Email Plugin Suite for *k*-Resilient Identity-Based Encryption and Searchable Encryption

Kah-Wai Chew and Swee-Huay Heng(✉)

Faculty of Information Science and Technology, Multimedia University, Melaka, Malaysia
shheng@mmu.edu.my

Abstract. Nowadays, most of the companies use their preferred email platform to perform their daily tasks such as exchanging of messages and sharing of documents. However, it is quite often to hear the news that someone's email account is being hacked. The malicious intention of the hackers is to hijack the users' personal information or company documents. Thus, cryptography plays an important role to provide more a secure email platform. Identity-based encryption (IBE) is a public key cryptosystem that uses a unique identifier such as email address or IP address to generate the user public key and it relies on a trusted third party to generate the corresponding user private key. As compared to conventional public key infrastructure, users now no longer need to distribute their public keys to others. Meanwhile, searching over encrypted emails could be a challenging task. In order to overcome this, the concept of public key encryption with keyword search (PEKS) was introduced to enable searching for the encrypted keywords without revealing anything about the original email. This research focuses on implementing and integrating both IBE and PEKS with encryption and searching functionalities into an email platform via an email plugin. More precisely, this project is to design an email plugin suite using *k*-resilient identity-based encryption and *k*-resilient searchable encryption in order to provide a more secure email platform. The performance analysis shows promising result.

Keywords: *k*-resilient · Identity-based encryption · Searchable encryption · Email plugin

1 Introduction

Due to the impact of the covid-19 pandemic, people nowadays have gradually changed their lifestyles. As the country is being locked down, people have to move everything online. As a result, covid-19 pandemic has accelerated digitisation as we can see that online users have increased dramatically whether it is in teaching, workplace or business. However, because a large number of people are forced to switch online due to the epidemic, they still need time to adapt to the online mode while picking up the new computer knowledge. Therefore, this will cause some ignorant computer users to accidentally expose themselves in the danger of the online world. Hence, some easy-to-use and good defence software should be introduced.

S.-C. Haw and K. Sonai Muthu (Eds.): CITIC 2022, 10, pp. 120–132, 2022.
https://doi.org/10.2991/978-94-6463-094-7_10

Be it online study or working from home, secure, easy and efficient communication are the most important elements to be considered. In most of the cases, email is the official platform used. However, there is always question on whether the email platforms used are secure enough. In particular, it is of utmost importance to ensure that the content of the communication would not be leaked out.

Encryption is definitely the best way to protect the email content. In general, encryption is divided into two types, which are private key encryption and public key encryption. For a private key encryption, the sender and the receiver will share the same secret key. They will use the shared secret key to encrypt and decrypt the message each time they send or receive a message. For a public key encryption, the sender and the receiver will have their own public and private key pair. The private key should be kept by them while the public key will be distributed to the public. Before transmitting a message, the sender will use the receiver's public key to encrypt the message. Once received, the receiver will use his own private key to decrypt the massage. Everyone except the receiver cannot read the message because the they do not know the receiver's private key. Both private and public key encryption complement each other as they have their own pros and cons.

Most of the time, emails are sent and kept in unencrypted format. The popular encryption technologies, such as SSL, can safeguard a communication from eavesdroppers during transmission. However, emails that are stored on the server are still vulnerable. The server stores emails, enables remote access and also enables easy backups for the email users. Thus, the server must secure the email content. However, once the server is compromised, the server entails a compromise of the users' emails, which the users cannot simply avoid. While public key encryption such as PGP can protect email content, this is not a feasible solution on its own because it keeps the email headers so that the messages can be sent correctly. As a result, an attacker can violate a user's privacy to some extent by discovering who the user is sending to. More crucially, PGP requires that the correspondents use the programme too but then public key encryption is not widely used [1].

Meanwhile, the concept of identity-based cryptography was introduced by Shamir [2] in 1984 to simplify certificate management. It is another public key cryptosystem that uses a unique identifier such as email address or IP address to generate the user public key and it relies on a trusted third-party to generate the corresponding user private key. As compared to the conventional public key infrastructure, identity-based users now no longer need to distribute their public keys to others because they public keys can be generated from the unique identifiers. This is in a way more secure because if there are only a finite number of users, then the third party's master secret can be destroyed once all users have been issued with their private keys. Thus, identity-based encryption (IBE) is recommended to be implemented in the email platform as email address can be used as a unique identifier. Furthermore, IBE encrypts all parts of the email including header, body and attachment to make sure the email is protected until it is received and decrypted by the corresponding user.

