JOURNAL OF COMMUNICATIONS AND NETWORKS, VOL. 10, NO. 1, MARCH 2008

Many-to-One Encryption and Authentication Scheme and Its Application

Xi-Jun Lin, Chuan-Kun Wu, and Feng Liu

Abstract: **This paper is to study a subclass of group-oriented cryptographic scheme: Many-to-one encryption and authentication scheme. The many-to-one encryption and authentication scheme is to solve a practical problem, i.e., the scenario that the number of the receivers is very small compared with the number of the senders and a receiver may serve millions of senders. Compared with the traditional methods, the burdens of the receiver and the KGC are reduced greatly. How to revoke a sender from his receiver's legitimate sender group is also proposed and it is efficient compared with some traditional methods. The proposed scheme is proven in the random oracle models. The computational complexity of our scheme is independent of the number of the senders. At the end of the paper, an example is given to show how to use our scheme in online software registration and update.**

Index Terms: **Authentication, dynamic accumulator, encryption, group-oriented cryptography, identity, many-to-one, pairing.**

I. INTRODUCTION

The concept of group-oriented cryptography was first introduced by Desmedt in [1], and has been developed in [2]–[6]. It has been developed to apply to distributed systems and has some advantages over the traditional individual user-based schemes. At first, two examples are given to show why we write this paper.

Today how to filter spam emails is a hard problem. Unfortunately, many spam emails (e.g., advertisements) are designed to be filter-resistant. If some encryption scheme can help to filter the spam emails, it would be very useful where confidential information is often communicated which needs to be encrypted.

How to publish softwares securely is another hard problem. Some software companies provide build-in online registration function to verify whether the software user used is genuine. The online registration should be provided with confidentiality and authentication to assure the information sent to the registration server is correct. The information may contain the serial number published with the software, the parameters of user's hard disk and the medium access control (MAC) address etc. After the verification is successful, the user can use the full version software, otherwise the limited version. For example, in the anti-virus softwares, only the full version software can download the new virus database from the online update server.

Manuscript received December 15, 2006; approved for publication by Huaxiong Wang, Division I Editor, July 25, 2007.

X.-J. Lin and F. Liu are with the State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, and Graduate School of Chinese Academy of Sciences, Beijing, email : {linxj77, liufeng}@is.iscas.ac.cn.

C.-K. Wu is with the State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing, email : ckwu@is.iscas.ac.cn.

This work was supported by National 973 project of China no.2007CB807902 and National 863 project of China no.2006AA01Z423.

From above two examples we summarize as follows: 1) The message sent to a server should be encrypted; 2) authentication should also be provided to make the server know who the sender is, that is, one sender cannot personate another; 3) compared with the number of the senders, the number of the servers is very small and each server serves millions of senders, we call it many-to-one scenario; 4) only the ciphertext constructed by a legitimate sender can be decrypted and verified successfully by his server. For example, for an online registration server, the softwares with the correct production serial numbers are its legitimate senders, and for a secure email server, an email sender who has the correct email address is its legitimate sender; 5) the sender revocation should also be provided. Revoking a legitimate sender does not affect the computational complexity of the server and other senders.

In a general way, there are two methods to achieve it: 1) Each legitimate sender shares a distinct key with the server; 2) any legitimate sender shares the same public key and the server has the private key, and each legitimate sender has a distinct signature key to authenticate himself to the server. However, there are some disadvantages in the above two methods. In the first method, the server has to keep too many secret keys since the number of the legitimate senders is very large, and it is a time costing operation to find the suitable key for decryption, which makes the scheme less useful. In the second method, the server has to keep too many keys for verifying the signature. Someone may think that if an identity-based signature is used, the server does not keep any key for verification, but how to revoke an identity is also a hard problem of identity-based cryptosystem [7]. There are some other paradigms such as certificate-based public key schemes [8] and certificateless public key schemes [9], [10], etc., but we can claim that in this many-to-one scenario, these schemes have some similar disadvantages as above. It is significant to propose a new paradigm under this many-to-one scenario to over these disadvantages.

In this paper, we design a subclass of group-oriented scheme called many-to-one encryption and authentication scheme for this scenario. In the proposed paradigm, the server (i.e., the receiver) has unique key for decryption and authentication, compared with the first method above, the key management of the receiver becomes much simpler. Each legitimate sender has a distinct key for constructing a ciphertext and a method to revoke a legitimate sender is also employed, compared with the second method above, the revocation is easy to achieve. Since that issuing encryption keys to the legitimate senders is a burdensome work for a receiver, a key generation center (KGC) is employed to setup the system and issue the encryption keys to the legitimate senders. The receiver first needs to register information of its private key to the KGC, and nominates its legitimate senders.

Its private key cannot be extracted from these information by the KGC. The task of the KGC is to verify the identities and issue encryption keys to the legitimate senders. The encryption key of each legitimate sender is distinct. After receiving the encryption keys, the legitimate senders can send ciphertexts to their receiver. The receiver always uses its unique private key to decrypt and authenticate them. The identity which is unique for each sender is also involved in authentication. The concept and the signification of the identity is similar to that in the certificateless public key schemes. Note that a sender is legitimate for one receiver, but may not be legitimate for another receiver.

In general, our many-to-one encryption and authentication scheme should have the following properties:

- Confidentiality: A ciphertext generated by a legitimate sender can only be decrypted by his receiver.
- Soundness: Apart from the KGC, only the legitimate senders can construct valid ciphertexts for their receiver. The illegitimate senders can generate a valid ciphertext only with negligible probability.
- Unique private key: Only one private key for decryption, integrity verification and authentication is holden by a receiver. The advantage is to reduce the receiver's burden of key management.
- No key escrow: A valid ciphertext can only be decrypted and authenticated by its receiver. A valid ciphertext means a ciphertext constructed with the encryption key issued by the KGC honestly.
- No public keys: Besides the system parameters, there are no keys which have to be published in many-to-one scheme. The advantage is that no public key directory is needed, which can save the KGC's storage and reduce its burden. There may be parameters published for each receiver, which does not increase the burden of the KGC very much without publishing the senders' information since compared with the number of the senders the number of the receivers is very small.

In the many-to-one scenario, how to revoke a legitimate sender efficiently is a hard problem.There have been some methods to achieve it. In some systems, white list or black list is used to filter the senders, but the time of matching the sender increases with the length of the list. It is not practical if the number of the senders is very large. The method to revoke a public key certificate in the public key system can be also used in the sender revocation problem, such as CRL (certificate revocation list), OCSP (online certificate status protocol). CRL is a very inefficient proposal, which is similar as the black list. There are also some refinements to this method, such as delta CRLs, but the transmission costs and the infrastructural costs necessary to enable the transmission are still quite high. Another method can be derived from OCSP, the KGC responds to a sender status query by generating a fresh signature on the sender's current status. It increases transmission costs to a single signature per query, and it also is susceptible to DoS attack. The subset covers model [11] can be employed to solve the sender revocation problem but the KGC should maintain a tree structure which will increase the burden of the KGC. A method can be derived from "Novomodo" proposed by Micali [12], [13] which has some advantages over CRL and OCSP while in each period the KGC should compute a new hashing value for each unrevoked sender, if the number of the unrevoked senders is large the computational cost of the KGC in each period is also high even the computational complexity of hashing is small.

In this paper, an easier method to provide sender revocation is proposed. A sender who is revoked from the receiver's group cannot construct a valid ciphertext any more. The computational complexity does not increase with the number of the senders. In our method, revoking a receiver and his legitimate sender group is also easy.

This paper is organized as follows: Some definitions and assumptions are given in Section II. In Section III, the definition and security models of many-to-one scheme are introduced. In Section IV, the proposed scheme and its security are introduced. How to revoke a legitimate sender is proposed in Section V and in Section VI how to apply this scheme is proposed followed by the last section to conclude our work.

II. PRELIMINARIES

In this section, we describe the concepts and definitions of bilinear pairing and some assumptions.

A. Bilinear Pairing

Let $\mathcal{G}$ be an additive group of prime order p, $\mathcal{F}$ be a multiplicative group of the same order. Bilinear pairing is a map $\hat{e} : \mathcal{G} \times \mathcal{G} \longrightarrow \mathcal{F}$ which satisfies the following properties:

- Bilinearity: Given any $P, Q \in \mathcal{G}$ and $a, b \in Z_p^*$, we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab} = \hat{e}(abP, Q)$, etc.
- Non-Degeneracy: There exists a $P \in \mathcal{G}$ such that $\hat{e}(P, P) \neq 1$.
- Computability: $\hat{e}(P, Q)$ can be computed in polynomial time.

