Chaavi: A Privacy Preserving architecture for Webmail Systems

Karthick Ramachandran, Hanan Lutfiyya and Mark Perry

Department of Computer Science
University of Western Ontario
London, Ontario, Canada
Email: {kramach, hanan, markp}@csd.uwo.ca

***Abstract*—The last two decades have seen major innovations in the Internet and transformation of the way people do business, communicate and live. Concomitant with the Internet bringing the advantages of new services, is a growing awareness of threats to Privacy that the Internet can enable. When considered in this context, the Cloud Computing paradigm requires users forgive disturbing levels of trust by users in the servers that hold their information. There is a pressing need for innovative architectures to allow the user to rely on the server with little or no need for trust in the service provider. In this work, we give an introduction of privacy issues in Cloud Computing and discuss the state of art in the privacy enhancing technologies that can be used for Cloud Computing. We focus on webmail services and propose a privacy preserving architecture in which users can retain their mail in the servers of their service providers in a cloud without compromising functionality or privacy. We benchmark our system and present the results showing that it is feasible to architect a privacy preserving solution for webmail systems.**

***Keywords*-privacy-preserving; webmail; encrypted search.**

I. Introduction

Cloud Computing is a model of computing in which the users can rent infrastructure, platform or software services from other vendors without requiring the physical access to the rented service [18]. There are three main types of cloud offerings: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IaaS offers virtualized instances of bare machines leaving the installation and customization of softwares including the Operating System to cloud computing customers. In PaaS, an application framework is provided to the customers for developers to develop their software with. A SaaS provider offers a particular application as a web service, which customers can customize to their needs. The Cloud Service Provider (CSP) focuses on infrastructure and software expertise and aims to optimize their utility by providing centralized services for one or many clients. The benefit to the cloud service client (CSC) is that the cost associated with the underlying infrastructure and software services needed to support the CSC's application is reduced. There are two reasons for the cost reduction. One reason is that the underlying infrastructure and software services are shared among CSCs. The second reason is that since a CSP manages data, it can use creative business models like Contextual Advertising Model [16] for generating revenue by delivering advertisements to users based on the data. For example, webmail services such as Google can provide Gmail for free. As a result, Cloud Computing has been widely adopted. MarketsandMarkets [17] estimates that the cloud computing global market will increase from $12.1 billion (US) to $37.8 billion (US) in 2015 at a compound annual growth rate of 26.2 percent.

In spite of this widespread adoption, organizations are still wary of storing their sensitive data with a CSP. Privacy risk remains a major concern in the cloud computing environment [11].

The definition of privacy that we use was defined by Warren et al. [23] in 1890. Warren et al. described privacy as the ”right to be let alone” with the focus on protecting individuals and is recognized in Convention for the Protection of Human Rights and Fundamental Freedoms. There are a variety of ways that the privacy of data can be compromised in a cloud service environment [4]. This includes the following:

1) Sharing of data with an unauthorized party: The Cloud provider could compromise the confidentiality of the data by sharing the data that it stores with unauthorized parties. This can go against the terms and conditions of the service and will qualify as a breach of security and contract. The end user may never be aware of such a breach.

2) Corruption of data stored: The Cloud Computing provider's root access to physical machines allows the Cloud Provider to have access that allows the Cloud Provider to modify/delete data. The Cloud Provider could tamper with the data making the data non-usable or modify the data in a way that system cannot detect the modification. This poses a serious threat to the integrity of the application.

3) Malicious Internal Users: The employee of a Cloud Computing Provider who has root access to these physical machines, could access the data and use it for their own advantage.

4) Data Loss or Leakage: When a virtual machine is used in an infrastructure, it poses a variety of security issues [10] which could lead to a compromise of the data. Moreover, when the facility that hosts the user's data is subjected to a natural calamity, it could risk the loss of the user's data.

5) Account or Service Hijacking: Another risk for the Cloud Computing provider is, if the service is hijacked, or the computer is hacked into by an intruder, the hacker will have access to data.

This work focuses on the following threats: (a) Sharing with an unauthorized party, b) Malicious internal users, and c) Account or service hijacking. Our work applies to the class of cloud services that stores data and provide searching as its primary functionality. This includes services such as webmail, collaborative document authoring (Google documents) and private blogs. The example used throughout this paper is webmail.

We proposed Chaavi, a webmail infrastructure that builds on the public/private key model to encrypt email with a custom implementation of encrypted indices for keyword searches using the server's infrastructure. Chaavi is the first system that addresses the above threats in a real working environment.

