University of Wollongong

Research Online

Faculty of Engineering and Information
Sciences - Papers: Part A

Faculty of Engineering and Information
Sciences

1-1-2013

Verifiable and anonymous encryption in asymmetric bilinear maps

Hui Cui

University of Wollongong, hc892@uowmail.edu.au

Yi Mu

University of Wollongong, ymu@uow.edu.au

Man Ho Au

University of Wollongong, aau@uow.edu.au

Follow this and additional works at: <https://ro.uow.edu.au/eispapers>

Part of the [Engineering Commons](#), and the [Science and Technology Studies Commons](#)

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Verifiable and anonymous encryption in asymmetric bilinear maps

Abstract

Consider a practical scenario: an untrusted gate-way is required to verify all the incoming information encrypted via an encryption scheme, while the sender does not want to reveal any information about the plaintext and the privileged user to the gateway. That is, the gateway distributes the information to a predefined group of users and only the privileged user can open the message. To solve this problem, we need an access control mechanism to allow certain specification of the access control policies while protecting the users' privacy. With this scenario in mind, we propose the notion of verifiable and anonymous encryption where a verification function is added to the ciphertext, which captures the security requirements of the confidentiality of the plaintext and the anonymity of the privileged user. We present two specific constructions of our framework under the setting of asymmetric bilinear pairings in this paper. Our first scheme is proven confidential and anonymous under a weaker security model in the random oracle model, and our second one is built on the basis of a zero knowledge proof of knowledge under a strong security game.

Keywords

bilinear, asymmetric, maps, encryption, verifiable, anonymous

Disciplines

Engineering | Science and Technology Studies

Publication Details

Cui, H., Mu, Y. & Au, M. (2013). Verifiable and anonymous encryption in asymmetric bilinear maps. Proceedings - 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013 (pp. 704-711). IEEE.

Verifiable and Anonymous Encryption in Asymmetric Bilinear Maps

Hui Cui

*School of Computer Science and Software Engineering
University of Wollongong
Wollongong, NSW 2522, Australia
hc892@uowmail.edu.au*

Yi Mu, Man Ho Au

*School of Computer Science and Software Engineering
University of Wollongong
Wollongong, NSW 2522, Australia
ymu,aau@uow.edu.au*

Abstract—Consider a practical scenario: an untrusted gateway is required to verify all the incoming information encrypted via an encryption scheme, while the sender does not want to reveal any information about the plaintext and the privileged user to the gateway. That is, the gateway distributes the information to a predefined group of users and only the privileged user can open the message. To solve this problem, we need an access control mechanism to allow certain specification of the access control policies while protecting the users' privacy. With this scenario in mind, we propose the notion of verifiable and anonymous encryption where a verification function is added to the ciphertext, which captures the security requirements of the confidentiality of the plaintext and the anonymity of the privileged user. We present two specific constructions of our framework under the setting of asymmetric bilinear pairings in this paper. Our first scheme is proven confidential and anonymous under a weaker security model in the random oracle model, and our second one is built on the basis of a zero knowledge proof of knowledge under a strong security game.

Keywords—Verification, Anonymity, Access control, Zero-knowledge proof of knowledge.

I. INTRODUCTION

The distribution and availability of digital information in modern life and work lead new opportunities for providing support to the individuals. This ubiquity of information also creates new challenges in the protection of both the provided information and the privacy of its users, which requires the access control mechanisms should allow some specification of the access control policies and protect the privacy of the users at the same time.

To address practical scenarios in which an untrusted gateway is involved for verification in encryption systems, we put forward the notion of verifiable and anonymous encryption as our access control mechanism, which is a three-party protocol enabling an outside sender to transmit a ciphertext to an inside user (i.e. this user is in a group composed of a large number of users) under the verification of gateway W (determines whether to broadcast this ciphertext or not). Unlike general encryption schemes, because of the involvement of an untrusted gateway W , in this case, collusion attack maybe possible between gateway W and the corrupted users: (1) gateway W may collude with the corrupted users to obtain the message content; (2)

gateway W may collude with the corrupted users to guess the identity of the privileged user. All in all, a verifiable and anonymous encryption system should maintain confidentiality, anonymity while preventing collusion attack between gateway W and the corrupted users, such that a sender can transmit information to an inside user under gateway W securely.

In order to provide a verifiable and anonymous encryption scheme, we design a secure framework for our problem. In our model, when a sender, say Alice, wants to transmit a message to an internal user, say Bob, she encrypts her message with a verifiable and anonymous encryption algorithm to generate a ciphertext which is composed of two parts: encryption of the message and verification of the receiver's identity. Gateway W checks whether Bob is an inside user with the verification part while Bob decrypts the ciphertext with the encryption part.

In our first construction, we make use of a trusted third party KGC [2] to generate the partial private keys for the users (i.e. the KGC does not have access to the private keys of the users), and then allow the users to generate their own (full) private keys (from partial private keys). Also, we demand every user in the system to generate a trapdoor and send it to gateway W through a public broadcast channel, which assures the unlinkability between the users and the trapdoors. Gateway W checks the validity of the trapdoors with the public keys of the users, and maintains them in a trapdoor list (without identity information). Once a ciphertext comes in, gateway W checks it using the trapdoor list storing on its side. Seemingly, our problem has been addressed well. However, in the above construction, regarding the security reduction, on the one hand, the adversaries are not allowed to issue the public key query on user ID_i^* , where ID_i^* is the identity in the challenge phase. Nevertheless, this limitation is a bit restrict such that it can only be applied in some special circumstances. On the other hand, its security is reduced in the random oracle model, which may be insecure in some cases [17].

To overcome these drawbacks, we resort to zero-knowledge proof of knowledge [14] to hide the message and the identity information of the privileged user, such that the adversaries could be given both identities and public

keys of all the users while achieving the confidentiality of the message content and the anonymity of the privileged user. In our second construction, we use a trusted certificate authority (CA) to issue certificates, which are actually the signatures of the users' public keys and identity information generated by CA with its master key. Now the verification part of the ciphertext is replaced by a signature based on a non-interactive zero-knowledge proof of knowledge, which in our case is consisted of the plaintext and the certificate of the privileged user, and gateway W checks the validity of this zero-knowledge proof of knowledge. The tough problem here is how to generate the certificates for the users in an efficient way under the setting of asymmetric bilinear groups. Fortunately, we found that [1] has presented an efficient structure-preserving signature in asymmetric bilinear maps, for which the proof is extractable and therefore yields an efficient non-interactive zero-knowledge proof of knowledge.

