2018

A Certificateless One-Way Group Key Agreement Protocol for End-to-End Email Encryption

Jyh-Haw Yeh
Boise State University

Srisarguru Sridhar
Boise State University

Gaby G. Dagher
Boise State University

Hung-Min Sun
National Tsing Hua University

Ning Shen
Boise State University

See next page for additional authors

 https://doi.org/10.1109/PRDC.2018.00014

A certificateless one-way group key agreement protocol for end-to-end email encryption

Jyh-haw Yeh
Dept. of Computer Science
Boise State University, Boise, USA
jhyeh@boisestate.edu

Srisarguru Sridhar
Dept. of Computer Science
Boise State University, Boise, USA
srisargurusridhar@u.boisestate.edu

Gaby Dagher
Dept. of Computer Science
Boise State University, Boise, USA
gabydagher@boisestate.edu

Hung-Min Sun[1,2]
[1]*Dept. of Computer Science*
National Tsing Hua University, Taiwan
[2]*Research Center for Information Technology Innovation, Academia, Sinica*, Taiwan
hmsun@cs.nthu.edu.tw

Ning Shen
Dept. of Computer Science
Boise State University
Boise, USA
ningshen@u.boisestate.edu

Kathleen Dakota White
Dept. of Applied Mathematics
University of North Carolina at Asheville
Asheville, USA
dwhite2@unca.edu

***Abstract*—Over the years, email has evolved into one of the most widely used communication channels for both individuals and organizations. However, despite near ubiquitous use in much of the world, current information technology standards do not place emphasis on email security. Not until recently, webmail services such as Yahoo's mail and Google's gmail started to encrypt emails for privacy protection. However, the encrypted emails will be decrypted and stored in the service provider's servers. If the servers are malicious or compromised, all the stored emails can be read, copied and altered. Thus, there is a strong need for end-to-end (E2E) email encryption to protect email user's privacy. In this paper, we present a certificateless one-way group key agreement protocol with the following features, which are suitable to implement E2E email encryption: (1) certificateless and thus there is no key escrow problem and no public key certificate infrastructure is required; (2) one-way group key agreement and thus no back-and-forth message exchange is required; and (3) n-party group key agreement (not just 2- or 3-party). This paper also provides a security proof for the proposed protocol using "proof by simulation". Finally, efficiency analysis of the protocol is presented at the end of the paper.**

***Index Terms*—End-to-end email encryption; One-way group key agreement; Certificateless PKC**

I. Introduction

This paper proposes a Certificateless One-way Group Key Agreement (CLOW-GKA) protocol for end-to-end (E2E) email encryption. The main motivations are that (1) email is the most used communication channel between individuals and organizations; (2) most organizations do not emphasize the importance of email security; and (3) most email servers will backup messages you sent to ensure delivery and thus the possibility of exposing backup messages to public is a real threat if the messages are not E2E encrypted.

E2E email encryption schemes are built on top of Public Key Cryptosystems (PKC). The evolution of the public key cryptography from the traditional PKC to the Identity-based PKC (ID-PKC) and then to the Certificateless PKC (CL-PKC) provides a more suitable cryptosystem for researchers to design secure protocols for E2E email encryption.

The traditional PKC is still used mostly around the world till now, though its certificate infrastructure is not easy to manage. In traditional PKC, each participant assigns their own public and private keys. Thus, it requires a trusted Certificate Authority (CA) to authenticate each participant's public key. The additional server for CA and the management (creation, revocation and storage) of all the certificates is a major issue the traditional PKC suffers.

PGP (Pretty Good Privacy) [1]–[3] probably is the most popular E2E email encryption scheme used in the world now. If a person wishes to use PGP to send a secure email, he/she needs to:

1) encrypt the email using the International Data Encryption Algorithm (IDEA) [4],
2) find and confirm the email receivers' public keys by verifying the corresponding public key certificates, and
3) encrypt the IDEA encryption key using the email receivers' public keys.

The encrypted email along with the encrypted IDEA key can then be sent over a regular emailing system. Upon receiving a PGP-encrypted email, each of the email receivers needs to

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

PGP's public key certificate infrastructure uses a distributed trust model called *Web Of Trust* to solve the key validation problem. In *Web Of Trust*, any user can issue certificates for any other users and users get the public keys from someone they trusts. This scheme is flexible and leave the individual user making the trust decisions. However, this certificate infrastructure still has some drawbacks, such as limited trust levels, limited validity levels, and counter-intuitive key validation [35]. The emailing processes listed above are fairly

easy for one-to-one emailing, but fail to provide an intuitive solution with multiple receivers. In that case, the sender needs to perform the key verification process for each email receiver, and also perform the encryption process multiple times with different keys.

Shamir [5] was the first one to propose the idea of Identity-based Public Key Cryptosystem ID-PKC to eliminate the certificate infrastructure. In ID-PKC, the participants' public keys are generated directly from their identities through some public known hash functions. Thus, every participant can directly derive other participants' public keys from their identities without having to verify the validity of public keys. Since Shamir's paper [5], many ID-PKC schemes [6]–[10] have been proposed in literature. However, ID-PKC schemes require a Key Generation Center (KGC) to generate and distribute private keys for participants. This approach inevitably introduces the key escrow problem, in which if the KGC is malicious or compromised, the whole system will be compromised (since the KGC knows participants' private keys).

In 2003, Al-Riyami et al. [11] introduced a first Certificateless Public Key Cryptosystem (CL-PKC) to solve the key escrow problem without having to have the public key certificate infrastructure. Following his paper, many certificateless protocols [12]–[15] were proposed for various applications, including secure encryption, signature, or key agreement. For application like email encryption, certificateless signature schemes obviously are not suitable. Certificateless encryption schemes are not suitable for email encryption either since for an email with multiple receivers the message or the encryption key may need to be encrypted multiple times, once by each email receiver's public key so that each receiver can use his/her private key to decrypt the message (or decrypt the encryption key first and then decrypt the message). Therefore, a secure key agreement protocol should be the best choice to implement E2E email encryption if the protocol is a one-way and is an n-party protocol. Unfortunately, none of the certificateless key agreement protocols proposed in literature were designed for E2E email encryption and thus are not one-way nor n-party.

The proposed CLOW-GKA protocol in this paper is a key agreement protocol with the following features that are required to implement a good E2E email encryption scheme.

- **Certificateless:**
  - **No public key certificate infrastructure is required:** Certificateless means there is no need to have certificate authority signed certificates to verify public keys. That is, users can use some known public information and/or identities to verify public keys directly.
  - **No key escrow problem:** There is no server (such as KGC) will know any user's private key and thus cannot derive any encryption key to decrypt emails. Any protocol used to implement E2E email encryption should not have the key escrow problem.
- **One-way:** Most existing group key agreement protocols require several rounds of message exchanges before all participants can reach a group key. This approach obviously does not work for E2E email encryption because not all participants are online when the email sender sends emails. The proposed CLOW-GKA protocol requires no back-and-forth message exchange. The sender only needs to encrypt the email using a specially constructed group key and then attach a set of Key Derivation Keys (KDKs) in the email, one KDK for each email receiver. Upon receiving the email, each receiver will based on his/her own private key, the KDK and the sender's public key to derive the group key (i.e., the encryption key).
- **n-party:** Most ID-based or certificateless-based group key agreement protocols in literature were designed for 2-party or 3-party only, probably because they are not one-way protocols and thus the amount of back-and-forth broadcast messages may become un-manageable if the group size is n (especially when n is big). The proposed CLOW-GKA has no such limitation and can support arbitrary n parties.

