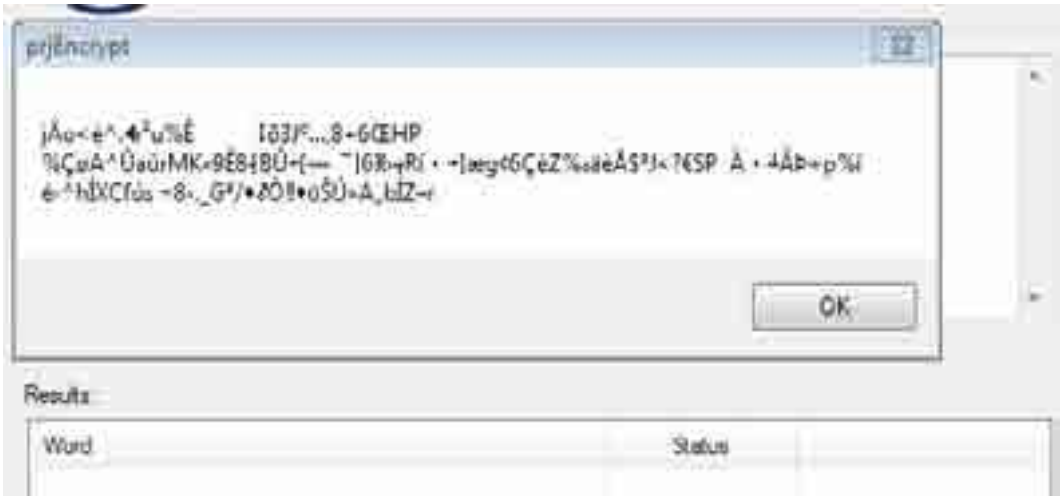


Figure 15: Result of Encrypting "Fadi" word by Using Sender's Public Key.

Another encryption must be tested in order to ensure that the query is encrypted two times by two different keys. Figure 16 shows the result of encryption the query for second time by using client's private key.

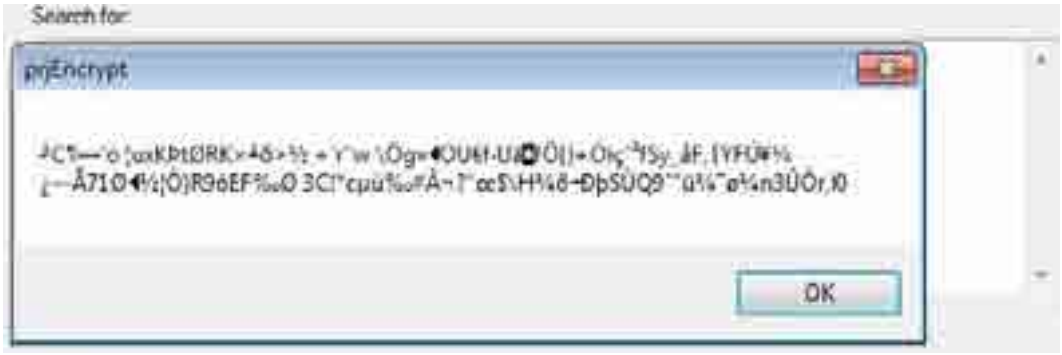


Figure 16: The Result of Encryption the Query for Second Time.

4.3 Test of the Whole Program

The whole program will be tested in order to check the program meets the proposed objectives and working correctly. Before starting with testing the program, it's needed to create the server's database. This test will be implemented by creating the server's database by inserting data that represents the data which the client will

query. The database contains “Fadi” but it doesn’t contain “Eshtaiwi”. Figure 17 shows the server’s database.

id	enc_id	enc_num
767	95584	1
768	95212	1
769	94884	1
770	51762	1
771	59735	1
772	59322	1

Figure 17: Server’s Database.

The client will query the server’s database by searching in the database on “Fadi” word and the results were perfect. Figure 18 and 19 show the result of query the server’s database by “Fadi” word.

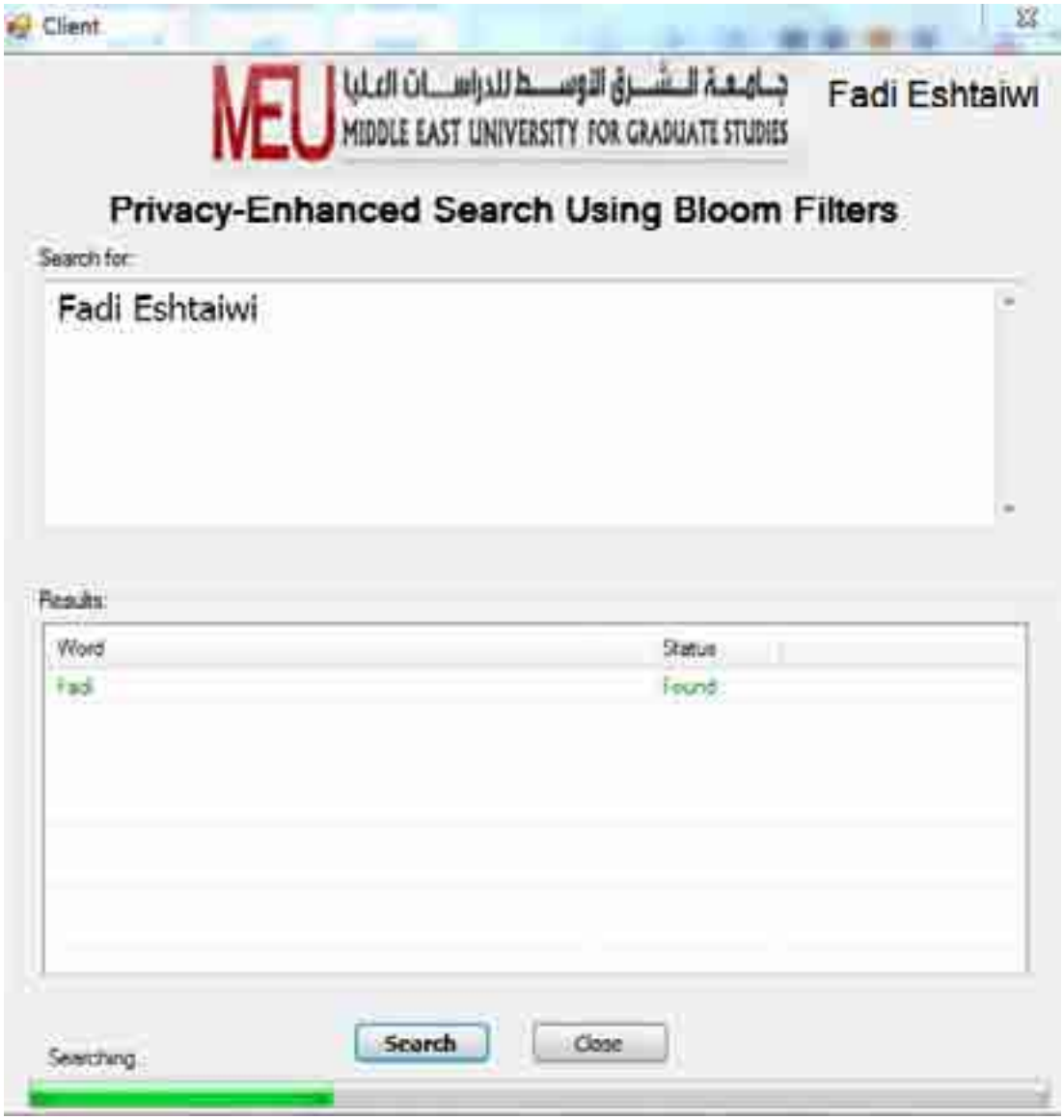


Figure 18: Searching on the Server's Database about "Fadi Eshtaiwi" word.

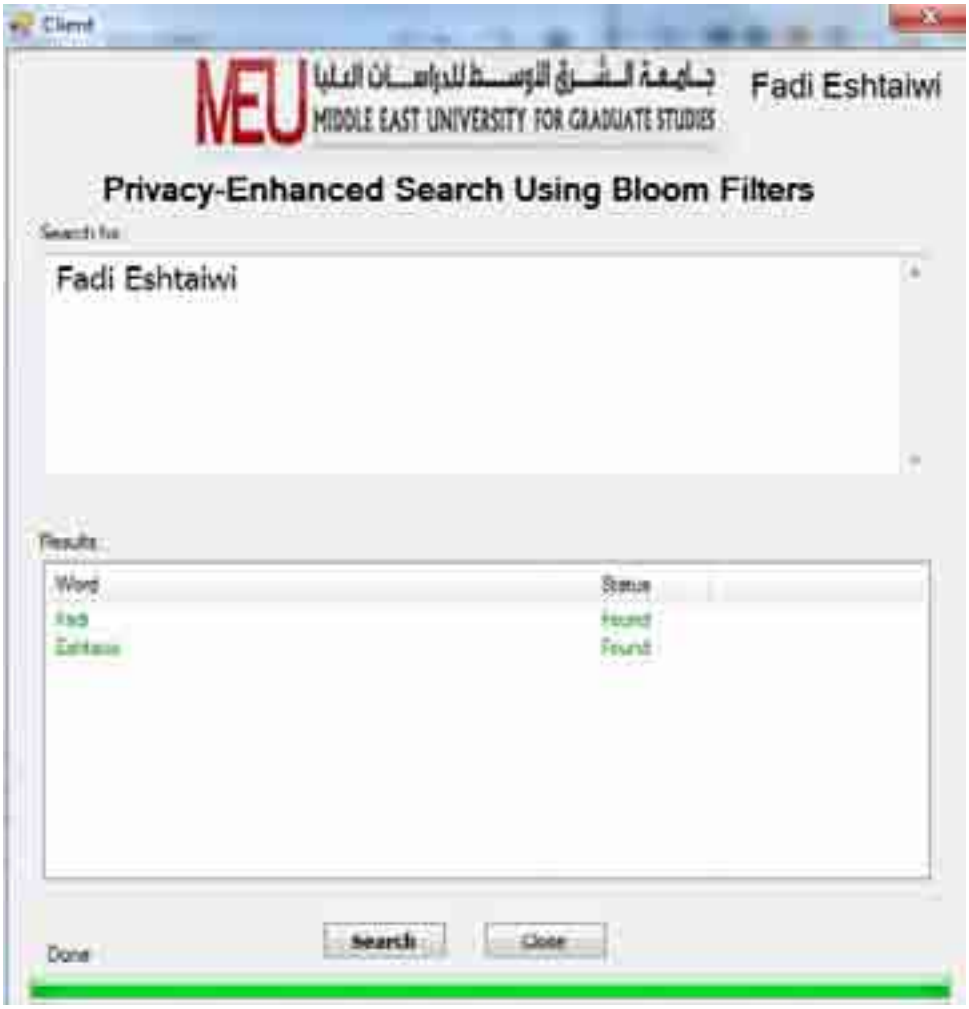


Figure 19: The Result of Query Server's Database on "Fadi Eshtaiwi".

The client also can query the server's database about "Fadi Eshtaiwi", the server's database just contains "Fadi Eshtaiwi" but it doesn't contain "MIDDEL". Let us see how the program will act. Figure 20 shows the result of "Fadi Eshtaiwi MIDDEL" query.

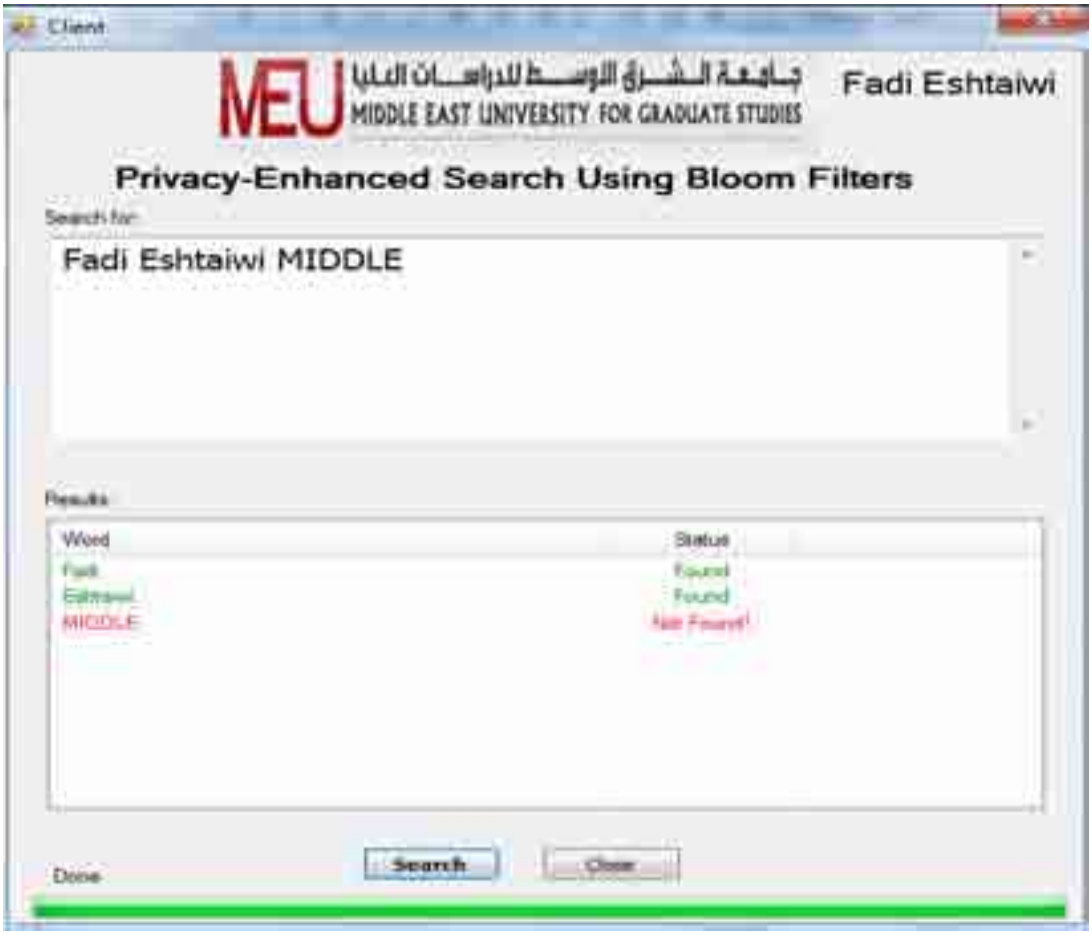


Figure 20: The Result of "Fadi Eshtaiwi MIDDLE" Query.

