1-1-2014

P2P Email Encryption by An Identity-Based One-Way Group Key Agreement Protocol

Jyh-haw Yeh
Boise State University

Fiona Zeng
Boise State University

Thomas Long
Boise State University

 doi: 10.1109/PADSW.2014.7097879

P2P email encryption by an identity-based one-way group key agreement protocol

Jyh-haw Yeh, Fiona Zeng, Thomas Long
Dept. of Computer Science Boise State University
Boise, Idaho 83725, USA
jhyeh@boisestate.edu
fionazeng@u.boisestate.edu
thomaslong@u.boisestate.edu

***Abstract*— As a result of high-tech companies such as Google, Yahoo, and Microsoft offering free email services, email has become a primary channel of communication. However, email service providers have traditionally offered little in the way of message privacy protection. This has made emails, of which billions are sent around the world on any day, an attractive data source for personal identity information thieves. Google was one of the first companies to provide substantial email privacy protection when they began using the HTTPS always-on option to encrypt messages sent through their email service, Gmail. Unfortunately, Gmail's encryption option does not offer true point-to-point encryption since the encrypted emails are decrypted and stored in plaintext form on Google's servers. This type of approach poses a security vulnerability which is unacceptable to security-minded users such as highly sensitive government agencies and private companies. For these users, true point-to-point encryption is needed. This paper introduces an identity-based one-way group key agreement protocol and describes a point-to-point email encryption scheme based on the protocol. Both the security proofs and the efficiency analysis, with experimental results, of the new scheme are provided.**

Keywords: P2P encryption; Identity-based encryption; One-way group-key agreement; Bilinear pairings

I. Introduction

Email has evolved into one of the most important and widely used communication channels for both individuals and organizations. However, despite email's ubiquity in almost all parts of the world, current industry standards do not emphasize email security. In fact, most emails are currently transmitted as plain text across the Internet and other public networks. Additionally, email servers often backup messages in order to ensure the message's delivery in the face of a network failure. Since attackers can potentially read, copy, and alter every un-encrypted email sent over networks or stored on a mail server, there is an urgent need for point-to-point (P2P) email encryption.

Among the most popular webmail services; namely Hotmail, Yahoo mail, and Google's Gmail; Gmail has done the most to protect users' privacy by using the https (http secure) always-on option to encrypt emails as they travel between a web browser and Gmail servers. This procedure helps protect data from being eavesdropped on by third parties during data transmission at the cost of higher CPU usage and latency. However, it does not offer protection against attackers who are able to gain access, either physically or remotely, to Google servers since all emails are decrypted and stored in plain text form on those servers. In this case, the attacker would be able to read and/or alter every email on the server; P2P encryption is needed to prevent such an attack. The following sequence of events describes the process used by Gmail to send an email:

Sender writes email and clicks "Send" $\Longrightarrow$ *Sender's browser encrypts email and transmits it to Gmail Server* $\Longrightarrow$ *Gmail server decrypts and stores email* $\Longrightarrow$ *Gmail server re-encrypts and sends email to recipient* $\Longrightarrow$ *Recipient's browser loads and decrypts email*

This https procedure relies on the Transport Layer Secure (TLS) protocol [1] to achieve session key agreement between the sender and Gmail server and the Gmail server and the email's ultimate recipient. The two session key agreements are needed since the Gmail server must decrypt each message using the session key shared with the sender and later re-encrypt and send the message using the session key shared with the email recipient. This can be quite expensive since each TLS key agreement requires the transmission of several back-and-forth handshake messages.

In contrast, P2P encryption requires a single encryption by the sender and a single decryption by the recipient; no server side encryption or decryption is needed. The following sequence of events illustrates the sending of an email using P2P encryption:

Sender writes email and clicks "Send" $\Longrightarrow$ *Sender's browser encrypts email and transmits it to email server* $\Longrightarrow$ *email server stores email* $\Longrightarrow$ *email server sends email to recipient* $\Longrightarrow$ *Recipient's browser loads and decrypts email*

P2P encryption compares favorably to the Gmail https protocol since it does not require server side encryption/decryption and can thus forego the two TLS handshakes. In theory, the adoption of a P2P email encryption scheme would reduce server load and thus latency.

P2P encryption requires the adoption of one of the fol-

lowing approaches:

1) Before two users communicate, they establish and share an encryption key which they will use for future communications.
2) Each email user determines a private/public key set and publishes their public key.
3) The email sender and recipient find a way to agree on an encryption key without the need for an information exchanging handshake since the recipient might not be online when the email is sent.

The first approach does not scale well since it requires sharing a different key with each potential email recipient. The second approach has been adopted by the well-known PGP (Pretty Good Privacy) [2], [3] secure email protocol, which requires each PGP user to have his/her own public and private key pair. In PGP, an email sender needs to know the public key of the email's recipient prior to the encryption of the message. This is often achieved by querying a public-key certificate authority (CA) to retrieve (or verify) the recipient's public key. While this approach scales well, it has some serious disadvantages such as:

1) all participants need to decide upon and publish public keys.
2) it is difficult to find a trusted third party to act as the CA.
3) tracking valid and revoked certificates requires extra work on behalf of the CA.
4) public key encryption is computationally expensive.