In addition to the above features, unlike other CL-PKC protocols, the proposed CLOW-GKA does not need to use secret channels for key distribution. In order to eliminate the key escrow problem, most CL-PKC protocols require the KGC to distribute a partial private key to each participant through some secret channels. The secrecy of the partial private key is important because if a third party gets hold of the partial private key, then they can create a fake public keys for the user to whom the partial private key belongs to. These fake public keys will also be able to pass the public key verification. A detailed explanation of this problem is discussed in Section IV-C. The proposed CLOW-GKA uses the binding-blinding technique [16] to eliminate this secret channel requirement. That is, the KGC can send a quantity, in which the partial private key is embedded within it, to the participant using any public channel. Only the participant receiving the quantity can recover the partial private key.

Whether the proposed E2E email encryption scheme is secure relies on the security of the underlying CLOW-GKA protocol. In Section 5, we provide a security proof for the protocol using a "proof by simulation" methodology.

This paper is organized in six sections. Section 2 briefly describes some related work in literature. Section 3 provides some preliminaries, including the elliptic curve cryptography and bilinear pairings. Section 4 presents the CLOW-GKA protocol for E2E email encryption. Section 5 proves the security of the protocol, followed by analyzing the efficiency in Section 6. The paper is concluded in Section 7.

II. Related work

Other than PGP, the authors found only one previous work [17], referred as YZL's protocol, in literature aimed at designing a key agreement protocol specific for E2E email encryption. However, this work proposed to use an identity-based protocol for E2E email encryption and thus has the key escrow problem.

Currently, some commercial email encryption software products available in the market today such as LuxSci SecureLine [18], ProtonMail [19] and Mailvelope [20] are email encryption software based on OpenPGP [3]. ProtonMail is an email service provider like Google which uses OpenPGP. Mailvelope is a chrome extension which can be integrated within the email service provider's interface. Mailvelope also has a firefox addon. LuxSci SecureLine also uses OpenPGP and provides end-to-end email encryption. In addition to the well-known certificate management problem, the disadvantages of using PGP/OpenPGP, as described earlier, are its limited trust and validity levels and its counter-intuitive key validation using certificates [35]. In our proposed CLOW-GKA, eliminating the need of certificates is one of the desired properties for building an end-to-end email encryption system.

Shamir [5] was the first person to propose the idea of Identity-based Public Key Cryptosystem ID-PKC, where every participants' public keys are generated from their identities through some public known hash functions. Thus, no public key verification is required. However, ID-PKC schemes require a Key Generation Center (KGC) to assign private keys for participants and this inevitably introduces the key escrow problem. Though the ID-PKC has the key escrow problem, some commercial email encryption software were still built based on ID-PKC such as HPE Secure email [21], Trend Micro email [22], DataMotion SecureMail [23], and Proofpoint [24].

In 2003, Al-Riyami et al. [11] designed the first CL-PKC (certificateless public key cryptosystem) to eliminate the key escrow problem by making the KGC to generate only a partial private key for each participant. Upon receiving a partial private key from KGC, each participant can then use it along with his/her private key to construct the public key. In this way, even a compromised or malicious KGC, with only the knowledge of the partial private key, will not be able to derive any participant's private key and thus the key escrow problem can be prevented. However, in their scheme the partial private keys have to be distributed to participants using some secret channels. In addition, the paper, based on their CL-PKC setting, has only sketched a 2-party (not n-party) key agreement protocol. Furthermore, the protocol is not one-way. That is, it requires back-and-forth message exchanges for both parties to agree on a key. For E2E email encryption, this key agreement protocol obviously is not suitable.

To summarize, most email encryption software products currently in the market still use the public key certificates for key management (for example, PGP/OpenPGP). Some newer email encryption software use ID-PKC to eliminate the burden of key management. However, they suffer on the key escrow problem. The proposed CLOW-GKA scheme aims at eliminating both problems based on certificateless protocols. In addition, our approach will be based on key agreement protocols, where the email is encrypted by a key which can be derived by all email participants. Thus, there is no need for multiple encryptions for an email with multiple receivers.

Table 1 provides a brief functionality comparison among the proposed CLOW-GKA protocol, YZL's (Yeh, Zeng and Long's) ID-PKC protocol [17] and Al-Riyami et al's CL-PKC [11], where the CLOW-GKA is a protocol designed for E2E email encryption, the YZL's protocol is designed for email encryption but has the key escrow problem and the Al-Riyami's CL-PKC is the first certificateless cryptosystem.

TABLE I
FUNCTIONALITY COMPARISON AMONG CLOW-GKA, YZL'S ID-PKC, AND AL-RIYAMI ET AL'S CL-PKC

	CLOW-GKA	YZL's ID-PKC	Al-Riyami's CL-PKC
Without Key Escrow problem	• YES	• NO	• YES
Without public key certificate infrastructure	• YES	• YES	• YES
One-way	• YES	• YES	• NO
n-party	• YES	• YES	• NO

III. PRELIMINARY

In this section, we describe the cryptographic basics of elliptic curves and bilinear pairings that are used frequently in ID-based and certificateless-based cryptosystems.

A. Elliptic curves

Elliptic curve cryptosystem (ECC) [25], [26] is a popular cryptosystem used in many different cryptographic protocols. To set up an ECC, based on the National Security Agency's recommendation [27], an appropriate prime curve is $E_p(a,b) : y^2 = x^3 + ax + b$ over a prime p and a base point (generator) P with an order q (i.e., $q \times P = O$), where $(4a^3 + 27b^2) \neq 0 \bmod p$ and O is the infinity point. An ECC $E_p(a,b)$ is an additive group with the following operation rules. For all points $A, B, C \in E_p(a,b)$,

1) $A + O = A$.
2) If $A = (x_A, y_A)$, then $-A = (x_A, -y_A)$ and thus $A + (x_A, -y_A) = O$.
3) Given $A = (x_A, y_A)$ and $B = (x_B, y_B)$ with $A \neq -B$, another point $C = A + B = (x_C, y_C)$ can be calculated by the rules below:

$$x_C = (\lambda^2 - x_A - x_B) \bmod p$$
$$y_C = (\lambda(x_A - x_C) - y_A) \bmod p$$

where

$$\lambda = \begin{cases} \frac{y_B - y_A}{x_B - x_A} \bmod p & \text{if } A \neq B \\ \frac{3x_A^2 + a}{2y_A} \bmod p & \text{if } A = B \end{cases}$$

4) Multiplication is defined as repeated additions. For example, for some positive integer k, the multiplication $k \cdot A = A + A + \ldots + A$, i.e., A adds itself $k - 1$ times.