Inquiry by words	**Results**
Fadi	Found
Eshtaiwi	Found
MIDDLE	Not Found

Table 5 : Results of inquiry three words of the database.

4.4 Critical Evaluation

This type of evaluation is concerned with the ability to critically evaluate the information. In this project, PROMPT approach will be used to critical evaluation of information which refers to Provenance, Relevance, Objectivity, Method, Presentation, and Timeliness.

1- **Provenance**: this project depends on many ideas in the enhancement of privacy field. Many researchers studied how to enhance the privacy by using bloom filters and hash functions. Bellovin and Cheswick (2005) enhanced the privacy by using bloom filter and hash functions. This project developed this idea by many features starting with the dummy values and the reordering process of query. There are many sources used in this project such as RSA code.

2- **Relevance:** the quality of information that is used in this project is high level quality because all information was gathered from scientific papers and conferences. It also include the date of publishing thee papers .All requirements of this project are clearly identified. In addition, all information is related to the subject of this project (enhancement the privacy for search process) and the aim that is trying to do (remove third party).

3- **Objectivity:** this project met its objectives and they are followed correctly in order to meet the main aim. All information doesn't conflict with each other and they are tending to implement all the requirements.

4- **Methods:** the methods applied in this project are "bottom-up approach", where the project is divided into sub system and each one will do specific function such as the hash function that will hash the query in both server and client. The method was implemented as it was deigned and it given the supposed results.

5- **Presentation:** all information in this project was presented in clear structure based on the project-handbook. Presentation includes font color, font size, font type, images and diagrams.

6- Timeliness: it is concerned with the date of producing or publishing the data in this project and it depends on the information need.

3.5 Critical Evaluation of This Project:

This project met the proposed aim which is removing the third part and enhances the privacy in search process. There are many activities that were followed in order to achieve the main aim. They include many processes beginning with gathering data and revision of the literature review in this field. All requirements are fully defined and then designed to be implemented by using Microsoft VB.bet and SQL server. The result of each process was tested to ensure that every function was built correctly and gives the specific outputs. The entire system was tested with many quires and the results were correctly.

Chapter 5

Results

This project is designed to implement a new suggested system in order to remove the third party by using two programs: VB.net and SQL server 2005. In this project, client could query server's database and search for anything without knowing anything about what is the client is looking for. Hash function is used to represent all the data in both client and server. The client queries the database for "Fadi Eshtaiwi" and the results were found for "Fadi" and "Eshtaiwi" and not found for "MIDDEL".

Microsoft VB.net has many built functions that have been used in this project. Object.GetHash is one of these built functions.It was very a useful function because it was used in both client and server. VB.bet also has the ability to communicate with the SQL server 2005 in order to build the server's database. Con.ConnectionString is used to make this connection.Classes in the VB.net were used to build the common functions that are used in both client and server such as the encryption, decryption and hash functions. Calculating the spent time of query for Fadi is done in three ways and see how the size of document has effects on the time.The following table shows the query's spent time. The results of spent time for querying multiple size of documents as shown in Table 6.

Table 6: Query Time.

Search in	**Spent time**
200 words	00:00:26
400 words	00:01:36
1000 words	00:02:50

Searching for "Fadi" in document with size 200 required for 26 milliseconds to find it. While searching for the same word in document with size 1000 words required for two seconds and 50 milliseconds. So, while the document is increasing, the required time for query will increase too much. Adding to the server's database too much of data may occur down in the server. So, the program is designed to prevent the copy and paste from

any document into the server in order to enhance the security. The collision rate can be shown in the bloom filter but in this project there were no collisions because there were three hash functions which are used in this project. So, the database is built in a way to prevent to duplicate the hash value. So, it's impossible to find two words that have the same hash values in the database.

The results of this project will be as result of querying the server database and the query is about words generated by the client. This project aimed to enhance the privacy of searching process between the two parties by using bloom filters. The client's query is " Fadi Eshtaiwi ", applied on this project in order to check that the system working as it is designed. The database has the following data: search, using, bloom, filter and "Fadi". After decrypting the query at the server's side, the query will consist of:

1- Hash values of the query. Figure 21, 22 and 23 show the result of hashing "Fadi". Figure 24, 25 and 26 show the result of hashing "Eshtaiwi".

Figure 21: The Result of First Hash Functions of Fadi.

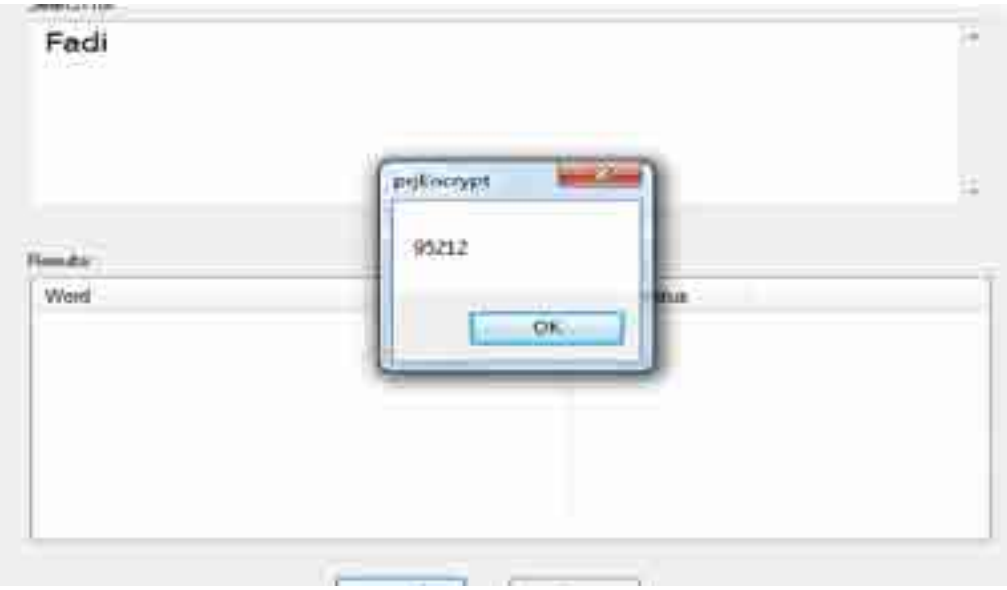


Figure 22: The Result of Second Hash Functions of Fadi.

Figure 23: The Result of Third Hash Functions of Fadi.

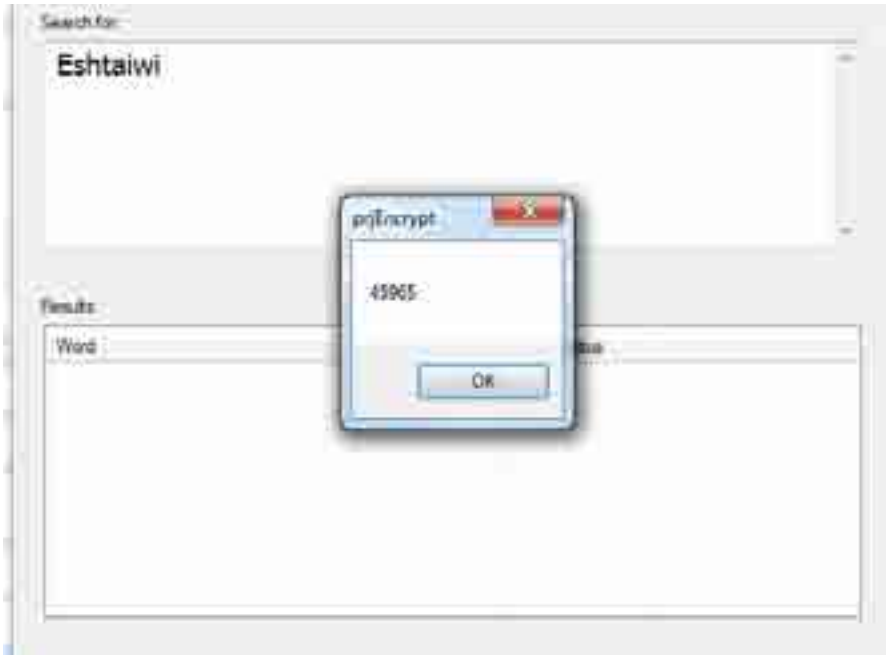


Figure 24:The Result of First Hash Functions of Eshtaiwi.

Figure 25: The Result of Second Hash Functions of Eshtaiwi.

Figure 26: The Result of Third Hash Functions of Eshtaiwi.

Table 7 shows the result of hash functions for three times with "Fadi" and "Eshtaiwi" word.

Table 7: Hash Function Results.

Hash functions	**Results**
1st Hash function of "Fadi"	95584
2nd Hash process of "Fadi"	95212
3rd Hash process of "Fadi"	94884
1st Hash function of "Eshtaiwi"	45965
2nd Hash process of "Eshtaiwi"	45135
3rd Hash process of "Eshtaiwi"	44573

2- Dummy Values and Reorder the Query

The query will contain also the dummy values and then the program will reorder the query. So the result if the query that will enter the server's database and search on it is shown in the figure 27 and 28. Figure 27 shows the dummy values added to the query after the reordering process of "Fadi" and Figure 28 shows the dummy values added to the query after the reordering process of "Eshtaiwi".

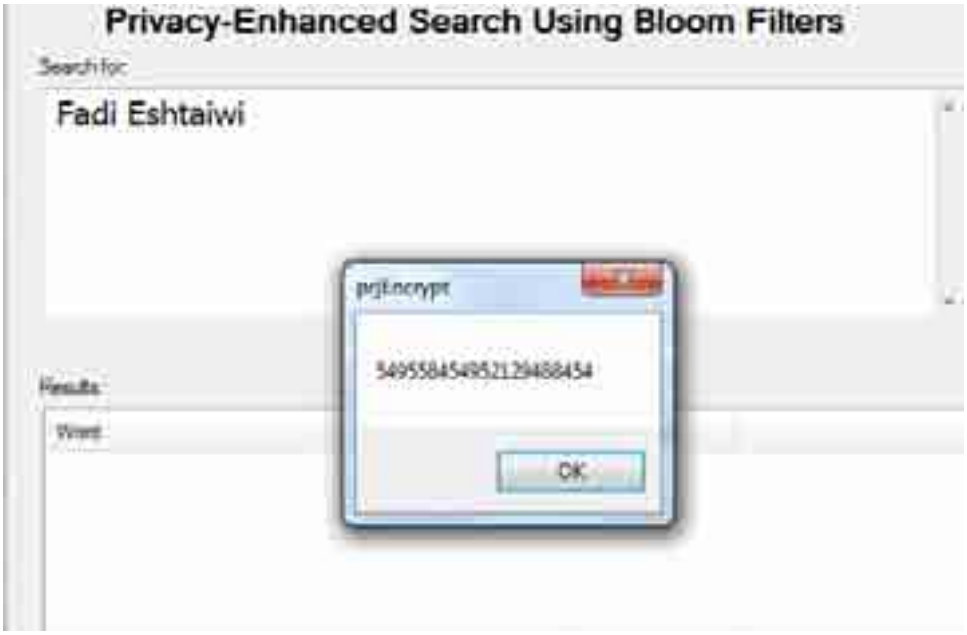


Figure 27: Hash Vales and Dummy Values after the Reorder Process of “Fadi” Query.