Related Work. Verifiable encryption has the property that the validity of ciphertext can be verified without knowledge of private key, and it has been used to construct solutions for fair exchange [3] [4], escrow schemes [19], signature sharing schemes [12] and publicly verifiable secret sharing [22]. The concept of verifiable encryption was first introduced by Stadler [22] with the cut-and-choose methodology in the context of publicly verifiable sharing schemes in 1996. Then, Asokan et al. [3] proposed a more general form of verifiable encryption with perfect separability for the purpose of fair exchange of signatures in 1998. Bao et al. [4] gave a verifiable encryption scheme without using the cut-and-choose methodology, but it failed to provide semantic security [13]. Camenisch et al. [9] proposed an anonymous verifiable encryption scheme which did not use the cut-and-choose methodology in 2001, but the prover needs to know the private key of the receiver. Camenisch et al. [10] introduced a verifiable encryption system that provides chosen ciphertext security and avoids inefficient cut-and-choose proofs in 2003; however, it requires to use Paillier encryption function [18].

The notion of zero-knowledge proof was put forward by Goldwasser, Micali and Rackoff in [14]. In a zero-knowledge proof protocol, a prover convinces a verifier that a statement is true, while the verifier learns nothing except the validity of the assertion. A proof of knowledge [5] is a protocol where the verifier is convinced that the prover knows a certain quantity w satisfying some kinds of relation R with respect to a commonly known string x . If this can be done in such a way that the verifier learns nothing besides the validity of the statement, this protocol is called a zero-knowledge proof of knowledge (ZKPoK) protocol [14]. That is, in zero-knowledge proof of knowledge (ZKPoK) protocols, a prover convinces a verifier that some statement is true, while the verifier learns nothing except the validity of the statement.

Hitherto, various efficient ZKPoK protocols about knowledge of discrete logarithms and their relations have been proposed [11] [8] [9] [10], of which some are used in the anonymous systems to prove their possession of certificates for authentication without revealing or certificates, and some are turned into non-interactive form, called signature of knowledge (SPK) [11].

The remainder of this paper is organized as follows. Section II defines the algorithms and security requirements of a verifiable and anonymous encryption system, and introduces the generic bilinear group model and the complexity assumptions that our proof of security depends on. Section III details our first verifiable and anonymous encryption scheme in asymmetric bilinear maps, and proves its confidentiality and anonymity under the random oracle model. Section IV presents another construction on the basis of zero-knowledge proof of knowledge, as well as its security, after pointing a shortage of the security reduction of the verifiable and anonymous encryption system in Section III. Section V concludes our paper and leaves some open problems.

II. PRELIMINARIES

In this section, we first present a formal framework of verifiable and anonymous encryption. Then we describe how an adversary will be allowed to interfere in the protocol with games between an adversary algorithm and a challenger algorithm. Furthermore, we review the definitions of bilinear pairings, and the complexity assumptions.

A. Definition

Let $S = \{ID_1, \dots, ID_n\}$ be the user set in the verifiable and anonymous encryption system, where ID_i ($i \in \{1, \dots, n\}$) represents the identity information (name, student ID, etc.) of user ID_i . Our framework is specified by the following algorithms.

- $\text{Setup}(k) \rightarrow (params, msk, (pk_G, sk_G), \mathcal{M}, \mathcal{C})$: Taking a security parameter k as input, this algorithm outputs the common system parameters $params$, the master key msk , the public key and private key pair (pk_G, sk_G) for gateway W , the description of the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Generally speaking, this algorithm is run by the KGC.
- $\text{Make-Partial-USKey}(params, msk, ID_i) \rightarrow psk_i$: Taking the common system parameters $params$, the master key msk , and the identity information ID_i as input, this algorithm outputs a partial private key psk_i for user ID_i . Generally speaking, this algorithm is run by the KGC and its output is sent to user ID_i through a confidential and authentic channel.
- $\text{Set-USValue}(params, ID_i) \rightarrow x_i$: Taking the common system parameters $params$, and the identity information ID_i as input, this algorithm outputs a secret value x_i for user ID_i .

- $\text{Make-USKey}(params, psk_i, x_i) \rightarrow sk_i$: Taking the common system parameters $params$, the partial private key psk_i , and the secret value x_i as input, this algorithm outputs a (complete) private key sk_i for user ID_i .
- $\text{Make-UPKey}(params, x_i) \rightarrow pk_i$: Taking the common system parameters $params$, and the secret value x_i as input, this algorithm outputs a public key pk_i for user ID_i .

Both Make-USKey and Make-UPKey are run by user ID_i itself, after running Set-USValue , and they share the same secret value x_i . Separating them means that there is no need for a temporal requirement on the generation of public and private keys in the scheme. Usually, user ID_i is the only one in possession of x_i and sk_i , and x_i will be chosen at random from a suitable and large set.

- $\text{Make-Trapdoor}(params, x_i) \rightarrow T_i$: Taking the common parameters $params$, and the secret value x_i as input, this algorithm outputs a trapdoor T_i for user ID_i .
- $\text{Encrypt}(params, ID_i, pk_i, pk_G, M) \rightarrow C$: Taking the common parameters $params$, the identity $ID_i \in S$ with the corresponding public key pk_i , the public key pk_G of gateway W , and a message $M \in G_T$ as input, this algorithm outputs a ciphertext C .
- $\text{Verify}(params, L_T, sk_G, C) \rightarrow C$: Taking the common system parameters $params$, the trapdoor T_i , the private key sk_G of gateway W , and the ciphertext C as input, this algorithm outputs a ciphertext C in case of success, or $\perp$ in case of failure.
- $\text{Decrypt}(params, sk_i, C) \rightarrow M$: Taking the common parameters $params$, the private key sk_i of user ID_i , and the ciphertext C as input, this algorithm outputs a message M in case of being a member of privileged receivers, or $\perp$ in case of being a member of non-privileged receivers.