There are many ECC encryption and signature algorithms available in literature. All these algorithms are based on the hardness assumption of the Elliptic Curve Discrete Logarithm Problem that states:

- Elliptic Curve Discrete Logarithm Problem (ECDLP): Given two points A and B on a curve $E_p(a, b)$ with an order q, if $B = r \cdot A$ for some $r \in Z_q^*$, it is computational hard to find r.

B. Bilinear pairings

Bilinear pairing has found recent use in various encryption and signature schemes [6]–[10], [28]–[30]. A symmetric bilinear pairings cryptosystem is described briefly below.

Let $(G_1, +)$ and $(G_2, \times)$ be two cyclic groups of the same prime order q, and let P 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:** $\forall\, A, B, C \in G_1$, and $\forall\, r_1, r_2 \in Z_q^*$,
  $e(A, B) = e(B, A)$
  $e(A, B + C) = e(A, B)e(A, C)$
  $e(r_1 A, r_2 B) = e(A, B)^{r_1 r_2} = e(r_2 A, r_1 B)$
- **Non-degeneracy:** If P is a generator of G_1, then $e(P, P)$ is a generator of G_2.
- **Computability:** $\forall\, A, B \in G_1$, there exists a polynomial-time algorithm to efficiently compute the bilinear mapping $e(A, B)$.

An elliptic curve is a typical example that can be used as the G_1 group. The security of most bilinear mapping based cryptographic algorithms is related to the hardness assumption of the following bilinear problems, which are:

- Computational Bilinear Diffie-Hellman Problem (CBDHP): Given $A, r_1 A, r_2 A$, for $r_1, r_2 \in Z_q^*$, compute $r_1 r_2 A$.
- Decision Bilinear Diffie-Hellman Problem (DBDHP): Given $A, r_1 A, r_2 A, r_3 A$, for $r_1, r_2, r_3 \in Z_q^*$, and an element $g \in G_2$, decide if $g = e(A, A)^{r_1 r_2 r_3}$.
- One Way Bilinearity Problem (OWBP): Given a random element $g \in G_2$, find a pair $(A, B) \in G_1$ such that $e(A, B) = g$.

There is no algorithm available currently which is able to efficiently solve these hard problems.

IV. The CLOW-GKA protocol for E2E email encryption

This section proposes the CLOW-GKA (certificateless one-way group key agreement) protocol and describes its application to E2E email encryption.

A. KGC servers

For the proposed CLOW-GKA protocol to allow an email sender and a group of n (≥ 1) email receivers to agree on a key for E2E email encryption, all of the email participants must register through a same KGC (key generation center). This can be easily achieved by letting the email service provider (such as Google) act as the KGC. For some security sensitive organizations, for example government agencies may want to establish their own email system for their employees only, the agency can also have a dedicated server serves as the KGC.

However, no matter in which cases (public or internal), the KGC will not be able to know any participant's private key, and thus not able to derive any agreed group key (i.e., email encryption key) and subsequently decrypt any of the emails if the KGC is compromised. Thus, the proposed CLOW-GKA protocol can implement a truly E2E email encryption scheme, i.e., no other entity (except the email sender and receivers) is able to decrypt the emails.

B. CLOW-GKA for E2E email encryption

In this section, we formally present the CLOW-GKA protocol for E2E email encryption. Within this protocol, several sub-protocols will be called and executed. The entities participating in the protocol are the KGC and the email users, where each email user can play different roles either as an email sender or as an email receiver.

CLOW-GKA protocol: E2E email encryption

Assume an email sender with an identity ID_0 would like to send an E2E encrypted email to $n \geq 1$ email receivers with identities $ID_i, \forall\, 1 \leq i \leq n$. This protocol is based on a cryptosystem with a set of parameters set up by the KGC. The KGC, ID_0, and all ID_i need to execute the steps described in this protocol.

1) The KGC server defines a cryptosystem and publishes a set of parameters $\langle G_1, G_2, e, P, P_{pub}, q, H_1, H_2 \rangle$, where parameters G_1, G_2, e, P and q were defined in Section III-B. The KGC also chooses a master secret $s \in Z_q^*$ and then computes $P_{pub} = sP$ as the system's public key. The KGC chooses a hash function $H_1 : \{0, 1\}^* \rightarrow G_1$. The map-to-curve and map-to-point algorithms from Weil pairings in [6], [9] are such functions that can map users' identities to points on an elliptic curve (the group G_1). Finally the KGC chooses another hash function $H_2 : Z_q^* \rightarrow Z_q^*$ that will be used in generating key generation keys. Secure Hash (SHA) algorithms are appropriate hash functions that can be used in this protocol.
2) Each email user, including ID_0 and all ID_i, $\forall\, 1 \leq i \leq n$, needs to have an email account before sending/receiving emails. Each email user executes the **Sub-protocol 1** to register an email account through the KGC. In Sub-protocol 1, each email user will generate his/her private and public keys.
3) Each email user, including ID_0 and all ID_i, publishes his/her public key along with his/her identity in a public directory.
4) The email sender ID_0 executes the **Sub-protocol 2** to send an encrypted email to all email receivers ID_i.
5) Each email receiver ID_i executes the **Sub-protocol 3** to receive and decrypt the email.

C. Email account registration

All email users need to register through the KGC first before they can send or receive any E2E encrypted email.

Sub-protocol 1: Email account registration

Input: An email user with an identity $ID_j, \forall\, 0 \leq j \leq n$, and a random number $r_j \in Z_q^*$.
Output: ID_j's private key $s_j \in Z_q^*$ and public key $\langle P_j, R_j \rangle$, where both P_j and $R_j \in G_1$.

A user ID_j registers an email account and generates his/her private and public keys through the KGC by the following steps:

1) ID_j computes and sends $r_j Q_j$ to KGC, where r_j is a random secret and $Q_j = H_1(ID_j)$.
2) KGC computes $D_j = s r_j Q_j$ and sends it back to ID_j through any public channel, in which a partial private key sQ_j is embedded in D_j.
3) ID_j chooses his/her private key s_j and then computes his/her public key $\langle P_j, R_j \rangle$, where

$$P_j = s_j P \tag{1}$$

$$R_j = s_j^{-1} r_j^{-1} D_j = s_j^{-1} r_j^{-1} r_j s Q_j = s_j^{-1} s Q_j \tag{2}$$

Note that in the step 2 of the Sub-protocol 1, unlike other ID-based or certificateless-based protocols, the proposed CLOW-GKA protocol does not require secret channels for the KGC to distribute the partial private key sQ_j. We use the binding-blinding technique to avoid the requirement of secret channels. Rather than sending the partial private key sQ_j directly, the KGC sends $D_j = s r_j Q_j$ to ID_j through any public channel, where only the ID_j knows the random secret r_j. Based on the hardness assumption of the Elliptic Curve Discrete Logarithm problem, it is hard for everyone (except the KGC and the user ID_j) to derive the partial private key sQ_j even he/she eavesdrops D_j from the public channel. The secrecy of the partial private key sQ_j is important to the security of the protocol since if a third party is able to eavesdrop sQ_j, then this party can create a fake public key for ID_j. This malicious party just needs to choose his/her own random secret s'_j, and based on the eavesdropped sQ_j, to generate $\langle P'_j, R'_j \rangle$ and then claims that this fake public key is ID_j's public key.

