Publicly Verifiable Privacy-Preserving Group Decryption

Bo Qin[1,3,4], Qianhong Wu[2,3,4], Willy Susilo[4], and Yi Mu[4]

[1] Department of Mathematics, School of Science,
Xi'an University of Technology, Xi'an, China
qinboo@xaut.edu.cn
[2] School of Computer, Wuhan University,
Wuhan City, China
[3] Department of Computer Engineering and Mathematics,
Universitat Rovira i Virgili, Tarragona, Spain
qianhong.wu@urv.cat
[4] School of Information Technology and Computer Science
University of Wollongong
Wollongong NSW 2522, Australia
{wsusilo,ymu}@uow.edu.au

Abstract. Anonymity is one of the main concerns in group-oriented cryptography. However, most efforts, for instance, group signatures and ring signatures, are only made to provide anonymity on the sender's side. There are merely few works done to ensure anonymity in a cryptographic sense on the recipient's side in group-oriented communications. This paper formalizes the notion of *group decryption (GD)*. It can be viewed as an analog of group signatures in the context of public key encryptions. In this notion, a sender can encrypt a committed message intended to any member of a group, managed by a group manager, while *the recipient of the ciphertext remains anonymous*. The sender can convince a verifier about this fact without leaking the plaintext or the identity of the recipient. If required, the group manager can verifiably open the identity of the recipient. We propose an efficient GD scheme that is proven secure in the random oracle model. The overhead in both computation and communication is *independent of the group size*. A full ciphertext is about 0.2K bytes in a typical implementation and the scheme is practical.

Keywords: Group-oriented cryptography, Group decryption, Bilinear pairing, Anonymity.

1 Introduction

Anonymity is the main concern in group-oriented cryptography. It has attracted a lot of attentions in the context of digital signatures and been extensively studied in the literature, such as group signatures [8], ring signatures [17], etc. However, these types of anonymous signatures only provide anonymity on the sender's side in the communication. Few works have been done to ensure anonymity on the recipient's side using cryptographic primitives.

M. Yung, P. Liu, and D. Lin (Eds.): Inscrypt 2008, LNCS 5487, pp. 72–83, 2009.
© Springer-Verlag Berlin Heidelberg 2009

There exist applications where the recipients' anonymity is useful. For instance, in a privacy enhanced optimistic fair exchange scenario, two parties exchanging the secrets may not want to reveal their identities to the third party. This notion may also be useful in identity escrow and transactions over the Internet. In [11], Kiayias *et al.* illustrated other examples of recipient anonymity such as anonymous trusted third party escrow, secure oblivious retriever storage and ad hoc access structure group signature. This paper concentrates on the *identity privacy of recipients* in group-oriented public key encryption.

1.1 Our Contribution

In this paper, we formalize the notion of group decryption, in which a sender can verifiably send a message to a receiver in a group. The identity of the receiver is kept anonymous to any one except the group manager who can convincingly trace the intended receiver in the case of dispute.

We explain our basic ideas here. First, we let the sender commit the message to be sent. Before encrypting the message, the sender also randomizes the receiver's public key and the corresponding certificate, such that the randomized public key corresponds to the same secret key as the original one. Furthermore, any one can verify that the randomized certificate is still a signature of the randomized public key, but no one can link them with the original public key and certificate except the intended receiver and the group manager. Then the sender encrypts the message using the randomized public key of some group member. Finally, the sender just proves that the last encryption operation takes the committed message as input and the intended receiver can use its original secret key to decrypt it. For a practical implementation, we have to find proper encryption schemes and methods to generate the receiver's public keys and their certificates allowing repeated randomization. More specifically, our contribution includes the following aspects.

- *Model and security definitions.* We formalize the notion of the group decryption. It can be viewed as an analog of group signatures in the context of public key encryptions. In this notion, a sender can encrypt a committed message to any intended group member managed by a group manager while *the recipient of the ciphertext remains anonymous.* The sender can convince a verifier about this fact without leaking the plaintext or the identity of the recipient. If required, the group manager can *verifiably open* the identity of the recipient.
- *Efficient implementation.* We design our scheme with the original ElGamal encryption in the context of pairing groups and the CL^+-signature [1,2] to generate the certificates of the group members' public keys. The scheme is proven semantically secure against chosen ciphertext attackers in the random oracle model [5]. The overhead in both computation and communication is independent of the group size and the scheme is practical.

1.2 Related Work

In [12,13,14], a similar notion of custodian-hiding verifiable encryption was presented. A sender can verifiably encrypt a message using a chosen public key from a public key list but the intended recipient is anonymous. There is no group manager to administer the potential recipients. Their notion is designed for ad-hoc applications and each ciphertext has to contain the public key list of potential recipients. Their instantiations suffer from a linear cost in both communication and computation in addition to the public key list in each ciphertext. In the case of dispute, no group manager can revoke the anonymity of the receiver.

More recently, Kiayias *et al.* [11] independently presented a primitive referred to as *group encryption* and an efficient instantiation achieving the security goals similar to ours. Their basic idea is to let the sender first commit the message to be sent. Then the sender encrypts the message using the intended receiver's public key. The sender also encrypts the receiver's public key as well as the associated certificate from the group manager using the open manager's public key. Finally, the sender proves to a verifier that it has behaved honestly in a zero-knowledge manner. For a practical implementation, proper underlying encryption schemes have to be found to enable an easy zero-knowledge proof. They realize a scheme with a cramer-shoup variation of the Paillier cryptosystem and obtain a CCA-2 secure scheme without using random oracles. The zero-knowledge proof is interactive. It can be converted into a non-interactive one using the Fiat-Shamir transformation [9] but the security now relies on the random oracle model.