The rest of paper is organized as following. A motivating example of webmail services is described in Section II. Section III presents some of state of the art in preserving privacy for cloud computing services. Section IV reviews background and related work for searching on encrypted data. Section V presents the architecture of Chaavi system. The implementation details are discussed in Section VI. Section VII presents the experiments conducted to study the system and we conclude by stating our contribution and future work in Section VIII.

II. Motivating Example: Webmail Services

Webmail services offer user convenience. With a username, password, and Internet access users, are not tied to any particular equipment or location. Webmail services primarily offer the following functionality:

1) Mail Storage
2) Organization of mail
3) Keyword Searching

For (1) and (2), the service provider need not know the exact content of the mail. However, for performing a plaintext keyword search on email the user needs the service provider to know the content of the mail, so that the cloud provider's infrastructure can be used to index the mail content, which can in turn be used for the search process.

The usage of webmail services, has the following shortcomings:

1) The need to trust the service provider (e.g., Google, Yahoo, or Microsoft) as the mail is stored as plain-text in the service providers' servers (or using single key encryption). The mail is then prone to insider attacks (anyone with the access control will be able to read the mails).
2) There is an assumption that the provider is honest, and the security level is sufficient.
3) When the mail is transferred from one domain to another, it is transmitted through SMTP [19]. SMTP as a protocol does not support encryption. Technologies like Transport Layer Security [9] are used to transfer mail to other domains. However, the data is still protected only up to the layer at which it reaches the target mail server. Once it reaches the target mail server, the mail is again prone to insider attacks in the new domain.

To address such problems, various client encryption systems, such as Pretty Good Privacy (PGP) [26], have been developed. However, encryption using PGP make the mail non-searchable in the web server.

III. Related Work

Privacy Enhancing Technologies (PET) can be used by the developers of the application to enhance the individuals privacy in an application development environment. In this section, we survey state of the art in PET.

Homomorphic Functions: Homomorphic encryption schemes refer to asymmetric encryption techniques, where algebraic operations on plain text can be performed directly on a respective cipher text. This was first introduced by Goldwasser et al. [12], where the authors performed modular addition of two bits using multiplication of ciphertexts (Quadratic Residuosity Problem). The best result so far is a scheme by Boneh et al. [7], where additions are freely performed on encrypted domain. This still remains in the theoretical realm as more advanced abstractions need to be created for using homomorphic functions in practical applications.

Privacy By Secure Computation: The objective of secure computation is to evaluate a function f that takes inputs from two parties A and B without revealing the exact inputs to each other. The Yaos protocol [25] provides some of the basic techniques to perform a computation in a secure way without revealing the inputs. The Yaos protocol forces the expression of a computation problem in terms of logical circuit using gates. The input of each gate is randomly encrypted and then the final resulting output is decrypted to get the exact answer of the computation. The encryption and the decryption is done at the client's end. The expression of a simple problem using the Yaos protocol is found to be complex. Applications that typically reside in the cloud (e.g., mail) are too complex for this.

Privacy By Using Secure CoProcessors: Secure coprocessors are currently the only realistic way to perform general-computing even when an adversary has direct physical access to the server. In our case the adversary could be the cloud service provider itself. It is a very limited computer with ROM, RAM and battery backup for persistent storage and an ethernet card. When installed in a computer, coprocessors can be seen as a secure area inside a computer, which even the main processor cannot access. Privacy as a Service [13] recognizes these factors and proposes a system architecture in which a coprocessor is installed in every Cloud Computing system. The data loaded into the cloud is classified based on its significance and security by the cloud user (No Privacy, Privacy with Trusted Provider, Privacy with Non-Trusted Provider). The data tagged with Privacy with Non-Trusted Provider level is processed by the secure co processor. Secure co-processors needs a separate hardware installation in each server. Also co-processors are expensive and are not yet economical to be used in a cloud computing environment.

Privacy By Encryption: Privacy can be enforced by encrypting all the data that is stored in the cloud. The main issue is that the cloud can be only used for storage of the data. As the data will be unrecognizable to the cloud service provider, it will not be possible for the cloud service provider to process the data nor to perform some number crunching tasks. Searchable encryption uses an algorithm which allows users to encrypt the data and then provides the server with trapdoor information [6], so that the server can search for a given string through the searchable encryption algorithm. This part is discussed in detail in Section IV-C.

Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data [8] proposes a new encryption scheme for keyword search over encrypted data in cloud computing environment with privacy and performance requirements.

In our work we achieve privacy by encryption by using searchable encryption scheme for a webmail software. Our focus is to study how this the encryption schemes can be engineered in a real working environment.

IV. Background

In this section, we review the basic elements common to webmail infrastructures. We also present an introduction to PGP and searchable encryption.

A. Mail Architecture

The webmail infrastructure is responsible for end to end delivery of email. Figure 1 presents architectural components and protocols typically used to support webmail applications.

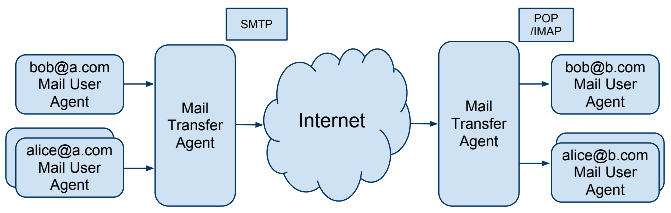
SMTP
POP /IMAP
bob@a.com Mail User Agent
alice@a.com Mail User Agent
Mail Transfer Agent
Internet
Mail Transfer Agent
bob@b.com Mail User Agent
alice@b.com Mail User Agent

Figure 1. Email Architecture

1) Components: This subsection describes the architectural components.

Mail User Agent: The Mail User Agent (MUA) is used to manage a user's email. It acts on behalf of the user to send and receive mail from the Mail Transfer Agent (MTA). Popular MUAs include Microsoft Outlook, Mozilla Thunderbird, Apple Mail. In a webmail system, the MUA runs in the server and the pages are rendered as HTML pages for the browser.

Mail Transfer Agent: The Mail Transfer Agent (MTA) transfers messages from one server to another. It receives email either from another MTA or MUA. The transmission of email follows standardized protocols for message transfers.

2) Protocols: This subsection describes commonly used protocols.

Simple Mail Transfer Protocol (SMTP): SMTP refers to the standard for the transfer of messages from one server to another. It is used by MUA to relay mail through MTA and it is also used by MTA to send and receive mail between other MTAs. SMTP as a standard does not encrypt messages (unless Transport Layer Security encryption is used).

Post Office Protocol (POP) / Internet Mail Access Protocol (IMAP): POP/IMAP are email retrieval protocols that specify standards for downloading messages from the MTA for MUA. Examples of use is found with support for POP version 3 and IMAP as provided by Gmail.

3) Privacy Threats: In webmail systems, there is a server for webmail introduced into the standard mail system (Figure 1). It acts as the Mail User Agent for a number of users and manages email for all the users. The MUA, unlike the standard model (Figure 1), is centralized at the server. The webmail server uses POP/IMAP to download messages from MTA.

There are several privacy concerns with respect to email systems. If the connection to the webmail server is not secured using Hypertext Transfer Protocol Secure (HTTPS) all the data between a user's browser and the server will be in plain text. SMTP, unless used with Transport Layer Security (TLS) layer, is insecure. Even if the TLS layer is used, the mail will still be accessible by the owner of the MTA, through which the mail is routed. This is because TLS is designed to protect data in an insecure network (like Internet) and not from the communicating parties. Some of the security threats involved in email systems are identified by Kangas et al. [14], and Kaufman et al. [15]. These are detailed below.

Eavesdropping: When email is unencrypted, potential hackers who have access to network packets flowing through the network will be able to read the email sent. This can be achieved by enabling the promiscuous mode on ethernet cards.

Identity Theft: If the user's username and password is obtained, then hackers have full access to all the email content. Such password information can be obtained by eavesdropping on the network.

Invasion of Privacy: The recipient of the mail is able to get more information from the email header information than what the sender intends to reveal. For example, the header will reveal the sender's SMTP IP address and subject of the email sent.

Message Modification: Anyone who has administrator access to the webmail server can modify the messages stored in the server. It is not always possible for a recipient to determine that email has been tampered with.

False Messages: It is relatively easy to create false messages and send it as if it is from any person (as evidenced by spam).

Message Replay: Akin to message modification, the message created by user can be saved and sent again and again.

Unprotected Backups: Messages are stored in plain-text on SMTP servers, and backups will also contain complete copies of the messages. Even when the user deletes a message from the server, the backup will still hold the content.

Repudiation: As email messages can be forged (for example see your spam box), there is no way of validating that the email has been in-fact sent by a particular person. This has serious implications in business communications, electronic commerce.