D. Sending an E2E encrypted email

The email sender ID_0 needs to execute the Sub-protocol 2 below to send an E2E encrypted email to a group of n email receivers $ID_i, \forall\, 1 \leq i \leq n$.

Sub-protocol 2: Sending an E2E encrypted email

To send an E2E encrypted email to n email receivers $ID_i, \forall\, 1 \leq i \leq n$, the sender ID_0 needs to perform the following steps:

1) ID_0 obtains each email receiver ID_i's public key $\langle P_i, R_i \rangle$ from the public directory.
2) ID_0 verifies each ID_i's public key by checking the equality of Equation 3 below.

$$e(P_i, R_i) \stackrel{?}{=} e(P_{pub}, Q_i) \tag{3}$$

since

$$\begin{aligned} e(P_i, R_i) &= e(s_i P, s_i^{-1} s Q_i) = e(sP, s_i s_i^{-1} Q_i) \\ &= e(P_{pub}, Q_i). \end{aligned}$$

If the equality checking in Equation 3 returns true, the validity of the public key $\langle P_i, R_i \rangle$ is verified.
3) ID_0 chooses a random number $r \in Z_q^*$ and computes

$$x_j = e(H_2(r) Q_0, s_0 P_j), \quad \forall\, 0 \leq j \leq n \tag{4}$$

4) ID_0 computes n key derivation keys y_i's, one for each email receiver ID_i, where

$$y_i = x_0 \oplus x_1 \oplus \ldots \oplus x_{i-1} \oplus x_{i+1} \oplus \ldots \oplus x_n \tag{5}$$

5) ID_0 computes the group key (will be used to encrypt the email message)

$$K = x_0 \oplus x_1 \oplus \ldots \oplus x_n \tag{6}$$

6) ID_0 uses the group key K to encrypt the email message and then sends the encrypted email along with the key derivation keys and the random number $(y_1, y_2, \ldots, y_n, r)$ as an attachment to all n email receivers ID_i's.

For Equation 3, the email sender only needs to use the KGC's public information P_{pub} and the email receiver's identity ID_i (or $Q_i = H_1(ID_i)$) to verify the validity of the ID_i's public key $\langle P_i, R_i \rangle$. This certificateless public-key verification capability (i.e., Any user can verify another user's public key without the need to have a third party to serve as a certificate authority) is an important feature for a well-designed certificateless protocol.

E. Receiving an E2E encrypted email

Upon receiving an E2E encrypted email from the email sender ID_0, each email receiver ID_i, $\forall\, 1 \leq i \leq n$, needs to execute the Sub-protocol 3 to derive the group key K and then use it to decrypt the message.

Sub-protocol 3: Receiving an E2E encrypted email

Each email receiver $ID_i, \forall\, 1 \leq i \leq n$, performs the following three steps:

1) ID_i first obtains the email sender ID_0's public key $\langle P_0, R_0\rangle$ from the public directory.
2) ID_i checks the validity of ID_0's public key $\langle P_0, R_0\rangle$ based on Equation 3.
3) ID_i derives the group key K using his/her own private key s_i, the key derivation key y_i, and the email sender ID_0's identity Q_0 ($= H_1(ID_0)$) and public key P_0. Equation 7 below describes this key derivation process.

$$K = y_i \oplus e(s_i Q_0, H_2(r)P_0) \tag{7}$$

since

$$\begin{aligned} & y_i \oplus e(s_i Q_0, H_2(r)P_0) \\ &= y_i \oplus e(H_2(r)Q_0, s_i P_0) \\ &= y_i \oplus e(H_2(r)Q_0, s_i s_0 P) \\ &= y_i \oplus e(H_2(r)Q_0, s_0 P_i) \\ &= y_i \oplus x_i \\ &= (x_0 \oplus \ldots \oplus x_{i-1} \oplus x_{i+1} \oplus \ldots x_n) \oplus x_i \\ &= K \end{aligned}$$

4) Finally, ID_i can decrypt the email using the just derived group key K.

In Equations 4 and 7, the reason of using $H_2(r)$, instead of r, to (re-)generate each x_j is to prevent a potential impersonation attack. The attack works as follows: If $x_j = e(rQ_0, s_0P_j)$, any receiver, for example ID_1, who can calculate his/her own x_1 and thus is able to compute each x_i ($1 \le i \le n$) from the transcript $(y_1, y_2, \ldots, y_n, r)$ from ID_0 (the sender). Then ID_1 just chooses a new random number r' to re-generate valid x'_j and y'_i as

$x'_j = e(rQ_0, s_0P_j)^{r'} = e(rr'Q_0, s_0P_j), \forall 0 \le j \le n$ and
$y'_i = x'_0 \oplus x'_1 \oplus \ldots \oplus x'_{i-1} \oplus x'_{i+1} \oplus \ldots \oplus x'_n \ \forall 1 \le i \le n$

Thus, ID_1 is able to impersonate ID_0 and send group emails by attaching the new transcript $(y'_1, y'_2, \ldots, y'_n, rr')$.

However if $x_j = e(H_2(r)Q_0, s_0P_j)$, the attack cannot be successful. Applying the same attack the receiver ID_1 first chooses a random number r' and re-generate x'_j as

$$x'_j = e(H_2(r)Q_0, s_0P_j)^{r'} = e(r'H_2(r)Q_0, s_0P_j)$$

Then in the transcript $(y'_1, y'_2, \ldots, y'_n, r'')$ the receiver ID_1 needs to give a random number r'' that makes $H_2(r'') = r'H_2(r)$. Based on the preimage resistance of hash functions, it is computationally hard to find such r''.

F. Example

For demonstration purpose, this section will give a walk through example. In this example, the email sender ID_0 is sending an email to $n = 3$ receivers ID_1, ID_2, and ID_3.