Identity-based cryptography (IBC) is a promising solution to these issues. In IBC, everyone's public key is generated from a unique identifier; for instance, an individual's email address. A trusted key distribution center (KDC) uses a cryptographic algorithm to calculate the private key for a public key and sends the pair of keys to the participant. Since both the public key generation algorithm and the input to the algorithm (i.e. email address) are publicly available, anyone is able to calculate another's public key on the fly without needing to querying a server. Studies have shown that an IBC system requires a significantly less complex infrastructure (fewer servers and easier installation) and lower operating costs and user productivity losses (one-fifth and one-third of the values, respectively) compared to a typical public-key system. [6]

This paper proposes an identity-based one-way group key agreement protocol for email encryption. The proposed protocol allows email participants to agree on a symmetric key using encryption algorithms, such as AES [4], which are far more efficient than those involving an IBC public key.

The paper is organized as follows. Section II describes related work. Section III gives the cryptographic background needed to understand this paper. It includes a brief introduction to bilinear pairings, the associated cryptographically hard problems, and a typical identity-based cryptosystem. Section IV describes the identity-based one-way group key agreement protocol. Section V shows security proofs for the protocol. Section VI provides an efficiency analysis of the proposed P2P email encryption scheme and includes a detailed comparison with the well-known PGP program. Section VII presents the performance results of the new scheme. Finally, Section VIII summarizes the proposed work and concludes the paper.

II. Related work

PGP is a secure protocol which enables P2P email encryption. If a person wishes to use PGP to send a secure email, he/she needs to:

1) encrypt the email using the IDEA encryption algorithm [5],
2) find and verify the email recipient's RSA public key [7],
3) encrypt the IDEA encryption key using the email recipient's RSA public-key.

The encrypted email can then be sent over a regular emailing system. Upon receiving a PGP-encrypted email, a user needs to

1) use his/her RSA private key to get the IDEA encryption key,
2) use the IDEA key to decrypt the email.

The emailing processes listed above are all fairly easy to implement with the exception of finding and verifying an email recipient's public keys. While getting someone's public key is fairly straightforward, often requiring the querying of a public directory, verifying the received key requires a CA's signature of endorsement. This is the main disadvantage of public-key cryptosystems, including PGP, and is even more of a problem when an email is sent to multiple recipients. In that case, the sender needs to perform the troublesome public-key verification process for each recipient. The proposed identity-based one-way group key agreement protocol does not require the public-key verification process and its associated costs.

III. Cryptographic background

A. Bilinear pairings

Bilinear pairing is a popular cryptosystem which has found recent use in various efficient encryption and signature schemes [8], [9], [10], [11], [12], [13], [14]. A symmetric bilinear pairings cryptosystem is described briefly in this section.

Let $(G_1, +)$ and $(G_2, \times)$ be two cyclic groups of the same prime order, q, and let B be the generator of the additive group G_1, and $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear mapping if it has the following properties:

Bilinearity:

$$\forall X, Y, Z \in G_1, \text{ and } \forall a, b \in Z_q^*,$$

$$e(X,Y) = e(Y,X)$$
$$e(aX,bY) = e(X,Y)^{ab} = e(bX,aY)$$
$$e(X,Y+Z) = e(X,Y)e(X,Z)$$

Non-degeneracy:

If B is a generator of G_1, then $e(B,B)$ is a generator of G_2.

Computability:

$\forall X, Y \in G_1$, there exists a polynomial-time algorithm to efficiently compute the bilinear mapping $e(X,Y)$.

An elliptic curve is a typical example for the G_1 group [15], [16]. The security of most bilinear mapping based cryptographic schemes is related to the difficulty of the bilinear variants of the Diffie-Hellman problems, which are:

- Computational Bilinear Diffie-Hellman Problem (CBDHP): Given X, aX, bX for $a, b \in Z_q^*$, compute abX.
- Decision Bilinear Diffie-Hellman Problem (DBDHP): Given X, aX, bX, cX for $a, b, c \in Z_q^*$, and an element $g \in G_2$, decide if $g = e(X,X)^{abc}$.

Currently, there is no known algorithm which is able to efficiently solve these hard problems.

B. Identity-based cryptosystem

In a typical public-key cryptosystem, a trusted third party must serve as a CA by providing public-key endorsement services for registered users. The disadvantages of having such certificate authorities were discussed earlier in this paper. Identity-based cryptography is remarkable in that it does not require a CA. Instead, a public known hash function can be used to derive a participant's public key from the participant's identity token.