B. Pretty Good Privacy

PGP was created by Zimmermann et al. [26], in 1991 to address the security issues with email. PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and public-key cryptography. Each public-key is bound to an email address. It serves as the verification mechanism for the origin of the email. As the email is encrypted using the private key of the user and the encrypted version is sent into the network, it addresses many security issues of the email infrastructure. For webmail systems, software such FireGPG [1] provide a browser extension that implements PGP. As PGP support enhances the security of the email system by encrypting the mails, the mail becomes unreadable by server. Hence the server cannot perform keyword searches on the mail.

C. Searchable Encrypted Data

Public Key Encryption with Keyword Search (PEKS) [6] is one of the seminal works in the area of making encrypted data searchable. The authors of PEKS propose to encrypt the message using the Public-Private key infrastructure. Along with this cipher text a Public-Key Encryption with Keyword Search (PEKS) of each keyword (the words that make up the message) is appended to the final message. To send a message M with keywords W_1, W_2, ... W_m the following information is transmitted to the server:

$$E_{A_{pub}}(M) \parallel PEKS(A_{pub}, W_1) \parallel ... \parallel PEKS(A_{pub}, W_m)$$

where A_{pub} is the public key of the user, $E_{A_{pub}}(M)$ is the encrypted message, $PEKS$ is the function that encrypts the keywords using A_{pub}. To test whether a word W is a part of the message, a user supplies $PEKS(A_{pub}, W)$ along with a trapdoor function T_w to the server, that can test whether $W = W'$ (W' being the keywords that are stored in the encrypted form in the server). If $W \neq W'$ the server learns nothing more about W'.

Public Key Encryption with Keyword Search Revisited [5] identifies some of the issues with the original PEKS and proposed a provably secure algorithm. The authors argue that if in PEKS the server starts learning the trapdoor then there can be a categorization of mail formed just based on the learned trapdoor information. The trapdoor information is the extra information sent to the server along with the encrypted keyword for the server to test for the existence of a keyword.

The authors also identify that in PEKS there is an assumption that the communication channel between the sender and the server is secure. To enable secure communication through insecure channels the authors propose a Secure Channel Free Public Key Encryption with Keyword Search (SCF-PEKS), that uses a server's public-private key pair for communication.

V. Architecture

This section describes the various components of Chaavi. Figure 2 gives the overall architecture of the system.

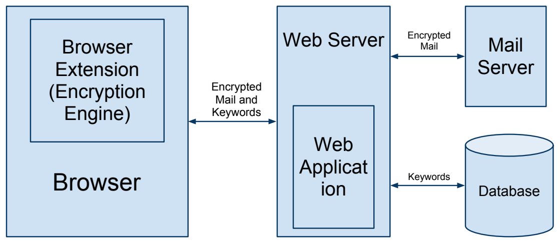

Figure 2. Chaavi - Architecture

A. Browser

The browser is responsible for rendering the pages created by the web application. Its default behavior can be modified or enhanced by using extensions in the browsers. Modern browsers such as Mozilla Firefox, Google Chrome provide functionality to write extensions and install the extensions locally.

B. Browser Extension

A browser extension is used in Chaavi to encrypt the secure message sent to the server. It is also used to decrypt the messages that are sent from the server. Additionally it has key generation and key management functionality. The extension is composed of the following modules.

Public-Private Key Generation: As stated earlier, Chaavi uses a public/private key model for securely communicating messages. In a public/private key model, a public-private key pair is generated when the system is initiated for the first time, for a particular user. The messages encrypted by the public key can be decrypted only by use of the private key. The public key as the name implies is shared in a public forum.

Keyword Encryption Key Generation: Public-Private key pair is used for secure message communication. A symmetric key is also generated to encrypt the individual keywords present in the mail. A symmetric algorithm (unlike the Public-Private key) is used here as the keywords need not be decrypted by anyone else other than the sender of the message.

Key Management: Key management is performed using a graphical user interface (GUI). The GUI enables the user to add or delete the public keys of the recipients with whom the user wants to communicate through mails.

Encryption: The functionality of the encryption module is to encrypt the messages that are sent to the server from the browser. It also extracts and encrypts the individual keywords in the message. The encryption module is triggered from the web application when the user submits a mail to send it to the web server. This module encrypts the message using the recipients's public key and the keywords with the keyword encryption key.

Decryption: When an encrypted message is sent from the server to the browser, the decryption module decrypts the messages using the private key of the user that is generated during system initialization.