Modified Weil pairing [14] is this kind of map, and it is employed in our proposed scheme. The existence of the bilinear pairing has two implications as follows:

Menezes *et al.* [15] show that the discrete log problem in group $\mathcal{G}$ is no harder than the discrete log problem in $\mathcal{F}$. Let $P, Q \in \mathcal{G}$, the discrete log problem in $\mathcal{G}$ is to find an element $a \in \mathrm{Z}_{\mathrm{p}}^*$, such that $Q = aP$. Let $g = \hat{e}(P, P)$, and $h = \hat{e}(P, Q)$, we can see that $h = g^a$ according to bilinearity of map $\hat{e}$. Then, by Non-degeneracy we know that g and h are the elements of order p in group $\mathcal{F}$. So we can reduce the discrete log problem in $\mathcal{G}$ to the problem in $\mathcal{F}$.

DDH (Decision Diffie-Hellman) problem is easy: DDH problem in $\mathcal{G}$ is easy. Given $P, aP, bP, cP \in \mathcal{G}$, we can check whether or not it is true that $c = ab \pmod p$ from $\hat{e}(P, cP) = \hat{e}(aP, bP)$. However, the CDH (Computational Diffie-Hellman) problem, BDH (Bilinear Diffie-Hellman) problem and BDDH (Bilinear Decision Diffie-Hellman) problem are still hard.

B. Assumption

Let $\mathcal{G}$ be a bilinear additive group of prime order p and $\mathcal{F}$ be a multiplicative group of the same order. Let P be a generator of $\mathcal{G}$.

Definition 1 [Bilinear Decision Diffie-Hellman Problem (BDDH) [16]]: Given a tuple $(P, xP, yP, zP) \in \mathcal{G}$ and $T \in \mathcal{F}$

randomly as input, output $b \in \{0, 1\}$. An algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving BDDH problem in $\mathcal{G}$ if

$|Pr[\mathcal{B}(P, xP, yP, zP, \hat{e}(P, P)^{xyz}) = 0] - Pr[\mathcal{B}(P, xP, yP, zP, T) = 0]| \geq \epsilon$.

where the probability is over the random choice of $x, y, z \in Z_p^*$, the random choice of $T \in \mathcal{F}$, and the random bits of $\mathcal{B}$.

Assumption 1: We say that the BDDH assumption holds in $\mathcal{G}$ if no polynomial time algorithm has advantage at least ϵ in solving BDDH problem in $\mathcal{G}$.

Assumption 2[d-CAA (Collusion Attack Algorithm with d traitors) [17] Assumption]: For an integer d, and $s \in Z_p^*$ randomly, $P \in \mathcal{G}$, given $(\mathcal{G}, P, sP, (h_1, \frac{1}{h_1+s}P), \ldots, (h_d, \frac{1}{h_d+s}P))$, where $h_i \in Z_p^*$ randomly and distinct for $1 \leq i \leq d$, computing $(h, \frac{1}{h+s}P)$ for some $h \in Z_p^*$ but $h \notin \{h_1, \ldots, h_d\}$ is hard.

III. MANY-TO-ONE ENCRYPTION AND AUTHENTICATION SCHEME

In this section, we present a formal definition of the many-to-one encryption and authentication scheme and the security models.

A. Definition

Definition 2: A many-to-one encryption and authentication scheme is specified by six algorithms.

- **Setup**: This algorithm takes security parameter k as input and returns the system parameters $params$ and MK. $params$ includes a description of the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. This algorithm is run by the KGC. $params$ are publicly and authentically available, and MK is kept as the master key by the KGC.
- **Private-Key-Extract**: This algorithm takes $params$ and a receiver's identity ID_S as inputs, and returns a private key SK_S. SK_S is used to decrypt and authenticate the ciphertexts. Usually this algorithm is run by the receiver himself.
- **Information-Extract**: This algorithm takes $params$, a receiver's private key SK_S and his identity ID_S as inputs, and returns the information $Info_S$ with respect to SK_S. This algorithm is run by the receiver and $Info_S$ is transported to the KGC over an authentic channel. SK_S should not be extracted from $Info_S$ by the KGC and other persons.
- **Encryption-Key-Extract**: This algorithm takes $params$, MK, $Info_S$, an identity $ID_A \in \{0, 1\}^*$ of a legitimate sender A, and his receiver's identity $ID_S \in \{0, 1\}^*$, as inputs. It returns the encryption key $EK_{A,S}$ for the legitimate sender. Usually this algorithm is run by the KGC and its output is transported to A over a confidential and authentic channel.
- **Encrypt**: This algorithm takes $params$, $EK_{A,S}$ and a message $M \in \mathcal{M}$ as inputs. It returns a ciphertext $C \in \mathcal{C}$. This algorithm is usually run by the legitimate sender A for his receiver S.
- **Decrypt**: This algorithm takes $params$, SK_S, the sender's identity ID_A, his receiver's identity ID_S and $C \in \mathcal{C}$ as inputs. It returns a message $M \in \mathcal{M}$ or a message indicating a decryption failure (if the ciphertext is not in the correct form or not sent by a legitimate sender whose identity is ID_A). This algorithm is usually run by the receiver.

From above we say that the message M should result from applying algorithm **Decrypt** with $params$, SK_S, ID_A, and ID_S as inputs on a ciphertext C generated by using algorithm **Encrypt** with $params$, $EK_{A,S}$ as inputs on the message M.

The identity in many-to-one scheme is only used for authentication. Another feature is that only the receiver's legitimate senders can generate a valid ciphertext apart from the KGC. The special issuing key process of the KGC is to achieve the confidentiality, data integrity and authentication at one time since the ciphertext is constructed with the encryption key which contains the information with respect to the receiver's private key and the legitimate sender's identity. To distinguish each receiver, we assume that a receiver also has a unique identity.

B. Security Model

From the above definition, we can see that many-to-one encryption and authentication scheme should satisfy confidentiality and authentication. In real attack, two types of adversaries should be considered: an adversary equipped with the master key tries to extract the plaintext from a ciphertext such as an eavesdropping KGC, and an adversary who is not equipped with the master key but may be equipped with a receiver's private key tries to forge a ciphertext pretended from an legitimate sender to the receiver. In this section, we define adversaries for such a scheme. The standard definition for security of a public key encryption scheme involves indistinguishability of ciphertexts against an adaptive chosen ciphertext (IND-CCA2) attacker. Two models are derived from it for many-to-one scheme: the models for confidentiality and authentication respectively. In these models, there are two parties, the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$.

The following is a list of the actions that a general adversary $\mathcal{A}$ against a many-to-one scheme may carry out.

- **Extract private key of a receiver** S: $\mathcal{C}$ responds by running algorithm **Private-Key-Extract** to generate the private key SK_S of the receiver S.
- **Extract information of a receiver** S**'s private key**: $\mathcal{C}$ responds by first running algorithm **Private-Key-Extract** to generate the private key SK_S and then running algorithm **Information-Extract** to generate the information with respect to SK_S.
- **Extract encryption key of a sender** A **and his receiver** S: $\mathcal{C}$ responds by first running algorithm **Private-Key-Extract** and **Information-Extract** to generate $Info_S$ with respect to the receiver S and then running algorithm **Encryption-Key-Extract** to generate the encryption key for the sender A.
- **Encryption query for plaintext** M**, the sender** A **and his receiver** S: $\mathcal{C}$ responds by first running algorithm **Encryption-Key-Extract** to generate the encryption key $EK_{A,S}$, and then running algorithm **Encrypt** on plaintext M with $EK_{A,S}$.
- **Decryption query for ciphertext** C**, the sender** A **and his receiver** S: $\mathcal{C}$ responds by running algorithm **Private-Key-Extract** to obtain the private key SK_S, then running algo-

rithm **Decrypt** on ciphertext C with SK_S and the identity ID_A.

The first model is to consider the adversaries who are equipped with the master key such as an eavesdropping KGC. The goal of this type of adversary is to extract the plaintext from a ciphertext.

Model 1: We say that a many-to-one encryption and authentication scheme is secure against an adaptive chosen ciphertext attack if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:

- **Setup:** The challenger takes a security parameter k and runs the algorithm **Setup**, and gives the resulting system parameters $params$ and master key MK to $\mathcal{A}$.
- **Phase 1:** $\mathcal{A}$ issues a sequence of requests, each request being either private key extraction, information extraction, encryption key extraction or decryption query. These queries may be asked adaptively.
- **Challenge:** Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length messages $m_0, m_1 \in \mathcal{M}$ for challenge with a challenge identity pair (ID_A^*, ID_S^*), ID_A^* is a sender A's identity and ID_S^* is his receiver S's identity. The challenger picks a random bit $\beta \in \{0,1\}$ and constructs ciphertext C^* on m_β with the encryption key relative to A and S. C^* is output to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ issues a second sequence of requests as in Phase 1.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$ and wins the game if $\beta' = \beta$.
  The probability of success $Pr[\beta = \beta']$ to be computed over all possible choices of m_0, m_1, C^*. The advantage is defined as $\epsilon \in [\frac{-1}{2}, \frac{1}{2}]$ such that $Pr[\beta = \beta'] \geq \frac{1}{2} + \epsilon$ (The scheme is said to be secure if it is impossible to get a non-negligible positive advantage ϵ).