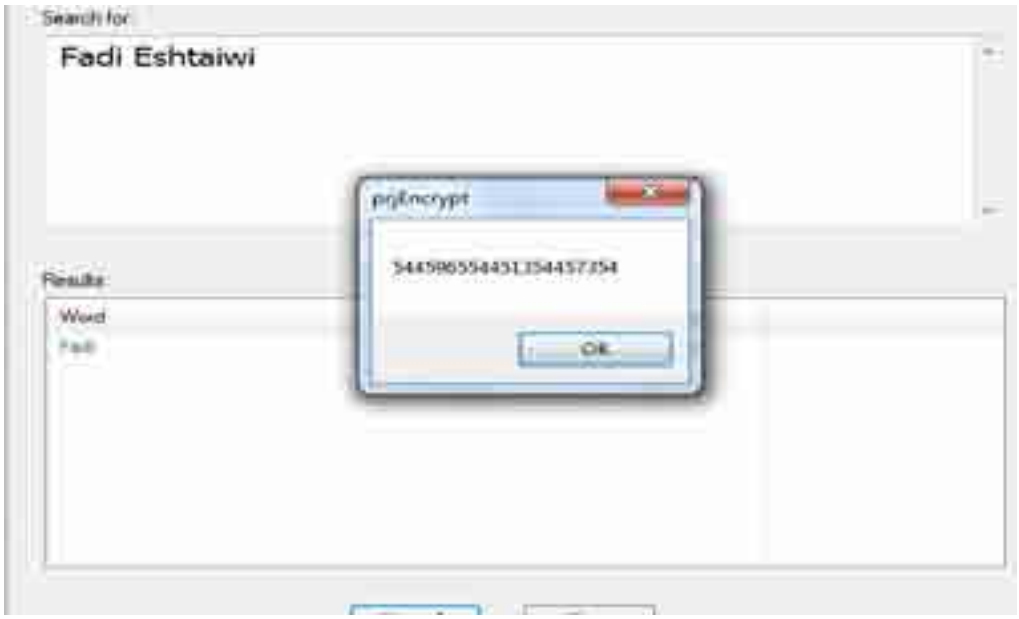


Figure 28: Hash Vales and Dummy Values after the Reorder Process of “Eshtaiwi” Query.

Table 8 shows the results of reordering process for query "Fadi" and "Eshtaiwi" added to dummy values.

Table 8: Reordering Results.

Reordered hash function with dummy values	Result
"Fadi" + Dummy values	549558454952129488454
"Eshtaiwi" + Dummy values	544596654451354457354

The server will not know what the client is searching about because the dummy values and reorder process will make the query ambiguous. Sever will perform the search process by dividing the query into five digits and return 1 if exists 1 or 0 if doesn't exist.

The result of querying "Fadi Eshtaiwi Middel" shows in Figure 29.

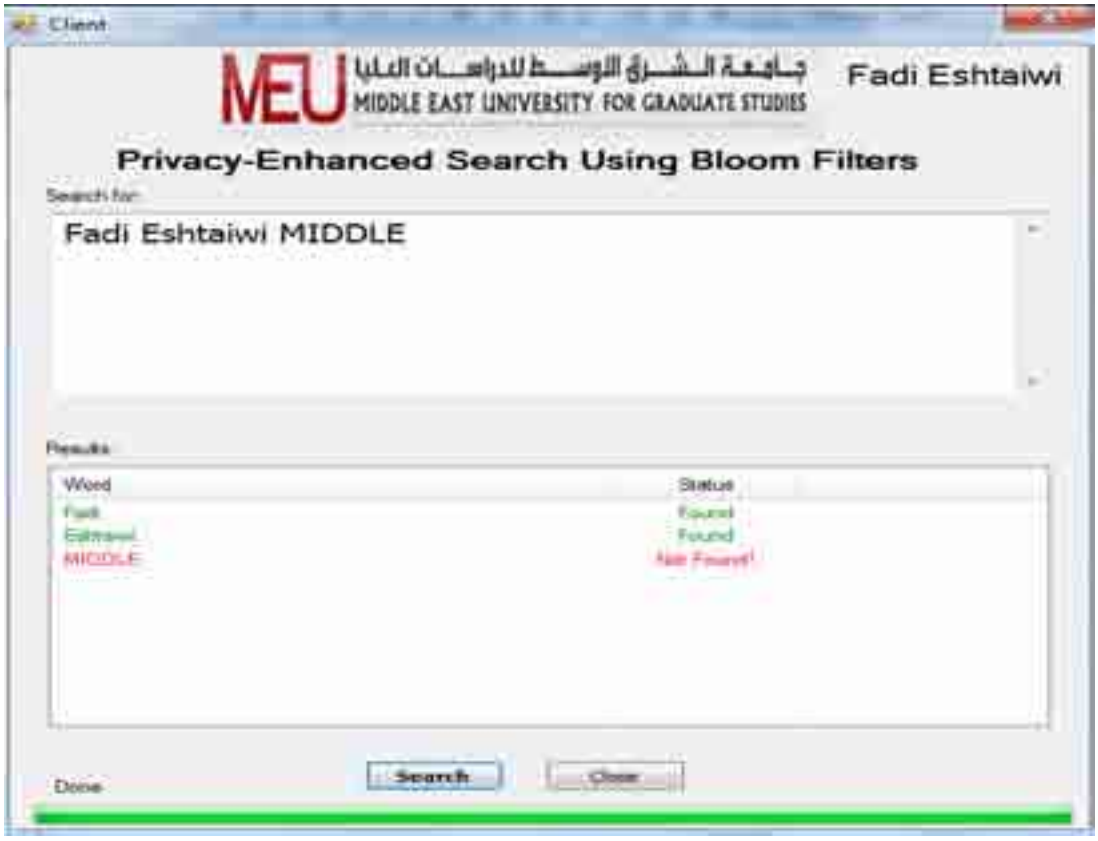


Figure 29: The Result of Querying Fadi Eshtaiwi Middle.

Collision Rate

Collision rate can be calculated in this project based on equation (1) and (2)

$$P_0 = (1 - \frac{1}{m})^{kn} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

$$P_{err} = (1 - P_0)^{k} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

Where:

n= number of primary Keys (the number of keys in this project are two)

m= size of document (number of words)

k= number of hash functions (number of used hash function are three)

And then calculate the error (P_{err}) which equal

The collision rate will be calculated for m=10 words as follows:

$$P_0 = (1 - \frac{1}{10})^{3*2} = 0.531441$$

$$P_{err} = (1 - 0.531441)^{3} = 0.0169$$

Equation 2 and 3 are implemented in the project and the results are shown in figures 30, 31 and 32.

Figure 30 shows the collision rate at the size equal 10 words.

Figure 30: Collision Rate at 10 words.

Figure 31 shows the collision rate at the size equal 30 words.

Figure 31: Collision Rate at 30 words.

Figure 32 shows the collision rate at the size equal 40 words.

Figure 32: Collision Rate at 40 words.

Marks of enhancing the privacy in this project .

First Mark:

Adding the dummy values:

This feature will enhance the privacy by making the server unable to know what the client is looking up. This will be useful if there are organizations in their databases.

Second Mark:

Reordering the process:

After hashing the query and adding dummy values, the query will be reordered in order to make the query is meaningless when it arrives to the server's database.

The previous two marks are shown in Figure33.

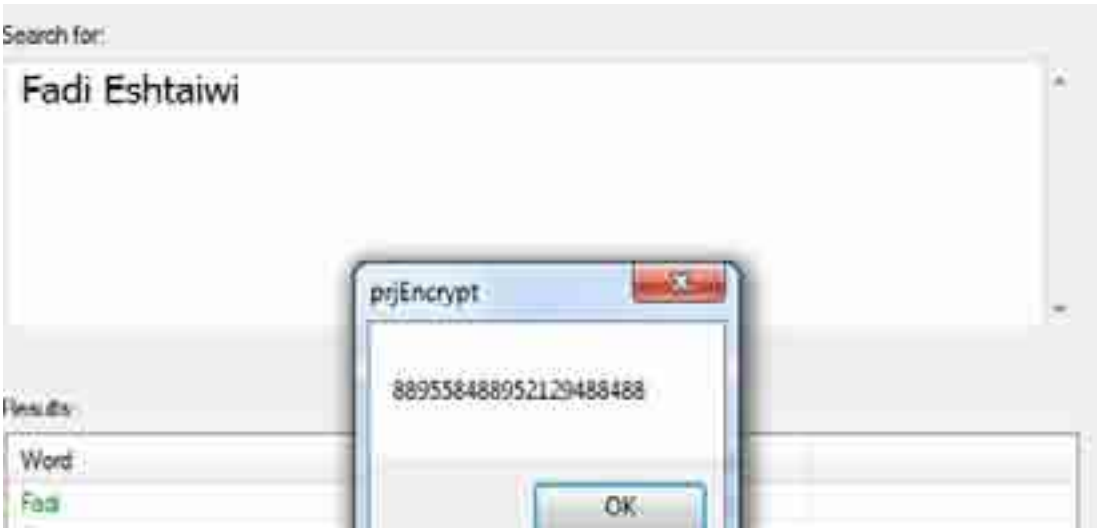


Figure 33: Marks of Enhancing the Privacy in this Project.

5.2 Code Discussion

This is the first code form will appear to the user in this project. There are three buttons in this interface. Client button was programmed by Button1_Click object to link the user to the client form.

Server button was programmed by Button2_Click object to link the user to the server form. `Public Class frmMain`

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim frm As New frmClient
        frm.Show()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Dim frm As New frmServer
        frm.Show()
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        Application.Exit()
    End Sub
End Class
```

Figure 34: Interface Buttons.

The function of Close button (Button3_Click) will exit from the application and stop for run process.

Client Code

The main idea to write an appropriate code to program the client is starting with design the client's interface.

Figure 35: Search Button.

In this interface, search button (btnSearch_Click) and when the client clicks the search button many procedures will be executed.

```
Public Class frmClient
    Dim objHash As New clsHash
    Dim dumValues(3) As Integer

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

        Me.Close()
    End Sub

    Private Sub btnSearch_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnSearch.Click
```

The following code will check if the client clicked Search button without inserting any word. If the user inserts data, the following code will be executed. Many variables and classes are identified to be used in many functions in the client side.

```
If txtClient.Text <> "" Then
            On Error Resume Next
            Dim cipherX As String
            Dim objCheck As New checkEncData
            Dim resp As String
            Dim returno As String
            Dim result As Boolean
```

pBar is identified for the searching time process, the default value of this object is zero. This bar will start loading when the user clicks on search button.


```
pBar.Value = 0

            If txtClient.Text <> String.Empty Then
```

```
Dim line As String = txtClient.Text
Dim arr() As String
arr = Split(line, " ")
Dim arrLength As Integer = CInt(arr.Length)
Dim numHashedTxt1(arrLength) As Integer
Dim numHashedTxt2(arrLength) As Integer
Dim numHashedTxt3(arrLength) As Integer

stlbl.Visible = True
pBar.Visible = True
stlbl.Text = "Searching..."
pBar.Minimum = 0
pBar.Maximum = 100

For i = 0 To (CInt(arr.Length - 1))

    pBar.Value = pBar.Value + 10 * (i + 1)
    '*******************************************
    '** Get Hash1
    '*******************************************
```

There is an array; the inserted data will be set in this array as string. The array will be the input of the hash procedure. There are three hash functions in clsHash.vb class (clsHash.vb will be explained later). Each word will pass through these hash functions. Each hash function various than the other one (each hash hash function has various code)

```
numHashedTxt1(i) = objHash.Hash1(arr(i))
```

Through execution of this code, there was a problem with the first digit of the hash value. This problem is shown when the first digit is equal zero and the program will drop it. So, this problem is solved by replacing the zero to 9 if the first digit is equal 0.

```
numHashedTxt1(i) = CInt(CStr(numHashedTxt1(i)).Replace("0", "9"))
```

The result of the hash function could be negative and that's not logical. The following code checks the result of hash function and if the hash value is less than zero, the hash value will be multiplied by -1.

```
                    If numHashedTxt1(i) < 0 Then
  numHashedTxt1(i) = numHashedTxt1(i) * -1
                        If numHashedTxt1(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt1(i).ToString.Length - 1
                                numHashedTxt1(i) =
CInt(CStr(numHashedTxt1(i)) & "1")
                            Next
                        End If

                        numHashedTxt1(i) =
Strings.Right(Convert.ToString(numHashedTxt1(i)),
Convert.ToString(numHashedTxt1(i)).Length -
(Convert.ToString(numHashedTxt1(i)).Length - 5))
                    Else
                        If numHashedTxt1(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt1(i).ToString.Length - 1
                                numHashedTxt1(i) =
CInt(CStr(numHashedTxt1(i)) & "1")
                            Next
                        End If
                        numHashedTxt1(i) =
Strings.Right(Convert.ToString(numHashedTxt1(i)),
Convert.ToString(numHashedTxt1(i)).Length -
(Convert.ToString(numHashedTxt1(i)).Length - 5))

                    End If

                    '*********************************************
                    '** Get Hash2
                    '*********************************************
                    numHashedTxt2(i) = objHash.Hash2(arr(i))
                    numHashedTxt2(i) =
CInt(CStr(numHashedTxt2(i)).Replace("0", "9"))
                    If numHashedTxt2(i) < 0 Then
                        numHashedTxt2(i) = numHashedTxt2(i) * -1
                        If numHashedTxt2(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt2(i).ToString.Length - 1
                                numHashedTxt2(i) =
CInt(CStr(numHashedTxt2(i)) & "1")
                            Next
                        End If
                        numHashedTxt2(i) =
Strings.Right(Convert.ToString(numHashedTxt2(i)),
Convert.ToString(numHashedTxt2(i)).Length -
(Convert.ToString(numHashedTxt2(i)).Length - 5))
                    Else
                        If numHashedTxt2(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt2(i).ToString.Length - 1
                                numHashedTxt2(i) =
CInt(CStr(numHashedTxt2(i)) & "1")
                            Next
                        End If
```

```
                        numHashedTxt2(i) =
Strings.Right(Convert.ToString(numHashedTxt2(i)),
Convert.ToString(numHashedTxt2(i)).Length -
(Convert.ToString(numHashedTxt2(i)).Length - 5))
                    End If



                    '*********************************************
                    '** Get Hash3
                    '*********************************************
                    numHashedTxt3(i) = objHash.Hash3(arr(i))
                    numHashedTxt3(i) =
CInt(CStr(numHashedTxt3(i)).Replace("0", "9"))
                    If numHashedTxt3(i) < 0 Then
                        numHashedTxt3(i) = numHashedTxt3(i) * -1
                        If numHashedTxt3(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt3(i).ToString.Length - 1
                                numHashedTxt3(i) =
CInt(CStr(numHashedTxt3(i)) & "1")
                            Next
                        End If
                        numHashedTxt3(i) =
Strings.Right(Convert.ToString(numHashedTxt3(i)),
Convert.ToString(numHashedTxt3(i)).Length -
(Convert.ToString(numHashedTxt3(i)).Length - 5))
                    Else
                        If numHashedTxt3(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt3(i).ToString.Length - 1
                                numHashedTxt3(i) =
CInt(CStr(numHashedTxt3(i)) & "1")
                            Next
                        End If
                        numHashedTxt3(i) =
Strings.Right(Convert.ToString(numHashedTxt3(i)),
Convert.ToString(numHashedTxt3(i)).Length -
(Convert.ToString(numHashedTxt3(i)).Length - 5))
                    End If

                Next i
```

```
'*********************************************
'** Add dummy values
'*********************************************
```

The query after hashing process, it's needed to generate dummy values. The following code will generate three values and each value is two digits. The result of dummy value will be saved to dumValues array.