C. Web Application

The webmail application provides graphical user interfaces for the users to read, send and search messages. It comprises of both server-side and client-side (browser) functionality.

When a user sends a message from the web application, the Encryption module encrypts the message and extracts and encrypts the keywords. The web application sends the encrypted message and keywords to the web server. On receiving the encrypted message and the keywords, at the server-side the application saves the encrypted message alongside the encrypted keywords in a database for future retrieval. The application then transfers the mail to the Mail Server (SMTP server) for the mail to be be delivered to recipient.

When the user wants to search for a particular keyword in their inbox, the encrypted keyword is sent to the server-side. The web application then searches for the mails corresponding to that particular encrypted word and then sends the encrypted mails back to the user.

D. Database

The mail storage and organizational functionality is already handled by the web application. One custom table, $search$ is added to the database which stores the $< message\_id, encrypted\_keyword >$ pair. This database is looked up when the user performs a keyword search.

E. Mail Server

The mail server sends and receives email communicated to it through the Internet. The mail server functionality is not modified by our system. The web application communicates with the mail server to send and receive messages.

VI. Implementation

The following software is used to implement the different components in the system:

- Browser - Google Chrome
- Browser Extension - Google Chrome using Javascript
- RSA encryption/decryption library from hanewin.net [3]
- AES encryption library [2]
- Web Application - Squirrelmail over PHP and MySQL
- Mail Server - Using the POP3 interface of the *csd.uwo.ca* mail server

The implementation details of individual modules of the system are detailed below.

A. Browser Extension

Public-Private Key Generation: The RSA algorithm [20] is used for the creation of keys. The key requires two large prime numbers as the input along with a random seed. All of these inputs are created by the extension randomly and provided as input for key generation. The keys are then stored locally along with the user name, for future retrieval in the local browser database.

AES Key Generation: The symmetric AES key algorithm is used to encrypt the individual keywords present in the mail. The AES key generation algorithm takes as input a random seed which is provided by requesting the user to move the mouse over the browser window. That generates some random co-ordinates which is then used to generate the key.

AES is a natural choice for the symmetric key algorithm as it has been analyzed extensively and used worldwide [24]. However, unlike PEKS [5], AES algorithm does not support trapdoor and hence it is susceptible to chosen plaintext attacks (The attacker has the capability to choose arbitrary plaintext and the corresponding cipher texts). Moreover the encryption of the keywords under AES negates the possibility of performing range searches (e.g., $10 < b < 20$) or similarity searches (name staring with ‘ka’).

Key Management: The GUI for key management is developed using the options functionality provided by the Chrome extension framework. It is used to insert the public keys of the recipients with whom the user wants to communicate. The private key of the user cannot be managed using this interface (the system automatically generates it when the user logs in for the first time). The keys are stored in the local storage database provided by HTML5.

Encryption: The user is provided with a HTML form from the web application which contains input fields to enter the recipient email address, subject and the contents of the mail. The form submission event ($onsubmit$ event) is associated with a custom submit event handler, which is hooked to the encryption module. The encryption module encrypts the contents of the mail using the user's public key and replaces the value in the field (contents of the mail) with the encrypted message. Along with this, the keywords in the message are extracted by the keyword extraction function and each keyword is encrypted using the AES key and stored in an object. This object is serialized in JSON (Javascript Object Notation) and sent to the server along with the encrypted message.

Decryption: When an encrypted message is sent from the server to the browser the server adds the attribute value $post - deencrypt$ to attribute $class$. The extension identifies these messages and decrypts the messages using the private key of the user. This decrypted message replaces the original encrypted message in the html page so that the user can see the message in the encrypted mail.

B. Web Application

An open source web application (Squirrelmail) is identified and it is modified for our application. Squirrelmail is responsible for storage and organization of the mails. Our custom module is developed in PHP and added to Squirrelmail to save the encrypted messages alongside the encrypted keywords and for the retrieval of the messages based on the given encrypted keyword.

VII. Experiments

The performance of algorithms used in Chaavi (Privacy Preserving Web Mail with Keyword Searches) is studied in terms of space and time consumed by the algorithm in the local client system. Even though the performance of the encryption algorithms has been studied before, we focus on the performance of our system. The results presented in this section are intended to provide some insight on the overhead provided by the algorithms in a browser based extension environment. Since encryption and decryption is performed in the client browser system, the encryption and decryption is independent of the number of users currently using the system. Hence, we focus on the performance of the encryption algorithms for a browser-based extension environment.