We require that a system is correct, meaning that for all $ID_i \in S$, if $(params, msk, (pk_G, sk_G), \mathcal{M}, \mathcal{C}) \leftarrow \text{Setup}(k)$, $psk_i \leftarrow \text{Make-Partial-USKey}(params, msk, ID_i)$, $x_i \leftarrow \text{Set-USValue}(params, ID_i)$, $sk_i \leftarrow \text{Make-USKey}(params, psk_i, x_i)$, $pk_i \leftarrow \text{Make-UPKey}(params, x_i)$, $T_i \leftarrow \text{Make-Trapdoor}(params, x_i)$, $C \leftarrow \text{Encrypt}(params, ID_i, pk_i, pk_G, M)$, $C \leftarrow \text{Verify}(params, L_T, sk_G, C)$, then $M = \text{Decrypt}(params, sk_i, C)$.

B. Security Model

In our model, since KGC is a trusted third party, we do not allow the adversaries to have access to master key, but we allow them to request partial private keys, or private keys, or both, for identities of their choices. The following is a list of the queries that an adversary algorithm $\mathcal{A}$ against verifiable and anonymous encryption may carry out. We define a challenger algorithm $\mathcal{B}$ to respond to these queries.

- 1) **Make-Partial-USKey** for user ID_i : Algorithm $\mathcal{B}$ responds by running algorithm $\text{Make-Partial-USKey}$ to

generate the partial private key sk_i for user ID_i .

- 2) **Make-USKey** for user ID_i : Algorithm $\mathcal{B}$ responds by running algorithm Make-USKey to generate the private key sk_i for user ID_i (first running Set-USValue for user ID_i if necessary).
- 3) **Make-UPKey** for user ID_i : We assume that public keys are available to algorithm $\mathcal{A}$. When receiving a public key request for user ID_i , algorithm $\mathcal{B}$ responds by running algorithm Set-UPKey to generate the public key pk_i for user ID_i (first running Set-USValue for user ID_i if necessary).
- 4) **Make-Trapdoor** for user ID_i : Algorithm $\mathcal{B}$ responds by running algorithm Make-Trapdoor to generate the trapdoor T_i for user ID_i .

Confidentiality and Anonymity. It is formalized by the indistinguishability of two games, and the adversary has to guess which plaintext and which identity has been encrypted in the challenge ciphertext. Note that we provide the adversary with the public and private key pair of gateway W , the trapdoor list and private information of some unprivileged users, which models a collusion between gateway W and corrupted users.

We define an anonymity and chosen plaintext attack for verifiable and anonymous encryption to ensure the confidentiality of the plaintext and the anonymity of the privileged user, which we call ANON-IND-CPA security. More precisely, confidentiality and anonymity are defined using a game between an adversary algorithm $\mathcal{A}$ and a challenger algorithm $\mathcal{B}$ that algorithm $\mathcal{A}$ cannot distinguish a ciphertext decrypted to one plaintext under one identity from a ciphertext decrypted to another plaintext under another identity.

- 1) **Initialization.** Algorithm $\mathcal{B}$ runs Setup to obtain the public parameter $params$ and the master key msk . Then, algorithm $\mathcal{B}$ generates the public and private key pair (pk_G, sk_G) for gateway W . Also, algorithm $\mathcal{B}$ generates the trapdoor list $L_T = \{T_1, \dots, T_n\}$, which stores the trapdoors of all the users in the system. Algorithm $\mathcal{B}$ gives the public parameter $params$, the public and private key pair (pk_G, sk_G) , and the trapdoor list L_T to algorithm $\mathcal{A}$ while keeping the master key msk to itself.
- 2) **Query Phase 1.** Algorithm $\mathcal{A}$ issues a sequence of queries, each query being either a $\text{Make-Partial-USKey}$ query, a Make-USKey query, a Make-UPKey query, or a Make-Trapdoor query on $ID_i \in S$. These queries may be issued adaptively.
- 3) **Challenge.** When algorithm $\mathcal{A}$ decides that Phase 2 is over, it outputs two messages $M_0^*, M_1^* \in \mathcal{M}$, two users $ID_0^*, ID_1^* \in S$ on which it wishes to be challenged. The only constraint is that ID_0^*, ID_1^* did not appear in Phase 2. To generate the challenge ciphertext, algorithm $\mathcal{B}$ chooses $d, e \in \{0, 1\}$, and runs

Encrypt on M_d^*, ID_e^* to obtain the ciphertext C^* . It sends C^* as the ciphertext to algorithm $\mathcal{A}$.

- 4) **Query Phase 2.** Algorithm $\mathcal{A}$ continues to adaptively issue Make-Partial-USKey queries, Make-USKey queries, Make-UPKey queries, Make-Trapdoor queries on $ID_i \in S \setminus \{ID_0^*, ID_1^*\}$, as in Phase 1.
- 5) **Guess.** Algorithm $\mathcal{A}$ outputs its guess $d', e' \in \{0, 1\}$ for d, e and wins the game if $d = d'$ and $e = e'$.

We refer to such an adversary algorithm $\mathcal{A}$ as an ANON-IND-CPA adversary. We define the advantage of the adversary algorithm $\mathcal{A}$ in attacking a verifiable and anonymous encryption scheme $\Pi = (\text{Setup}, \text{Make-Partial-USKey}, \text{Set-USValue}, \text{Make-USKey}, \text{Make-UPKey}, \text{Make-Trapdoor}, \text{Encrypt}, \text{Verify}, \text{Decrypt})$ as

$$\text{Adv}_{\Pi, \mathcal{A}} = |\Pr[d = d' \wedge e = e'] - 1/4|.$$

The probability is over the random bits used by the challenger and the adversary.

There is another stronger version of security, the ANON-IND-CCA security, where the adversary is not only allowed to issue the above queries adaptively, but also allowed to issue decryption queries.

C. Computational Assumption

Let G and $\hat{G}$ be two multiplicative cyclic groups of prime order q . Let g be a generator of G and $\hat{g}$ be a generator of $\hat{G}$, we define $\hat{e} : G \times \hat{G} \rightarrow G_T$ to be a bilinear map if it has the following properties [6] [15] [16]:

1. Bilinear: for all $g \in G$, $\hat{g} \in \hat{G}$ and $a, b \in \mathbb{Z}_q^*$, we have $\hat{e}(g^a, \hat{g}^b) = \hat{e}(g, \hat{g})^{ab}$.
2. Non-degenerate: $\hat{e}(g, \hat{g}) \neq 1$.