While identity-based cryptosystems do not require a CA, most implementations require a central server, called a key distribution center (KDC), to generate and distribute the public/private key pair for users when they first register. KDCs differ from CAs in that they do not need to up-and-running all the time and they don't need to provide public-key verification services for each communication.

The KDC needs to define a set of cryptographic parameters and make them publicly known. A typical identity-based cryptosystem setting is as follows:

1) The KDC chooses two cyclic groups $(G_1, +)$ and $(G_2, \times)$ of the same prime order q. Let B be a generator of the order q over G_1 and $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear mapping.
2) The KDC also chooses a cryptographic hash function $H : \{0,1\}^* \rightarrow G_1$ that can map a user's identity to a point in G_1. The map-to-curve and map-to-point algorithms from Weil pairings in [17], [18] are such functions.
3) The KDC selects a master secret $S \in Z_q^*$.
4) Finally, the KDC publishes the set of cryptographic parameters $\{G_1, G_2, q, B, e, H\}$.
5) Each registered user U_i will have a public key P_i.

$$P_i = H(ID_i) \in G_1 \quad (1)$$

where $ID_i \in \{0,1\}^*$ is the identity of the user U_i. The KDC calculates a private key S_i and sends it to U_i through a secure channel, where

$$S_i = S \times P_i \in G_1 \quad (2)$$

IV. One-way group key agreement protocol for P2P email encryption

This section proposes the identity-based one-way group key agreement protocol and describes its application to P2P email encryption.

A. KDC server

For the proposed one-way group key agreement protocol to allow an email sender and a group of recipients to agree on a key for P2P email encryption, all of the email's participants must have public-private keys issued by the same KDC. This can easily be achieved by letting the email service provider act as the KDC. In that case, when a user registered an account with the company, the company's email server would act as the KDC and would generate the user's public key and private key using the steps described in Section III-B. However, since the service provider knows the master secret key, S, they would be able to derive every user's private key. With the private key, they could determine the user's encryption key and subsequently decrypt any of their emails. The service provider would have strong incentives for decrypting user emails since it would allow for more efficient indexing, better spam detection, and more effective ad-targeting.

For this reason, highly sensitive organizations such as government agencies would probably want to have their own server running as the KDC to use their own set of identity-based cryptographic parameters. An agency's use of its own server as the KDC does not limit the email service providers which could be used by the agency's employees. On the contrary, it allows for secure P2P email encryption across most networks. Only the agency's KDC and the email's recipients (if they are registered users of the agency), who can use the agency's set of cryptographic parameters to determine the key for the email's encryption, will be able to decrypt the email.

B. Key generation by email sender

In any email application, a sender can email a message to a group of $n > 0$ recipients. Assume ID_0 is the email sender's identity (i.e. the sender's email address) and let ID_i, for $i = 1, 2, \ldots, n$, denote the identity for each of the n email recipients.

1) The email sender picks a random number $r \in Z_q^*$ and computes

$$x_i = e(S_0, rP_i) \in G_2, \forall i = 0, 1, 2, \ldots, n \quad (3)$$

where S_0 is the private key of the email sender ID_0 and $P_i = H(ID_i)$ is the public key of the email recipient ID_i.

2) The email sender generates the encryption key K by computing

$$K = \oplus_{\forall i=0,1,\ldots,n}(x_i) \quad (4)$$

3) The email sender also computes $y_i, \forall i = 1, 2, \ldots, n$, as follows.

$$y_i = \oplus_{\forall j \neq i}(x_j) \quad (5)$$

in other words,

$$y_i = x_0 \oplus x_1 \oplus \ldots \oplus x_{i-1} \oplus x_{i+1} \oplus \ldots \oplus x_n \quad (6)$$

4) The email sender encrypts the email using the secret key K and sends the encrypted email out along with $(r, y_1, y_2, \ldots, y_n)$.

C. Key re-generation by each email recipient

Upon receiving the email from ID_0, each recipient ID_i can compute the secret key K from y_i (which is attached in the email) and the public key $P_0 = H(ID_0)$ of the email sender ID_0 with the following equation:

$$K = y_i \oplus e(rP_0, S_i) \quad (7)$$

since

$$\begin{aligned} y_i \oplus e(rP_0, S_i) &= y_i \oplus e(rP_0, sP_i) \\ &= y_i \oplus e(sP_0, rP_i) \\ &= y_i \oplus e(S_0, rP_i) \\ &= y_i \oplus x_i \\ &= (\oplus_{\forall j \neq i}(x_j)) \oplus x_i \\ &= K \end{aligned}$$

D. Example

The security proofs in the next section distinguish between the case where an email has an even number of recipients and the case where an email has an odd number of recipients. Thus, in this section we will provide an example for each case. To start, assume an email sender, with identity ID_0, would like to send an email to two (an even number) recipients with identities ID_1 and ID_2. This scenario would be handled as follows:

1) The sender picks a random number r and computes

$$\begin{cases} x_0 = e(S_0, rP_0) \\ x_1 = e(S_0, rP_1) \\ x_2 = e(S_0, rP_2) \end{cases}$$

2) The sender generates the encryption key

$$K = x_0 \oplus x_1 \oplus x_2$$

3) The sender computes

$$\begin{cases} y_1 = x_0 \oplus x_2 \\ y_2 = x_0 \oplus x_1 \end{cases}$$

4) The sender encrypts the email using the key K and sends (r, y_1, y_2) along with the email.