In this model, there are some restrictions: ID_S^* should not be queried to extract its private key. The ciphertext C^* should not be queried to decryption query with the identities (ID_A^*, ID_S^*).

We also consider the adversaries who are not equipped with the master key, but may be equipped with a receiver's private key, want to forge the ciphertexts pretended from a legitimate sender to his receiver. The model is as follows:

Model 2: We say that a many-to-one scheme is secure against an adaptive chosen message attack if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:

- **Setup:** The challenger takes a security parameter k and runs the algorithm **Setup**, and gives the resulting system parameters $params$ to $\mathcal{A}$. The master key MK is kept secretly.
- **Queries:** $\mathcal{A}$ issues a sequence of requests, each request being either private key extraction, information extraction, encryption key extraction, encryption query or decryption query. These queries may be asked adaptively.
- **Challenge:** Once $\mathcal{A}$ decides that **Queries** phase is over, it outputs a new ciphertext C^* whose sender's identity is ID_A^* and receiver's identity is ID_S^*. $\mathcal{A}$ wins the game if C^* is a valid ciphertext.

There are also some restrictions: (ID_A^*, ID_S^*) should not be queried to extract encryption key, (C^*, ID_A^*, ID_S^*) should not be queried to decryption query and (M^*, ID_A^*, ID_S^*) should not be queried to the encryption query, where M^* is the plaintext of C^*. In this model, $\mathcal{A}$ can query the private key of ID_S^*. It is to model that even with a receiver's private key, without the master key, the adversaries cannot forge the ciphertexts for this receiver.

IV. PROPOSED SCHEME

In this section, we construct a many-to-one encryption and authentication scheme, which we call **MTO**. How to employ the sender revocation method in **MTO** is proposed in the next section. We let k be a security parameter given to the **Setup** algorithm. The algorithms are as follows:

- **Setup:** With the input k, this algorithm runs as follows:
  1. Generate a cyclic group $(\mathcal{G}, +)$ of prime order p and another cyclic group $(\mathcal{F}, \cdot)$ of the same order and the pairing $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$.
  2. Choose an arbitrary generator $P \in \mathcal{G}$.
  3. Pick $s \in Z_p^*$ randomly and set $P_0 = sP$.
  4. Choose cryptographic hash functions $H : \{0,1\}^* \rightarrow \{0,1\}^n$, $H_1 : \{0,1\}^* \rightarrow Z_p^*$, $H_2 : \mathcal{G} \times \{0,1\}^n \rightarrow Z_p^*$, $H_3 : \{0,1\}^* \rightarrow Z_p^*$ and $H_4 : \{0,1\}^* \rightarrow \mathcal{G}$.

  The system parameters are $params = (\mathcal{G}, \mathcal{F}, \hat{e}, n, P, P_0, H, H_1, H_2, H_3, H_4)$, and the master key $MK = s$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathcal{G} \times \mathcal{G} \times \{0,1\}^n$, where $\frac{1}{2^n}$ is negligible.
- **Private-Key-Extract:** This algorithm takes as input $params$ and a receiver's identity ID_S, and carries out the following steps to construct the receiver's private key:
  Select $a, b \in Z_p^*$ randomly and output $SK_S = (a, b)$
- **Information-Extract:** This algorithm takes as input $params$, a receiver's identity ID_S and his private key SK_S, and performs the following steps:
  Let $SK_S = (a, b)$ and output $Info_S = \hat{e}(aQ_S, bP)$, where $Q_S = H_4(ID_S)$.
- **Encryption-Key-Extract:** This algorithm takes as input $params$, the master key $MK = s$, $Info_S$, a legitimate sender's identity ID_A and his receiver's identity ID_S, and performs as follows:
  1. Compute $Q_S = H_4(ID_S)$ and $\gamma = H_3(Info_S, ID_A)$.
  2. Set $EK_1 = \frac{1}{H_1(ID_A)+s} Q_S$ and $EK_2 = Info_S^\gamma$.
  3. Output $EK_{A,S} = (EK_1, EK_2)$.
- **Encrypt:** This algorithm takes as input $params$, $EK_{A,S}$ and $M \in \mathcal{M}$, performs the following steps:
  1. Let $EK_{A,S} = (EK_1, EK_2)$ and pick $r_1 \in Z_p^*$ randomly.
  2. Compute $c = r_1 P$ and $r = H_2(c, M)$.
  3. Compute $u = (r + r_1) EK_1$ and $v = M \oplus H(EK_2^r)$.
  4. Output $C = (c, u, v)$ as the ciphertext.
- **Decrypt:** This algorithm takes $params$, the private key SK_S, the sender's identity ID_A, his receiver's identity ID_S and the ciphertext $C = (c, u, v)$, carries out as follows:
  1. Let $SK_S = (a, b)$, compute $Q_S = H_4(ID_S)$ and $\gamma = H_3(Info_S, ID_A)$
  2. Compute $\alpha = \frac{\hat{e}(u, ab\gamma(H_1(ID_A)P + P_0))}{\hat{e}(c, ab\gamma Q_S)}$, and then $M' = v \oplus H(\alpha)$.

If $\alpha = Info_S^{\gamma H_2(c,M')}$, output M' as result, otherwise reject it as an invalid ciphertext.

The correctness can be easily checked:

$$\begin{aligned}\alpha &= \frac{\hat{e}(u, ab\gamma(H_1(ID_A)P+P_0))}{\hat{e}(c, ab\gamma Q_S)} \\ &= \frac{\hat{e}((r+r_1)EK_1, ab\gamma(H_1(ID_A)P+P_0))}{\hat{e}(r_1P, ab\gamma Q_S)} \\ &= \frac{\hat{e}(\frac{r+r_1}{H_1(ID_A)+s}Q_S, ab\gamma(H_1(ID_A)+s)P)}{\hat{e}(r_1P, ab\gamma Q_S)} \\ &= \hat{e}(Q_S, P)^{rab\gamma} \\ &= Info_S^{\gamma r} \\ &= EK_2^r D\end{aligned}$$

and $Info_S = \hat{e}(aQ_S, bP)$ can be pre-computed before receiving any ciphertext from the legitimate senders.

A. Security

Lemma 1: Let hash functions H, H_1, H_2, H_3, and H_4 be random oracles. Suppose that there is no polynomially bounded algorithm that can solve BDDH assumption. Then the proposed scheme is IND-CCA2 secure.

Proof: Suppose that there is an algorithm $\mathcal{A}$ can break the proposed scheme in polynomial (in k) time $t(k)$ with non-negligible advantage $Adv(k)$, then we can construct an algorithm $\mathcal{B}$ to break BDDH problem in polynomial (in k) time $t'(k)$ with non-negligible advantage $Adv'(k)$. Any identity pair (ID_A,ID_S) in the proof means that ID_A is the sender's identity and ID_S is his receiver's identity. □

There is a challenger for BDDH problem by publishing parameters ($\mathcal{G}, \mathcal{F}$, P, xP, yP, zP, $\hat{e}$) and $T \in \mathcal{F}$. $\mathcal{B}$'s task is to answer 1 if $\hat{e}(P,P)^{xyz} = T$ or 0 if $\hat{e}(P,P)^{xyz} \neq T$ by calling $\mathcal{A}$ as a routine. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ constructs the parameters for the proposed scheme as follows:

Let integer $I \geq 1$. $\mathcal{B}$ picks $s \in Z_p^*$ randomly as the master key and set $P_0 = sP$ and sends ($\mathcal{G}, \mathcal{F}$, $\hat{e}, n, P, P_0, H, H_1, H_2, H_3, H_4$) to $\mathcal{A}$ as the system parameters, where H,H_1,H_2, H_3 and H_4 are controlled by $\mathcal{B}$ as random oracles. The lists *H-list*, H_1*-list*, H_2*-list*, H_3*-list* and H_4*-list* are maintained by $\mathcal{B}$ for answering these oracle queries made by $\mathcal{A}$. They are initialized as empty. s is also sent to $\mathcal{A}$.