However, there is always a trade-off for every security system. The user convenience will always be sacrificed for a more secure system. Once the email content is encrypted, users can no longer simply click and view the original content of the email. They are required to download and decrypt the email prior to read the original content. Thus, this

will bring inconvenience to the users since they have to look for the email they need without previewing the email content. In order to overcome this problem, the concept of public key encryption with keyword search (PEKS) was introduced by Boneh et al. [3]. PEKS allows users to search for encrypted keywords without revealing anything about the original email.

Although most of the email platforms have their own security scheme, some of the open source email platforms enable user to add on or plug in their own software. A plugin is a software component that can add on to an existing computer programme to add more features to it. User may write their own security scheme and plug into the email platform or download available plugin and install on the email platform. Users can then use it whenever they want to encrypt and send an email.

This research focuses on implementing and integrating both IBE and PEKS, with encryption and searching functionalities, into an email platform as an email plugin. More precisely, this project is to design an email plugin suite using *k*-resilient identity-based encryption [and *k*-resilient searchable encryption to provide a more secure email platform.

The *k*-resilient IBE proposed by Heng and Kurosawa [4], and the *k*-resilient PEKS proposed by Yau et al. [5] are chosen due to that the schemes are provably secure in the standard computational model. The schemes are also efficient as they can be implemented in the finite fields. Moreover, the security of the schemes is based on the intractability of the decisional Diffie-Hellman (DDH) problem, which is a well-studied problem.

2 Preliminaries

This section provides some preliminaries on IBE, PEKS and email plugin which will be used as the underlying building blocks in constructing a secure and efficient email plugin suite.

2.1 Identity-Based Encryption

IBE allows users to generate public key from a unique identifier such as their email address and the corresponding private key is generated by the trusted third party called private key generator (PKG). The master secret key will be kept by the PKG and the master public key will be distributed. Therefore, the communicating parties can encrypt messages (or verify signatures) without having to distribute keys among the various participants in advance. However, key escrow or key recovery problem is inherent due to the dependence on the PKG who knows all user private keys and this somehow limits its usage to closed group applications or corporate use. This key escrow feature may be desirable in circumstances where an audit trail is required, for instance, in monitoring employee activities or in resolving disputes.

In an IBE scheme, there are four polynomially bounded algorithms [4].

Setup: A probabilistic approach to build up all of the scheme's parameters used by the PKG. The Setup algorithm receives a security parameter and outputs the global system parameters and a master key. The system parameters will be made public, but only the PKG will have access to the master key.

Extract: A probabilistic approach that uses public identity to extract the corresponding private key. The Extract algorithm receives the master key and the user public identity and outputs the user private key.

Encrypt: A probabilistic approach that uses a public identity to encrypt a message. The Encrypt algorithm receives a message, a public identity, and the system parameters and outputs the ciphertext.

Decrypt: A deterministic approach that receives the ciphertext, the user private key and the system parameters and outputs the message.

2.2 Public Key Encryption with Keyword Search

PEKS is a public key cryptosystem which enables users to search for a specific keyword in the email without learning other things about the email. Mobile device is an indispensable device for everyone nowadays and it is convenient to carry along. However, mobile devices have a smaller storage as compared to a laptop or desktop computer. Therefore, PEKS allows users to search for a keyword such as "urgent" from the encrypted emails so that the users do not need to download all emails and waste the capacity of the devices.

By using PEKS, an email gateway is used to test whether the email contains the keyword searched by the user and then routes the email accordingly without having the ability to decrypt all of the messages. PEKS is very useful when the mail server stores many encrypted emails as PEKS can be employed to identify all emails containing the specific keywords. This not only saves storage capacity but also makes searching process very fast and easy.

A PEKS consists of three entities, namely, an email gateway, a sender and a receiver. Firstly, an encrypted email associated with the encrypted keyword w is sent by the sender Bob to the email gateway. Next, the receiver Alice sends a trapdoor associated with the searching keyword w to the email gateway. Then, the email gateway uses the PEKS mechanism to check whether the searching keyword w is the encrypted keyword associated with the email without learning other things about the email. Once authenticated, the email gateway will return the corresponding encrypted email to Alice. Therefore, PEKS can overcome the limitation of secure search over encrypted data.