We say that $(G, \hat{G})$ is a bilinear group if the group action in $(G, \hat{G})$ can be computed efficiently and there exists a group G_T and an efficiently computable bilinear map $\hat{e} : G \times \hat{G} \rightarrow G_T$ as above.

Decisional BDH. The decisional bilinear Diffie-Hellman (BDH) problem is that for any probabilistic polynomial-time algorithm, it is difficult to distinguish $(g, g^a, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, \hat{e}(g, \hat{g})^{abc})$ from $(g, g^a, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, Z)$, where $g \in G$, $\hat{g} \in \hat{G}$, $Z \in G_T$, $a, b, c \in \mathbb{Z}_q^*$ are chosen independently and uniformly at random.

Decisional DH. The decisional Diffie-Hellman (DH) problem is that for any probabilistic polynomial-time algorithm, it is difficult to distinguish (g, g^a, g^c, g^{ac}) from (g, g^a, g^c, Z) , where $g \in G$, $Z \in G$, $a, c \in \mathbb{Z}_q^*$ are chosen independently and uniformly at random.

III. PROPOSED SCHEME

In this section, we give a secure verifiable and anonymous encryption scheme based on the techniques in [2]. Besides, we provide the security reduction for this scheme in the random oracle model.

A. Description

Let $S = \{ID_1, \dots, ID_n\}$ be the recipient set in the system. Let $\hat{e} : G \times \hat{G} \rightarrow G_T$ be a bilinear map over bilinear groups $G, \hat{G}$ of prime order q with generators $g \in G, \hat{g} \in \hat{G}$ respectively. Our verifiable and anonymous encryption scheme in asymmetric bilinear maps consists of the following six algorithms.

- **Setup(k):** This algorithm takes a security parameter k as input. It runs as follows.
 - 1) Selects $s, \beta \in \mathbb{Z}_q^*$ and computes $g_1 = g^s, g_2 = \hat{g}^s, g_3 = \hat{g}^\beta$.
 - 2) Defines a hash function $H_1 : \{0, 1\}^* \rightarrow \hat{G}$.
 - 3) Chooses $x \in \mathbb{Z}_q^*$ uniformly at random and computes $X = g^x$. It sets (X, x) as the public key and private key pair for gateway W .

Thus, the common system parameters are $params = (g, g_1, \hat{g}, g_2, g_3, H_1)$. The master key is $msk = s \in \mathbb{Z}_q^*$. The message space is $\mathcal{M} \in G_T$ and the ciphertext space is $\mathcal{C} \in G_T$.

- **Make-Partial-USKey($params, msk, ID_i$):** This algorithm takes the common system parameters $params$, the master key msk , and the identity information ID_i as input. It outputs $psk_i = H_1(ID_i)^s$ as the partial private key associated for user ID_i .
- **Set-Secret-USValue($params, ID_i$):** This algorithm takes the common system parameters $params$, and an identity ID_i as input. It selects $x_i \in \mathbb{Z}_q^*$, and outputs x_i as the secret value for user ID_i .
- **Make-USKey($params, psk_i, x_i$):** This algorithm takes the common system parameters $params$, the partial private key psk_i , and the secret value x_i as input. It outputs a private key $sk_i = psk_i^{x_i} = H_1(ID_i)^{sx_i}$ for user ID_i .
- **Make-UPKey($params, x_i$):** This algorithm takes the common system parameters $params$, and the secret value x_i as input. It outputs a public key $pk_i = (X_i, Y_i)$ for user ID_i , where $X_i = g^{x_i}$ and $Y_i = g_1^{x_i} = g^{sx_i}$. Note that the validity of the public key pk_i can be checked by the equation $\hat{e}(X_i, g_2) = \hat{e}(Y_i, \hat{g})$.
- **Make-Trapdoor($params, x_i$):** This algorithm takes the common system parameters $params$, and the secret value x_i as input. It outputs a trapdoor $T_i = (\hat{X}_i, \hat{Y}_i)$ for user ID_i , where $\hat{X}_i = \hat{g}^{x_i}$ and $\hat{Y}_i = g_2^{x_i} = \hat{g}^{sx_i}$. Note that gateway W can check the validity of T_i by the equations $\hat{e}(X_i, \hat{g}) = \hat{e}(g, \hat{X}_i)$ and $\hat{e}(Y_i, \hat{g}) = \hat{e}(g, \hat{Y}_i)$, but it has no idea about user ID_i and the corresponding trapdoor T_i . We assume that gateway W stores all T_i in a trapdoor list L_T .
- **Encrypt($params, ID_i, pk_i, pk_G, M$):** This algorithm takes the common parameters $params$, the identity information $ID_i \in S$ with the corresponding public key $pk_i = (X_i, Y_i)$, the public key pk_G of gateway W , and a message $M \in G_T$ as input. It runs as follows.

- 1) Chooses $r \in Z_q^*$, computes $C_1 = g^r$ and

$$C_2 = M \cdot \hat{e}(Y_i, H_1(ID_i))^r,$$

$$C_3 = \hat{e}(X, g_3)^r \cdot \hat{e}(Y_i, \hat{g})^r.$$

- 2) Outputs the ciphertext $C = (C_1, C_2, C_3)$.

- **Verify**($params, L_T, sk_G, C$): This algorithm takes the common parameters $params$, the trapdoor list L_T , the private key sk_G of gateway W , and the ciphertext C as input. It parses the ciphertext C as (C_1, C_2, C_3) , and checks whether $\hat{e}(g_1, \hat{X}_i) = \hat{e}(g, \hat{Y}_i)$ and $C_3 = \hat{e}(C_1, g_3)^x \cdot \hat{e}(C_1, \hat{Y}_i)$. If both of the two equations hold, it outputs the ciphertext C . Otherwise, it outputs a failure symbol $\perp$.
- **Decrypt**($params, sk_i, C$): This algorithm takes the common system parameters $params$, the private key sk_i of user ID_i , and the ciphertext C as input. It outputs $M = C_2 / \hat{e}(C_1, sk_i) = C_2 / \hat{e}(C_1, H_1(ID_i)^{sx_i})$ or a failure symbol $\perp$.