Phase 1: At any time, $\mathcal{A}$ can make the following queries to $\mathcal{B}$:

- *H-query*: $\mathcal{A}$ sends α to $\mathcal{B}$ for query $H(\alpha)$. If there has been an item $[\alpha, h]$ in list *H-list*, $\mathcal{B}$ returns h to $\mathcal{A}$; Otherwise, $\mathcal{B}$ picks a random $h \in \{0,1\}^n$, stores $[\alpha, h]$ into *H-list* and outputs h to $\mathcal{A}$ as the answer.
- H_1*-query*: $\mathcal{A}$ sends ID to $\mathcal{B}$ for query $H_1(ID)$. $\mathcal{B}$ does as follows:
  If there has been an item $[ID, h_1]$ in list H_1*-list*, $\mathcal{B}$ returns h_1 to $\mathcal{A}$; Otherwise, $\mathcal{B}$ picks a random $h_1 \in Z_p^*$, stores $[ID, h_1]$ into H_1*-list* and outputs h_1 to $\mathcal{A}$ as the answer.
- H_2*-query*: $\mathcal{A}$ sends (c, M) to $\mathcal{B}$ for query $H_2(c, M)$. If there has been an item $[c, M, r]$ in list H_2*-list*, $\mathcal{B}$ returns r to $\mathcal{A}$; Otherwise, $\mathcal{B}$ picks a random $r \in Z_p^*$, stores $[c, M, r]$ into H_2*-list* and outputs r to $\mathcal{A}$ as the answer.
- H_3*-query*: $\mathcal{A}$ sends $(Info, ID)$ to $\mathcal{B}$ for query $H_3(Info, ID)$. If there has been an item $[Info, ID, \gamma]$ in list H_3*-list*, $\mathcal{B}$ returns γ to $\mathcal{A}$; Otherwise, $\mathcal{B}$ picks a random $\gamma \in Z_p^*$, stores $[Info, ID, \gamma]$ into H_3*-list* and outputs γ to $\mathcal{A}$ as the answer.
- H_4*-query*: $\mathcal{A}$ sends ID to $\mathcal{B}$ for query $H_4(ID)$. If ID is I-th distinct query, let $ID_I = ID$. If there has been an item $[ID, \theta]$ in list H_4*-list*, $\mathcal{B}$ returns θP to $\mathcal{A}$; Otherwise, $\mathcal{B}$ picks a random $\theta \in Z_p^*$, stores $[ID, \theta]$ into H_4*-list* and outputs θP to $\mathcal{A}$ as the answer.
- **Private Key Extraction**: $\mathcal{A}$ sends an identity ID_S to $\mathcal{B}$ for its private key, $\mathcal{B}$ answers it as follows:
  1. If $ID_S = ID_I$, abort with failure.
  2. Otherwise, perform as follows:
     If ID_S has not been queried, pick $a, b \in Z_p^*$ randomly and send them to $\mathcal{A}$ as the answer; otherwise, return the private key which has been generated with respect to ID_S to $\mathcal{A}$ as the answer.
- **Information Extraction**: If $\mathcal{A}$ wants to extract the information with respect to a receiver's private key, whose identity is ID_S, $\mathcal{B}$ does as follows:
  If $ID_S = ID_I$, output $Info_S = \hat{e}(xH_4(ID_I), yP)$. Otherwise, call **Private Key Extraction** with input ID_S to obtain the private key $SK_S = (a, b)$ and output $Info_S = \hat{e}(aH_4(ID_S), bP)$.
- **Encryption Key Extraction**: $\mathcal{A}$ sends an identity pair (ID_A, ID_S) to $\mathcal{B}$ for its encryption key. $\mathcal{B}$ answers it as follows:
  1. Obtain $Info_S$ by calling **Information Extraction** with input ID_S.
  2. Compute $Q_S = H_4(ID_S)$ and $\gamma = H_3(Info_S, ID_A)$
  3. Compute $EK_1 = \frac{1}{H_1(ID_A)+s}Q_S$ and $EK_2 = Info_S^\gamma$.
  4. Output $EK_{A,S} = (EK_1, EK_2)$
- **Decryption query**: $\mathcal{A}$ sends ciphertext $C = (c, u, v)$ with identity pair (ID_A,ID_S) to $\mathcal{B}$. $\mathcal{B}$ performs as follows:
  1. If $ID_S = ID_I$, perform as follows: Obtain $Info_I$ by calling *Information Extraction* with ID_I as input and γ by computing $H_3(Info_I, ID_A)$. If there exist $[\alpha, h]$ in *H-list* and $[c', M, r]$ in H_2*-list*, and they satisfy:
     $$c = c', \alpha = Info_I^{\gamma r} \text{ and } M = h \oplus v.$$
     output M as answer. Otherwise, reject C as an invalid ciphertext.
  2. If $ID_S \neq ID_I$, call *Private Key Extraction* to obtain the private key of ID_S and then call *Decrypt* algorithm to decrypt C.

Challenge: $\mathcal{A}$ sends two equal length messages M_0, M_1 and a challenge identity pair (ID_A^*, ID_S^*) to $\mathcal{B}$ for challenge. If ID_I has been assigned but $ID_S^* \neq ID_I$, $\mathcal{B}$ aborts with failure. Otherwise, $\mathcal{B}$ computes as follows:

1. If ID_I has not been assigned, then let $ID_I = ID_S^*$.
2. Pick $\beta \in \{0,1\}$ and $r_1 \in Z_p^*$ randomly, where (r_1P, M_β) has not been queried to H_3 query.
3. Obtain θ from H_4-list by computing $H_4(ID_I)$ and $Info_I$ by computing **Information Extraction** with input ID_I, and then compute $\gamma = H_3(Info_I, ID_A^*)$.
4. Compute $c^* = r_1P$, $u^* = \frac{1}{H_1(ID_A^*)+s}(\theta zP + r_1\theta P)$ and $v^* = M_\beta \oplus H(T^{\theta\gamma})$
5. Output $C^* = (c^*, u^*, v^*)$ as the challenge ciphertext.

It is clear that

$$\begin{aligned}u^* &= \frac{1}{H_1(ID_A^*)+s}(\theta zP + r_1\theta P) \\ &= \frac{z+r_1}{H_1(ID_A^*)+s}Q_I\end{aligned}$$

$$= (z + r_1)EK_1$$

and if $T = \hat{e}(P, P)^{xyz}$, we have

$$\begin{aligned} v^* &= M_\beta \oplus H(T^{\theta\gamma}) \\ &= M_\beta \oplus H(\hat{e}(xP, yP)^{\theta\gamma z}) \\ &= M_\beta \oplus H(Info_I^{\gamma z}) \\ &= M_\beta \oplus H(EK_2^z) \end{aligned}$$

where $EK_{A,I} = (EK_1, EK_2)$ is the encryption key with respect to (ID_A^*, ID_I) and we define $z = H_2(c^*, M_\beta)$ implicitly, so C^* is a valid ciphertext if $T = \hat{e}(P, P)^{xyz}$.

Phase 2: $\mathcal{A}$ can make any queries as in **Phase 1** after receiving C^*. The restriction is that ID_I should not be queried to extract its private key. The ciphertext C^* should not be queried to decryption query with the identity pair (ID_A^*, ID_I).

Guess: At the end of the game, $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$. If $\beta' = \beta$, $\mathcal{B}$ outputs 1 meaning $T = \hat{e}(P, P)^{xyz}$. Otherwise, it outputs 0 meaning $T \neq \hat{e}(P, P)^{xyz}$.

It is clear that the time $t'(k)$ is polynomial (in k) since the number of A's queries and the time of B's answers are all polynomial (in k). As long as (c^*, M_β) does not get queried to H_2 query, ID_I does not get queried to **Private Key Extraction** query in Phase 1 and $ID_S^* = ID_I$, the simulation is perfect. The advantage $Adv'(k) \geq Adv(k) \times \frac{p \cdot 2^n - q_{H_2}}{p \cdot 2^n q_{H_4}}$ which is non-negligible since $Adv(k)$ is non-negligible, where q_{H_2} is the number of H_2 oracle queries and q_{H_4} is the number of H_4 oracle queries.

Lemma 2: Let hash functions H, H_1, H_2, H_3 and H_4 be random oracles. Suppose that there is no polynomially bounded algorithm that can solve d-CAA assumption. Then the proposed scheme is secure against adaptive chosen message attack.

Proof: Suppose that there is an algorithm $\mathcal{A}$ which can forge a ciphertext in polynomial (in k) time $t(k)$ with non-negligible advantage $Adv(k)$, then we can construct an algorithm $\mathcal{B}$ to break d-CAA assumption in polynomial time $t'(k)$ with non-negligible advantage $Adv'(k)$. □

There is a challenger for d-CAA assumption. At first the parameters $(\mathcal{G}, \mathcal{F}, P, sP, (h_1, \frac{1}{h_1+s}P), \ldots, (h_d, \frac{1}{h_d+s}P))$ and $h \notin \{h_1, \cdots, h_d\}$ are generated by the challenger and published to $\mathcal{B}$. s is kept by the challenger secretly. We assume that d is bigger than the number of H_1 queries, which does not affect the correctness of the proof. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ constructs the system parameters for the proposed scheme as follows:

Let integer $I \geq 1$. $\mathcal{B}$ publishes $(\mathcal{G}, \mathcal{F}, \hat{e}, n, P, P_0 = sP, H, H_1, H_2, H_3, H_4)$ as the system parameters to $\mathcal{A}$, where H, H_2, H_3 and H_4 are controlled by $\mathcal{B}$ as in the proof of Lemma 1. H_1 is also controlled by $\mathcal{B}$ with list H_1-list which is initially empty.