5) The recipient with identity ID_1 computes

$$\begin{aligned} y_1 \oplus e(rP_0, S_1) &= x_0 \oplus x_2 \oplus e(rP_0, SP_1) \\ &= x_0 \oplus x_2 \oplus e(sP_0, rP_1) \\ &= x_0 \oplus x_2 \oplus e(S_0, rP_1) \\ &= x_0 \oplus x_2 \oplus x_1 \\ &= K \end{aligned}$$

6) The recipient with identity ID_2 computes

$$\begin{aligned} y_2 \oplus e(rP_0, S_2) &= x_0 \oplus x_1 \oplus e(rP_0, SP_2) \\ &= x_0 \oplus x_1 \oplus e(sP_0, rP_2) \\ &= x_0 \oplus x_1 \oplus e(S_0, rP_2) \\ &= x_0 \oplus x_1 \oplus x_2 \\ &= K \end{aligned}$$

Thus, both email recipients are able to derive the encryption key K which was generated by the email's sender.

Now, assume an email sender with identity ID_0 would like to send an email to three (an odd number) recipients with identities ID_1, ID_2, and ID_3. This scenario would be handled as follows:

1) The sender picks a random number r and computes

$$\begin{cases} x_0 = e(S_0, rP_0) \\ x_1 = e(S_0, rP_1) \\ x_2 = e(S_0, rP_2) \\ x_3 = e(S_0, rP_3) \end{cases}$$

2) The sender generates the encryption key

$$K = x_0 \oplus x_1 \oplus x_2 \oplus x_3$$

3) The sender computes

$$\begin{cases} y_1 = x_0 \oplus x_2 \oplus x_3 \\ y_2 = x_0 \oplus x_1 \oplus x_3 \\ y_3 = x_0 \oplus x_1 \oplus x_2 \end{cases}$$

4) The sender encrypts the email using the key K and sends (r, y_1, y_2, y_3) along with the email.

5) The recipient with identity ID_1 computes

$$\begin{aligned} y_1 \oplus e(rP_0, S_1) &= x_0 \oplus x_2 \oplus x_3 \oplus e(rP_0, SP_1) \\ &= x_0 \oplus x_2 \oplus x_3 \oplus e(sP_0, rP_1) \\ &= x_0 \oplus x_2 \oplus x_3 \oplus e(S_0, rP_1) \\ &= x_0 \oplus x_2 \oplus x_3 \oplus x_1 \\ &= K \end{aligned}$$

6) The recipient with identity ID_2 computes

$$\begin{aligned} y_2 \oplus e(rP_0, S_2) &= x_0 \oplus x_1 \oplus x_3 \oplus e(rP_0, SP_2) \\ &= x_0 \oplus x_1 \oplus x_3 \oplus e(sP_0, rP_2) \\ &= x_0 \oplus x_1 \oplus x_3 \oplus e(S_0, rP_2) \\ &= x_0 \oplus x_1 \oplus x_3 \oplus x_2 \\ &= K \end{aligned}$$

7) The recipient with identity ID_3 computes

$$\begin{aligned} y_3 \oplus e(rP_0, S_3) &= x_0 \oplus x_1 \oplus x_2 \oplus e(rP_0, SP_3) \\ &= x_0 \oplus x_1 \oplus x_2 \oplus e(sP_0, rP_3) \\ &= x_0 \oplus x_1 \oplus x_2 \oplus e(S_0, rP_3) \\ &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \\ &= K \end{aligned}$$

Thus, all three email recipients are able to derive the encryption key K which was generated by the email's sender.

V. Security analysis

In this section, we provide proofs showing that the encryption key K cannot be derived from the public information $\{y_1, y_2, \ldots, y_n\}$ alone. Let's define the encryption key K and all y_i's as sets, since we will consider these quantities as sets in our security analysis.

Definition V-1: *the encryption key K defined in Equation (4) is a set of elements $\{x_0, x_1, \ldots, x_n\}$ that are linked together by the $\oplus$ operator.*

Definition V-2: *A subset, s, of K is a subset of $\{x_0, x_1, \ldots, x_n\}$, where all elements in the subset are linked together by the $\oplus$ operator. Thus, the y_i's defined in Equation (5) are all subsets of K.*

To prove the security of the proposed scheme, we need to answer the question: What subsets of K are required in order to determine K?