```
    For c = 0 To 2
    Dim rnd1 As New Random
    Dim x As Integer = rnd1.Next(10, 99)
    dumValues(c) = x
Next


'*********************************************
'** Add results on the listview
'*********************************************
```

Based on the inserted data in the textbox, the hash values will be added to the dummy values. The result of adding hash values and dummy values will be saved into array and then reorder the whole values.

```
    Dim itm1 As New ListViewItem
            Dim numHashedTxt(numHashedTxt1.Length) As String

            For b As Integer = 0 To numHashedTxt1.Length - 2
                numHashedTxt(b) = dumValues(0).ToString &
numHashedTxt1(b).ToString & dumValues(1).ToString &
numHashedTxt2(b).ToString & numHashedTxt3(b).ToString &
dumValues(2).ToString

                Dim xHashed As String = String.Empty
                xHashed = numHashedTxt(b).ToString
```

```
'*********************************************
'** Generation public and private keys
'*********************************************
```

The following code will generate public and private keys for the client. GetKeys () is a function will call the globalVars.vb which is responsible for three procedures: generate keys, encryption and decryption.

```
GetKeys()
```

The following code will call encryption function from globalVars.vb class and this function is called EncryptRSA ().

xHashed = hash values + dummy valus + reorder process

xHashed will be passed to the EncryptRSA () in order to be encrypted and the result of this process encryption will be saved into cipherX variable.

```
cipherX = EncryptRSA(xHashed)
```

Now the client is finished the first stage which included:

- Client interface (textbox, search buttons and searching bar)
- Hashing the query
- Adding dummy value
- Reorder the query
- Encryption the query by using public and private keys.

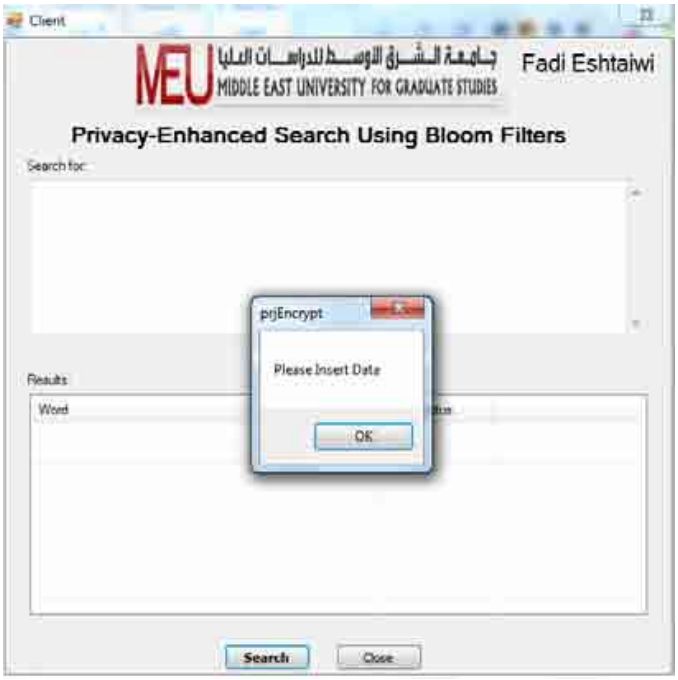
After finishing these steps, the query is ready to send to the server.

The following code will call many functions from checkEncData.vb class and this function is called objCheck.checkData (). By calling this function, decrypt the query, connection the client with server and search for the suery in the server's database. After finishing these steps, the function will return the result in 0 nd 1 format.

```
resp = objCheck.checkData(cipherX)


                    returno = objCheck.getOnes(resp)

                    Dim splitReturno() As String =
Strings.Split(returno, "|")

                    If splitReturno(0) = "1" Then
                        result = True
                    Else
                        result = False
                    End If

                    If result = False Then
                        itm1 = lstView.Items.Add(arr(b).ToString)
                        itm1.SubItems.Add("Not Found!")
                        itm1.SubItems(0).ForeColor = Color.Red
                    ElseIf result = True Then
                        itm1 = lstView.Items.Add(arr(b).ToString)
                        itm1.SubItems.Add("Found")
                        ' itm1.SubItems.Add(splitReturno(1))
                        itm1.SubItems(0).ForeColor = Color.Green
                    End If
                Next b

            End If

            stlbl.Text = "Done"
            pBar.Value = 100
        Else
```

The following is connected with the first process in the client code, if the user clicked search without inserting any data in the textbox, the application will show a message "please insert data"

```
MsgBox("Please Insert Data")
        End If
    End Sub
```




```
    Private Sub frmClient_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        Me.Height = 523
        lstView.Columns.Add("Word", 300, HorizontalAlignment.Left)
        lstView.Columns.Add("Status", 100, HorizontalAlignment.Center)
        ' lstView.Columns.Add("Table Name", 100,
HorizontalAlignment.Center)
    End Sub

End Class
```

There are many classes built in this project. Each class has many functions and we can call these functions.

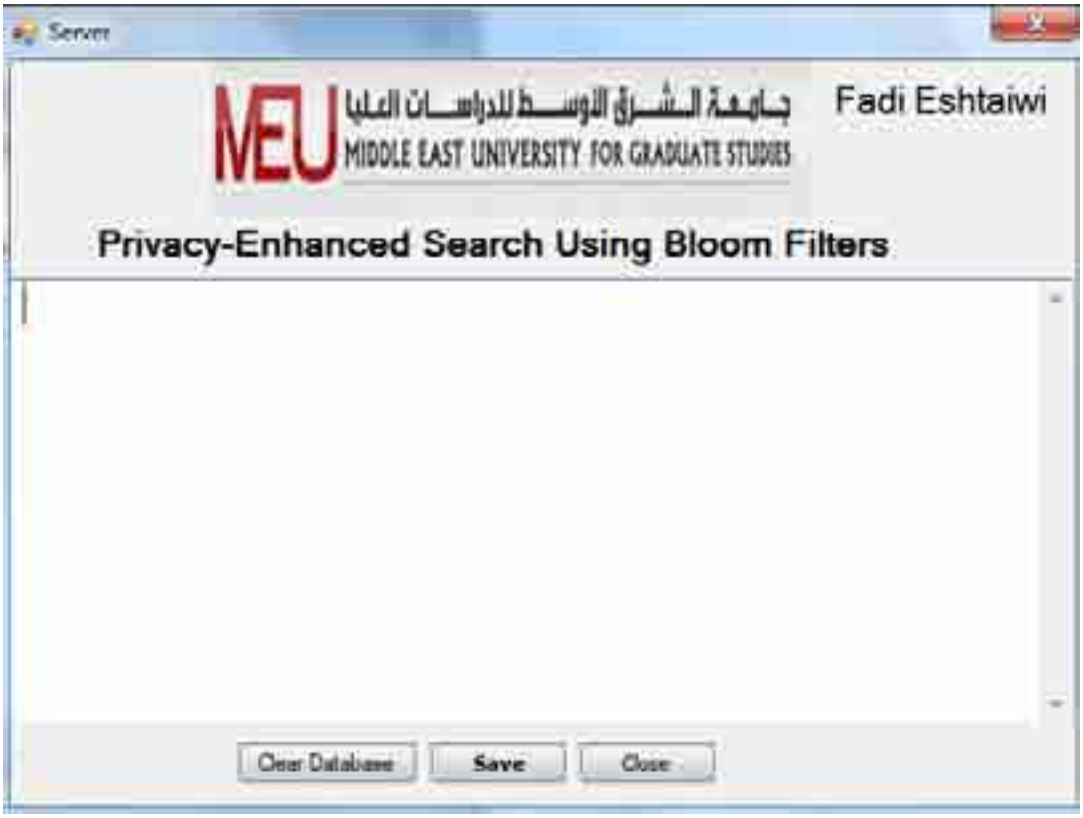
The following code is responsible for the hash processes. Hash function is a built function in the Microsoft visual basic and it had been used in this project. There are three hash functions and each one has different equations than the others.

The following equation is for the first hash function:

$$h = h + Asc(Mid(val, i, 1))$$

```
Public Class clsHash

'------------- get hash code 1 -------------

Public Function Hash1(ByVal val As String) As Int32

Return val.GetHashCode()

End Function

'------------- get hash code 2 -------------


 Public Function Hash2(ByVal val As String) As Int32

 Dim h As Integer
 Dim i As Integer
 h = 0

 For i = 1 To Len(val)

 h = h + Asc(Mid(val, i, 1))

 Next i

 Return (h + val.GetHashCode())

 End Function




'------------- get hash code 3 -------------


 Public Function Hash3(ByVal val As String) As Int32

 Dim h As Integer
 Dim i As Integer
 h = 0
 Dim values As Char() = val.ToCharArray()

 For i = 1 To Len(val)
 h = h + Asc(values(0)) + Asc(values(values.Length() - 1))
 Next i
 Return (h + val.GetHashCode())

 End Function
End Class
```

Designing the server interface required textbox and three buttons:



Textbox: will let the admin to insert data to the server's database.

Save button: save the inserted data into the server's database after hashing process.

Clear data: remove all the data that was inserted from the database.

Close: stop running the application and exit.

Programming the server required to identify many variables.

Server Code

```
Imports System.String
Imports System.Text
Imports Microsoft.VisualBasic
Imports System.IO
Imports System.Data
Imports System.Data.SqlClient
Imports System.Security.Cryptography
Imports System.Diagnostics
Imports System.Security

Public Class frmServer

    Dim objHash As New clsHash

    Private Sub btnEncrypt_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnEncrypt.Click

        If txtServer.Text <> "" Then
            Dim con As New SqlConnection()
            Dim com As New SqlCommand()

            ' -------- Get Random Name from Function Name GetRandomName
-------
            Dim TablName As String = "enctbl" 'getRandomName()
            ' Dim createTbl As String
```

To allow the admin to insert the data to the database, it's needed to establish a connection between the admin and database.

The following code will establish this connection.

```
            con.ConnectionString = "Data Source =(Local); Initial
Catalog = en_db; Integrated Security = True;"
```

After the establishment of the connection, this connection has to be open in order to transmit the data between the admin and the database.

```
            If con.State = Data.ConnectionState.Closed Then
                con.Open()
            End If


            com.Connection = con
```

```
            Dim line As String = txtServer.Text
            Dim arr() As String
            arr = Strings.Split(line, " ")

            For i = 0 To (CInt(arr.Length - 1))
                ' Dim hashedTxt As String
                Dim numHashedTxt1 As Integer
                Dim numHashedTxt2 As Integer
```

There is an array; the inserted data by admin will be set in this array as string. The array will be the input of the hash procedure. There are three hash functions in clsHash.vb class. Each word will pass through these hash functions. Each hash function is various than the other one (each hash hash function has various code).