Queries: At any time of the game, $\mathcal{A}$ can make following queries to $\mathcal{B}$:

- H, H_2, H_3, and H_4 are answered as in the proof of Lemma 1.
- H_1-*query*: $\mathcal{A}$ sends ID to $\mathcal{B}$ for query $H_1(ID)$. $\mathcal{B}$ does as follows:
  1. If ID is Ith distinct query, let $ID_1^* = ID$, and then output h as the answer.
  2. Otherwise, perform as follows:

     If there has been an item $[ID, h']$ in list H_1-*list*, $\mathcal{B}$ returns h' to $\mathcal{A}$; Otherwise, $\mathcal{B}$ picks $h' \in \{h_1, \ldots, h_d\}$ randomly, where there has not been an item $[\cdot, h']$ in H_1-list, stores $[ID, h']$ into H_1-*list* and outputs h' to $\mathcal{A}$ as the answer.
- **Private Key Extraction**: $\mathcal{A}$ sends an identity ID_S to $\mathcal{B}$ for its private key, $\mathcal{B}$ answers it as follows:
  If ID_S has not been queried, pick $a, b \in Z_p^*$ randomly and send them to $\mathcal{A}$ as the answer; Otherwise, return the private key which has been generated with respect to ID_S to $\mathcal{A}$ as the answer.
- **Information Extraction**: If $\mathcal{A}$ wants to extract the information with respect to a receiver's private key, whose identity is ID_S, $\mathcal{B}$ does as follows:
  Call **Private Key Extraction** with input ID_S to obtain the private key $SK_S = (a, b)$ and output $Info_S = \hat{e}(aH_4(ID_S), bP)$.
- **Encryption Key Extraction**: $\mathcal{A}$ sends an identity pair (ID_A, ID_S) to $\mathcal{B}$ for its encryption key. If $ID_A = ID_1^*$, $\mathcal{B}$ aborts with failure. Otherwise, $\mathcal{B}$ answers it as follows:
  1. Obtain θ from H_4-list by computing $H_4(ID_S)$ and $Info_S$ by calling **Information Extraction** with input ID_S.
  2. Obtain h_i by calling H_1 query with input ID_A, and get $R = \frac{1}{h_i+s}P$ from the pair $(h_i, \frac{1}{h_i+s}P)$ according to h_i.
  3. Set $EK_1 = \theta R$ and $EK_2 = Info_S^{H_3(Info_S, ID_A)}$
  4. Output $EK_{A,S} = (EK_1, EK_2)$
- **Encryption query**: $\mathcal{A}$ sends plaintext M and an identity pair (ID_A, ID_S) to $\mathcal{B}$ for its ciphertext. If $ID_A = ID_1^*$, $\mathcal{B}$ does as follows:
  1. Pick $x, r \in Z_p^*$ randomly.
  2. Obtain $Info_S$ by calling **Information Extraction** with ID_S as input. Compute $\gamma = H_3(Info_S, ID_A)$ and get θ from H_4-list by computing $H_4(ID_S)$.
  3. Compute $c = \theta^{-1}x(hP + P_0) - rP$ and store $[c, M, r]$ into H_2-list.
  4. Output $C = (c, u, v) = (c, xP, M \oplus H(Info_S^{r\gamma}))$ as the answer.

  It is clear that suppose that $c = r_1 P$, we have $u = \frac{r+r_1}{h+s}H_4(ID_S)$.
  If $ID_A \neq ID_1^*$, $\mathcal{B}$ constructs the ciphertext with the encryption key computed by calling **Encryption Key Extraction** with the input (ID_A, ID_S).
- **Decryption query**: $\mathcal{A}$ sends ciphertext $C = (c, u, v)$ with identity pair (ID_A, ID_S) to $\mathcal{B}$. $\mathcal{B}$ calls **Private Key Extraction** to obtain the private key of ID_S and then calls *Decrypt* algorithm to decrypt C.

Challenge: At the end of the game, $\mathcal{A}$ forges a new ciphertext $C^* = (c^*, u^*, v^*)$ with an identity pair (ID_A^*, ID_S^*). If ID_1^* has been assigned but $ID_A^* \neq ID_1^*$, $\mathcal{B}$ aborts with failure. Otherwise, $\mathcal{B}$ does as follows:

1. If ID_1^* has not been assigned, let $ID_1^* = ID_A^*$.
2. Obtain θ by computing $H_4(ID_S^*)$.
3. Obtain $Info_S^*$ by calling **Information Extraction** with ID_S^* as input and $\gamma = H_3(Info_S^*, ID_1^*)$. If there exist $[\alpha, h']$ in *H-list* and $[c^*, M^*, r]$ in H_2-*list*, and they satisfy:
   $$M^* = h' \oplus v^* \text{ and } \alpha = Info_S^{*\gamma r}$$
   then let $r_1^* = r$ and $\theta_1^* = \theta$.

In the above simulation, there are also some restrictions: (ID_1^*, ID_S^*) should not be queried to **Encryption Key Extraction**, (M^*, ID_1^*, ID_S^*) should not be queried to **Encryption query** and (C^*, ID_1^*, ID_S^*) should not be queried to **Decryption query**.

$\mathcal{B}$ empties all its lists and plays the above game with $\mathcal{A}$ again, and then at the end of the game, $\mathcal{B}$ gets new values r_2^* and θ_2^* from $\mathcal{A}$'s challenge ciphertext $C^{'*} = (c^{'*}, u^{'*}, v^{'*})$. If $c^* = c^{'*}$, $\mathcal{B}$ outputs $Y = (r_1^* - r_2^*)^{-1}(\theta_1^{*-1}u^* - \theta_2^{*-1}u^{'*})$ as the answer for the challenger. Otherwise, $\mathcal{B}$ aborts with failure.

With the fork theorem, it is clear that if $c^* = c^{'*}$, the randomness used in the two challenge ciphertexts are equal. We have $u^* = \frac{r_1^*+r_1}{h+s}\theta_1^* P$ and $u^{'*} = \frac{r_2^*+r_1}{h+s}\theta_2^* P$, and then $Y = \frac{1}{h+s}P$.

As long as $ID_A^* = ID_1^*$, ID_1^* does not get queried to *Encryption Key Extraction* query as the sender part, and $c^* = c^{'*}$, then the simulation is perfect. The time $t'(k)$ is polynomial (in k) and advantage $Adv'(k) \geq \frac{1}{\sqrt{q_{H_2} q_{H_1}}} \times Adv(k)$ which is non-negligible since $Adv(k)$ is non-negligible.

V. SENDER REVOCATION

In this section, we introduce our method of revoking a legitimate sender. The dynamic accumulator is employed [18]. At first, we introduce the concept of Strong RSA assumption and the dynamic accumulator followed by our revocation method.

A. Strong RSA Assumption

Fujisaki and Okamoto proposed a variation of the well-known RSA assumption: the so-called strong RSA assumption. Let k be a security parameter and let $G(k)$ denote the set of groups whose order has length k and consists of two prime factors of length $\frac{k-2}{2}$.

Definition 3[Strong RSA Problem]: Given G and $z \in G/\{\pm 1\}$, find a pair $(\tilde{u}, e) \in G \times Z$ such that $\tilde{u}^e = z$ and $e > 1$.

Let K denote a key-generator that on input 1^k outputs a $G \in G(k)$ and a $z \in G/\{\pm 1\}$.

Assumption 3[Strong RSA Assumption]: There exists a probabilistic algorithm K such that for all probabilistic polynomial-time algorithms A, all polynomials $p(\cdot)$, all sufficiently large k

$Pr[z = \tilde{u}^e \wedge e > 1 : (G, z) := K(1^k), (\tilde{u}, e) := A(G, z)] < \frac{1}{p(k)}$

In an implementation of the key-generator K, where G is chosen to be Z_N^* and N is an RSA modulus, the parameter z should not be chosen as a power in Z.