Theorem V-1: *A set of subsets of K, denoted by $G = \{s_1, s_2, \ldots, s_t\}$ for some positive integer t, can be joined with the $\oplus$ operator to yield K if and only if every element of K appears an odd number of times in G.*

Proof: Assume, for the sake of contradiction, that $d > 0$ elements of K, namely $\{x_{i_1}, x_{i_2}, \ldots, x_{i_d}\}$, appear an even (or zero) number of times in $G = \{s_1, s_2, \ldots, s_t\}$, where $s_1 \oplus s_2 \oplus \ldots \oplus s_t = K$. The remaining $n - d + 1$ elements, namely $\{x_{j_1}, x_{j_2}, \ldots, x_{j_{n-d+1}}\}$ must appear an odd number of times in G. Let $(\oplus x_i)^r$ represent the operation $x_i \oplus x_i \oplus \ldots \oplus x_i$, where x_i appears r times (x_i XOR'ed with itself $r - 1$ times). Note that

$$(x_i)^r = \begin{cases} x_i & \text{if } r \text{ is an odd number} \\ 0 & \text{if } r \text{ is an even number} \end{cases}$$

Now consider

$$\begin{aligned} G &= s_1 \oplus s_2 \oplus \ldots \oplus s_t \\ &= [(\oplus x_{i_1})^{\alpha_1} \oplus (\oplus x_{i_2})^{\alpha_2} \oplus \ldots \oplus (\oplus x_{i_d})^{\alpha_d}] \oplus \\ &\quad [(\oplus x_{j_1})^{\beta_1} \oplus (\oplus x_{j_2})^{\beta_2} \oplus \ldots \oplus (\oplus x_{j_{n-d+1}})^{\beta_{n-d+1}}] \\ &= x_{j_1} \oplus x_{j_2} \oplus \ldots \oplus x_{j_{n-d+1}} \end{aligned}$$

where each α_i is even $\forall i \in [1, d]$ and each β_j is odd $\forall j \in [1, n - d + 1]$.

Having assumed that $s_1 \oplus s_2 \oplus \ldots \oplus s_t = K$ and having shown that $s_1 \oplus s_2 \oplus \ldots \oplus s_t = x_{j_1} \oplus x_{j_2} \oplus \ldots \oplus x_{j_{n-d+1}}$, we can conclude that $K = x_{j_1} \oplus x_{j_2} \oplus \ldots \oplus x_{j_{n-d+1}}$. However, this could only be true if $d = 0$, which contradicts our previous assumption that $d > 0$. This contradiction proves the theorem. $\diamondsuit$

Theorem V-2: *A set of subsets of K, denoted by G, which can be combined via $\oplus$ to yield K cannot be constructed from a subset of $Y = \{y_1, y_2, \ldots y_n\}$, where the y_i's are those which are defined in Equation (5).*

Proof: From the given set Y, let's try to construct a set G which can be XORed to yield K. Obviously $G \neq \emptyset$ and $G \neq \{y_i\}, \forall i$ since in these cases G cannot be XORed to get K.

Case $|G|$ is even: Noting that x_0 appears once in every y_i (see Equation (6) and the example in Section IV-D), if $|G|$ is even then x_0 appears an even number of times in G. So according to Theorem V-1, G cannot be XORed to yield K.

Case $|G|$ is odd: In this case, every x_i such that $y_i \in G$ appears $|G| - 1$ times in G since x_i appears once in every $y_j \in G$ except y_i (See Equation (5)). Because $|G|$ is odd and is greater than 1, $|G| - 1$ must be even and non-zero. Thus, some elements x_i's of K appear an even number of times in G. Again, according to Theorem V-I G cannot be XORed to yield K.

Having covered all possible cases, we conclude that the encryption key K cannot be derived from the given set $Y = \{y_1, y_2, \ldots y_n\}$. $\diamondsuit$

VI. Efficiency

In this section we analyze the proposed P2P email encryption scheme from several perspectives. First, we analyze the server side computational cost of both user registration (i.e. public/private key generation) and email transactions. Next, we consider the computational cost from an email sender's perspective and an email recipient's perspective. We assume that the email was sent to n recipients. Finally, we analyze the increase in the size of emails which are subject to our P2P encryption scheme. Our analysis will use the following notation for operations associated with the P2P scheme:

PM	point multiplication in group G_1
BP	bilinear pairing
HASH	map-to-point hash algorithm [17], [18].

Before we begin, we note that if a practical elliptic curve E/F_3163 is used to implement the group G_1, then one BP operation requires $\approx 11,110$ modular multiplications in F_3163 [19]. Meanwhile, a PM operation of E/F_3163 requires only a few hundred modular multiplications in F_3163.

A. User registration

A new user registers an email account through the KDC (e.g., Google server). After a user chooses his/her email

address, the KDC uses the HASH and a PM to compute the public and private key pair for the user. These two key generations can be seen in Equations (1) and (2), respectively. Both operations are quite efficient and do not represent a significant cost.