Our construction also achieves traceability, i.e. the PKG can reveal the identities of privileged users if necessary.

- **Query-Trace**($params, C$): This algorithm takes the common system parameters $params$, and the ciphertext C as input. It outputs a trapdoor T_i via the Verify algorithm.
- **Trace**($params, pk_i, psk_i, T_i$): This algorithm takes the common parameters $params$, the public key pk_i , the partial private key psk_i , and the trapdoor T_i as input. It checks whether $\hat{e}(g, \hat{Y}_i) = \hat{e}(Y_i, \hat{g})$ and $\hat{e}(X_i, H_1(ID_i)^s) = \hat{e}(Y_i, H_1(ID_i))$ hold. If such ID_i is found, it outputs ID_i . Otherwise, it outputs a failure symbol $\perp$.

B. Security Analysis

We present the security reduction of our verifiable and anonymous encryption scheme by showing that it is confidential and anonymous under the Decisional BDH assumption in the random oracle model.

Theorem 1: The above scheme is confidential and anonymous in the random oracle model assuming that the Decisional BDH assumption holds in $(G, \hat{G})$.

Proof. Suppose there exists an algorithm $\mathcal{A}$ breaks the confidentiality and anonymity of our verifiable and anonymous scheme. Thus, we construct an algorithm $\mathcal{B}$ that solves the Decisional BDH problem, which is given as input a random tuple $(g, g^a, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, Z)$, and outputs 1 (Z is $\hat{e}(g, \hat{g})^{abc}$) or 0 (Z is a random element in G_T).

Initialization. To generate the system parameters, algorithm $\mathcal{B}$ runs as follows.

- Sets $g_1 = g^a$, $g_2 = \hat{g}^a$, $g_3 = \hat{g}^b$, and outputs $params = (g, g_1, \hat{g}, g_2, g_3, H_1)$ as the public parameter, where H_1 is a random oracle controlled by algorithm $\mathcal{B}$.

- Chooses $x \in Z_p^*$ uniformly at random and computes $X = g^x$. It outputs $(pk_G, sk_G) = (X, x)$ as the public and private key pair of gateway W .
- For each user $ID_i \in S$, chooses $x_i \in Z_q^*$, outputs $T_i = (\hat{X}_i, \hat{Y}_i) = (\hat{g}^{x_i}, g_2^{x_i})$ as a trapdoor.

Algorithm $\mathcal{B}$ sends $params, (pk_G, sk_G)$ and the trapdoor list $L_T = \{T_1, \dots, T_n\}$ to algorithm $\mathcal{A}$.

H_1 -queries. At any time algorithm $\mathcal{A}$ can query the random oracle on ID_i . To respond to these queries, algorithm $\mathcal{B}$ maintains a list L_H of tuples $(ID_i, H_1(ID_i), r_i, x_i, psk_i, sk_i, pk_i, T_i)$ which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on identity ID_i , if (ID_i, r_i) already exists in the list L_H , algorithm $\mathcal{B}$ responds with $H_1(ID_i)$; otherwise, algorithm $\mathcal{B}$ chooses $r_i \in Z_q^*$, outputs $H_1(ID_i) = g_3^{r_i}$, and completes the list L_H with $(ID_i, g_3^{r_i}, r_i, \cdot, \cdot, \cdot, \cdot, \cdot)$.

Phase 1. Algorithm $\mathcal{A}$ queries $ID_i \in S$ to a series of oracles, each of which is either Make-Partial-USKey, Make-USKey, Make-UPKey, or Make-Trapdoor.

- **Make-Partial-USKey:** If (ID_i, r_i, psk_i) already exists in the list L_H , algorithm $\mathcal{B}$ responds with $psk_i = g_1^{r_i}$. Otherwise, algorithm $\mathcal{B}$ chooses $r_i \in Z_q^*$, outputs $psk_i = g_1^{r_i}$, and completes the list L_H with $(ID_i, \cdot, r_i, \cdot, psk_i, \cdot, \cdot, \cdot)$. To see psk_i is a correct partial private key with respect to ID_i . Let $r'_i = r_i/b$, thus we have that $psk_i = g_1^{r_i} = g_3^{ar'_i} = (g_3^{r'_i})^a = H_1(ID_i)^a$.
- **Make-USKey:** If (ID_i, r_i, x_i, sk_i) already exists in the list L_H , algorithm $\mathcal{B}$ responds with $sk_i = g_1^{r_i x_i}$. Otherwise, algorithm $\mathcal{B}$ chooses $r_i, x_i \in Z_q^*$, outputs $sk_i = g_1^{r_i x_i}$, and completes the list L_H with $(ID_i, \cdot, r_i, x_i, \cdot, sk_i, \cdot, \cdot)$. To see sk_i is a correct private key with respect to ID_i . Let $r'_i = r_i/b$, thus we have that $sk_i = g_1^{r_i x_i} = (g_3^{ar'_i})^{x_i} = H_1(ID_i)^{ax_i}$.
- **Make-UPKey:** If (ID_i, x_i, pk_i) already exists in the list L_H , algorithm $\mathcal{B}$ responds with $pk_i = (X_i, Y_i) = (g^{x_i}, g_1^{x_i})$. Otherwise, algorithm $\mathcal{B}$ chooses $x_i \in Z_q^*$, outputs $pk_i = (X_i, Y_i) = (g^{x_i}, g_1^{x_i})$, and completes the list L_H with $(ID_i, \cdot, r_i, x_i, \cdot, \cdot, pk_i, \cdot)$.
- **Make-Trapdoor:** If (ID_i, x_i, T_i) already exists in the list L_H , algorithm $\mathcal{B}$ responds with $T_i = (\hat{X}_i, \hat{Y}_i) = (\hat{g}^{x_i}, g_2^{x_i})$. Otherwise, algorithm $\mathcal{B}$ chooses $x_i \in Z_q^*$, outputs $T_i = (\hat{X}_i, \hat{Y}_i) = (\hat{g}^{x_i}, g_2^{x_i})$, and completes the list L_H with $(ID_i, \cdot, \cdot, x_i, \cdot, \cdot, \cdot, T_i)$.