```
                '---------------------- Get Hash1 ----------------

                numHashedTxt1 = objHash.Hash1(arr(i))
                numHashedTxt1 = CInt(CStr(numHashedTxt1).Replace("0",
"9"))
                If numHashedTxt1 < 0 Then
                    numHashedTxt1 = numHashedTxt1 * -1
                    If numHashedTxt1.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt1.ToString.Length - 1
                            numHashedTxt1 = CInt(CStr(numHashedTxt1) &
"1")
                        Next
                    End If
                    numHashedTxt1 =
Strings.Right(Convert.ToString(numHashedTxt1),
Convert.ToString(numHashedTxt1).Length -
(Convert.ToString(numHashedTxt1).Length - 5))
                Else
                    If numHashedTxt1.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt1.ToString.Length - 1
                            numHashedTxt1 = CInt(CStr(numHashedTxt1) &
"1")
                        Next
                    End If
                    numHashedTxt1 =
Strings.Right(Convert.ToString(numHashedTxt1),
Convert.ToString(numHashedTxt1).Length -
(Convert.ToString(numHashedTxt1).Length - 5))

                End If
```

```
                '--------------------- Get Hash2 ---------------


                numHashedTxt2 = objHash.Hash2(arr(i))
                numHashedTxt2 = CInt(CStr(numHashedTxt2).Replace("0",
"9"))
                If numHashedTxt2 < 0 Then
                    numHashedTxt2 = numHashedTxt2 * -1
                    If numHashedTxt2.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt2.ToString.Length - 1
                            numHashedTxt2 = CInt(CStr(numHashedTxt2) &
"1")
                        Next
                    End If
                    numHashedTxt2 =
Strings.Right(Convert.ToString(numHashedTxt2),
Convert.ToString(numHashedTxt2).Length -
(Convert.ToString(numHashedTxt2).Length - 5))
                Else
                    If numHashedTxt2.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt2.ToString.Length - 1
                            numHashedTxt2 = CInt(CStr(numHashedTxt2) &
"1")
                        Next
                    End If
                    numHashedTxt2 =
Strings.Right(Convert.ToString(numHashedTxt2),
Convert.ToString(numHashedTxt2).Length -
(Convert.ToString(numHashedTxt2).Length - 5))
                End If
               '--------------------- Get Hash3 -----------------

                numHashedTxt3 = objHash.Hash3(arr(i))
                numHashedTxt3 = CInt(CStr(numHashedTxt3).Replace("0",
"9"))
                If numHashedTxt3 < 0 Then
                    numHashedTxt3 = numHashedTxt3 * -1
                    If numHashedTxt3.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt3.ToString.Length - 1
                            numHashedTxt3 = CInt(CStr(numHashedTxt3) &
"1")
                        Next
                    End If
                    numHashedTxt3 =
Strings.Right(Convert.ToString(numHashedTxt3),
Convert.ToString(numHashedTxt3).Length -
(Convert.ToString(numHashedTxt3).Length - 5))
                Else
                    If numHashedTxt3.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt3.ToString.Length - 1
                            numHashedTxt3 = CInt(CStr(numHashedTxt3) &
"1")
                        Next
```

```
                    End If
                    numHashedTxt3 =
Strings.Right(Convert.ToString(numHashedTxt3),



Convert.ToString(numHashedTxt3).Length -
(Convert.ToString(numHashedTxt3).Length - 5))

                End If

                Dim numHashedTxt As String
                numHashedTxt = numHashedTxt1.ToString &
numHashedTxt2.ToString & numHashedTxt3.ToString
```

After hashing the inserted data by the admin, the first hash value will be inserted into the database. This procedure will be implanted three times.

```
'---------------------- Insert Hash1 -----------------

Dim rd As SqlDataReader
Dim isExist As Boolean

com.CommandType = CommandType.Text
com.CommandText = "SELECT * FROM " & TablName & " WHERE enc_id ='" &
numHashedTxt1.ToString & "'"
                rd = com.ExecuteReader()
                isExist = Convert.ToBoolean(rd.Read)
                rd.Close()
```

The program will check the hash value before inserting it because if this hash value exists in the database, it will not save because it already exists.

```
If Not isExist Then
com.CommandType = Data.CommandType.Text
com.CommandText = "INSERT INTO " & TablName & "(enc_id, enc_num) VALUES
('" & numHashedTxt1.ToString & "', '1')"
com.ExecuteNonQuery()
End If

                '---------------------- Insert Hash2 ----------------

com.CommandType = CommandType.Text
```

```
com.CommandText = "SELECT * FROM " & TablName & " WHERE enc_id ='" &
numHashedTxt2.ToString & "'"
rd = com.ExecuteReader()
isExist = Convert.ToBoolean(rd.Read)




rd.Close()

If Not isExist Then
com.CommandType = Data.CommandType.Text
com.CommandText = "INSERT INTO " & TablName & "(enc_id, enc_num) VALUES
('" & numHashedTxt2.ToString & "', '1')"
                    com.ExecuteNonQuery()
                End If

                '--------------------- Insert Hash3 -----------------

                com.CommandType = CommandType.Text
com.CommandText = "SELECT * FROM " & TablName & " WHERE enc_id ='" &
numHashedTxt3.ToString & "'"
                rd = com.ExecuteReader()
                isExist = Convert.ToBoolean(rd.Read)
                rd.Close()

 If Not isExist Then
com.CommandType = Data.CommandType.Text
com.CommandText = "INSERT INTO " & TablName & "(enc_id, enc_num) VALUES
('" & numHashedTxt3.ToString & "', '1')"
                    com.ExecuteNonQuery()
                End If
            Next i

                '-----------------------------------------------------
```

After the admin finishing of inserting the data, the connection will be closed. The following code will close the connection between the admin and database.

```
            If con.State = Data.ConnectionState.Open Then
                con.Close()
            End If
        Else
            MsgBox("Please Insert Data")

        End If

        txtServer.Text = ""
```

```
    End Sub

        '------------------------------------------------------
```

If the admin clicked the close button, the following code will be executed and the result of this event is topping the process and closes the application.

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        'Dim frm As New frmMain
        'frm.Show()
        Me.Close()
```

If the admin clicked the Clear database button, the program will show the admin message box (Yes /No) in order to inform his request because all the data will be deleted. The following code will be executed and the result of this event is “Empty Database”.

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim con As New SqlConnection()
        Dim com As New SqlCommand()

If MsgBox("Are you sure you want to clear the database?",
MsgBoxStyle.YesNo, "Clear Database") = MsgBoxResult.No Then
            Exit Sub
        End If

con.ConnectionString = "Data Source =(Local); Initial Catalog = en_db;
Integrated Security = True;"

        If con.State = Data.ConnectionState.Closed Then
            con.Open()
        End If

com.Connection = con
com.CommandText = "DELETE FROM enctbl"
com.ExecuteNonQuery()

        con.Close()
```

```
    End Sub
End Class
```

checkEncData.vb is a class which has many functions. Each function will execute a specific process.

Functions include:

- Decryption the encrypted data by calling functions from globalVars.vb class.
- Search for the hash values in the database.
- Return the result (0 or 1).

checkEncData.vb

```
Imports Microsoft.VisualBasic
Imports System.Data.SqlClient

Public Class checkEncData
    Private deCipherX As String = String.Empty

    Public Function checkData(ByVal ciphered As String) As String
```

DecryptRSA() will get the encrypted data from the client's query and then decrypt it and return the plain text and put it in the variable (deCipherX).

```
        deCipherX = DecryptRSA()


        Dim tempnewDeCipherX1 As String = String.Empty
        Dim tempnewDeCipherX2 As String = String.Empty
```

```
        Dim FoundHashes(2) As String


        '*********************************************
        '** Dividing hash to three sets of five numbers
        '*********************************************

Dim divDeCipher(2) As String
Dim adds As Integer

For z As Integer = 0 To 2
adds = z * 5
divDeCipher(z) = Strings.Mid(newDeCipherX, adds + 1, 5)
Next
Dim con As New SqlConnection
Dim com As New SqlCommand
Dim rd As SqlDataReader
Dim xx As String = String.Empty
Dim dataTable() As String
Dim isExist As Boolean
```

After decryption the query, the program will open the connection and search for the hash values.

```
con.ConnectionString = "Data Source =(Local); Initial Catalog = en_db;
Integrated Security = True;"

        If con.State = Data.ConnectionState.Closed Then
            con.Open()
        End If

com.Connection = con
com.CommandText = "SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE' ORDER BY TABLE_NAME"
        rd = com.ExecuteReader

Try
While rd.Read
xx = xx & rd.GetString(0) & "|"
End While

rd.Close()
con.Close()
dataTable = Strings.Split(xx, "|")
```

```
'*****************************************
'** Searching Results
'*****************************************
```

The following code will search the hash values. If the hash values exist, the program will return 1. Otherwise, the program will return 0.

```
If con.State = Data.ConnectionState.Closed Then
con.Open()
End If

com.Connection = con

For j As Integer = 0 To dataTable.Length - 1
If dataTable(j) = String.Empty Then Exit For
For s As Integer = 0 To 2
com.CommandType = CommandType.Text
com.CommandText = "SELECT * FROM " & dataTable(j).ToString & " WHERE "
& dataTable(j).ToString & ".enc_id = '" & divDeCipher(s) & "'"
                    rd = com.ExecuteReader()
                    isExist = Convert.ToBoolean(rd.Read)
                    rd.Close()

                    If isExist Then
                        FoundHashes(s) = "1"
                    Else
                        FoundHashes(s) = "0"
                    End If
                Next


Dim myFoundHashes As String = String.Empty
For u As Integer = 0 To FoundHashes.Length - 1
myFoundHashes = myFoundHashes & FoundHashes(u).ToString
Next
Return myFoundHashes

Next

Catch ex As Exception
Throw ex
End Try
con.Close()
End Function

Public Function getOnes(ByVal foundHash As String) As String
Dim splitResult() As String = Strings.Split(foundHash, "|")

If InStr(splitResult(0), "0", CompareMethod.Text) Then
```

```
Return "0"
Else
Return "1" '|" & splitResult(1).ToString
End If
End Function
End Class
```

globalVars.vb

```
Imports System.IO
Imports System.Text
Imports Microsoft.VisualBasic

Public Module Globals
    Public text_to_encrypt As String
    Public pubKey, priKey As String
    Dim TDES As TripleDES
```

The following code will give value for both public and private keys to be used in the encryption process. These values are set by TripleDES.vb class based on specific equations.

```
  Public Sub GetKeys()
        Try
            pubKey = String.Empty
            priKey = String.Empty
            TDES = New TripleDES
            TDES.GetKeysForRSA(pubKey, priKey)
        Catch ex As Exception
            Throw ex
        End Try
End Sub
```

After getting the public and private keys from TripleDES.vb class, the program will use these keys in the encryption.

```
Public Function EncryptRSA(ByVal text As String) As String
Dim txtEnc As Byte()
frmServer.txtEnc.Text = String.Empty
Try
Dim tData As New StringBuilder
Dim arrlis As ArrayList
TDES = New TripleDES

text_to_encrypt = text
arrlis = TDES.EncryptRSA(text, pubKey)
```

```
For j As Integer = 0 To arrlis.Count - 1
txtEnc = CType(arrlis(j), Byte())

For i As Integer = 0 To txtEnc.Length - 1
frmServer.txtEnc.AppendText((Chr(txtEnc(i))))
Next i
Next j

 Dim innerString As String = frmServer.txtEnc.Text

Return innerString

Catch ex As Exception
Throw ex
End Try
End Function
```

The following code will be used in the decryption process of the encrypted data. This function will get the public and private keys from TripleDES.vb class. After getting the keys, the encrypted will pass through many procedures to decrypt the data. The details of RSA algorithm will be discussed in the TripleDES.vb class.

```
Public Function DecryptRSA() As String
Dim txtEnc As Byte()
frmClient.txtEnc.Text = String.Empty
Try
Dim tData As New StringBuilder
Dim arrlis As ArrayList
TDES = New TripleDES

arrlis = TDES.EncryptRSA(text_to_encrypt, pubKey)

For j As Integer = 0 To arrlis.Count - 1
txtEnc = CType(arrlis(j), Byte())
txtEnc = TDES.DecryptRSA(txtEnc, priKey)

For i As Integer = 0 To txtEnc.Length - 1
frmClient.txtEnc.AppendText((Chr(txtEnc(i))))
Next i
Next j

Dim innerString As String = frmClient.txtEnc.Text

Return innerString

Catch ex As Exception
Throw ex
End Try
```

```
End Function
End Module
```

In this class, public and private keys will be generated for both encryption and decryption processes. GetKeys()will call the GetKeysForRSA function.

The code of GetKeysForRSA function to provide the EncryptRSA function with the public and private keys.

TripleDES.vb

```
Imports System
Imports System.IO
Imports System.Security.Cryptography
Imports System.Text

Public Class TripleDES

Shared publicKey As String 'The public key only
Shared privateKey As String
Shared xmlKeys As String 'A combination of both the public and private
keys
Dim key3DES() As Byte = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20, 21, 22, 23, 24}
Dim key() As Byte = {1, 2, 3, 4, 5, 6, 7, 8}
Dim iv() As Byte = {65, 110, 68, 26, 69, 178, 200, 219}


Public Sub New()
Dim rsa As New RSACryptoServiceProvider
xmlKeys = rsa.ToXmlString(True)
publicKey = rsa.ToXmlString(False)
End Sub

Public Sub New(ByVal encType As String)
Select Case encType
Case "3DES"
Case "RSA"
Case Else
End Select
End Sub
```

After hashing the data, the result of hash values added to dummy values and reordering the whole query will be the input of the encryption function. The following code will do the encryption process. Public also will be encrypted.