B. Computational cost for an email sender

To send an email to n recipients, the sender is required to

1) calculate $x_0, x_1, x_2, \ldots, x_n$: From Equations (1) and (3) we see that each computation requires a HASH, PM, and BP operation.
2) derive the encryption key K. From Equation (4), this derivation requires $\oplus$ing all x_i's.
3) calculate $y_1, y_2, \ldots, y_n$: From Equation (5), the calculation of each y_i requires $\oplus$ing all x_j's $\forall j \neq i$.
4) AES encrypt the email using the encryption key K.

The bit-wise $\oplus$ operation is extremely efficient, making the costs for calculating K and y_i's negligible. From the email sender's perspective, the main computational cost stems from the AES encryption and the calculation of $X = \{x_0, x_1, \ldots, x_n\}$. Since each x_i calculation requires a HASH, a PM and a BP operation, the total cost of calculating X is $(n+1)$ HASH, $(n+1)$ PM and $(n+1)$ BP operations.

C. Computational cost for an email server

In contrast to the current Gmail behavior, the email server from the proposed scheme is not required to perform any decryption or encryption operations. However, encrypted emails have an opportunity cost associated with them since encryption makes spam filtering and history searches more difficult.

D. Computational cost for an email recipient

To receive an email, a recipient needs to

1) re-construct the encryption key using Equation (7). This requires One HASH, one PM and one BP operation (the $\oplus$ operation is again ignored).
2) AES decrypt the message using the re-constructed encryption key K.

We see that the computational cost of sending an email is linearly proportional to the number of recipients while the cost of receiving an email is constant.

E. Email size

The size of an email will increase as a result of using the proposed P2P email encryption scheme. The first, and least significant, cause of growth is the use of block ciphers such as AES. These ciphers often require full blocks and will consequently fill the last block with random bits if it is not full. This has the potential to add up to one block minus a bit to the email's size. Since most block sizes are relatively small, AES has a block size of 256 bits, the added size is usually not noticeable. A more significant increase in message size results from having to include the key agreement information, $(r, y_1, y_2, \ldots, y_n)$, with every email. If an elliptic curve is used to implement the bilinear pairing cryptosystem from the proposed P2P encryption scheme, each member of the key agreement information will be roughly the same size as the key of the selected curve. According to the National Institute of Standards and Technology, n-bit security (the security of a symmetric encryption scheme with an n-bit key) requires an elliptic curve with a key size $\approx 2n$ bits. So using an elliptic curve and 128-bit security would result in a $256 \times (n+1)$ bit increase in the size of an email sent to n recipients.

F. Comparison to the current Gmail setting

Table I provides a brief efficiency comparison between the proposed P2P-EES and the current Gmail's HTTPS always-on option.

TABLE I

EFFICIENCY COMPARISON BETWEEN P2P-EES AND THE GMAIL'S HTTPS ALWAYS-ON OPTION, ASSUMING THERE ARE n RECIPIENTS IN AN EMAIL

	P2P-EES	Gmail's HTTPS always-on
Email sender	• $(n+1)$ PM; • $(n+1)$ BP; • $(n+1)$ HASH; • Email encryption.	• TLS handshake with the email server; • Email encryption.
Email server	• None	• TLS handshake with the email sender; • Email decryption • TLS handshake with the email recipient; • Email re-encryption.
Email recipient	• 1 PM; • 1 BP; • 1 HASH; • Email decryption.	• TLS handshake with the email server; • Email decryption.
Email size	• Encrypted message along with all y_i's (the key agreement info).	• Encrypted message only.

VII. EXPERIMENTS

In addition to the theoretical performance analysis of the P2P-EES, we implemented the P2P-EES prototype and conducted experiments to measure the latency and storage requirements for server cryptosystem setup, user registration, and email transmission.

A. Implementation

The cryptosystem was implemented using a type A pairing [20], [25], which is constructed on a curve $E : y^2 = x^3 + x$ over the field F_q for some prime $q = 3 \bmod 4$. As a result, $E(F_q)$ contains $q+1$ points and $E(F_{q^2})$ contains $(q+1)^2$

points. The group $G = E(F_q)[r]$ is cyclic if r is an odd number and a factor of $q+1$. Consider the distortion map in [21]

$$\psi(x, y) = (x, iy) \tag{8}$$

where ψ maps points $(x, y) \in E(F_q)$ to points in $E(F_{q^2})\backslash E(F_q)$. If f denotes the Tate or Weil [22], [23], [24] pairing, the bilinear mapping $e : G \times G \rightarrow F_{q^2}$ can be defined by

$$e(P, Q) = f(P, \psi(Q)) \tag{9}$$

An implementation for this type of pairing has been suggested by [25] and is as follows:

1) An order r, of the group G, is chosen to be larger enough to avoid generic discrete logarithm attacks. $r = 160$ bits and $r = 256$ bits were used for our experiments.
2) Choose a random number h such that it is a multiple of 4 and $(hr)^2$ is large enough to resist finite field attacks. For example, if one desires q^2 to be 1024 bits long, then h must be ≈ 256 bits long (assuming $r = 256$ bits).
3) Repeat step 2 while $q = hr - 1$ is not prime.