The following algorithms define a PEKS scheme [5]:

KeyGen(s): It receives a security parameter s, and outputs a public key pk and a private key sk of the user.

PEKS(pk, w): It receives a keyword w and a public key pk, outputs a PEKS ciphertext C. During this process, the sender encrypts the keyword w and associates it with the email.

Trapdoor(sk, w): It receives a keyword w and a private key sk, outputs a trapdoor T_w. During this process, the receiver retrieves the encrypted emails associated with the keyword w from the email gateway.

Test(pk, C, T_w): It receives a public key pk, a PEKS ciphertext $C = PEKS(pk,w')$, and a trapdoor T_w and outputs "1" if $w = w'$.

2.3 Enigmail

Enigmail is a security add-on for Thunderbird, Interlink Mail & News and Postbox that works effortlessly. Enigmail's users can encrypt their emails by using the digital

signature, as well as decrypt the received emails, using OpenPGP. Enigmail is a free programme. The Mozilla Public License of Enigmail allows it to be freely used, modified, and distributed. Enigmail encrypts and signs emails using OpenPGP public keys. Enigmail is compatible with Unix, Windows, and also Mac operating system. Moreover, other email clients that support PGP/Inline and PGP/MIME can run Enigmail too. Furthermore, GNU Privacy Guard is in charge of the encryption.

Thunderbird and SeaMonkey, by default, use S/MIME for email encryption and signing, which is based on X.509 keys. Enigmail introduces a new system in which cooperative people can use a key issued by a trusted network. This approach relies on numerous people to verify the sender and receiver credentials' legitimacy. In theory, this improves security because it eliminates the reliance on centralised entities that could be hacked owing to security flaws.

Both of GNU Privacy Guard and Enigmail are open-source software and free for use. The most popular PGP setup is presently Enigmail with Thunderbird. However, Thunderbird's developers stated in October 2019 that the built-in encryption and signature capabilities of Enigmail add-on will be replaced by OpenPGP Thunderbird 78. It is because the coding of Thunderbird has changed and no longer support the legacy add-ons. Because this necessitated a complete rewriting of Enigmail, Patrick Brunschwig opted to support the Thunderbird team's native implementation [6].

There are several main features of Enigmail. Firstly, it supports for OpenPGP Interlink Mail & News and Postbox. Besides, up to version 68 of Mozilla Thunderbird is supported. Next, when sending an email, it encrypts and signs the email; when receiving an email, it decrypts and verifies the email. Furthermore, PGP/MIME (RFC 3156) and inline-PGP are supported for read only. Through Autocrypt, it is possible to discover and exchange keys automatically. Moreover, encryption and signing can be enabled or disabled automatically for each recipient. Then, the next feature of Enigmail is encrypting or signing attachments to inline PGP messages automatically. Lastly, Mozilla's Multiple Identities functionality is also supported.

Due to that it is an extension, users will not have to download anything to their computer, making the installation process much simpler. It allows users to generate encryption for each message they send and receive, ensuring that they cannot be altered by third parties throughout their journey. It has a highly advanced security system, which allows it to deactivate the HTML service because many assaults can occur through it, which is why emails using HTML are normally significantly more vulnerable. It can be configured to work with mail servers like Yahoo, Gmail, and Outlook. Besides, it aids in the generation of public and private keys. It provides users with the highest level of privacy and security when transferring data over the Internet. Currently, network communications have become a target for attackers. By using Enigmail, it can assist users in blocking them by ensuring the integrity of each of the web-based messages.

3 Proposed Email Plugin Suite

3.1 System Design

Figure 1 shows the process flow for sending email. Once user inputs the receiver email address, email header and the message and clicks on the encrypt button, the *Setup*,