All the experiments are executed in a Pentium IV Core 2 Duo processor using Google Chrome 5.0.375.99 beta.

A. Time Complexity

The following algorithms are studied with respect to the execution time.

- Key Generation
- Encryption and Decryption (RSA Algorithm)
- Keyword Encryption (AES Algorithm)

1) Key Generation: Key generation is expensive since it involves finding two large random prime numbers and finding a product of the prime numbers based on the given random seed. The length of keys (as measured by bits) can be of sizes: 128, 256, 512, 1024. The higher the number of bits used, the more difficult it is to break the key (According to Schneier et al. [21], for breaking AES with key size greater than or equal to 256-bit through brute force will require fundamental breakthroughs in physics and understanding of universe). However, generating larger keys is time consuming. We present the average time taken for key generation for different bit sizes in Figure 3.

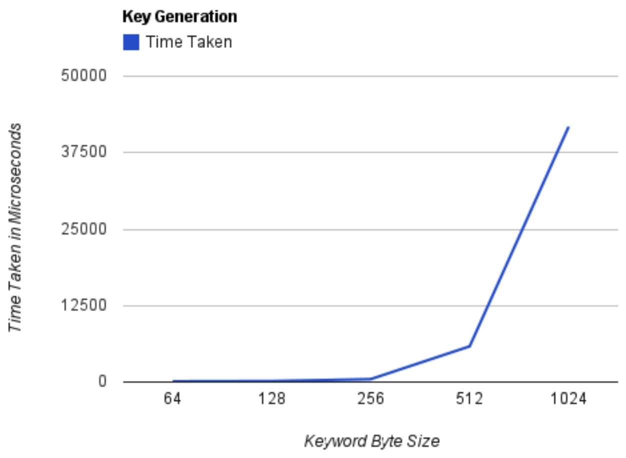

Figure 3. Key Generation

As can be seen the keyword bit size increases the creation time exponentially. The 1024 bit key generation takes around 41 seconds. However, as this is a one time activity (when the user sets up the system) the usability and inconvenience is minimal.

2) Encryption and Decryption: When the user wants to send an email the encryption module is executed each time, and the decryption module is activated when the user wants to read an email. This is a frequent activity and therefore more computation time spent on these modules will impact usability. The encryption and decryption algorithm is run over random data (which represents an email message) set using the Javascript library in Chrome browser. The performance of RSA algorithm is studied here in a browser environment. The following are the results using a 512 bit key.

It can be seen that at a relatively larger message size, around 212 KB, the time taken for encryption and decryption is less than 2 seconds. However as the message size increases in the order of megabytes, the time is around 16 seconds. A 67 MB message takes around 16 seconds to encrypt and 9 seconds to decrypt, which is still acceptable for sending such a large message. Moreover, most webmail systems have a limit of 10 MB on message sizes.

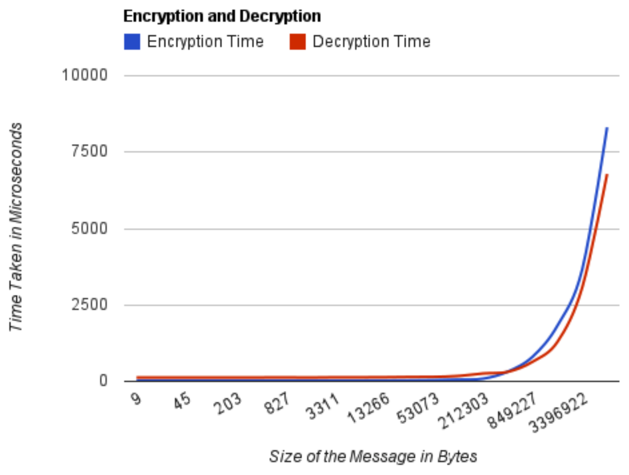

Figure 4. Encryption and Decryption

3) Keyword Encryption: In this phase the performance of AES algorithm is studied. Each word from the message is extracted and is encrypted using the AES algorithm. There is no decryption phase here, as the encrypted words are checked against each other.