B. Experimental hardware

The experiments were conducted on a machine with an Intel(R) Core(TM)i3CPU M330@2.13GHz processor, 4 GB RAM, and the 64-bit Windows 7 home premium operating system.

C. Cryptosystem setup

In a real application, the setup process needs to only be performed once for a specific set of (rBits,qBits,MKBits) values, where rBits is r's bit size, qBits is q's bit size, and MKBits is the Master Key S's bit size. For the performance testing, the setup process was ran multiple times with different (rBits,qBits,MKBits) values. Table II gives the execution time for determining the parameters for the pairing system, including the generation of the master key S.

TABLE II
SERVER PAIRING CRYPTOSYSTEM PARAMETERS SETUP

rBits	qBits	MKBits	Time (ms: millisecond)
160	256	256	907
160	256	512	923
256	512	256	978
256	512	512	1239

D. User registration

After the setup process, the server writes the cryptosystem's parameters and master key to files. Upon receiving a user registration request, a public and private key pair are generated for the user. The key pair generation process was described previously with Equations (1) and (2), where the user's email address is used as their unique ID. Table III shows the execution time of key pair generation during our experiments.

TABLE III
USER REGISTRATION - KEY PAIR GENERATION

rBits	qBits	MKBits	email address	Time (ms)
256	512	256	fiona201301@gmail.com	109
256	512	512	fiona201301@gmail.com	156
256	512	256	fionazeng@u.boisestate.edu	125
256	512	512	fionazeng@u.boisestate.edu	167

E. Email transmission

A Type A curve, with rBits = 256 and qBits = 512, was used to measure encryption and decryption times. The connection time is considered to be the time it takes for a client to connect to the email server when sending or opening an email.

1) Key derivation, encryption, and decryption for email messages with a single recipient: Table IV shows the sender's and recipient's network connection time; as well as the key derivation, encryption and decryption time; for emails containing only one recipient. The cipher text size is included for reference.

TABLE IV
CONNECTION (CONN.), KEY DERIVATION (DER.), ENCRYPTION (ENC.) AND DECRYPTION (DEC.) TIME FOR EMAILS WITH ONLY ONE RECIPIENT

Msg. (char)	Cipher size (char)	Sender			Recipient		
		Conn. (ms)	Der. (ms)	Enc. (ms)	Conn. (ms)	Der. (ms)	Dec. (ms)
524	875	4493	157	198	4926	153	698
3009	5670	4953	168	224	4583	172	813
10658	12944	5614	153	229	4922	144	935

There are two factors which contributed to the increase in cipher text size. First, the padding scheme used by AES (a block cipher) could have added a few bytes (no more than the 16 byte block size) of padding to the message. Second, and more importantly, a random number r and all y_i's needed to be appended at the end of the message as part of the cryptosystem. Therefore, the size of a cipher text is approximately the sum of the size of the message, the random number r, and all y_i's.

2) Encryption and decryption for email messages with multiple recipients: Emails were sent to multiple recipients to measure how the system scaled with the number of email recipients. The marginal increase in transmission time was consistent for larger numbers of recipients and so, for the sake of brevity, we show only the results of two and three recipient emails. The connection times were also omitted because of the lack of variance in the values (see Table IV for typical connection times). The results of the two and three recipient tests are shown in Tables V and VI.

TABLE V

KEY DERIVATION (DER.), ENCRYPTION (ENC.) AND DECRYPTION (DEC.) TIME FOR EMAILS WITH TWO RECIPIENTS

Msg. (char)	Sender		Recipient1		Recipient2	
	Der. (ms)	Enc. (ms)	Der. (ms)	Dec. (ms)	Der. (ms)	Dec. (ms)
524	335	202	103	681	98	662
3009	349	226	96	892	112	824
10658	389	276	101	1064	116	922

TABLE VI

KEY DERIVATION (DER.), ENCRYPTION (ENC.) AND DECRYPTION (DEC.) TIME FOR EMAILS WITH THREE RECIPIENTS

Msg. (char)	Sender		Recipient1		Recipient2		Recipient3	
	Der. (ms)	Enc. (ms)	Der. (ms)	Dec. (ms)	Der. (ms)	Dec. (ms)	Der. (ms)	Dec. (ms)
524	477	192	105	668	109	662	102	676
3009	498	212	96	876	112	824	107	864
10658	481	296	101	998	116	922	112	972

F. Results Summary

The experimental result can be summarized as follows:

1) Cryptosystem parameter setup, including the master key generation (rBits=256, qBits=512 and MK-Bits=512 bits) takes ≈ 1 second.
2) Key pair generation during user registration takes $\approx .1$ seconds.
3) Connecting to an email server using the Java Mail library requires 4 to 6 seconds. It takes a fraction of a second to encrypt or decrypt messages, and that speed (throughput per second) is independent of the email's size and recipient count.
4) The key derivation time is roughly the same for each recipient. However, the sender's key derivation time is directly proportional to the number of recipients. These results match the theoretical analysis from Table I.
5) The increase in cipher size is slight and independent of the plain text size, so the scheme has no storage concerns.