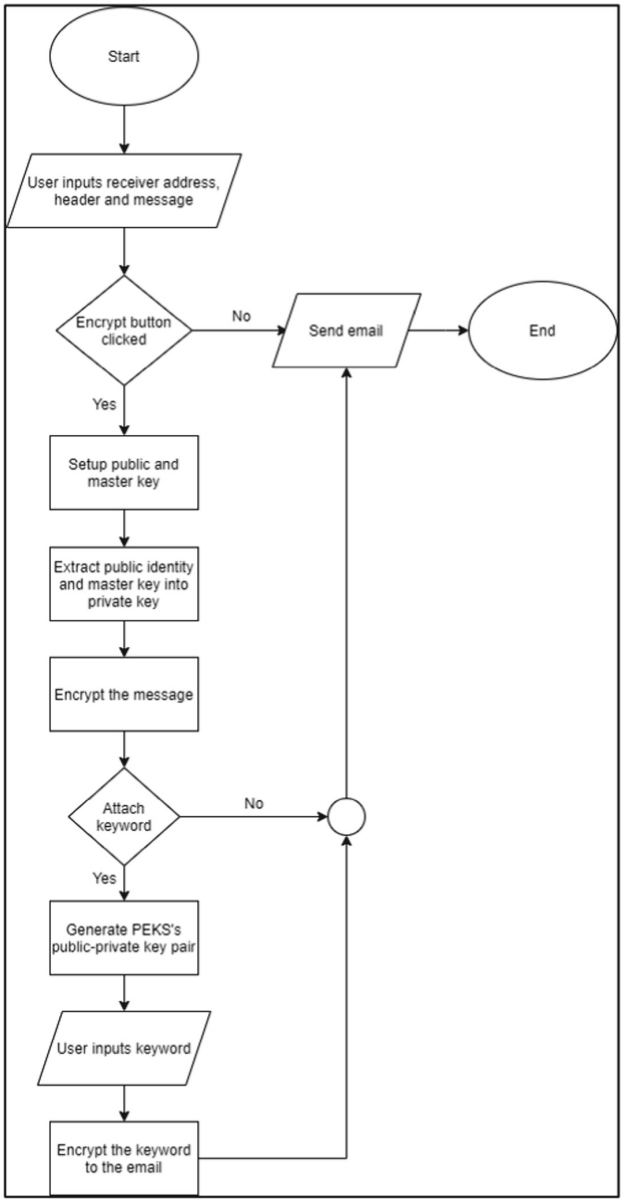

Fig. 1. Flowchart for sending email

Extract and *Encrypt* algorithms of the IBE will be run. Then, user can choose whether to associate a keyword to the email before sending. If the user decides to associate a keyword, then the *Keygen* and *Encrypt* algorithms of the PEKS will be run.

Figure 2 shows the process flow for receiving email. Once user receives email, the email will be stored together with other emails. Hence, user may use the search function

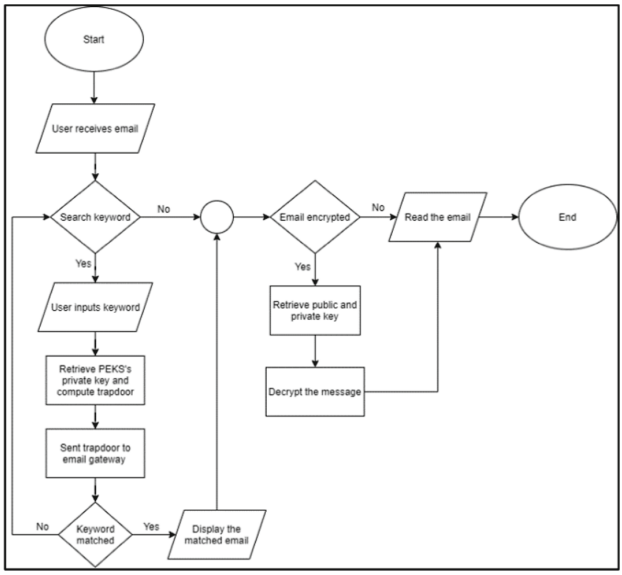

Fig. 2. Flowchart for receiving email

to search a keyword that associated to the email he wanted to look for. After keying in the keyword and pressing the search button, the *Trapdoor* and *Test* algorithms of the PEKS will be run. If a matching is found, then the searched email is returned to the user, else no results is found by the user. Next, in order to decrypt an email, the *Decrypt* algorithm of the IBE will be run so that user is able to retrieve the original message.

4 Implementation

4.1 k-Resilient IBE

The k-resilient IBE scheme is provably secure in the standard computational model which is a weaker model as compared to the random oracle model. It achieves the most desirable security notion for an encryption scheme which is the chosen ciphertext security.