1) **Cryptosystem setup:** The KGC sets up a cryptosystem with all the parameters as described in the step 1 of CLOW-GKA protocol in Section IV-B.
2) **Email account registration:** Each of the users ID_0, ID_1, ID_2 and ID_3 registers an email account through the KGC and during the registration, their public keys are generated.
3) **Public key directory:** All the users ID_0, ID_1, ID_2 and ID_3 publish their public keys in a public directory.
4) **Sending an encrypted email:**
   a) The email sender ID_0 accesses the public directory to obtain all email receiver ID_1's, ID_2's and ID_3's public keys $\langle P_1, R_1\rangle, \langle P_2, R_2\rangle, \langle P_3, R_3\rangle$.
   b) ID_0, based on Equation 3, verifies the validity of all three public keys by checking
   $e(P_1, R_1) \stackrel{?}{=} e(P_{pub}, Q_1)$
   $e(P_2, R_2) \stackrel{?}{=} e(P_{pub}, Q_2)$
   $e(P_3, R_3) \stackrel{?}{=} e(P_{pub}, Q_3)$
   c) ID_0 picks a random number r and computes
   $$\begin{cases} x_0 = e(H_2(r)Q_0, s_0P_0) \\ x_1 = e(H_2(r)Q_0, s_0P_1) \\ x_2 = e(H_2(r)Q_0, s_0P_2) \\ x_3 = e(H_2(r)Q_0, s_0P_3) \end{cases}$$
   d) ID_0 computes three key derivation keys as follows.
   $$\begin{cases} y_1 = x_0 \oplus x_2 \oplus x_3 \\ y_2 = x_0 \oplus x_1 \oplus x_3 \\ y_3 = x_0 \oplus x_1 \oplus x_2 \end{cases}$$
   e) ID_0 generates the group (encryption) key
   $$K = x_0 \oplus x_1 \oplus x_2 \oplus x_3$$
   f) ID_0 encrypts the email using the group key K and sends (y_1, y_2, y_3, r) along with the email.
5) **Receiving an encrypted email:**
   a) Each email receiver ID_1, ID_2 or ID_3 accesses the public directory to obtain ID_0's public key $\langle P_0, R_0\rangle$.
   b) By Equation 3, each email receiver ID_1, ID_2 or ID_3 verifies the sender ID_0's public key $\langle P_0, R_0\rangle$ by checking $e(P_0, R_0) \stackrel{?}{=} e(P_{pub}, Q_0)$.
   c) ID_1 computes
   $$\begin{aligned} & y_1 \oplus e(s_1Q_0, H_2(r)P_0) \\ &= x_0 \oplus x_2 \oplus x_3 \oplus e(H_2(r)Q_0, s_1s_0P) \\ &= x_0 \oplus x_2 \oplus x_3 \oplus e(H_2(r)Q_0, s_0P_1) \\ &= x_0 \oplus x_2 \oplus x_3 \oplus x_1 = K \end{aligned}$$
   ID_2 computes
   $$\begin{aligned} & y_2 \oplus e(s_2Q_0, H_2(r)P_0) \\ &= x_0 \oplus x_1 \oplus x_3 \oplus e(H_2(r)Q_0, s_2s_0P) \\ &= x_0 \oplus x_1 \oplus x_3 \oplus e(H_2(r)Q_0, s_0P_2) \\ &= x_0 \oplus x_1 \oplus x_3 \oplus x_2 = K \end{aligned}$$
   ID_3 computes
   $$\begin{aligned} & y_3 \oplus e(s_3Q_0, H_2(r)P_0) \\ &= x_0 \oplus x_1 \oplus x_2 \oplus e(H_2(r)Q_0, s_3s_0P) \\ &= x_0 \oplus x_1 \oplus x_2 \oplus e(H_2(r)Q_0, s_0P_3) \\ &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 = K \end{aligned}$$
   d) Finally all three email receivers can decrypt the encrypted email using the group key K.

V. Security analysis

A. Adversarial model

CLOW-GKA protocol is secure against static and non-colluding adversaries in the semi-honest adversarial model, also called *honest-but-curious*. The adversary obtains the internal state of the corrupted party, including the transcripts of all messages received. It correctly follows the protocol specification while attempting to learn private information from other parties. In the rest of this section, we prove the security of our protocol in the setting of semi-honest adversarial model.

B. Privacy by simulation

According to the simulation paradigm [33], a protocol is secure against semi-honest adversaries if whatever can be computed by a party participating in the protocol can be computed based merely on its input and output. If the view of a party is simulatable based only on the party's input and output, then whatever the adversary can learn from the input and output of the corrupted party, can be learned without the execution of the protocol. If the view each party in the protocol is simulatable based only on the it's input and output, then we can conclude that the protocol is privacy-preserving, and consequently secure.

*1) **Definition of security**:*

- Let $f = (f_i, ..., f_m)$ be a m-ary probabilistic polynomial-time functionality and let Π be a multi-party protocol for computing f.
- The view of the *i*th party $(i \in 1, 2, .., m)$ during an execution of Π on $x = (x_1, ..., x_m)$ and security parameter n is denoted by $\mathsf{view}_i^{\Pi}(x, n)$ and equals $(w, r_i, m_{i,1}, \ldots, m_{i,t})$ where $w \in x = (x_1, ..., x_m)$ (its value depending on the value of i), r_i equals the contents of the *i*th party's internal random tape, and $m_{i,j}$ represents the *j*th message that it received.
- The output of the *i*th party during an execution of Π on $x = (x_1, ..., x_m)$ and security parameter n is denoted by $\mathsf{output}_i^{\Pi}(x, n)$ and can be computed from its own view of the execution. We denote the joint output of all parties by $\mathsf{output}_i^{\Pi}(x, n) = (\mathsf{output}_1^{\Pi}(x, n), ..., \mathsf{output}_m^{\Pi}(x, n))$

In the case where the functionality f is deterministic (our protocol is deterministic too), a simpler definition can be used. Specifically, we do not need to consider the joint distribution of the simulatorŁ output with the protocol output. Rather we separately require correctness, meaning that

$$\{\mathsf{output}^{\Pi}(x, n)\}_{x \in (\{0,1\}*); n \in N} \stackrel{c}{\equiv} \{f(x, n)\}_{x \in (\{0,1\}*); n \in N}$$

and in addition, that there exist probabilistic polynomial-time algorithm (simulator) S such that for each $I \subseteq \{1, \ldots, m\}$:

$$\{S_I(1^n, x_I, f_I(x))\}_{x \in (\{0,1\}*); n \in N} \stackrel{c}{\equiv} \{\mathsf{view}_I^{\Pi}(x, n)\}_{x \in (\{0,1\}*); n \in N}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability. □

According to the security definition above, it is sufficient to show that we can effectively simulate the view of each party in the two phases of CLOW-GKA protocol given only the input and output of that party, in order to prove the protocol is secure.

1. User Registration Phase:

Theorem 1: *Let f be a polynomial-time two-party single-output functionality that takes as input two private keys s_j from user ID_j and D_j from KGC, and outputs a public key pair $\langle P_j, R_j \rangle$. Then Sub-protocol 1 securely computes f in the presence of static semi-honest adversaries.*

Proof. In Sub-protocol 1, a user with an identity ID_j first sends a parameter to KGC. Then the KGC computes the partial private key and sends it back to ID_j, which uses it to computes its public key pairs. The interaction between KGC and ID_j is secure due to the binding-blinding technique which has the hardness assumption of the Elliptic Curve Discrete Logarithm problem [32]. According to the simulation paradigm [33], A protocol is secure against static semi-honest adversaries if whatever can be computed by a party participating in the protocol can be computed based on its input and output only.