VIII. CONCLUSION

This paper proposed a secure and efficient identity-based one-way group key agreement protocol which can be integrated with an email service application to provide P2P encryption. The scheme's security was proved in Section V and its efficiency was analyzed in Section VI, in which Table I gave a detailed theoretic comparison between the proposed P2P email encryption scheme and the current Gmail HTTPS always-on option. Experiments were conducted to measure the performance of the proposed P2P-EES in Section VII.

REFERENCES

[1] T. Dierks, E. Rescorla (August 2008). "The Transport Layer Security (TLS) Protocol, Version 1.2." http://tools.ietf.org/html/rfc5246

[2] P. Zimmermann, "Why I wrote PGP?" https://www. philzimmermann.com/EN/essays/WhyIWrotePGP.html, 1991.

[3] The International PGP Homepage, "PGP Documentation, " http://www.pgpi.org/doc/

[4] "Advanced Encryption Standard (AES)," National Institute of Standards and Technology (NIST), FIPs 197, 2001.

[5] X. Lai and J. Massey, "A Proposal for a New Block Encryption Standard," EUROCRYPT, pp. 389-404, 1990.

[6] "Total Cost of Ownership of Voltage IBE 3X Lower than PKI in Ferris Research Study," Press Releases Voltage Security, May 30, 2006. http://www.voltage.com/pressreleases/PR060530.htm

[7] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, Vol. 21, No. 2, pp. 120-126, 1978.

[8] H.Y. Lin, T.S. Wu, S.K. Huang, Y.S. Yeh, "Efficient proxy signcryption scheme with provable CCA and CMA security," Computers and Mathematics with Applications Vol. 60, No. 7, pp. 1850-1858, 2010.

[9] H. Elkamchouchi, Y. Abouelseoud, "A new proxy identity-based signcryption scheme for partial delegation of signing rights," Cryptology ePrint Archive, Report 2008/041, 2008.

[10] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, M. Scott, "Efficient algorithms for pairing-based cryptosystems," CRYPTO'02, pp. 354-368, 2002.

[11] D. Boneh, M. Franklin, "Identity-based encryption from the Weil pairing," CRYPTO'01, pp. 213-229, 2001.

[12] D. Boneh, B. Lynn, H. Shacham, "Short signature from the Weil pairing," Advances in Cryptography - ASIACRYPT'01, Springer-Verlag, pp. 514-532, 2001.

[13] C. Gentry, A. Silverberg, "Hierarchical ID-based cryptography," Advances in Cryptography - ASIACRYPT'02, Springer-Verlag, pp. 548-566, 2002.

[14] F. Zhang, K. Kim, "ID-based blind signature and ring signature from pairings," Advances in Cryptography - ASIACRYPT'02, Springer-Verlag, pp. 533-547, 2002.

[15] N. Koblitz, "Elliptic Curve Cryptosystems," Mathematics of Computation, Vol. 48, pp. 203-209, 1987.

[16] V. Miller, "Use of Elliptic Curves in Cryptography," CRYPTO'85, pp. 417-426, 1985.

[17] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in CRYPTO'01, Springer, LNCS 2139, pp. 213-229, 2001.

[18] X. Yi, "An Identity-based Signature Scheme from the Weil Pairing," IEEE Communications Letter, Vol. 7, No. 2, pp. 76-78, 2002.

[19] P.S.L.M. Barreto, B. Lynn, M. Scott, "On the selection of pairing-friendly groups," Selected Areas in Cryptography, Springer-verlag, 2003.

[20] "Bilinear Pairing Parameters Generators," JPBC Library, http://gas.dia.unisa.it/projects/jpbc/docs/ecpg.html#TypeA

[21] D. Page, N.P. Smart, and F. Vercauteren, "A comparison of MNT curves and supersingular curves," Cryptology ePrint Archive, 2004. http://eprint.iacr.org/

[22] G. Frey, M. Muller, and H.G. Ruck, "The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems," Transactions of Information Theory, Vol. 45, pp. 1717-1719, 1999.

[23] A. Joux, "The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems," Proceedings of 5th Algorithmic Number Theory Symposium, Lecture Notes in Computer Science. Springer-Verlag, 2002.

[24] D. Boneh and M. Franklin, "Identity-based Encryption from the Weil Pairing," Lecture Notes in Computer Science, Vol. 2139, 2001.

[25] B. Lynn, "On the implementation of Pairing-based Cryptosystems," Ph.D Dissertation, Stanford University, 2007.