The k-resilient IBE consists of the following algorithms [4].

Setup: Firstly, a random multiplicative group $G \subset Z_p^*$ of prime order q and two generators $g_1, g_2 \in G$ are selected. Next, choose six random k-degree polynomials over Z_q, which are $f_1(x) \stackrel{\text{def}}{=} \sum_{t=0}^{k} a_t x^t$, $f_2(x) \stackrel{\text{def}}{=} \sum_{t=0}^{k} a_t^{'} x^t$, $h_1(x) \stackrel{\text{def}}{=} \sum_{t=0}^{k} b_t x^t$, $h_2(x) \stackrel{\text{def}}{=} \sum_{t=0}^{k} b_t^{'} x^t$, $p_1(x) \stackrel{\text{def}}{=} \sum_{t=0}^{k} d_t x^t$ and $p_2(x) \stackrel{\text{def}}{=} \sum_{t=0}^{k} d_t^{'} x^t$. Then, $A_t = g_1^{a_t} g_2^{a_t^{'}}$, $B_t = g_1^{b_t} g_2^{b_t^{'}}$ and $D_t = g_1^{d_t} g_2^{d_t^{'}}$, are computed for $t = 0, \ldots, k$ and a hash function H is chosen at random from the collision-resistant hash functions. Lastly, it outputs the system parameters $params = \langle g_1, g_2, A_0, \ldots, A_k, B_0, \ldots, B_k, D_0, \ldots, D_k, H \rangle$ and master-key $\langle f_1, f_2, h_1, h_2, p_1, p_2 \rangle$ but the master key is only known to the PKG.

Extract: It receives the public identity $ID \in Z_q$ and outputs the user private key $SK_{ID} = \langle f_{1,ID}, f_{2,ID}, h_{1,ID}, h_{2,ID}, p_{1,ID}, p_{2,ID}, \rangle$.

Encrypt: To encrypt a message m:

1. $r_1 \leftarrow_R Z_q$
2. $u_1 \leftarrow g_1^{r_1}$
3. $u_2 \leftarrow g_2^{r_1}$
4. $A_{ID} \leftarrow \prod_{t=0}^{k} A_t^{ID^t}$, $B_{ID} \leftarrow \prod_{t=0}^{k} B_t^{ID^t}$, $D_{ID} \leftarrow \prod_{t=0}^{k} D_t^{ID^t}$
5. $s \leftarrow D_{ID}^{r_1}$
6. $c \leftarrow m \cdot s$
7. $\alpha \leftarrow H(u_1, u_2, c)$
8. $v_{ID} \leftarrow A_{ID}^{r_1} \cdot B_{ID}^{r_1 \alpha}$
9. $C \leftarrow \langle u_1, u_2, c, v_{ID} \rangle$

Decrypt: To decrypt a ciphertext C:

1. $\alpha \leftarrow H(u_1, u_2, c)$
2. Test if $v_{ID} \leftarrow u_1^{f_{1,ID}+h_{1,ID}\alpha} \cdot u_2^{f_{2,ID}+h_{2,ID}\alpha}$, stop if false.
3. $s \leftarrow u_1^{p_{1,ID}} \cdot u_2^{p_{2,ID}}$
4. $m \leftarrow c \cdot s^{-1}$

4.2 k-Resilient PEKS

Similarly, the k-resilient PEKS scheme is provably secure in the standard computational model and it achieves the most desirable security notion for a PEKS scheme which is chosen keyword security.

The k-resilient PEKS consists of the following algorithms [5].

KeyGen(s): To generate the pk and sk key pair:

1. By generator $g_1 \in G$ and order q, a group G is selected. $r \leftarrow_R Z_q$ is chosen and $g_2 = g_1^a$ is computed.
2. Two random k-degree polynomials are selected over Z_q, $p_1(x) = \sum_{t=0}^{k} d_t x^t$, $p_2(x) = \sum_{t=0}^{k} d_t' x^t$.
3. $D_t = g_1^{d_t} g_2^{d_t'}$ is computed for $t = 0, \ldots, k$.
4. $sk = \langle p_1, p_2 \rangle$ And $pk = \langle g_1, g_2, D_0, \ldots, D_k \rangle$ is returned.

PEKS(pk, w): To encrypt a keyword w:

1. $r \leftarrow_R Z_q$ is chosen.