B. Dynamic Accumulator

The dynamic accumulator is first presented in [18] which is derived from the construction due to Barić and Pfitzmann [19]. The differences from [19] are that 1) the domain of the accumulated values consists of prime numbers only; 2) a method is presented for deleting values from the accumulator, i.e., a method constructing a dynamic accumulator; 3) the authors give efficient algorithms for deleting a user and updating a witness. The construction of the dynamic accumulator is as follows:

1. $\mathcal{F}_k$ is the family of functions that correspond to exponentiating modulo safe-prime products drawn from the integers of length k. $\mathcal{X}_k$ is the set of the primes which are accumulated and suppose that $\mathcal{X}_k = \{x_1, \ldots, x_h\}$. Choosing $f \in \mathcal{F}_k$ amounts to choosing a random modulus $N = \tilde{p}\tilde{q}$ of length k, where $\tilde{p} = 2p' + 1, \tilde{q} = 2q' + 1$ (k is security parameter) and $\tilde{p}, p', \tilde{q}, q'$ are all primes.
2. Denote $f(\tilde{u}, x) = \tilde{u}^x \pmod N$, where the value $x \in \mathcal{X}_k$, $\tilde{u} \in QR_N$ and $\tilde{u} \neq 1$. Note that $f(f(\tilde{u}, x_1), x_2) = \tilde{u}^{x_1 x_2} \pmod N$.
3. Update of the accumulator value. Denote $\tilde{v}'$ as the new accumulator value when a prime is added or deleted from the accumulator value $\tilde{v}$. Adding a prime $x \notin \mathcal{X}_k$ to the accumulator value $\tilde{v}$ can be done as $\tilde{v}' = f(\tilde{v}, x) = \tilde{v}^x \pmod N$ and $\mathcal{X}_k$ is updated as $\mathcal{X}_k \bigcup \{x\}$. Deleting a value $x \in \mathcal{X}_k$ from the accumulator value is as follows. $\tilde{v}' = D((\tilde{p}, \tilde{q}), \tilde{v}, x) = \tilde{v}^{x^{-1} \pmod{(\tilde{p}-1)(\tilde{q}-1)}} \pmod N$, and $\mathcal{X}_k$ is updated as $\mathcal{X}_k \backslash \{x\}$.
4. Update of witness: Denote $\tilde{u}'$ as the new witness when the witness $\tilde{u} \in QR_N$ is updated. Updating the witness $\tilde{u}$ after $x \notin \mathcal{X}_k$ has been added can be done by $\tilde{u}' = f(\tilde{u}, x) = \tilde{u}^x \pmod N$. In case, $x \neq \hat{x}$ ($\tilde{u}^{\hat{x}} = \tilde{v} \pmod N$) has been deleted from the accumulator, the witness $\tilde{u}$ can be updated as follows: By the extended GCD algorithm, one can compute the integers a, b such that $a\hat{x} + bx = 1$ and then $\tilde{u}' = \tilde{u}^b \tilde{v}^{'a}$ ($\tilde{v}'$ is the new accumulator value). Let us verify that $f(\tilde{u}', \hat{x}) = \tilde{u}^{'\hat{x}} \pmod N = \tilde{v}'$:
$(\tilde{u}^b \tilde{v}^{'a})^{\hat{x}} = ((\tilde{u}^b \tilde{v}^{'a})^{\hat{x}x})^{x^{-1}} = ((\tilde{u}^{\hat{x}})^{bx}(\tilde{v}^{'x})^{a\hat{x}})^{x^{-1}} = (\tilde{v}^{bx}\tilde{v}^{a\hat{x}})^{x^{-1}} = \tilde{v}^{x^{-1}} = \tilde{v}' \pmod N$.

As Mentioned in [18], adding or deleting several values at once can be done simply by letting $\hat{x}$ be the product of the added or deleted values. This also holds with respect to updating the witness. More precisely, let π_a be the product of the x's to add to and π_d be the ones to delete from the accumulator value $\tilde{v}$. Then, the new accumulator value $\tilde{v}' = \tilde{v}^{\pi_a \pi_d^{-1}} \pmod{(\tilde{p}-1)(\tilde{q}-1)} \pmod N$. If $\tilde{u}$ was the witness that x was contained in $\tilde{v}$ and x was not removed from the accumulator, i.e., x cannot divide π_d, then $\tilde{u}' = \tilde{u}^{b\pi_a}\tilde{v}^{'a} \pmod N$ is a witness that x is contained in $\tilde{v}'$, where a and b satisfy $ax + b\pi_d = 1$ and are computed using the extended GCD algorithm.

The following lemma and its proof are given in [18]:

Lemma 3: Under the strong RSA assumption, the above construction is a secure dynamic accumulator.

C. Our Method

A many-to-one encryption and authentication scheme with sender revocation is proposed in this subsection by extending **MTO** with the dynamic accumulator. The algorithms **Private-Key-Extract** and **Information-Extract** are the same as those in **MTO**. The KGC performs as follows:

- **Setup**: In addition to setting up the system parameters $params^{MTO}$ and the master key MK of **MTO**, the KGC creates the public modulus $N = \tilde{p}\tilde{q}$ for the accumulator, where $\tilde{p}$ and $\tilde{q}$ are two safe primes with the security parameter k, chooses $\tilde{u} \in QR_N$ and $\tilde{s} \in Z_{\phi(N)}^*$ randomly. Two hash functions $H_5 : \{0,1\}^* \rightarrow \{0,1\}^k$ and $H_6 : \{0,1\}^k \rightarrow \{0,1\}^k$ are also chosen.
For a sender A, the KGC chooses unique prime $x_A \in Z_{\phi(N)}^*$ randomly as his secret value.

For a receiver S the KGC compute $\tilde{v}_S = \tilde{u}^{x_1 x_2 \cdots x_l} \pmod N$ and $\lambda_S = \tilde{v}_S^{\tilde{s}} \pmod N$, where x_i is the receiver's legitimate sender's secret value and $l > 1$. For the receiver S, the KGC also sets up (empty for now) public archives E_S^{add} for storing the secret values that correspond to added S's legitimate senders and E_S^{delete} for storing the secret values that correspond to deleted S's legitimate senders.

$params = (params^{MTO}, N, H_5, H_6)$ are published as the system public parameters, and $(MK, \tilde{p}, \tilde{q}, \tilde{s})$ are kept as the system master key, $\tilde{u}$ is also kept secretly. $\{\tilde{v}_S, \lambda_S\}$ is also published as the receiver S's parameter and can be changed during the running of the system. Since the number of the receivers is very small, managing these parameters does not increase the burden of the KGC very much.

- **Encryption-Key-Extract**: The KGC computes as in **MTO** and gets the encryption key (EK_1, EK_2) for a sender A whose receiver is S, and then computes as follows:
  Suppose that S's information is $Info_S$ and γ is computed as in **MTO**.
  1. Pick a distinct prime $x_A \in \{x_1, \ldots, x_l\}$.
  2. Compute $\tilde{u}_A = \tilde{v}_S^{x_A^{-1}} \pmod N$. It is clear that $\tilde{v}_S = \tilde{u}_A^{x_A} \pmod N$.
  3. Choose a random value $r \in \mathrm{Z}^*_{\phi(\mathrm{N})}$ and compute $\eta_A = x_A(r + H_5(Info_S^{\gamma} \| ID_A)) \pmod{\phi(N)}$.
  4. Compute $\omega_A = r + \tilde{s}H_6(\eta_A) \pmod{\phi(N)}$.
  5. Output $EK_{A,S} = (EK_1, EK_2, x_A)$ and $PUB_{A,S} = (\tilde{u}_A, \eta_A, \omega_A)$. $EK_{A,S}$ is the encryption key. $PUB_{A,S}$ is an identity verification certificate which is not kept secretly, and it acts as a part of the ciphertext actually.
- **Join**: If a new legitimate sender is added to the group of the receiver S, and his secret value is the prime $\hat{x}$, the KGC chooses a randomness $r \in \mathrm{Z}^*_{\phi(\mathrm{N})}$ and adds $\delta = r\hat{x} \pmod{\phi(N)}$ to E_S^{add}, at the same time $\tilde{v}_S$ and λ_S are updated as $\tilde{v}_S^{\delta} \pmod N$ and $\lambda_S^{\delta} \pmod N$ respectively.
- **Revoke**: When a legitimate sender is revoked from the group of the receiver S, and his secret value is the prime $\hat{x}$, the KGC adds $\hat{x}$ to E_S^{delete}, at the same time $\tilde{v}_S$ and λ_S are updated as $\tilde{v}_S^{\hat{x}^{-1}} \pmod N$ and $\lambda_S^{\hat{x}^{-1}} \pmod N$ respectively.
  The archives E_S^{add} and E_S^{delete} are not part of any public key, i.e., the receiver is not required to read them for any verification purposes. An entry in the archive is called "new" if it was entered after the last time a sender whose secret value is x_A performed an update on $\tilde{u}_A$. The sender A can update $\tilde{u}_A$ as follows:
  1. Let y denote the old value of $\tilde{u}_A$.
  2. For all new $\delta_j \in E_S^{add}$, $\tilde{u}_A = y^{\prod \delta_j} \pmod N$.
  3. For all new $\hat{x}_j \in E_S^{delete}$, if $x_A \neq \hat{x}_j$, the sender himself can update $\tilde{u}_A$ as follows:

  There exist two integers a and b such that $ax_A + b\prod \hat{x}_j = 1$ since x_A and $\prod \hat{x}_j$ are co-prime each other. The sender can compute $\tilde{u}_A = (y^b \tilde{v}_S^a) \pmod N$. It is clear that $\tilde{v}_S = \tilde{u}_A^{x_A} \pmod N$ according to section V.
  From section V, the sender who has $\hat{x}_j \in E_S^{delete}$ cannot update his $\tilde{u}_j$.
  Suppose that a sender's encryption key is $EK_{A,S} = (EK_1, EK_2, x_A)$ and identity verification certificate is $PUB_{A,S} = (\tilde{u}_A, \eta_A, \omega_A)$. After updating his $\tilde{u}_A$, he constructs a ciphertext to his receiver by running the following algorithm:
- **Encrypt**: The sender first constructs (c, u, v) with (EK_1, EK_2) as in the algorithm **Encrypt** of **MTO**, and then outputs $C = (c, u, v, \tilde{u}_A, \eta_A, \omega_A)$ as the ciphertext.
  After receiving a ciphertext $C = (c, u, v, \tilde{u}_A, \eta_A, \omega_A)$ from a sender A, the receiver S verifies and decrypts it by running the following algorithm:
- **Decrypt**:
  1. Check $\tilde{u}_A \neq \tilde{v}_S$, $\tilde{u}_A \neq \lambda_S$ and $\eta_A > 1$, and then compute $\alpha' = \frac{\tilde{u}_A^{\eta_A}}{\tilde{v}_S^{H_5(Info_S^{\gamma} \| ID_A)}} \pmod N$.
  2. Check whether $\alpha' \lambda_S^{H_6(\eta_A)} = \tilde{v}_S^{\omega_A} \pmod N$.

If all checks are successful, the sender is not revoked, and then the receiver decrypts (c, u, v) as in the algorithm **Decrypt** of **MTO**. Otherwise, reject the ciphertext as an invalid one.

Indeed, we have

$$\alpha' = \frac{\tilde{u}_A^{\eta_A}}{\tilde{v}_S^{H_5(Info_S^{\gamma} \| ID_A)}} = \frac{\tilde{v}_S^{r + H_5(Info_S^{\gamma} \| ID_A)}}{\tilde{v}_S^{H_5(Info_S^{\gamma} \| ID_A)}} = \tilde{v}_S^{r} \pmod N$$

and

$$\alpha' \lambda_S^{H_6(\eta_A)} = \tilde{v}_S^{r} \tilde{v}_S^{\tilde{s}H_6(\eta_A)} = \tilde{v}_S^{\omega_A} \pmod N.$$

From the attributes of the dynamic accumulator, it is clear that if a sender is revoked, i.e. his secret prime x_i is added into the archive E_S^{delete}, the check will not be successful, because he *cannot update his* u_i.

To revoke a receiver S, the KGC only removes his parameters $\{\tilde{v}_S, \lambda_S\}$ and the archives $(E_S^{add}, E_S^{delete})$. The senders can know that the receiver S is revoked by visiting his archives and finding that they are removed.

It is clear that to verify a sender status, the receiver does not ask the KGC and only checks the identity verification certificate sent with the ciphertext. To add or revoke a sender, the KGC only changes the receiver's parameters and his archives and no online computation has to be provided by the KGC. The computational complexity is very lower than the other methods. Actually the computational cost is distributed to each sender, which reduces the burdens of the KGC and the receivers.

D. Security Analysis

Lemma 4: Let hash functions $H, H_1, \ldots, H_6$ be random oracles. Suppose that there is no polynomially bounded algorithm that can solve the BDDH problem, then the accumulator added scheme is IND-CCA2 secure.

Proof: The proof is similar to that of Lemma 1. In the setup phase, besides the parameters of **MTO**, $\mathcal{B}$ generates the parameters $(\tilde{p}, \tilde{q}, \tilde{u}, \tilde{s}, H_5, H_6)$ for the dynamic accumulator. With $(\tilde{p}, \tilde{q}, \tilde{u}, \tilde{s}, H_5, H_6)$, $\mathcal{B}$ can easily answer the queries which the dynamic accumulator is involved in. In the challenge phase, it is easy to construct the challenge ciphertext $C^* = (c^*, u^*, v^*, \tilde{u}^*, \eta^*, w^*)$, where (c^*, u^*, v^*) are constructed as in the challenge phase of Lemma 1 and $(\tilde{u}^*, \eta^*, w^*)$ are constructed with $(\tilde{p}, \tilde{q}, \tilde{u}, \tilde{s}, H_5, H_6)$ (adding or deleting a member is easy with these parameters). The guess phase is the same as in Lemma 1 since $(\tilde{u}^*, \eta^*, w^*)$ have nothing to do with the challenge plaintexts. The advantage remains the same as that in Lemma 1. □

From the above we see that the accumulator does not reduce the confidentiality of the scheme. More precisely, we have

Lemma 5: Let hash functions $H, H_1, \cdots, H_6$ be random oracles. Suppose that there is no polynomially bounded algorithm that can solve the *d-CAA* problem and the Strong RSA problem, and the dynamic accumulator is secure, then the accumulator added scheme is secure against adaptive chosen message attack.

Proof: At first, $\mathcal{B}$ picks a random bit $b \in \{0,1\}$. 1) If $b = 0$, Given the parameters of d-CAA problem, $\mathcal{B}$ generates parameters $(\tilde{p}, \tilde{q}, \tilde{u}, \tilde{s}, H_5, H_6)$ for the dynamic accumulator, and then performs as in Lemma 2 (the outputs of each queries related to the dynamic accumulator can be easily constructed since $(\tilde{p}, \tilde{q}, \tilde{u}, \tilde{s}, H_5, H_6)$ are known). If (c^*, u^*, v^*) in the challenge ciphertext $(c^*, u^*, v^*, \tilde{u}^*, \eta^*, \omega^*)$ satisfies the restrictions of the Model 1, $\mathcal{B}$ performs and outputs as in Lemma 2; otherwise, $\mathcal{B}$ aborts with failure. 2) If $b = 1$, Given the parameters $(N, z \in QR_N)$ of the Strong RSA problem, $\mathcal{B}$ picks a distinct prime x. Let $\tilde{u} = z^x \pmod N$, and for each receiver S, let $\tilde{v}_S = \tilde{u}^{\prod x_i} \pmod N$ and $\lambda_S = z^{\prod x_i} \pmod N$, where $\{x_i\}$ are the secret values with respect to S. In this case, $\tilde{s} = x^{-1} \pmod{\phi(N)}$ implicitly. $\mathcal{B}$ generates the parameters of d-CAA problem, then constructs and publishes the parameters of the system, and then answers the queries as in Lemma 2. Additionally, to answer a distinct H_6 query, $\mathcal{B}$ picks θ_A randomly and outputs $x\theta_A$ as the answer (it is easy to see that $\tilde{s}H_6(\eta_A) = \theta_A$ under the modulus $\phi(N)$). To answer a distinct **Encryption Key Extraction** query, $\mathcal{B}$ picks ω_A randomly and computes $\eta_A = x_A(\omega_A - \theta_A + H_5(Info_S^{\gamma}))$. Adding or deleting a member is also easy with the system parameters. Suppose that the forged ciphertext is $(c^*, u^*, v^*, \tilde{u}^*, \eta^*, \omega^*)$, where $\eta^*, \omega^* \in Z^*_{\phi(N)}$ (η^* is odd since $\phi(N) = 4p'q'$) and the challenge receiver is S. If there exists x_i ($\{x_i\}$ are the secret values with respect to S) such that $\tilde{u}^{*x_i} = \tilde{v}_S \pmod N$, where $1 \le i \le l$, then abort with failure; otherwise, performs as follows: compute $t = \omega^* + H_5(Info_S^{\gamma} \| ID_A) - \tilde{s}H_6(\eta^*)$ (we have $\tilde{u}^{*\eta^*} = \tilde{v}_S^t \pmod N$). Since η^* and t are two integers, let $d = gcd(\eta^*, tx\prod x_i)$ (d is also odd). If $gcd(d, \phi(N)) \ne 1$, then $d = p', q'$ or $p'q'$ and $\mathcal{B}$ can easily construct a forgery for the Strong RSA problem; otherwise ($gcd(d, \phi(N)) = 1$), perform as follows:

Compute $\tilde{\eta} = \frac{\eta^*}{d}$ and $\tilde{t} = \frac{tx\prod x_i}{d}$ (we have $\tilde{u}^{*\tilde{\eta}} = z^{\tilde{t}} \pmod N$). If $\tilde{\eta} = 1$, abort with failure; else if $\tilde{t} = 1$, output $(\tilde{u}^*, \tilde{\eta})$ as the forgery for the Strong RSA problem; otherwise, compute two integers a and b such that $a\tilde{\eta} + b\tilde{t} = 1$, let $y = \tilde{u}^{*b}z^a \pmod N$, and output $(y, \tilde{\eta})$ as the forgery. Indeed, we have $y^{\tilde{\eta}} = (\tilde{u}^{*b}z^a)^{\tilde{\eta}} = z \pmod N$. The advantage $Adv'(k) \ge \frac{1}{2\sqrt{q_{H_2}q_{H_1}}} \times Adv(k)$ is non-negligible since $Adv(k)$ is non-negligible ($Adv(k)$ is the same as in Lemma 2). □

VI. APPLICATIONS

In this section, as an example, how to employ the proposed scheme in online software registration and update is introduced. At first, the software company setups the system. A private key is distributed to the online server and two archives E^{add} and E^{delete} are also constructed which is initially empty. Each copy of the software has a distinct product serial number which is used as its identity. The software is published with its encryption key and identity verification certificate. The encryption key can be stored in a read-only equipment such as smart card. The identity verification certificate and the software are stored in a CD-ROM since this certificate needs not be kept secretly.

After the software is installed in the user's computer, the online registration function is activated to ask the user input his personal information. The personal information and the parameters of the computer such as hard disk parameters and MAC address are encrypted with its encryption key. The software visits the archives E^{add} and E^{delete} to update its identity verification certificate, and sends the encrypted information, the updated identity verification certificate and the product serial number to the online server. If the verification of the ciphertext (including the verification of the identity verification certificate) is successful, these information is stored into the database for further service, and then the full version can be used by the user.

For a full version software, to update its data, it does as follows:

1. Visit the archives E^{add} and E^{delete}. If there are new entries, it updates its identity verification certificate.
2. A ciphertext (including the product serial number and the changed identity verification certificate) on a random temporary key is sent to the server.

After receiving the ciphertext, the server performs as follows:

1. It verifies the identity verification certificate. If the verification is successful, the software has not been revoked. Otherwise, reject the ciphertext as an invalid one.
2. Decrypt the ciphertext with its private key. If the decryption is successful, that is, the ciphertext is sent by its legitimate copy of the software and it is not tampered, the new data can be downloaded by the software with the protection of the temporary key. Otherwise, the server rejects the ciphertext as an invalid one.

To expedite the decryption, the server can store his parameters $(\tilde{v}, \lambda)$ to the cache. If there are no new entries in the archives, it decrypt the ciphertext with its cached parameters. Otherwise with the new parameters.

When a new copy of software is published, the company should change the server's parameters $(\tilde{v}, \lambda)$ and add its randomized secret prime δ to the server's archive E^{add}. If a copy of software is revoked, the company should change the server's parameters $(\tilde{v}, \lambda)$ and add its prime to the server's archive E^{delete}. It is clear that adding or revoking a copy of software is very easy, which does not increase the burden of the company and the server very much.

VII. CONCLUSIONS

In this paper, a new paradigm, many-to-one encryption and authentication scheme is introduced. The goal of this scheme is to reduce the burden of key management of a receiver and the computational complexity of a sender when confidentiality, integrity and authentication should be provided under the many-to-one scenario, that is, compared with the number of the senders, the number of the receivers is very small and each receiver may serve many senders such as the servers for online software registration and update. The concept, definition and security models are given and a method of revoking the legitimate senders is also proposed for the many-to-one scheme.

REFERENCES

[1] Y. Desmedt,"Society and group oriented cryptography: A new concept," in *Proc. CRYPTO'87*, LNCS 293, 1988, pp. 120–127.

[2] C.C. Chang and H.C. Lee,"A new generalized group-oriented cryptosystem without trusted centers," in *IEEE J. Sel. Areas Commun.*, vol. 11, no. 5, pp. 725–729, 1993.

[3] L. Harn, "Group-oriented (t,n) threshold digital signature and digital multisignature," *Proc. IEE Computers and Digital Techniques*, vol. 141, no. 5, pp. 307–313, 1994.

[4] L. Harn and S. Yang,"Group-oriented undeniable signature schemes without the assistance of a mutually trusted party," in *Proc. AUSCRYPT'92*, LNCS 718, 1993, pp. 133–142.

[5] C.H. Lin, C.T. Wang, and C.C. Chang, "A group-oriented (t,n) undeniable signature scheme without trusted centers," in *Proc. Information Security and Privacy*, LNCS 1172, 1996, pp. 266–274.

[6] C.K. Wu and V. Varadharajan,"Many-to-one algorithms and group signatures," in *Proc. ACSC'99*, 1999, pp. 432–444.

[7] A. Shamir,"Identity-based cryptosystems and signature schemes," in *Proc. CRYPTO'84*, LNCS 196, 1985, pp. 48–53.

[8] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *Proc. EUROCRYPT 2003*, LNCS 2656, 2003, pp. 272–293.

[9] S.S. Al-Riyami,"Cryptographic schemes based on elliptic curve pairings," Ph.D. thesis, University of London, 2004.

[10] S.S. Al-Riyami and K.G. Paterson,"Certificateless public key cryptography," in *Proc. ASIACRYPT 2003*, LNCS 2894, 2003, pp. 452–473.

[11] D. Naor, M. Naor, and J. Lotspiech,"Revocation and tracing schemes for stateless receivers," in *CRYPTO 2001*, LNCS 2139, 2001, pp. 41–62.

[12] S. Micali, "Efficient certificate revocation," MIT Laboratory for Computer Science, Tech. Rep., 1996, TM-542b.

[13] S. Micali, "Novomodo: Scalable certificate validation and simplified PKI management," in *1st Annual PKI Research Workshop*, 2002.

[14] E.R. Verheul,"Evidence that XTR is more secure than supersingular elliptic curve cryptosystems," in *Proc. EUROCRYPT 2001*, LNCS 2045, 2001, pp. 195–210.

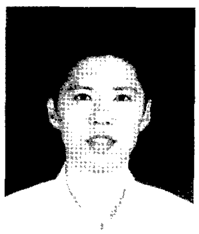
[15] A. Menezes,T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," *IEEE Trans. Inf. Theory*, vol. 39, pp. 1639–1646, 1993.

[16] A. Joux, "A one round protocol for tripartite Diffie-Hellman," in in *Proc. ANTS IV*, LNCS 1838, 2000, pp. 385–394.

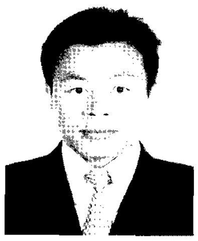
[17] S. Mitsunari, R. Sakai and M. Kasahara, "A new traitor tracing," *IEICE Trans*, vol. E85-A, no. 2, pp. 481–484, 2002.

[18] J. Camenisch and A. Lysyanskaya,"Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Proc. CRYPTO 2002*, LNCS 2442, 2002, pp. 61–76.

[19] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *Proc. EUROCRYPT' 97*, LNCS 1233, 1997, pp. 480–494.



Xi-Jun Lin received his Bachelor degree in Computer Science from Qingdao Technological University in 2000, and Master degree in Computer Science from Ocean University of China in 2004. Since 2004, he is a Ph.D. candidate of Institute of Software, Chinese Academy of Sciences. His research interests include Cryptography and Information Security.



Chuan-Kun Wu received his Bachelor degree in Science from Qufu Normal University in 1995, and Master degree in Science in 1988, and Ph.D. degree in Engineering in 1994, both from Xidian University, China. He did his postdoc fellowship at Queensland University of Technology in 1995, was a research fellow at University of Western Sydney in 1997, and was a lecturer/senior lecturer at the Australian National University. Since 2003, he has been a professor at the Institute of Software, Chinese Academy of Sciences. His research interests include Cryptography and Information Security. He served as a PC member for many international conferences related to cryptography and information security, and was the initiator for the International Workshop (named as conference since 2005) on Cryptography and Network Security (CANS). He serves as an associate editor for IEEE Communications Letters since 2002, and has been on the editorial board of International Journal of Network Security.

Feng Liu received his Bachelor degree in Computer Science in 2003 from Shandong University, China. And he is a Ph.D. Candidate of Institute of Software, Chinese Academy of Sciences and Graduate School of Chinese Academy of Sciences from 2003. His research interests include visual cryptography and design and analysis of network security protocols.