Challenge. Algorithm $\mathcal{A}$ outputs two messages $M_0^*, M_1^* \in G_T$, and two users $ID_0^*, ID_1^* \in S$, where ID_0^*, ID_1^* did not appeared in Phase 2. Algorithm $\mathcal{B}$ executes as follows.

- Responds to H_1 -queries to obtain r_i^* , for $i \in \{0, 1\}$. Let $(ID_i^*, g_3^{r_i^*}, r_i^*, \cdot, \cdot, \cdot, \cdot, \cdot)$ be the corresponding tuple on the list L_H .
- Selects $d, e \in \{0, 1\}$, sets $C_1^* = g^c$, and computes

$$C_2^* = M_d^* \cdot Z, \quad C_3^* = \hat{e}(g^c, g_3)^x \cdot \hat{e}(g^c, g_2)^{x_e^*},$$

where $x_e^* \in \{x_0^*, x_1^*\}$.

- Outputs the challenge ciphertext $C^* = (C_1^*, C_2^*, C_3^*)$.
To see this, let $x_e^* = 1/r_e^*$, we have that

$$\begin{aligned} C_2^* &= M_d^* \cdot Z = M_d^* \cdot \hat{e}(g^{a \cdot x_e^*}, \hat{g}^{b \cdot r_e^*})^c \\ &= M_d^* \cdot \hat{e}(Y_e^*, H_1(ID_e^*))^c, \\ C_3^* &= \hat{e}(g^c, g_3)^x \cdot \hat{e}(g^c, g_2)^{x_e^*} \\ &= \hat{e}(X, g_3)^c \cdot \hat{e}(g^{a \cdot x_e^*}, \hat{g})^c \\ &= \hat{e}(X, g_3)^c \cdot \hat{e}(Y_e^*, \hat{g})^c. \end{aligned}$$

Hence, when Z equals $\hat{e}(g, \hat{g})^{abc}$, then C^* is a valid encryption of M_d^* of ID_e^* chosen by algorithm $\mathcal{A}$. On the other hand, when Z is uniform and independent in G_T , then C^* is independent of γ in the view of algorithm $\mathcal{A}$.

Phase 2. Algorithm $\mathcal{A}$ continues to adaptively query $ID_i \in S \setminus \{ID_0^*, ID_1^*\}$ to oracles Make-Partial-USKey, Make-USKey, Make-UPKey, or Make-Trapdoor. Algorithm $\mathcal{B}$ responds as in Phase 1.

Guess. Finally, algorithm $\mathcal{A}$ outputs $d', e' \in \{0, 1\}$. If $d = d'$ and $e = e'$, algorithm $\mathcal{B}$ outputs 1 meaning algorithm $\mathcal{A}$ wins the game. Otherwise, algorithm $\mathcal{B}$ outputs 0.

Let ϵ be the advantage that algorithm $\mathcal{A}$ breaks the confidentiality of the above game. We can see that if $Z = \hat{e}(g, \hat{g})^{abc}$, the simulation is exactly the same as the real attack, and algorithm $\mathcal{A}$ will output $d' = d$ and $e' = e$ with probability $1/4 + \epsilon$. Else, if Z is uniformly random, then algorithm $\mathcal{A}$'s advantage is nil and thus output $d' = d$ and $e' = e$ with probability $1/4$. Thus, we have that algorithm $\mathcal{B}$'s probability in solving the decisional BDH problem is

$$\begin{aligned} \Pr[\mathcal{A}(g, g^a, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, Z)] &= 1/2 \cdot (1/4 + \epsilon) + 1/2 \cdot 1/4 \\ &= 1/4 + \epsilon/2. \end{aligned}$$

This completes the proof of Theorem 1.

IND-CCA2 Secure Scheme. The result of Boneh and Katz [7] can be applied to the above ANON-IND-CPA secure verifiable and anonymous encryption scheme, thus we can obtain a verifiable and anonymous encryption scheme that is provably ANON-IND-CCA secure in random oracle by just making $\text{MAC}_i = H(ID_i \parallel M \parallel C)$ as the addition to the ciphertext.

IV. DISCUSSIONS

In the security proof of the above scheme, the adversaries are not allowed to issue the public key query on user ID_i^* where ID_i^* is the identity in the challenge phase. Otherwise, the vicious users in the system can easily obtain the identity of the privileged user when colluding with the untrusted gateway W . To some extent, this constraint is not very reasonable, as in most cases, the users are required to publicize both their identities and public keys on a public bulletin. That is, this system cannot be widely used in practical environment. In this section, we propose a new

scheme with a stronger security in which the adversaries could query the public keys and the certificates of all the users in the system.

A. Construction

In this construction, we use a trusted certificate authority (CA) to issue certificates, which actually are signatures of the users' public keys generated by CA with its master key, for users. Note that if a user accidentally reveals its secret key or an attacker actively compromises it, the user itself may request revocation of its certificate. Alternatively, the user's organization may request revocation if the user leaves the company or changes position and is no longer entitled to use the key.

Now in our verifiable and anonymous encryption system under the setting of asymmetric bilinear maps, the public keys and the identity information of the users in the system might be made available to any sender. To encrypt a message M to user ID_i , the sender makes use of its public key pk_i to compute a ciphertext C_i , and generates a signature of (Cert_i, C_i) based on a proof of knowledge in [14], where Cert_i is the certificate of user ID_i .

Let $S = \{ID_1, \dots, ID_n\}$ be the recipient set in the system. Let $\hat{e} : G \times H \rightarrow G_H$ be a bilinear map over bilinear groups G, H of prime order q with generators $g \in G, h \in H$ respectively. Let $H : \{0, 1\}^* \rightarrow Z_q^*$ be a collision resistant hash function. Our verifiable and anonymous encryption scheme consists of the following six algorithms.