Case 1 - User with identity ID_j is corrupted. We construct a simulator $\mathcal{S}_1$ that is given private key s_j and public key $\langle P_j, R_j \rangle$ as inputs, and it generates the view of ID_j in Sub-protocol 1. We need to prove that:

$$\{\mathcal{S}_1(s_j, \langle P_j, R_j \rangle)\}_{s_j, P_j, R_j \in \{0,1\}^*} \stackrel{c}{\equiv} \{\mathsf{view}_{ID_j}^{\Pi}(s_j, s)\}_{s_j, s \in \{0,1\}^*} \quad (8)$$

where $\mathcal{S}_1(s_j, \langle P_j, R_j \rangle)$ is the simulator as defined above and Π denotes Sub-protocol 1, by proving that each simulated message received by ID_j in Sub-protocol 1 is indistinguishable from a message received in the real world. In Step 2 of Sub-protocol 1, ID_j receives $D_j = sr_jQ_j$. Therefore, $\mathcal{S}_1$ must generate D_j and send it to ID_j such that ID_j is able to obtain $\langle P_j, R_j \rangle$. However, $\mathcal{S}_1$ cannot honestly generate D_j as it does not know KGC's master key s. The crux of this proof is in showing that D'_j generated by $\mathcal{S}_1$ is computationally indistinguishable from D_j that ID_j receives in the real protocol execution. $\mathcal{S}_1$ uniformly chooses random number r'_j that ID_j would use to generate r_jQ_j (in Step 1). Next, we know that

$$R_j = s_j^{-1} r'^{-1}_j D_j \quad (9)$$

So rewriting Equation 9 we get

$$D_j = R_j s_j r'_j$$

Since $\mathcal{S}_1$ knows R_j, s_j and r'_j, $\mathcal{S}_1$ is able to compute D_j without knowing s and sends it to ID_j. ID_j can now generate $\langle P_j, R_j \rangle$ as it does in the real execution.

Case 2 KGC is corrupted. We construct a simulator $\mathcal{S}_2$ that is given master key of KGC s and public key

$\langle P_j, R_j \rangle$ as inputs, and generates the view of KGC in Sub-protocol 1. We need to prove that:

$$\{\mathcal{S}_2(s, \langle P_j, R_j \rangle)\}_{s, P_j, R_j \in \{0,1\}^*} \stackrel{c}{\equiv} \{\mathsf{view}^{\Pi}_{KGC}(s_j, s)\}_{s_j, s \in \{0,1\}^*} \quad (10)$$

where $\mathcal{S}_2(s, \langle P_j, R_j \rangle)$ is the simulator as defined above and Π denotes Sub-protocol 1. In Step 1 of Sub-protocol 1, KGC receives $r_j Q_j$ from ID_j. $\mathcal{S}_2$ computes $Q_j = H_1(ID_j)$. Next, it uniformly chooses random number r'_j, computes $r'_j Q_j$ and sends it to KGC. $r_j Q_j$ is computationally indistinguishable from $r'_j Q_j$ as both r_j and r'_j were randomly chosen from a uniform distribution. ♢

2. Email Sending and Receiving Phase:

Theorem 2: *Let f:* $\{0,1\}^* \times ... \times \{0,1\}^* \rightarrow \{0,1\}^*$ *be a polynomial-time multi-party single-output functionality, and let Sub-protocol 1 be a secure user registration protocol in the presence of static semi-honest adversaries. Sub-protocol 2 and Sub-protocol 3 securely computes f in the presence of static semi-honest adversaries.*

Proof: In Sub-protocol 2, a sender with identity ID_0 sends an encrypted email with some parameters to n email receivers $ID_i, \forall 1 \leq i \leq n$. In Sub-protocol 3, each receiver ID_i receives an encrypted email with some parameters, derives the decryption key, and then decrypts the email. In Step 2 of both Sub-protocol 2 and Sub-protocol 3, the user verifies the validity of other user's public key, e.g., ID_0 verifies for the validity of ID_i's public key pair $\langle P_i, R_i \rangle$ by checking Equation 3, i.e., $e(P_i, R_i) \stackrel{?}{=} e(P_{pub}, Q_i)$ since $e(P_i, R_i) = e(s_i P, s_i^{-1} s Q_i) = e(sP, s_i s_i^{-1} Q_i) = e(P_{pub}, Q_i)$.

Although parameters P_i, R_i, P_{pub}, Q_i are all public, no secret information can be derived from these parameters due to the hardness assumption of the Elliptic Curve Discrete Logarithm problem [32]. Note that since in Sub-protocol 2 and Sub-protocol 3 the sender does not receive any message, there is no need to simulate the view of the sender. We only need to simulate the view of one receiver (let us assume it to be ID_1) in each sub-protocol to prove they are secure according to the simulation paradigm.

Case 3 - Receiver with identity ID_1 is corrupted. In this case, we construct a simulator $\mathcal{S}_3$ that is given encryption key K and private key s_1 of ID_1 as inputs, and generates the view of ID_1 in Sub-protocol 2 and Sub-protocol 3. $\mathcal{S}_3$ will simulate sender ID_0 and other receivers $ID_i, \forall 2 \leq i \leq n$. We need to prove that:

$$\{\mathcal{S}_3(s_1, K)\}_{s_1, K \in \{0,1\}^*} \stackrel{c}{\equiv} \{\mathsf{view}^{\Pi}_{ID_1}(s_1, (x_i, \forall 0 \leq i \leq n))\}_{s_1, (x_i, \forall 0 \leq i \leq n) \in \{0,1\}^*} \quad (11)$$

where $\mathcal{S}_3(s_1, K)$ is the simulator as defined above and Π denotes Sub-protocol 2 and Sub-protocol 3 combined. $\mathcal{S}_3$ uniformly chooses random number r' that ID_0 would choose. Notice that ID_0 first computes x_0 and $x_i, \forall 1 \leq i \leq n$ using its private key s_0 in the real execution. $\mathcal{S}_3$ cannot honestly compute x_0 and $x_i, \forall 1 \leq i \leq n$ because it does not know private key s_0 of ID_0. Next, ID_0 computes $y_i, \forall 1 \leq i \leq n$ and K and encrypts the email using K and sends $(y_1, y_2, \ldots, y_n, r)$ along with the encrypted email in the real protocol execution. The crux of this proof is in showing that $(y_1, y'_2, \ldots, y'_n, r')$ along with the encrypted email (generated by $\mathcal{S}_3$) is computationally indistinguishable from the real $(y_1, y_2, \ldots, y_n, r)$ along with the encrypted email that ID_1 receives in the real protocol execution. We know that:

$$x_1 = e(H_2(r')Q_0, s_0 P_1)$$

$\mathcal{S}_3$ computes x_1 with input (s_1, K) as follows:

$$\begin{aligned} x_1 &= e(s_1 Q_0, H_2(r') P_0) \\ &= e(H_2(r') Q_0, s_1 s_0 P) \\ &= e(H_2(r') Q_0, s_0 P_1) \text{ (as computed by } ID_0) \end{aligned}$$

$\mathcal{S}_3$ now knows s_1, Q_0, r', P_0, and $\mathcal{S}_3$. Since:

$$\begin{aligned} y_1 &= x_0 \oplus x_2 \oplus \ldots \oplus x_n \\ K &= x_0 \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n \end{aligned}$$