```
Public Function Encrypt(ByVal plainText As String, ByVal encType As

String) As Byte()
Dim utf8encoder As UTF8Encoding = New UTF8Encoding
Dim inputInBytes() As Byte = utf8encoder.GetBytes(plainText)
Dim tdesProvider As Object
Dim cryptoTransform As Object
Select Case encType
Case "DES"
tdesProvider = New DESCryptoServiceProvider

cryptoTransform = tdesProvider.CreateEncryptor(Me.key, Me.iv)
Case "3DES"
tdesProvider = New TripleDESCryptoServiceProvider
cryptoTransform = tdesProvider.CreateEncryptor(Me.key3DES, Me.iv)
End Select

Dim encryptedStream As MemoryStream = New MemoryStream
Dim cryptStream As CryptoStream = New CryptoStream(encryptedStream,
cryptoTransform, CryptoStreamMode.Write)

cryptStream.Write(inputInBytes, 0, inputInBytes.Length)
cryptStream.FlushFinalBlock()
encryptedStream.Position = 0

Dim result(encryptedStream.Length - 1) As Byte
encryptedStream.Read(result, 0, encryptedStream.Length)
cryptStream.Close()
Return result
End Function
```

The encrypted data will be encrypted and extract the public and private keys in the server and client. The decryption process in this project is involved in many places. The client will call this function two times and the server once.

```
Public Function Decrypt(ByVal inputInBytes() As Byte, ByVal decType As
String) As String
Dim utf8encoder As UTF8Encoding = New UTF8Encoding

Dim tdesProvider As Object
Dim cryptoTransform As Object
Select Case decType
Case "DES"
tdesProvider = New DESCryptoServiceProvider

cryptoTransform = tdesProvider.CreateDecryptor(Me.key, Me.iv)
```

Collision Rate calculation code

```
Dim m As String = (arr.Length - 1)
Dim aa As String = (1 - (1 - (1 / m)) ^ 6) ^ 3
```

Chapter 6

Conclusions

Bloom filters (server's database in this project) can be built by using hashing function which provides the hash values to the server's database and then it will be secured. The proposed system can solve the third party problem. There are three eight encryption/decryption processes in this project which mean the security is very high level. Through this project, the third party has been removed and the client has the ability to enter the server's database without knowing what the client is looking for. RSA is used in this project in order to encrypt the query because it's a very strong algorithm technique and no one can crack the query after encrypt it by two different keys. The size of the document can affect the performance of this project. So if the document size is increased, the required time to find the query will increase. Using three hash functions will decrease or prevent the collision rate in this project and the program is not designed to have a word with the same hash values. "Eshtaiwi" word doesn't exist in the server's database, so the program return is not found. Copy and Paste is not supported in this project because may be any client can attach a huge number of data and this will occur down in the server. SQL server 2005 is very good software to build the database and it can be used in many applications and the server's database in this project is dynamic and that is a good feature that can be provided by SQL server 2005. So the administrator can insert data as possible as (this is determined by SQL server program).

The proposed system will enhance the privacy during the search process in the bloom filters. Information security and privacy is a very important issue for any organizations and companies especially when they are sharing the data between them. It is necessary when there are two or more parties that do not trust each other and they are share data between them. For example if there are two intelligence agencies that share the data and they wish to let each party query the other databases without disclosing the query and know what the other is searching for specific data. (Bellovin and Rescorla , 2005).

Chapter 7

Recommendations

- Implement this project in the real world especially in two parties that don't fully trust each other.
- This project is designed by using VB.net so it's recommended to use another software package because VB.net in the server interface doesn't allow the admin to perform copy/past feature and the admin must insert the data by typing which will take long time.
- This project uses three hash functions. So, it's recommended to use two hash functions rather three. Each word in this project will yield three hash values, so the number of hash values is huge and requires a huge of database. SQL server 2005 is limited space.
- The server's database in this project is built by typing the text in the specific textbox. So it's recommended to change the method of inserting data to the server's database by uploading files. This feature will allow the admin to upload the data as file whereas *.txt, .doc* and *.pdf*. This feature wasn't adopted in this project in order to prevent attaching huge file which brings the server down.

Appendix

Menu Code

```
Public Class frmMain

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim frm As New frmClient
        frm.Show()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Dim frm As New frmServer
        frm.Show()
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        Application.Exit()
    End Sub
End Class
```

Client Code

```
Public Class frmClient
    Dim objHash As New clsHash
    Dim dumValues(3) As Integer

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        ' Dim frm As New frmMain
        ' frm.Show()
        Me.Close()
    End Sub

    Private Sub btnSearch_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnSearch.Click
        If txtClient.Text <> "" Then
            Dim timee As String = Date.Now
            On Error Resume Next
            Dim cipherX As String
            Dim objCheck As New checkEncData
            Dim resp As String
            Dim returno As String
            Dim result As Boolean

            pBar.Value = 0

            If txtClient.Text <> String.Empty Then

                Dim line As String = txtClient.Text
                Dim arr() As String
                arr = Split(line, " ")
                Dim arrLength As Integer = CInt(arr.Length)
                Dim numHashedTxt1(arrLength) As Integer
                Dim numHashedTxt2(arrLength) As Integer
                Dim numHashedTxt3(arrLength) As Integer

                stlbl.Visible = True
                pBar.Visible = True
                stlbl.Text = "Searching..."
                pBar.Minimum = 0
                pBar.Maximum = 100

                For i = 0 To (CInt(arr.Length - 1))

                    pBar.Value = pBar.Value + 10 * (i + 1)

                    '*********************************************
                    '** Get Hash1
                    '*********************************************
                    numHashedTxt1(i) = objHash.Hash1(arr(i))
                    numHashedTxt1(i) =
CInt(CStr(numHashedTxt1(i)).Replace("0", "9"))
                    If numHashedTxt1(i) < 0 Then
                        numHashedTxt1(i) = numHashedTxt1(i) * -1
                        If numHashedTxt1(i).ToString.Length < 5 Then
```

```
                            For y As Integer = 1 To
numHashedTxt1(i).ToString.Length - 1
                                numHashedTxt1(i) =
CInt(CStr(numHashedTxt1(i)) & "1")
                            Next
                        End If
                        numHashedTxt1(i) =
Strings.Right(Convert.ToString(numHashedTxt1(i)),
Convert.ToString(numHashedTxt1(i)).Length -
(Convert.ToString(numHashedTxt1(i)).Length - 5))
                    Else
                        If numHashedTxt1(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt1(i).ToString.Length - 1
                                numHashedTxt1(i) =
CInt(CStr(numHashedTxt1(i)) & "1")
                            Next
                        End If
                        numHashedTxt1(i) =
Strings.Right(Convert.ToString(numHashedTxt1(i)),
Convert.ToString(numHashedTxt1(i)).Length -
(Convert.ToString(numHashedTxt1(i)).Length - 5))

                    End If


                    '*********************************************
                    '** Get Hash2
                    '*********************************************
                    numHashedTxt2(i) = objHash.Hash2(arr(i))
                    numHashedTxt2(i) =
CInt(CStr(numHashedTxt2(i)).Replace("0", "9"))
                    If numHashedTxt2(i) < 0 Then
                        numHashedTxt2(i) = numHashedTxt2(i) * -1
                        If numHashedTxt2(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt2(i).ToString.Length - 1
                                numHashedTxt2(i) =
CInt(CStr(numHashedTxt2(i)) & "1")
                            Next
                        End If
                        numHashedTxt2(i) =
Strings.Right(Convert.ToString(numHashedTxt2(i)),
Convert.ToString(numHashedTxt2(i)).Length -
(Convert.ToString(numHashedTxt2(i)).Length - 5))
                    Else
                        If numHashedTxt2(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt2(i).ToString.Length - 1
                                numHashedTxt2(i) =
CInt(CStr(numHashedTxt2(i)) & "1")
                            Next
                        End If
                        numHashedTxt2(i) =
Strings.Right(Convert.ToString(numHashedTxt2(i)),
Convert.ToString(numHashedTxt2(i)).Length -
(Convert.ToString(numHashedTxt2(i)).Length - 5))
```

```
                    End If


                    '*********************************************
                    '** Get Hash3
                    '*********************************************
                    numHashedTxt3(i) = objHash.Hash3(arr(i))
                    numHashedTxt3(i) =
CInt(CStr(numHashedTxt3(i)).Replace("0", "9"))
                    If numHashedTxt3(i) < 0 Then
                        numHashedTxt3(i) = numHashedTxt3(i) * -1
                        If numHashedTxt3(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt3(i).ToString.Length - 1
                                numHashedTxt3(i) =
CInt(CStr(numHashedTxt3(i)) & "1")
                            Next
                        End If
                        numHashedTxt3(i) =
Strings.Right(Convert.ToString(numHashedTxt3(i)),
Convert.ToString(numHashedTxt3(i)).Length -
(Convert.ToString(numHashedTxt3(i)).Length - 5))
                    Else
                        If numHashedTxt3(i).ToString.Length < 5 Then
                            For y As Integer = 1 To
numHashedTxt3(i).ToString.Length - 1
                                numHashedTxt3(i) =
CInt(CStr(numHashedTxt3(i)) & "1")
                            Next
                        End If
                        numHashedTxt3(i) =
Strings.Right(Convert.ToString(numHashedTxt3(i)),
Convert.ToString(numHashedTxt3(i)).Length -
(Convert.ToString(numHashedTxt3(i)).Length - 5))
                    End If

                Next i
                MsgBox(timee)
                '*********************************************
                '** Randomize dummy values
                '*********************************************
                For c = 0 To 2
                    Dim rnd1 As New Random
                    Dim x As Integer = rnd1.Next(10, 99)
                    dumValues(c) = x
                Next

                '*********************************************
                '** Add results on the listview
                '*********************************************
                Dim itm1 As New ListViewItem
                Dim numHashedTxt(numHashedTxt1.Length) As String

                For b As Integer = 0 To numHashedTxt1.Length - 2
                    numHashedTxt(b) = dumValues(0).ToString &
numHashedTxt1(b).ToString & dumValues(1).ToString &
```

```
numHashedTxt2(b).ToString & numHashedTxt3(b).ToString &
dumValues(2).ToString

                    Dim xHashed As String = String.Empty
                    xHashed = numHashedTxt(b).ToString

                    GetKeys()
                    cipherX = EncryptRSA(xHashed)
                    MsgBox(cipherX)
                    ' text_to_encrypt = objCheck.checkData(cipherX)
                    resp = objCheck.checkData(cipherX) 'DecryptRSA()

                    returno = objCheck.getOnes(resp)

                    Dim splitReturno() As String =
Strings.Split(returno, "|")

                    If splitReturno(0) = "1" Then
                        result = True
                    Else
                        result = False
                    End If

                    If result = False Then
                        itm1 = lstView.Items.Add(arr(b).ToString)
                        itm1.SubItems.Add("Not Found!")
                        itm1.SubItems(0).ForeColor = Color.Red
                    ElseIf result = True Then
                        itm1 = lstView.Items.Add(arr(b).ToString)
                        itm1.SubItems.Add("Found")
                        ' itm1.SubItems.Add(splitReturno(1))
                        itm1.SubItems(0).ForeColor = Color.Green
                    End If
                Next b

            End If

            stlbl.Text = "Done"
            pBar.Value = 100


        Else
            MsgBox("Please Insert Data")
        End If
    End Sub

    Private Sub frmClient_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        Me.Height = 523
        lstView.Columns.Add("Word", 300, HorizontalAlignment.Left)
        lstView.Columns.Add("Status", 100, HorizontalAlignment.Center)
        ' lstView.Columns.Add("Table Name", 100,
HorizontalAlignment.Center)
    End Sub

End Class
```

Hash Code

```
Imports Microsoft.VisualBasic
Imports System.IO

Public Class clsHash

'------------- get hash code 1 -------------

Public Function Hash1(ByVal val As String) As Int32

Return val.GetHashCode()

End Function

'------------- get hash code 2 -------------


 Public Function Hash2(ByVal val As String) As Int32

 Dim h As Integer
 Dim i As Integer
 h = 0

 For i = 1 To Len(val)

 h = h + Asc(Mid(val, i, 1))

 Next i

 Return (h + val.GetHashCode())

 End Function

'------------- get hash code 3 -------------


 Public Function Hash3(ByVal val As String) As Int32

 Dim h As Integer
 Dim i As Integer
 h = 0
 Dim values As Char() = val.ToCharArray()

 For i = 1 To Len(val)
 h = h + Asc(values(0)) + Asc(values(values.Length() - 1))
 Next i
 Return (h + val.GetHashCode())

 End Function
End Class
```

Server Code

```
Imports System.String
Imports System.Text
Imports Microsoft.VisualBasic
Imports System.IO
Imports System.Data
Imports System.Data.SqlClient
Imports System.Security.Cryptography
Imports System.Diagnostics
Imports System.Security

Public Class frmServer

    Dim objHash As New clsHash

    Private Sub btnEncrypt_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnEncrypt.Click

        If txtServer.Text <> "" Then
            Dim con As New SqlConnection()
            Dim com As New SqlCommand()

            ' -------- Get Random Name from Function Name GetRandomName
-------
            Dim TablName As String = "enctbl" 'getRandomName()
            ' Dim createTbl As String

            con.ConnectionString = "Data Source =(Local); Initial
Catalog = en_db; Integrated Security = True;"
            If con.State = Data.ConnectionState.Closed Then
                con.Open()
            End If

            '------------- Create Table In database ----------------

            com.Connection = con


            Dim line As String = txtServer.Text
            Dim arr() As String
            arr = Strings.Split(line, " ")

            For i = 0 To (CInt(arr.Length - 1))
                ' Dim hashedTxt As String
                Dim numHashedTxt1 As Integer
                Dim numHashedTxt2 As Integer
                Dim numHashedTxt3 As Integer