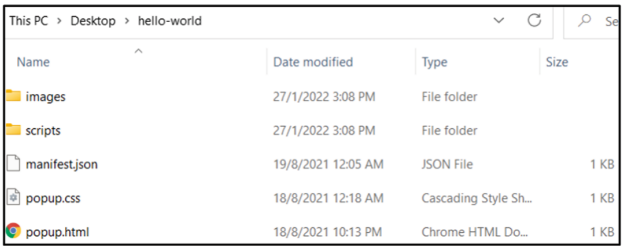

Fig. 3. A simple "*hello-world*" firefox plugin

2. $u_1 = g_1^r$ is computed.
3. $u_2 = g_2^r$ is computed.
4. $S_w = (\prod_{t=0}^{k} D_t^{w^t})^r$ is computed.
5. Ciphertext $C = \langle u_1, u_2, s_w \rangle$ is returned.

Trapdoor(sk, w): It uses a keyword w and a private key sk to compute trapdoor $T_w = \langle p_1(w), p_2(w) \rangle$.

Test(pk, C, T_w): It uses a ciphertext C, a public key pk and a trapdoor T_w to test if $s_w = u_1^{p_1(w)} \cdot u_2^{p_2(w)}$, outputs "1" if yes.

4.3 Plugin

Figure 3 shows the files needed to develop a simple "*hello-world*" Mozilla Firefox plugin. *Mainfest.json* is a JSON file and it is the most important component for a plugin which will be run first as it specifies all details of the plugin such as the manifest version, plugin name, description, icon, content scripts and also the background scripts. Then, the image of the icon will be stored in the images folder, and all the JavaScript will be stored in the scripts folder. The *popup.html* and *popup.css* specify how the plugin looks like.

Once all files are ready, zip the whole folder and rename the extension of the zipped folder to.*xpi*. XPI is an abbreviation for Cross-Platform Install. Its file extension called zippy, is a Mozilla browser extension archive file used to enhance the functionality of Mozilla products. Then, install the plugin on Mozilla Firefox.

5 System Output and Performance

5.1 System Output

Figure 4 shows the sample output of the programme when sending an email. When sending an email, sender will be asked to key in their email address, receiver email address, content of the email and also a keyword to be attached to the email. Then, the system will encrypt the email by using the receiver public key which is generated from their email address. Then, the content of the email and also the keyword will be encrypted and sent.

Fig. 4. Programme output of sending email

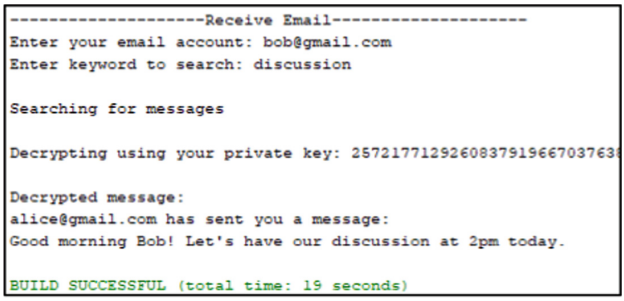

Fig. 5. Programme output of receiving email

Figure 5 shows the sample output of the programme when receiving an email. Upon receiving an email, receiver will be asked to key in their email address and the keyword that is used to search. Then, the system will search whether is there any email for the receiver that matched with the keyword. If found, the system will decrypt the email using the receiver private key and then display the original message.

5.2 Performance

In order to determine the efficiency of the proposed email plugin suite, the run-time performance of the system is being analysed in terms of encrypting, decrypting and searching. To ensure the accuracy of the result, we are using one test machine for the whole testing process. The machine that is being used is Intel ®Core i5-8250U CPU@ 3.4 GHz running on a 64-bit Windows OS. The run-time of the schemes is being recorded from 50 runs with different message and keyword sizes.

Figures 6, 7 and 8 show the scatter plot of run-time for 50 runs for encryption, decryption and searching respectively. As a result, the encryption run-time of this system is between 200 to 300 ms, the decryption run-time is between 100 to 200 ms and the search run-time is between 50 to 100 ms. The difference of the value of the 50 test results is below 0.1 s. It shows that both of these schemes are actually working with a good performance.