$\mathcal{S}_3$ computes y_1 as follows:

$$\begin{aligned} y_1 &= K \oplus x_1 \\ &= x_0 \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n \oplus x_1 \\ &= x_0 \oplus x_2 \oplus \ldots \oplus x_n \text{ (as computed by } ID_0) \end{aligned}$$

$\mathcal{S}_3$ now has K, r', y_1. It can now encrypt email using K as in real execution. Since $\mathcal{S}_3$ uses the same K as in real execution the encrypted email generated by $\mathcal{S}_3$ will be computationally indistinguishable from the encrypted email generated in the real execution. Even though $\mathcal{S}_3$ has all the parameters that receiver ID_1 needs to derive back K and decrypt the email we still need to make sure the message $(y_1, y'_2, \ldots, y'_n, r')$ along with the encrypted email is computationally indistinguishable from the real execution message. So $\mathcal{S}_3$ has to compute $y'_2, \ldots, y'_n$. But $\mathcal{S}_3$ does not know x_0 and $x_i, \forall 2 \leq i \leq n$. Therefore, $\mathcal{S}_3$ randomly chooses n-1 elliptic curve points and applies bilinear mapping to generate x'_0 and $x'_i, \forall 2 \leq i \leq n$, from which it computes $y'_2, \ldots, y'_n$ as shown in Equation 5. Next, $\mathcal{S}_3$ sends $(y_1, y'_2, \ldots, y'_n, r')$ along with the encrypted email to receiver ID_1. Since x'_0 and $x'_i, \forall 2 \leq i \leq n$ were randomly generated and based on the One-way Bilinearity problem [34], $y'_2, \ldots, y'_n$ are computationally indistinguishable from the real $y_2, \ldots, y_n$. ♢

Theorem 3: *Let* Π *be a polynomial-time multi-party protocol which is CLOW-GKA protocol: E2E email encryption and let Sub-protocol 1, Sub-protocol 2 and Sub-protocol 3 be secure in the presence of static semi-honest adversaries, then* Π *securely executes in the presence of static semi-honest adversaries.*

Proof: By Theorem 1 it is possible to securely execute Sub-protocol 1 and by Theorem 2 it is possible to securely

execute Sub-protocol 2 and Sub-protocol 3 assuming the existence of static semi-honest adversaries. Furthermore, recall that CLOW-GKA protocol: E2E email encryption is made up of the sub-protocol blocks mentioned above. Combining these facts with Theorem 3 we can conclude that CLOW-GKA protocol: E2E email encryption is secure in the presence of static semi-honest adversaries.

VI. Efficiency analysis

In this section we analyze our proposed CLOW-GKA protocol: E2E email encryption scheme. First, we analyze the computational cost for each of our protocol blocks. Secondly, we analyze the efficiency with respect to email size. Finally, we compare the efficiency of our protocol with other related protocols with the use of a table. In our analysis we use the following notations for the operations associated with our CLOW-GKA protocol.

PM	Point multiplication in group G_1
BP	Bilinear pairing
HASH1	[6], [9] defined map-to-point hash algorithm as $H : \{0,1\}^* \rightarrow G_1$
HASH2	This hash function is defined as $H : Z_q^* \rightarrow Z_q^*$
HASH3	[11] defined this hash function as $H : G_2 \rightarrow \{0,1\}^n$, where n is the bit length of plaintexts

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 [29]. Also, a PM operation of E/F_3163 requires only a few hundred modular multiplications in F_3163.

A. Computational cost of the protocol

In this subsection we are going to analyze the computational cost of the main protocol block and its sub-protocol blocks.

1) CLOW-GKA protocol: E2E email encryption:

In this protocol block, the KGC defines a cryptosystem and publishes a set of parameters. The KGC uses only one PM operation to compute the system's public key.

2) Sub-protocol 1:

Each email user needs to register an email account through the KGC. The user first uses a HASH1 and a PM operation to send a parameter to KGC. The KGC again uses a PM operation to compute the partial private key and sends it back to the user. The user computes its public key pair using 2 PM operations as shown in Equations 1 and 2. So the total computational cost for each user is (4 PM + 1 HASH1).

3) Sub-protocol 2:

To send an email to n recipients, the sender needs to

1) verify n recipients' public key by Equation 3. This requires 1 HASH1 and 2 BP operations for verifying each recipient's public key.
2) calculate $x_0, x_1, \ldots, x_n$: By Equation 4, each computation needs 1 HASH2, 2 PM, and 1 BP operations.
3) calculate $y_1, y_2, \ldots, y_n$: By Equation 5, the calculation of each y_i requires $\oplus$ing all x_j's $\forall\, j \neq i$.
4) derive the encryption key K. By Equation 6, this derivation requires $\oplus$ing all x_i's.
5) AES encrypt the email by 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 public key verification, AES encryption and the calculation of $X = \{x_0, x_1, \ldots, x_n\}$. Since each x_i calculation requires 2 PM, 1 HASH2, and 1 BP operations and each public key verification requires 1 HASH1 and 2 BP operations, the total cost for this block is n HASH1, $n+1$ HASH2, $(2n+2)$ PM and $(3n+1)$ BP operations.

4) Sub-protocol 3:

To receive an email, a recipient needs to

1) verify the sender's public key by Equation 3. This requires 1 HASH1 and 2 BP operations.
2) re-construct the group key using Equation 7. This requires 1 HASH2, 2 PM and 1 BP operation (the $\oplus$ operation is again ignored).
3) AES decrypt the message using the re-constructed group key K.

So the total cost is 1 HASH1, 1 HASH2, 2 PM and 3 BP operations. 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.

B. Email size analysis

Our proposed CLOW-GKA for E2E email encryption is an $n-$party protocol and so the size of an email will increase depending on the number of recipients. The most important factor for the email size increase is due to the inclusion of key derivation keys $(y_1, y_2, \ldots, y_n)$. Each key derivation key will have the same size as the key of the selected curve, since our scheme uses an elliptic curve to implement the group G_1 in the bilinear pairings. An elliptic curve with a key size of $\approx 2k$ bits is required for the security of a symmetric encryption scheme with an k-bit key according to the National Institute of Standards and Technology. So the size increase of an email to n recipients would be $\approx 256 \times n$ bits when using an elliptic curve with a 128-bit symmetric encryption security strength.

C. Efficiency comparison with different protocols

In Section II Related Work, we have selected two other protocols besides the proposed CLOW-GKA for the comparison of functionality in Table I. Table II below provides a brief efficiency comparison among these three protocols again.