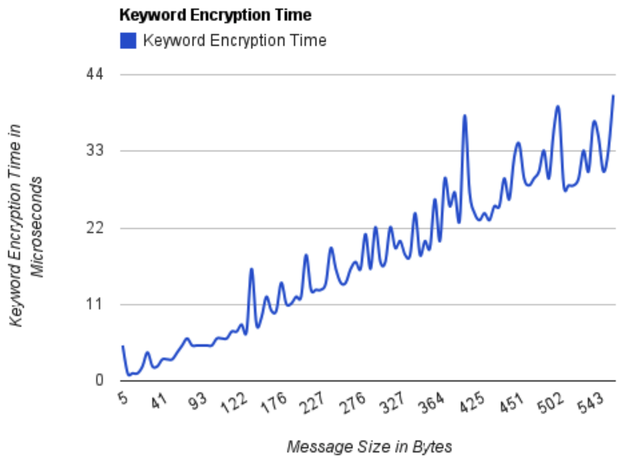

Figure 5. Keyword Encryption Time

It can been seen that there is a linear relationship between the message size and time taken for encrypting keywords. It has to be also noted that when there are duplicate words the encryption is not done twice. However, in these experiments each word was generated at random with a random size (with maximum as 25 bytes). The probability of the same word repeating is very low for this case.

B. Space Complexity

In our study of the space complexity, we were interested in the following:

1) Increase in size of the keyword index
2) Increase in the size of the final mail

1) Impact of increase in size on the keyword index: The AES algorithm is executed over the generated keywords and the impact of the size of the encrypted keywords on execution time is examined. There is close to a 10 times increase in the generated encrypted keywords compared to the keyword's actual size. This can pose a design challenge at the database level on how to store these keywords for efficient lookups at the server level.

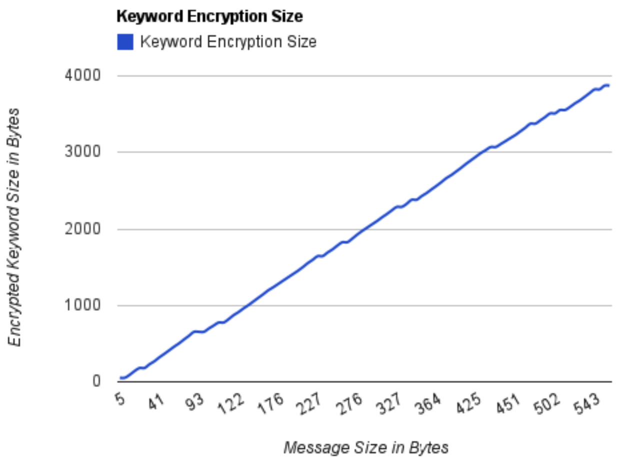

Figure 6. Keyword Encryption Size

2) Impact of increase on Final Message size: Here we study the total increase in the email size. The email that is sent to the server of the recipient will be in this format and the any increase in size, will increase the overall network traffic.

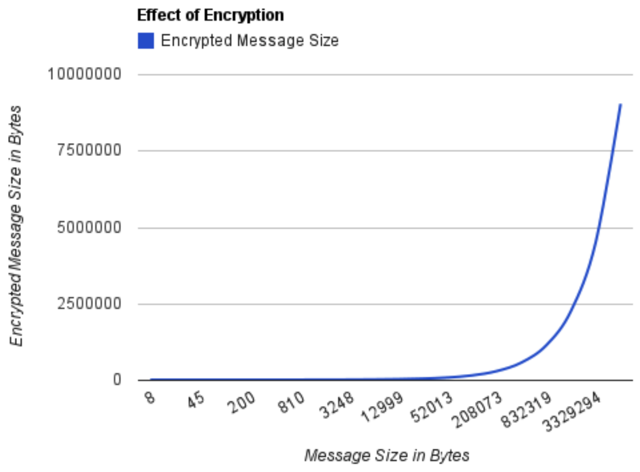

Figure 7. Message Size

It can be seen from the graph (Figure 7) that initially, when the message is transferred, there is not much of an increase in the encrypted message size (8 bytes to 186 bytes, 18 bytes to 199 bytes, 404 bytes to 722 bytes). However as the size increases beyond 4MB there is a steep increase in the difference between the message size and encrypted message (4MB to 5MB, 8MB to 11MB, 66MB to 90MB). On average, there is a 3 times increase in size when encrypted using RSA. This is another major factor that has to be taken into consideration while using this system.

VIII. Conclusion

We proposed a privacy preserving architecture for our webmail system, that enables secure communication of messages using a public/private key model and privacy preserving keyword search functionality using AES key encryption algorithm.