- **Setup(k):** This algorithm takes a security parameter k as input. It chooses $u, v, w, z \in Z_q^*$ and computes $U = g^u, V = h^v, W = h^w, Z = h^z$. Thus, the common system parameters are $params = (g, h)$. The master secret key is $msk = (u, v, w, z) \in Z_q^*$ and the master public key is $mpk = (U, V, W, Z)$. The message space is $\mathcal{M} \in G$ and the ciphertext space is $\mathcal{C} \in G$. Usually, this algorithm is run by the CA.
- **Make-UKey($params, ID_i$):** This algorithm takes the common system parameters $params$, and the identity information $ID_i \in H$ as input. It chooses $x_i \in Z_q^*$, and computes $Y_i = g^{x_i}$. It outputs a public and private key pair $(pk_i, sk_i) = (Y_i, x_i)$ for user ID_i . Usually, this algorithm is run by user ID_i .
- **Certificate($params, ID_i, pk_i, msk$):** This algorithm takes the common parameters $params$, the identity information ID_i , the public key $pk_i = Y_i$, and the master secret key $msk = (u, v, w, z)$ as input. It chooses $s \in Z_q^*$, and computes

$$R_i = g^s, \quad S_i = g^{z-sw} Y_i^{-w}, \quad T_i = (h \cdot ID_i^{-u})^{\frac{1}{s}}.$$

It outputs $\text{Cert}_i = (R_i, S_i, T_i)$ as a certificate for user ID_i .

Usually, this algorithm is run by the CA. Note that here we use the optimal structure-preserving signature

proposed in [1], which is very efficient in asymmetric bilinear groups.

- **Encrypt**($params, pk_i, Cert_i, M$): This algorithm takes the common parameters $params$, the public key $pk_i = Y_i$, the certificate $Cert_i$, and a message $M \in \mathcal{M}$ as input. It runs as follows.

1) Chooses $r \in Z_q^*$, and computes $C_1 = g^r$, $C_2 = M \cdot Y_i^r$.

2) Generates a signature for $(Cert_i, C_1, C_2)$ based on the zero-knowledge proof of knowledge, which in fact is based on the non-interactive version of the Schnorr's proof system presented in [20], [21].

– Chooses $\beta_0, \beta_1, \beta_Y, \beta_M, \beta_R, \beta_S, \beta_T, \beta_D \in Z_q^*$, $g_1, g_2 \in G$ and $h_1, h_2 \in H$, and computes

$$\begin{aligned} A_0 &= g_1^{\beta_0} g_2^{\beta_Y}, & A_M &= M \cdot g_1^{\beta_M}, \\ A_Y &= Y_i \cdot g_1^{\beta_Y}, & B_0 &= h_1^{\beta_1} h_2^{\beta_T}, \\ B_R &= R_i \cdot g^{\beta_R}, & B_S &= S_i \cdot g^{\beta_S}, \\ B_T &= T_i \cdot h_1^{\beta_T}, & B_D &= ID_i \cdot g^{\beta_D}. \end{aligned}$$

– Chooses $K_0, K_1, K_2, K_3, K_4, K_5, K_Y, K_M, K_R, K_S, K_T, K_D, K_r \in Z_q^*$, and computes

$$\begin{aligned} E_r &= g^{K_r}, & E_0 &= g_1^{K_0} g_2^{K_Y}, \\ E_1 &= A_0^{-K_r} g_1^{K_2} g_2^{K_3}, & E_2 &= A_Y^{-K_r} g_1^{K_3} g_2^{K_M}, \\ F_0 &= h_1^{K_1} h_2^{K_T}, & F_1 &= B_1^{-K_r} h_1^{K_4} h_2^{K_5}, \\ F_2 &= \hat{e}(g_1, W)^{K_Y} \hat{e}(g, V)^{K_R} \hat{e}(g, h)^{K_S}, \end{aligned}$$

$F_3 = \hat{e}(U, h)^{K_D} \hat{e}(B_R, h_1)^{K_T} \hat{e}(g, B_T)^{K_R} \hat{e}(g, h_1)^{K_T}$.

– Computes

$$\begin{aligned} c &= H(C_1, C_2, E_r, E_0, E_1, E_2, F_0, F_1, F_2, F_3), \\ d_r &= K_r - cr, & d_0 &= K_0 - c\beta_0, \\ d_1 &= K_1 - c\beta_1, & d_2 &= K_2 - c\beta_0r, \\ d_3 &= K_3 - c\beta_Yr, & d_4 &= K_4 - c\beta_1\beta_R, \\ d_5 &= K_5 - c\beta_T\beta_R, & d_Y &= K_Y - c\beta_Y, \\ d_M &= K_M - c\beta_M, & d_R &= K_R - c\beta_R, \\ d_S &= K_S - c\beta_S, & d_T &= K_T - c\beta_T, \\ d_D &= K_D - c\beta_D. \end{aligned}$$

Thus, we obtain C_3 , a signature of knowledge $\mathbb{M} = (Cert_i, C_1, C_2)$ as

$$\text{SPK} \left\{ \begin{array}{l} (r, \beta_M, \beta_0, \\ \alpha_0, \alpha_1, \beta_Y, \\ \beta_D, \beta_T, \\ \alpha_T, \alpha_Y) \end{array} \left| \begin{array}{l} A_0 = g_1^{\beta_0} g_2^{\beta_Y} \wedge \\ 1 = A_0^{-r} g_1^{\alpha_0} g_2^{\alpha_Y} \wedge \\ C_1 = g^r \wedge \\ \frac{A_M}{C_2} = A_Y^{-r} g_1^{\alpha_Y} \cdot g_1^{\beta_M} \\ \wedge B_0 = h_1^{\beta_1} h_2^{\beta_T} \wedge \\ 1 = B_0^{-\beta_R} h_1^{\alpha_1} h_2^{\alpha_T} \wedge \\ \frac{\hat{e}(B_R, V) \hat{e}(B_S, h) \hat{e}(A_Y, W)}{\hat{e}(g, Z)} \\ = \hat{e}(g, W)^{\beta_Y} \\ \hat{e}(g, V)^{\beta_R} \hat{e}(g, h)^{\beta_S} \\ \wedge \frac{\hat{e}(U, B_D) \hat{e}(B_R, B_T)}{\hat{e}(g, h)} = \\ \hat{e}(U, h)^{\beta_D} \hat{e}(B_R, h_1)^{\beta_T} \\ \hat{e}(g, B_T)^{\beta_R} \hat{e}(g, h_1)^{\alpha_T} \end{array} \right. \right\} (\mathbb{M}).$$

where $\alpha_0 = \beta_0 \cdot r$, $\alpha_Y = \beta_Y \cdot r$, $\alpha_1 = \beta_1 \cdot \beta_R$, $\alpha_T = \beta_T \cdot \beta_R$.