                '--------------------- Get Hash1 ----------------

                numHashedTxt1 = objHash.Hash1(arr(i))
                numHashedTxt1 = CInt(CStr(numHashedTxt1).Replace("0",
"9"))
                If numHashedTxt1 < 0 Then
                    numHashedTxt1 = numHashedTxt1 * -1
```

```
                    If numHashedTxt1.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt1.ToString.Length - 1
                            numHashedTxt1 = CInt(CStr(numHashedTxt1) &
"1")
                        Next
                    End If
                    numHashedTxt1 =
Strings.Right(Convert.ToString(numHashedTxt1),
Convert.ToString(numHashedTxt1).Length -
(Convert.ToString(numHashedTxt1).Length - 5))
                Else
                    If numHashedTxt1.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt1.ToString.Length - 1
                            numHashedTxt1 = CInt(CStr(numHashedTxt1) &
"1")
                        Next
                    End If
                    numHashedTxt1 =
Strings.Right(Convert.ToString(numHashedTxt1),
Convert.ToString(numHashedTxt1).Length -
(Convert.ToString(numHashedTxt1).Length - 5))

                End If

                '--------------------- Get Hash2 ---------------

                numHashedTxt2 = objHash.Hash2(arr(i))
                numHashedTxt2 = CInt(CStr(numHashedTxt2).Replace("0",
"9"))
                If numHashedTxt2 < 0 Then
                    numHashedTxt2 = numHashedTxt2 * -1
                    If numHashedTxt2.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt2.ToString.Length - 1
                            numHashedTxt2 = CInt(CStr(numHashedTxt2) &
"1")
                        Next
                    End If
                    numHashedTxt2 =
Strings.Right(Convert.ToString(numHashedTxt2),
Convert.ToString(numHashedTxt2).Length -
(Convert.ToString(numHashedTxt2).Length - 5))
                Else
                    If numHashedTxt2.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt2.ToString.Length - 1
                            numHashedTxt2 = CInt(CStr(numHashedTxt2) &
"1")
                        Next
                    End If
                    numHashedTxt2 =
Strings.Right(Convert.ToString(numHashedTxt2),
Convert.ToString(numHashedTxt2).Length -
(Convert.ToString(numHashedTxt2).Length - 5))
                End If
```

```
                '--------------------- Get Hash3 -----------------

                numHashedTxt3 = objHash.Hash3(arr(i))
                numHashedTxt3 = CInt(CStr(numHashedTxt3).Replace("0",
"9"))
                If numHashedTxt3 < 0 Then
                    numHashedTxt3 = numHashedTxt3 * -1
                    If numHashedTxt3.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt3.ToString.Length - 1
                            numHashedTxt3 = CInt(CStr(numHashedTxt3) &
"1")
                        Next
                    End If
                    numHashedTxt3 =
Strings.Right(Convert.ToString(numHashedTxt3),
Convert.ToString(numHashedTxt3).Length -
(Convert.ToString(numHashedTxt3).Length - 5))
                Else
                    If numHashedTxt3.ToString.Length < 5 Then
                        For y As Integer = 1 To
numHashedTxt3.ToString.Length - 1
                            numHashedTxt3 = CInt(CStr(numHashedTxt3) &
"1")
                        Next
                    End If
                    numHashedTxt3 =
Strings.Right(Convert.ToString(numHashedTxt3),
Convert.ToString(numHashedTxt3).Length -
(Convert.ToString(numHashedTxt3).Length - 5))

                End If

                Dim numHashedTxt As String
                numHashedTxt = numHashedTxt1.ToString &
numHashedTxt2.ToString & numHashedTxt3.ToString



                '--------------------- Insert Hash1 ---------------

                Dim rd As SqlDataReader
                Dim isExist As Boolean

                com.CommandType = CommandType.Text
                com.CommandText = "SELECT * FROM " & TablName & " WHERE
enc_id ='" & numHashedTxt1.ToString & "'"
                rd = com.ExecuteReader()
                isExist = Convert.ToBoolean(rd.Read)
                rd.Close()

                If Not isExist Then
                    com.CommandType = Data.CommandType.Text
                    com.CommandText = "INSERT INTO " & TablName &
"(enc_id, enc_num) VALUES ('" & numHashedTxt1.ToString & "', '1')"
                    com.ExecuteNonQuery()
                End If
```

```
                '--------------------- Insert Hash2 ---------------

                com.CommandType = CommandType.Text
                com.CommandText = "SELECT * FROM " & TablName & " WHERE
enc_id ='" & numHashedTxt2.ToString & "'"
                rd = com.ExecuteReader()
                isExist = Convert.ToBoolean(rd.Read)
                rd.Close()

                If Not isExist Then
                    com.CommandType = Data.CommandType.Text
                    com.CommandText = "INSERT INTO " & TablName &
"(enc_id, enc_num) VALUES ('" & numHashedTxt2.ToString & "', '1')"
                    com.ExecuteNonQuery()
                End If

                '--------------------- Insert Hash3 -----------------

                com.CommandType = CommandType.Text
                com.CommandText = "SELECT * FROM " & TablName & " WHERE
enc_id ='" & numHashedTxt3.ToString & "'"
                rd = com.ExecuteReader()
                isExist = Convert.ToBoolean(rd.Read)
                rd.Close()

                If Not isExist Then
                    com.CommandType = Data.CommandType.Text
                    com.CommandText = "INSERT INTO " & TablName &
"(enc_id, enc_num) VALUES ('" & numHashedTxt3.ToString & "', '1')"
                    com.ExecuteNonQuery()
                End If

                '-------------------------------------------------------

            Next i


            If con.State = Data.ConnectionState.Open Then
                con.Close()
            End If
        Else
            MsgBox("Please Insert Data")

        End If

        txtServer.Text = ""
    End Sub

    Function getRandomName() As String
        Dim ref As String
        Dim random1 As New Random
        Dim dt As DateTime = DateTime.Now

        If dt.Millisecond.ToString.Length = 1 Then
            ref = ref + "00" + dt.Millisecond.ToString

        ElseIf dt.Millisecond.ToString.Length = 2 Then
```

```
            ref = ref + "0" + dt.Millisecond.ToString

        Else

            ref = ref + dt.Millisecond.ToString
        End If

        ref = ref + random1.Next(10000, 99999).ToString

        Return ref
    End Function

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        'Dim frm As New frmMain
        'frm.Show()
        Me.Close()
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim con As New SqlConnection()
        Dim com As New SqlCommand()

        If MsgBox("Are you sure you want to clear the database?",
MsgBoxStyle.YesNo, "Clear Database") = MsgBoxResult.No Then
            Exit Sub
        End If

        con.ConnectionString = "Data Source =(Local); Initial Catalog =
en_db; Integrated Security = True;"
        If con.State = Data.ConnectionState.Closed Then
            con.Open()
        End If

        com.Connection = con
        com.CommandText = "DELETE FROM enctbl"
        com.ExecuteNonQuery()

        con.Close()
    End Sub
End Class
```

checkEncData.vb

```
Imports Microsoft.VisualBasic
Imports System.Data.SqlClient

Public Class checkEncData
    Private deCipherX As String = String.Empty

    Public Function checkData(ByVal ciphered As String) As String

        deCipherX = DecryptRSA()
        Dim newDeCipherX As String = String.Empty
        Dim tempnewDeCipherX1 As String = String.Empty
        Dim tempnewDeCipherX2 As String = String.Empty
        Dim FoundHashes(2) As String


        newDeCipherX = Strings.Right(deCipherX, deCipherX.Length - 2)
'**Cut front dummy
        newDeCipherX = Strings.Left(newDeCipherX, newDeCipherX.Length -
2) '**Cut rare dummy
        tempnewDeCipherX1 = Strings.Left(newDeCipherX,
newDeCipherX.Length - 12) '**Front
        tempnewDeCipherX2 = Strings.Right(newDeCipherX,
newDeCipherX.Length - 7) '**End
        newDeCipherX = tempnewDeCipherX1 & tempnewDeCipherX2 '** Pure
Hash

        '*******************************************
        '** Dividing hash to three sets of five numbers
        '*******************************************
        Dim divDeCipher(2) As String
        Dim adds As Integer

        For z As Integer = 0 To 2
            adds = z * 5
            divDeCipher(z) = Strings.Mid(newDeCipherX, adds + 1, 5)
        Next

        Dim con As New SqlConnection
        Dim com As New SqlCommand
        Dim rd As SqlDataReader
        Dim xx As String = String.Empty
        Dim dataTable() As String
        Dim isExist As Boolean

        con.ConnectionString = "Data Source =(Local); Initial Catalog =
en_db; Integrated Security = True;"
        If con.State = Data.ConnectionState.Closed Then
            con.Open()
        End If

        com.Connection = con
```

```
        com.CommandText = "SELECT TABLE_NAME FROM
INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' ORDER BY
TABLE_NAME"
        rd = com.ExecuteReader

        Try
            While rd.Read
                xx = xx & rd.GetString(0) & "|"
            End While

            rd.Close()
            con.Close()

            dataTable = Strings.Split(xx, "|")

            '*****************************************
            '** Searching Results
            '*****************************************
            If con.State = Data.ConnectionState.Closed Then
                con.Open()
            End If

            com.Connection = con

            For j As Integer = 0 To dataTable.Length - 1
                If dataTable(j) = String.Empty Then Exit For
                For s As Integer = 0 To 2
                    com.CommandType = CommandType.Text
                    com.CommandText = "SELECT * FROM " &
dataTable(j).ToString & " WHERE " & dataTable(j).ToString & ".enc_id =
'" & divDeCipher(s) & "'"
                    rd = com.ExecuteReader()
                    isExist = Convert.ToBoolean(rd.Read)
                    rd.Close()

                    If isExist Then
                        FoundHashes(s) = "1"
                    Else
                        FoundHashes(s) = "0"
                    End If
                Next

                Dim myFoundHashes As String = String.Empty
                For u As Integer = 0 To FoundHashes.Length - 1
                    myFoundHashes = myFoundHashes &
FoundHashes(u).ToString
                Next

                '** Returning hash search result for this word with the
table name before
                Return myFoundHashes '& "|" & dataTable(j).ToString
'EncryptRSA(myFoundHashes & "|" & dataTable(j).ToString)

            Next

        Catch ex As Exception
            Throw ex
```

```
        End Try

        con.Close()

    End Function

    Public Function getOnes(ByVal foundHash As String) As String
        Dim splitResult() As String = Strings.Split(foundHash, "|")

        If InStr(splitResult(0), "0", CompareMethod.Text) Then
            Return "0"
        Else
            Return "1" '|" & splitResult(1).ToString
        End If
    End Function
End Class
```

globalVars.vb

```
Imports System.IO
Imports System.Text
Imports Microsoft.VisualBasic

Public Module Globals
    Public text_to_encrypt As String
    Public pubKey, priKey As String
    Dim TDES As TripleDES

    Public Sub GetKeys()
        Try
            pubKey = String.Empty
            priKey = String.Empty
            TDES = New TripleDES
            TDES.GetKeysForRSA(pubKey, priKey)
        Catch ex As Exception
            Throw ex
        End Try
    End Sub

    Public Function EncryptRSA(ByVal text As String) As String
        Dim txtEnc As Byte()
        frmServer.txtEnc.Text = String.Empty
        Try
            Dim tData As New StringBuilder
            Dim arrlis As ArrayList
            TDES = New TripleDES

            text_to_encrypt = text
            arrlis = TDES.EncryptRSA(text, pubKey)

            For j As Integer = 0 To arrlis.Count - 1
                txtEnc = CType(arrlis(j), Byte())

                For i As Integer = 0 To txtEnc.Length - 1
                    frmServer.txtEnc.AppendText((Chr(txtEnc(i))))
                Next i
            Next j

            Dim innerString As String = frmServer.txtEnc.Text

            Return innerString

        Catch ex As Exception
            Throw ex
        End Try
    End Function

    Public Function DecryptRSA() As String
        Dim txtEnc As Byte()
        frmClient.txtEnc.Text = String.Empty
        Try
            Dim tData As New StringBuilder
            Dim arrlis As ArrayList
```

```
            TDES = New TripleDES

            arrlis = TDES.EncryptRSA(text_to_encrypt, pubKey)

            For j As Integer = 0 To arrlis.Count - 1
                txtEnc = CType(arrlis(j), Byte())
                txtEnc = TDES.DecryptRSA(txtEnc, priKey)

                For i As Integer = 0 To txtEnc.Length - 1
                    frmClient.txtEnc.AppendText((Chr(txtEnc(i))))
                Next i
            Next j

            Dim innerString As String = frmClient.txtEnc.Text

            Return innerString

        Catch ex As Exception
            Throw ex
        End Try
    End Function
End Module
```

TripleDES.vb

```
Imports System
Imports System.IO
Imports System.Security.Cryptography
Imports System.Text

Public Class TripleDES

    Shared publicKey As String 'The public key only
    Shared privateKey As String
    Shared xmlKeys As String 'A combination of both the public and
private keys
    Dim key3DES() As Byte = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24}
    Dim key() As Byte = {1, 2, 3, 4, 5, 6, 7, 8}
    Dim iv() As Byte = {65, 110, 68, 26, 69, 178, 200, 219}