Our approach requires every client to install an extension to their browser and the cloud computing provider to modify their webmail application to support encrypted keyword search. Even though technically this is a possible solution, economically a cloud provider might not prefer this approach. Most of the business models in web application are built around the contextual advertising model, where the cloud provider relies on the user's data to deliver the relevant advertisements to the user. In our case as the data is encrypted in the server, the cloud provider will not have access to the user's data. Works such as Toubiana et al. [22], try to address this problem by offloading the keyword extraction in contextual advertising to the client browser. Approaches like [22] needs to be modified for our architecture so that our system remains economically viable.

Unlike in PEKS [5], our system does not use a trapdoor function. This makes our system more susceptible to chosen plaintext attacks. If a recipient of a mail is also a potential attacker, the recipient can eavesdrop the encrypted keyword information sent from the sender to the server, and make a guess on what keyword represents the encrypted cipher by analyzing a number of mails sent to the recipient (attacker) from the same sender. However, our contribution is the proposal of the framework. The encryption algorithms used can be modified to utilize more secure alternatives in our architecture.

In our performance study, we see a considerable increase in the size of the message and the keywords after encryption. This will have a direct effect in the database storage and the keyword look up time.

We have also not implemented the functionality to add the incoming messages to the encrypted search database. Future work should address this. Future work also involves detailed study on the strength of the encryption, support to range and similarity searches, improvements to the algorithms used whilst maintaining performance.

Acknowledgements

The authors would like to thank the IBM Center of Advanced Studies and NSERC for their funding.

References

[1] http://getfiregpg.org/s/home. Online at 27th June 2011.

[2] http://www.hanewin.net/encrypt/aes/aes.htm. Online at 27th July 2011.

[3] http://www.hanewin.net/encrypt/rsa/rsa.htm. Online at 27th June 2011.

[4] Top threats to cloud computing v1.0. Cloud Security Alliance.

[5] J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. *Computational Science and Its Applications–ICCSA 2008*, pages 1249–1259, 2008.

[6] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology-Eurocrypt 2004*, pages 506–522. Springer, 2004.

[7] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. *Theory of Cryptography*, pages 325–341, 2005.

[8] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data. In *IEEE INFOCOM*, 2011.

[9] T. Dierks. The transport layer security (tls) protocol version 1.2. 2008.

[10] T. Garfinkel and M. Rosenblum. When virtual is harder than real: Security challenges in virtual machine based computing environments. In *Proceedings of the 10th conference on Hot Topics in Operating Systems-Volume 10*, page 20. USENIX Association, 2005.

[11] R. Gellman. Privacy in the clouds: Risks to privacy and confidentiality from cloud computing. In *World Privacy Forum*, pages 1–26, 2009.

[12] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[13] W. Itani, A. Kayssi, and A. Chehab. Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures. In *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 711–716. IEEE, 2009.

[14] E. Kangas and L. President. The Case for Email Security. *Published as a Lux Scientiae Article, available at http://luxsci. com/extranet/articles/email-security. html (accessed 1 May 2007)*, 2004.

[15] L. Kaufman. Data security in the world of cloud computing. *IEEE Security and Privacy*, 7(4):61–64, 2009.

[16] D. Kenny and J. Marshall. Contextual marketing–the real business of the Internet. *Harvard Business Review*, 78(6):119, 2000.

[17] MarketsandMarkets.com. Cloud computing market - global forecast (2010 -2015).

[18] P. Mell and T. Grance. The nist definition of cloud computing. *National Institute of Standards and Technology, Information Technology Laboratory*, Version 15, 10-7-09:2, 2009.

[19] J. Postel. RFC821: Simple mail transfer protocol, 1982.

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

[21] B. Schneier. Snake oil. crypto-gram newsletter (http://www.schneier.com/crypto-gram-9902.htmlsnakeoil) [online on 05th september 2011], February., 1999.

[22] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *17th Annual Network & Distributed System Security Symposium, San Diego, CA, USA*. Citeseer, 2010.

[23] S. Warren and L. Brandeis. The right to privacy. *Harvard Law Review*, pages 193–220, 1890.

[24] H. B. Westlund. Nist reports measurable success of advanced encryption standard - news briefs - national institute of standards and technology - brief article. *Journal of Research of the National Institute of Standards and Technology*, 2002.

[25] A. Yao. Protocols for secure computations. *Proceedings of the 23rd Annual IEEE Symposium on ...*, Jan 1982.

[26] P. Zimmermann. *The official PGP user's guide.* MIT Press, May 1995.