Al-Riyami et al's CL-PKC (certificateless public key cryptosystem) [11] is not an n-party scheme. For fair comparison, we assume that it performs one CL-PKC encryption for each of the n email recipients. In Table II, our CLOW-GKA protocol is slower than those in YZL's and Al-Riyami's protocols by a constant factor (about 2 times slower in average). Though the

TABLE II
COMPARISON AMONG CLOW-GKA, YZL'S ID-PKC, AND AL-RIYAMI ET AL'S CL-PKC, ASSUMING THERE ARE n RECIPIENTS IN AN EMAIL

	CLOW-GKA	YZL's ID-PKC	Al-Riyami's CL-PKC
System Setup	• 1 PM	• None	• 1 PM
Key distribution and generation	• 1 HASH1 • 4 PM	• 1 HASH1 • 1 PM	• 1 HASH1 • 2 BP • 4 PM
Encryption (Sending email)	• $(2n+2)$ PM • $(3n+1)$ BP • (n) HASH1 • $(n+1)$ HASH2 • Symmetric key email encryption	• $(n+1)$ PM • $(n+1)$ BP • $(n+1)$ HASH1 • Symmetric key email encryption	(n) CL-PKC email encryptions contain: • (n) HASH1 • (n) HASH3 • $(3n)$ BP • $(2n)$ PM
Decryption (Receiving email)	• 2 PM • 3 BP • 1 HASH1 + 1 HASH2 • Symmetric key email decryption	• 1 PM • 1 BP • 1 HASH1 • Symmetric key email decryption	CL-PKC email decryption contains • 1 BP • 1 HASH3
Message size	• Encrypted message plus all y_i's (the key agreement info)	• Encrypted message plus all y_i's (the key agreement info)	• (n) CL-PKC encrypted message

proposed CLOW-GKA has a minor performance disadvantage, it should still be a better choice for E2E email encryption since it provides a better functionality than others (see Table I).

VII. CONCLUSION

To protect email privacy, an end-to-end email encryption scheme is required, where only the email sender and receivers can decrypt the emails. In this paper, we proposed a certificateless one-way group key agreement protocol with nice features which are suitable for implementing end-to-end email encryption schemes. These features are (1) certificateless (and thus no public-key certificate infrastructure is required and does not have key escrow problem) , (2) one-way, and (3) n-party. This paper also gives a security proof of the proposed protocol using a "proof by simulation" method. This method is relatively new and thus very few examples (or case studies) can be found in literature. The proof presented in this paper can be a good reference for future researchers who would like to use this methodology for security proof. Finally, the efficiency of the protocol is analyzed at the end of the paper.

REFERENCES

[1] P. Zimmermann, "Why I wrote PGP?" 1991. https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html
[2] The International PGP Homepage, "PGP Documentation," http://www.pgpi.org/doc/
[3] The OpenPGP homepage, http://openpgp.org
[4] X. Lai and J. Massey, "A Proposal for a New Block Encryption Standard," EUROCRYPT, pp. 389-404, 1990.
[5] A. Shamir, "Identity-based Cryptosystems and Signature Schemes," CRYPTO'84, LNCS 196, pp. 47-53, 1985.
[6] D. Boneh, M. Franklin, "Identity-based encryption from the Weil pairing," CRYPTO'01, pp. 213-229, 2001.
[7] C. Gentry, A. Silverberg, "Hierarchical ID-based cryptography," Advances in Cryptography - ASIACRYPT'02, Springer-Verlag, pp. 548-566, 2002.
[8] F. Zhang, K. Kim, "ID-based blind signature and ring signature from pairings," Advances in Cryptography - ASIACRYPT'02, Springer-Verlag, pp. 533-547, 2002.
[9] X. Yi, "An Identity-based Signature Scheme from the Weil Pairing," IEEE Communications Letter, Vol. 7, No. 2, pp. 76-78, 2002.
[10] H. Elkamchouchi, Y. Abouelseoud, "A new proxy identity-based signcryption scheme for partial delegation of signing rights," Cryptology ePrint Archive, Report 2008/041, 2008.
[11] S. Al-Riyami and K. Paterson, "Certificateless public key cryptography," Advances in Cryptology - ASIACRYPT'03, pp. 1-40, 2003.
[12] H. Xiong, F. Li, and Z. Qin, "Certificateless threshold signature secure in the standard model," Information Sciences, Vol. 237, pp. 73-81, 2013.
[13] R. Tso, X. Huang, and W. Susilo, "Strongly secure certificateless short signatures," Journal of Systems and Software, vol. 85, no. 6, pp. 1409-1417, 2012.
[14] J. Baek, R. Safavi-Naini, and W. Susilo, "Certificateless public key encryption without pairing," 8th International Information Security Conference (ISC 2005), LNCS 3650, pp. 134-148, 2005.
[15] G. Yang and C.H. Tan, "Strongly secure certificateless key exchange without pairing," ASIACCS'11, pp.71-79, 2011.
[16] M.L. Das, "A key escrow-free identity-based signature scheme without using secure channel," Cryptologia, Vol. 35, No. 1, pp. 58-72, 2011.
[17] J.H. Yeh, F. Zeng, and T. Long, "E2E email encryption by an identity-based one-way group keyagreement protocol," 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2014), pp. 760-767, 2014.
[18] LuxSci SecureLine, 2011, https://luxsci.com/extranet/secure-email.html
[19] ProtonMail, 2013, https://protonmail.com/security-details
[20] Mailvelope, 2014, https://www.mailvelope.com/en/
[21] HPE Secure Email, 2015, https://www.voltage.com/products/email-security/hpe-securemail/
[22] Trend Micro email, 2008, http://www.trendmicro.com/us/enterprise/network-web-messaging-security/email-encryption/
[23] Data-Motion SecureMail, 2010, https://www.datamotion.com/products/securemail/
[24] Proofpoint Email Protection, 2005, https://www.proofpoint.com/us/products/email-protection
[25] N. Koblitz, "Elliptic Curve Cryptosystems," Mathematics of Computation, Vol. 48, pp. 203-209, 1987.
[26] V. Miller, "Use of Elliptic Curves in Cryptography," CRYPTO'85, pp. 417-426, 1985.
[27] National Security Agency, "Recommended Set of Advanced Cryptography Algorithms - Suite B", 2005.
[28] 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.
[29] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, M. Scott, "Efficient algorithms for pairing-based cryptosystems," CRYPTO'02, pp. 354-368, 2002.
[30] D. Boneh, B. Lynn, H. Shacham, "Short signature from the Weil pairing," Advances in Cryptography - ASIACRYPT'01, Springer-Verlag, pp. 514-532, 2001.
[31] O Goldreich "Foundations of Cryptography," Cambridge University Press,New York, NY, USA, Vol. 2, 2004.
[32] D. Hankerson, A. Menezes "Elliptic Curve Discrete Logarithm Problem," Encyclopedia of Cryptography and Security Springer US, Boston, MA, pp. 397-400, 2011.
[33] C. Hazay, Y. Lindell "Efficient Secure Two-Party Protocols," Springer-Verlag Berlin Heidelberg, Vol. 2, 2010.
[34] M. B. Barbosa "Identity Based Cryptography From Bilinear Pairings," Universidade do MinhoCampus de Gualtar Braga Portugal, 2005.
[35] R. Haenni, J Jonczy "A new approach to PGP's web of trust," EEMA 2007: European e-Identity Coference, Paris, France, 2007