- 3) Outputs the ciphertext $C = (C_1, C_2, C_3)$.

Usually, this algorithm is run by any sender who is given pk_i and $Cert_i$ of the users in the system.

- **Verify**($params, mpk, C$): This algorithm takes the common parameters $params$, the master public key mpk , and the ciphertext C as input. It parses the ciphertext C as (C_1, C_2, C_3) , and checks the validity of C_3 .

– Computes

$$\begin{aligned} B_1 &= \frac{\hat{e}(B_R, V) \hat{e}(B_S, h) \hat{e}(A_Y, W)}{\hat{e}(g, Z)}, \\ B_2 &= \frac{\hat{e}(U, B_D) \hat{e}(B_R, B_T)}{\hat{e}(g, h)}, \\ E_r &= C_1^c g^{d_r}, & E_0 &= A_0^c g_1^{d_0} g_2^{d_Y}, \\ E_1 &= A_0^{-d_r} g_1^{d_2} g_2^{d_3}, & E_2 &= \left(\frac{A_M}{C_2}\right)^c A_Y^{-d_r} g_1^{d_3} g_2^{d_M}, \\ F_0 &= B_0^c h_1^{d_1} h_2^{d_T}, & F_1 &= B_0^{-d_R} h_1^{d_4} h_2^{d_5}, \\ F_2 &= B_1^c \hat{e}(g_1, W)^{d_Y} \hat{e}(g, V)^{d_R} \hat{e}(g, h)^{d_S}, \\ F_3 &= B_2^c \hat{e}(U, h)^{d_D} \hat{e}(B_R, h_1)^{d_T} \hat{e}(g, B_T)^{d_R} \hat{e}(g, h_1)^{d_5}. \end{aligned}$$

– If $c = H(C_1, C_2, E_r, E_0, E_1, E_2, F_0, F_1, F_2, F_3)$, it outputs the ciphertext C . Otherwise, it outputs $\perp$.

Usually, this algorithm is run by gateway W .

- **Decrypt**($params, sk_i, C$): This algorithm takes the common system parameters $params$, the private key sk_i of user ID_i , and the ciphertext C as input. It outputs $M = C_2 \cdot C_1^{-x_i}$ or a failure symbol $\perp$.

Usually, this algorithm is run by user ID_i .

B. Security Proof

Theorem 2: The above scheme is ANON-IND-CCA secure assuming that the Decisional DH assumption holds in G and SPK is a secure zero-knowledge proof of knowledge system.

Proof. Given a ciphertext $C^* = (C_1^*, C_2^*, C_3^*)$, under the Decisional DH assumption, it is clear that algorithm $\mathcal{A}$ cannot obtain the plaintext M and the public key Y_i used in C^* from the encryption part (C_1^*, C_2^*) . On the other hand, because of the security properties on SPK [11], algorithm $\mathcal{A}$ has negligible probability to learn the identity of the privileged user from the verification part C_3^* . Also, C_3^* can be regarded as an one-time signature scheme built from a proof of knowledge system such that achieves ANON-IND-CCA security. Note that as C_3^* is based on a zero-knowledge proof of knowledge scheme, algorithm $\mathcal{A}$ acquires nothing even it colludes with CA. We omitted the details here.

V. CONCLUSIONS AND OPEN PROBLEMS

In this paper, we introduce a new primitive called verifiable and anonymous encryption, and propose two specific schemes in the setting of asymmetric bilinear maps, where we authorize an untrusted gateway W to verify whether the privileged user of an outside ciphertext is an inside member, but gateway W could not learn the content of the message and the identity information of the privileged user.

In our first verifiable and anonymous encryption scheme in asymmetric bilinear pairings, gateway W stores the trapdoors generated by the users in the system in a trapdoor

list. When it receives an outside ciphertext, it decides to broadcast it or not by verifying this ciphertext with the trapdoor list. Seemingly, it solves our problem; however, in this construction, the adversaries are not allowed to issue a public key query on ID_i^* where ID_i^* is the identity in the challenge phase, which is too strict to be widely applied in the real world, where the users are usually required to publicize their identity information and the corresponding public keys.

In our second verifiable and anonymous encryption scheme, we consider making use of a zero-knowledge proof of knowledge to generate a signature for verification in our second scheme, thus the scheme could be secure when the adversaries are given both the identity information and the public keys of all the users in the system. Though zero-knowledge proof of knowledge is very useful to achieve verifiable and anonymous encryption, we can see that it is still not very efficient even the structure of the certificates in our system are optimal structure-preserving in asymmetric bilinear maps.

In both our verifiable and anonymous encryption schemes, only one privileged user is allowed in a protocol run. Hence, we leave as an open problem the solution of building efficient verifiable and anonymous broadcast encryption systems, which are secure with short size or constant size of the ciphertext in the random oracle model or the standard model.

REFERENCES

- [1] M. Abe, J. Groth, K. Haralambiev, and M. Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 649–666. Springer, 2011.
- [2] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.
- [3] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer, 1998.
- [4] F. Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *CARDIS*, volume 1820 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 1998.
- [5] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.
- [6] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–219. Springer-Verlag, 2001.
- [7] D. Boneh and J. Katz. Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.
- [8] F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer, 2000.
- [9] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001.
- [10] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.
- [11] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.
- [12] M. K. Franklin and M. K. Reiter. Verifiable signature sharing. In *EUROCRYPT*, volume 921 of *Lecture Notes in Computer Science*, pages 50–63. Springer, 1995.
- [13] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
- [14] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
- [15] A. Joux. A one round protocol for tripartite diffie-hellman. In *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, 2000.
- [16] A. Joux and K. Nguyen. Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups. *J. Cryptology*, 16(4):239–247, 2003.
- [17] D. Naccache. Secure and practical identity-based encryption. *IACR Cryptology ePrint Archive*, 2005:369, 2005.
- [18] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
- [19] G. Poupard and J. Stern. Fair encryption of rsa keys. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 172–189. Springer, 2000.
- [20] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1990.
- [21] C.-P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.
- [22] M. Stadler. Publicly verifiable secret sharing. In *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer, 1996.