    Public Sub New()
        Dim rsa As New RSACryptoServiceProvider
        xmlKeys = rsa.ToXmlString(True)
        publicKey = rsa.ToXmlString(False)
    End Sub

    Public Sub New(ByVal encType As String)
        Select Case encType
            Case "3DES"
            Case "RSA"
            Case Else
        End Select
    End Sub

    Public Function Encrypt(ByVal plainText As String, ByVal encType As
String) As Byte()
        Dim utf8encoder As UTF8Encoding = New UTF8Encoding
        Dim inputInBytes() As Byte = utf8encoder.GetBytes(plainText)
        Dim tdesProvider As Object
        Dim cryptoTransform As Object
        Select Case encType
            Case "DES"
                tdesProvider = New DESCryptoServiceProvider

                cryptoTransform = tdesProvider.CreateEncryptor(Me.key,
Me.iv)
            Case "3DES"
                tdesProvider = New TripleDESCryptoServiceProvider
                cryptoTransform =
tdesProvider.CreateEncryptor(Me.key3DES, Me.iv)
        End Select


        Dim encryptedStream As MemoryStream = New MemoryStream
        Dim cryptStream As CryptoStream = New
CryptoStream(encryptedStream, cryptoTransform, CryptoStreamMode.Write)
```

```
        cryptStream.Write(inputInBytes, 0, inputInBytes.Length)
        cryptStream.FlushFinalBlock()
        encryptedStream.Position = 0

        Dim result(encryptedStream.Length - 1) As Byte
        encryptedStream.Read(result, 0, encryptedStream.Length)
        cryptStream.Close()
        MsgBox(result)
        Return result
    End Function

    Public Function Decrypt(ByVal inputInBytes() As Byte, ByVal decType
As String) As String
        Dim utf8encoder As UTF8Encoding = New UTF8Encoding

        Dim tdesProvider As Object
        Dim cryptoTransform As Object
        Select Case decType
            Case "DES"
                tdesProvider = New DESCryptoServiceProvider

                cryptoTransform = tdesProvider.CreateDecryptor(Me.key,
Me.iv)
            Case "3DES"
                tdesProvider = New TripleDESCryptoServiceProvider
                cryptoTransform =
tdesProvider.CreateDecryptor(Me.key3DES, Me.iv)
        End Select

        Dim decryptedStream As MemoryStream = New MemoryStream
        Dim cryptStream As CryptoStream = New
CryptoStream(decryptedStream, cryptoTransform, CryptoStreamMode.Write)
        cryptStream.Write(inputInBytes, 0, inputInBytes.Length)
        'cryptStream.Flush()
        cryptStream.FlushFinalBlock()
        decryptedStream.Position = 0


        Dim result(decryptedStream.Length - 1) As Byte
        decryptedStream.Read(result, 0, decryptedStream.Length)
        cryptStream.Close()
        Dim myutf As UTF8Encoding = New UTF8Encoding
        Return myutf.GetString(result)
    End Function

    Public Function EncryptRSA(ByVal plainText As String, ByVal key As
String) As ArrayList
        Dim rsa As New RSACryptoServiceProvider
        rsa.UseMachineKeyStore = True
        Dim EncryptedStrAsByt() As Byte
        Dim encrAl As New ArrayList
        Dim maxLimit As Integer = 58
        rsa.FromXmlString(key)
        Dim strs() As String
        If (plainText.Length > maxLimit) Then

            Dim splitText As String
```

```
            For counter As Integer = 0 To plainText.Length
                If (plainText.Length - counter < maxLimit) Then
                    splitText = plainText.Substring(counter,
plainText.Length - counter)
                Else
                    splitText = plainText.Substring(counter, maxLimit)
                End If
                EncryptedStrAsByt =
rsa.Encrypt(System.Text.Encoding.Unicode.GetBytes(splitText), False)
                encrAl.Add(EncryptedStrAsByt)
                counter = counter + (maxLimit - 1)
            Next

        Else
            EncryptedStrAsByt =
rsa.Encrypt(System.Text.Encoding.Unicode.GetBytes(plainText), False)
            encrAl.Add(EncryptedStrAsByt)
        End If

        Return encrAl
    End Function

    Public Function DecryptRSA(ByVal inputBytes() As Byte, ByVal key As
String) As Byte()
        Dim rsa As New RSACryptoServiceProvider
        rsa.UseMachineKeyStore = True
        rsa.FromXmlString(key)

        Dim DecryptedStrAsByte() As Byte = rsa.Decrypt(inputBytes,
False)
        publicKey = String.Empty
        privateKey = String.Empty
        Return DecryptedStrAsByte

    End Function

    Public Shared Function Encrypt(ByVal plainText As String, _
                                   ByVal passPhrase As String, _
                                   ByVal saltValue As String, _
                                   ByVal hashAlgorithm As String, _
                                   ByVal passwordIterations As Integer,
_
                                   ByVal initVector As String, _
                                   ByVal keySize As Integer) _
                           As String
        Dim initVectorBytes As Byte() =
Encoding.ASCII.GetBytes(initVector)

        Dim saltValueBytes As Byte() =
Encoding.ASCII.GetBytes(saltValue)
        Dim plainTextBytes As Byte() =
Encoding.UTF8.GetBytes(plainText)

        Dim password As PasswordDeriveBytes = New
PasswordDeriveBytes(passPhrase, _
                                            saltValueBytes, _
                                            hashAlgorithm, _
```

```
                                            passwordIterations)

        Dim keyBytes As Byte() = password.GetBytes(keySize / 8)

        Dim symmetricKey As RijndaelManaged = New RijndaelManaged

        symmetricKey.Mode = CipherMode.CBC
        symmetricKey.Padding = PaddingMode.PKCS7

        Dim encryptor As ICryptoTransform =
symmetricKey.CreateEncryptor(keyBytes, initVectorBytes)
        Dim memoryStream As MemoryStream = New MemoryStream


        Dim cryptoStream As CryptoStream = New
CryptoStream(memoryStream, _
                                        encryptor, _
                                        CryptoStreamMode.Write)

        cryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length)

        cryptoStream.FlushFinalBlock()

        Dim cipherTextBytes As Byte() = memoryStream.ToArray()
        memoryStream.Close()
        cryptoStream.Close()

        Dim cipherText As String =
Convert.ToBase64String(cipherTextBytes)
        Encrypt = cipherText
    End Function

    Public Shared Function Decrypt(ByVal cipherText As String, _
                                   ByVal passPhrase As String, _
                                   ByVal saltValue As String, _
                                   ByVal hashAlgorithm As String, _
                                   ByVal passwordIterations As Integer,
_
                                   ByVal initVector As String, _
                                   ByVal keySize As Integer) _
                               As String
        Dim initVectorBytes As Byte() =
Encoding.ASCII.GetBytes(initVector)

        Dim saltValueBytes As Byte() =
Encoding.ASCII.GetBytes(saltValue)
        Dim cipherTextBytes As Byte() =
Convert.FromBase64String(cipherText)
        Dim password As PasswordDeriveBytes = New
PasswordDeriveBytes(passPhrase, _
                                            saltValueBytes, _
                                            hashAlgorithm, _
                                            passwordIterations)
        Dim keyBytes As Byte() = password.GetBytes(keySize / 8)

        Dim symmetricKey As RijndaelManaged = New RijndaelManaged
```

```
        symmetricKey.Mode = CipherMode.CBC
        symmetricKey.Padding = PaddingMode.PKCS7

        Dim decryptor As ICryptoTransform =
symmetricKey.CreateDecryptor(keyBytes, initVectorBytes)

        Dim memoryStream As MemoryStream = New
MemoryStream(cipherTextBytes)

        Dim cryptoStream As CryptoStream = New
CryptoStream(memoryStream, _
                                        decryptor, _
                                        CryptoStreamMode.Read)
        Dim plainTextBytes As Byte()
        ReDim plainTextBytes(cipherTextBytes.Length)

        Dim decryptedByteCount As Integer =
cryptoStream.Read(plainTextBytes, _
                                                0, _
                                                plainTextBytes.Length)

        memoryStream.Close()
        cryptoStream.Close()


        Dim plainText As String =
Encoding.UTF8.GetString(plainTextBytes, _
                                            0, _
                                            decryptedByteCount)
        Decrypt = plainText
    End Function

    Public Sub GetKeysForRSA(ByRef pubKey As String, ByRef priKey As
String)
        Try
            Dim rsa As New RSACryptoServiceProvider
            rsa.UseMachineKeyStore = True
            pubKey = rsa.ToXmlString(False)
            priKey = rsa.ToXmlString(True)
            rsa = Nothing
        Catch ex As Exception
            Throw ex
        End Try
    End Sub

End Class
```

References

- Aimeur, E., Gambs, S., and Ho, A. (2010). Towards a privacy-enhanced social networking site. *Proceedings of ARES*, 172-179.

- Bellare, M., Boldyreva, A. and Adam O'Neill, A., (2007). Deterministic and efficiently searchable encryption, *CRYPTO '07 Proceedings. Lecture Notes in Computer Science*, 4622, 535–552.

- Bellovin, M. and Rescorla, K. (2005). Deploying. *Deploying a new hash algorithm*. Technical Report CUCS-036-05, Dept. of Computer Science, Columbia University.

- Bellovin, M. and Cheswick, W. (2004). *Privacy-enhanced searches using encrypted bloom filters.* Columbia University, 1-16, Technical Report CUCS-034-07.

- Bellovin, M. and Rescorla, K. (2007). *Privacy-enhanced searches using encrypted bloom filters*, Technical Report CUCS-034-07.

- Bonch, D. Dicrescenzo, G. (2004). *Public key encryption with keyword search*, Published by Stanford University, Stanford, USA.

- Curtmola, R., Garay, J., Kamara, S. and Ostrovsky, R. (2005). Searchable symmetric encryption: improved definitions and efficient constructions. Published by Department of Computer Science, Johns Hopkins University, USA.

- Caralli, R. and Wilson, W. (2004). The challenges of security management. Networked systems survivability program, SEI. [cited 2007 12th March] .

- Davis, T. (2003) *RSA Encryption***,** published by Hill, NY.

- Doumen, J. Brinkman, R. Feng, L. Hartel, P.H. and Jonker, W. (2004). Efficient tree search in encrypted data. University of Twente, Enschede, the Netherlands, 1-10.

- Fisher, D. RSA (2010). Experts expect several ciphers to be cracked soon. [online] available: http://threatpost.com/en_us/blogs/experts-expect-several-ciphers-be-cracked-soon-030210 [accessed September 16, 2010] .

- FIPs. (1995). Secure hash standard, Federal Information Processing Standards Publication.

- Gou, C. Zhao, R. and Diao, J. (2010). A load-balancing scheme based on bloom filters, IEEE. Issue Date: 22-24 Jan. 2010, Future Networks, 2010. *ICFN '10. Second International Conference*, 404 – 407.

- Kammuller, F. and Kammuller, R. (2009). Enhancing privacy implementations of database enquiries. Issue Date: 24-28 May. 2009, Venice, Mestre, Italy.

- Kleinjung, T. Aoki, K. Franke, J. Lenstra, A. K. Thomé, E. Bos, J. W. Gaudry, P. Kruppa, A. Montgomery, P.L. Osvik, D.A. te Riele, H. Timofeev, A. Zimmermann, P.(2010). *Factorization of a 768-bit RSA modulus*.

- Leeuw, K.M. and Jan Bergstra, J. (2007). *The History of Information Security: A Comprehensive Handbook*, Elsevier Science, 251-284.

- Li, J. Krohn, M. Mazi`eres, D. and Shasha, D. (2004). Secure untrusted data repository (SUNDR). USENIX Association. NYU Department of Computer Science, 1-15 .

- Needham, R. and Schroeder, M. (1978). Using encryption for authentication in large networks of computers, *Communications of the ACM*, *21*(12).

- Needham, R. and Schroeder, M. (1987). Authentication revisited, *ACM Operating Systems Review*, *21*(1).

- Pesante, L. (2008) Introduction to information security, Carnegie Mellon University, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 1-3.

- Prince, A. (2002). *Murach's Beginning Visual Basic .NET*. (1st edition), Mike Murach & Associates.

- Rivest, R., Shamir, A. and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM, 21*(2), 120-126.

- SANS Institute, (2001). *History of encryption: Version 2*, published by: SANS.

- Schneier, B. (1996). *Applied cryptography: protocols, algorithms, and source code in C*, (2nd edition), NJ: John Wiley & Sons .

- Schneier, B. (2005) *Managed security monitoring: Network security for the 21st century*, NJ: John Wiley & Sons .

- Shiraki, T. Teranishi, Y. Takeuchi, S. Harumoto, K. and Nishio, S. (2009). ***A*** *Bloom filter-based user search method based on movement records for P2P network*, IEEE, National Institute of Information and Communications Technology, 1- 4.

- Song, D., Wagner, D. and Perrig, A. (2000). Practical techniques for searches on encrypted data, *in Proc. 2000 IEEE Symp. On Security and Privacy (SP '00)*, Los Alamitos, CA: IEEE Computer Society, 2000, 44-55.

- Stamp, M. (2006). *Information Security.* NJ: John Wiley and Sons.

- Willis, T. (2004). *Beginning VB.NET Databases*. (1st edition), Wrox Press.