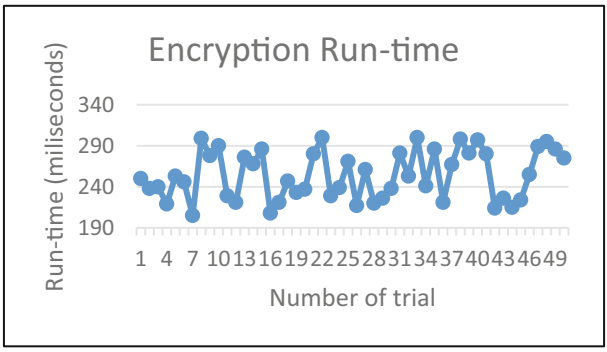

Fig. 6. Encryption run-time

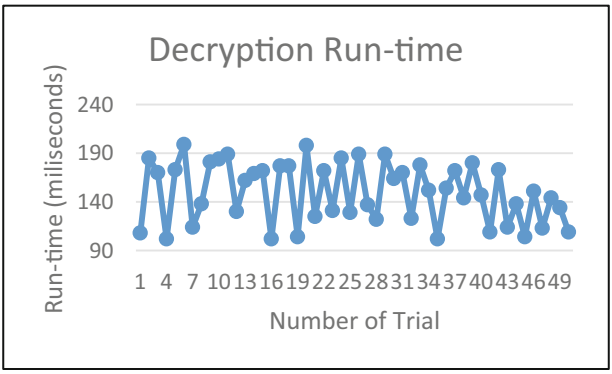

Fig. 7. Decryption run-time

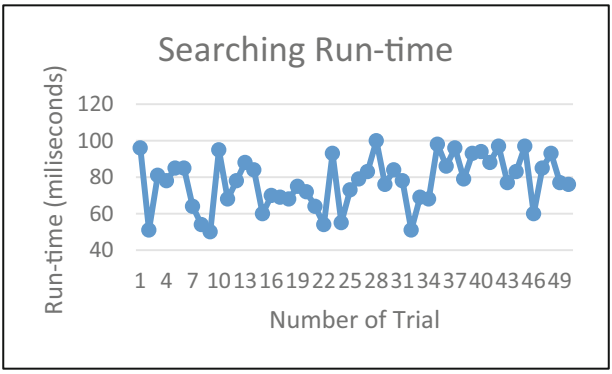

Fig. 8. Searching run-time

6 Conclusion

In conclusion, an email plugin suite for *k*-resilient IBE and PEKS has been implemented successfully with good performance. This email plugin with both encryption and searching functionalities provides an alternative choice of secure email platform to email users. Immediate future works include enhancing the existing package to be more user friendly and exploring possible email plugin suite with other available email platforms.

Acknowledgement. This work was supported by the Ministry of Higher Education of Malaysia's Fundamental Research Grant Scheme (FRGS/1/2018/ICT04/MMU/01/01).

References

1. Aviv, A. J., Locasto, M. E., Potter, S., & Keromytis, A. D. (2007). SSARES: Secure Searchable Automated Remote Email Storage. Proceedings - Annual Computer Security Applications Conference, ACSAC, 129–138. DOI: https://doi.org/10.1109/ACSAC.2007.30
2. Shamir, A. (1985). Identity-Based Cryptosystems and Signature Schemes. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS 196, 47–53. DOI: https://doi.org/10.1007/3-540-39568-7_5
3. Boneh D., Di Crescenzo G., Ostrovsky R., & Persiano G. (2004) Public Key Encryption with Keyword Search. In: Cachin C., Camenisch J.L. (eds) Advances in Cryptology - EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027, 506–522. Springer, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-540-24676-3_30
4. Heng, S.-H., & Kurosawa, K. (2004). *k*-Resilient Identity-Based Encryption in the Standard Model. In: Okamoto, T. (eds) Topics in Cryptology – CT-RSA 2004. Lecture Notes in Computer Science, vol 2964, 67–80. Springer, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-540-24660-2_6
5. Yau, W.-C., Heng, S.-H., Tan, S.-Y., Phan, Raphael C.-W., & Goi, B.-M (2012). Efficient Encryption with Keyword Search in Mobile Networks. Security and Communication Networks, vol 5, issue 12, 412–1422. DOI: https://doi.org/10.1002/sec.505
6. Patrick, B. (2021). What is Enigmail? Enigmail. https://www.enigmail.net/index.php/en/home