We compares our work with the related schemes in functionalities and efficiency. The schemes in [12,13,14] are similar with ours in functionalities except that the receivers in their schemes cannot be traced in the case of dispute. In efficiency, their schemes suffer from a linear overhead related to the group size while our scheme requires only a constant overhead. The group encryption scheme [11] achieves similar security goals but is less efficient. Note that the schemes in [12,13,14] can all be seen as an analog of (threshold) ring signatures in the encryption context and hence fall into one category. The scheme in [14] relies on the cut-and-choose method in zero-knowledge and is less efficient. The scheme in [12] improves the scheme in [14]. It is non-interactive and does not depend on the cut-and-choose method. The scheme in [13] is a threshold version of [12]. Hence, we only compare our scheme with [11] and [12] for clarity. The detailed comparison is given in the following table.

In the above table, ROM represents the random oracle model. $\lambda_N \geq 1024$ and $\lambda_p \geq 171$ are security parameters. **E** and **P** denote exponentiation operation and pairing map, respectively. Note that we do not differentiate a

Table 1. Comparison of related schemes

	Comp.	Length	Interaction	Trace	Security	Assumptions
[12]	$(20n+2)$**E**s	$\approx (16n+2)\lambda_N$	No	No	ROM	Standard
[11]	90**E**s	$\approx 75\lambda_N$	Yes	Yes	Standard	Non-standard
Ours	16**P**s+18**E**s	$\approx 12\lambda_p$	No	Yes	ROM	Standard

multi-exponentiation (or multi-pairing) with a single exponentiation (or a single pairing) as they have similar complexity [7]. Both our scheme and the schemes in [12,13] depend on a random oracle model. The security proof of [11] uses a standard model but it relies on non-standard computational assumptions. Table 1 shows that our scheme is more efficient than those in [11,12].

2 Modeling Group Decryption

In this section, we formalize the notion of the group decryption. It allows a sender to verifiably encrypt a committed message to any group member while the intended recipient remains anonymous. In the case of dispute, the anonymity can be verifiably revoked by the group manager.

2.1 Group Decryption Algorithms

A group decryption (GD) scheme involves four parties, a group manager who administers the group and traces the recipients when necessary, a group of registered members who anonymously receive messages from senders, a sender who might be one of the group members or an outsider and has secret messages to be sent to the registered members, and a verifier who can verify the GD protocol without secret inputs. A GD scheme consists of the following procedures.

- ParaGen: It is a polynomial time algorithm which takes as input a security parameter λ, outputs the system-wide parameter π.
- GKeyGen: It is a polynomial time algorithm which takes as input the system parameter π, outputs the group public and secret key pair (gpk, gsk).
- UKeyGen: It is a polynomial time algorithm which takes as input the system parameter π, outputs a user's public and secret key pair (upk, usk). It outputs independent key pairs in different runs with the same input π.
- Join: It is a polynomial time interactive algorithm between a user $\mathcal{U}$ who wants to join a group and the group manager $\mathcal{GM}$. $\mathcal{U}$ has input usk while $\mathcal{GM}$ has input gsk. The common input is (π, gpk, upk). The user has outputs (mpk, msk) which are the public and secret key pair of $\mathcal{U}$ as a legitimate group member. $\mathcal{GM}$ outputs an updated local database which includes a tracing trapdoor $T_{\mathcal{U}}$ corresponding to the group member $\mathcal{U}$. The tracing trapdoors form a tracing list L_T secretly maintained by the group manager. $T_{\mathcal{U}}$ and $T_{\mathcal{U}'}$ may be identical or not for $\mathcal{U} \neq \mathcal{U}'$. All the legitimate group members' public keys $\{mpk\}$ form a public key list L_{pk}.
- Encrypt: It is a polynomial time algorithm which takes as input a secret message m in the structured message space, the intended group members's public key mpk in the public key list and the system parameter π, outputs a ciphertext c in the ciphertext space.
- EnVerify: It is a polynomial time algorithm which takes as input a ciphertext c, the system parameter π, the group public key gpk and the public key list of the group members, outputs a bit 1 or 0 to represent whether the ciphertext is valid or not.

- Decryption: It is a polynomial time algorithm which takes as input a valid ciphertext c, the system parameter π, the intended group member $\mathcal{U}$'s public key mpk and secret key msk, outputs a message m in the message space.
- Trace: It is a polynomial time interactive algorithm between the group manager $\mathcal{GM}$ and a verifier. The group manager and the verifier have common inputs (π, gpk, L_{pk}, c), where c is a valid ciphertext defined by the encryption procedure. The group manager has secret inputs (gsk, L_T). After the interactive algorithm is run, the manager $\mathcal{GM}$ outputs mpk which represents the identity of the intended recipient of c and completes a zero-knowledge proof that the recipient has been traced correctly, while the verifier outputs a bit 1 or 0 to represent that the Trace procedure has been correctly run or not.

A GD scheme is said to be *correct* if all the parties follow the scheme honestly, the EnVerify algorithm outputs 1, the Decryption algorithm outputs the correct message and the verifier in the Trace procedure outputs 1.

2.2 Adversarial Model

We model the adversaries in GD schemes with the following oracles to which the adversaries can query. These oracles are maintained by a challenger.

- UKeyGen Oracle. For the i-th ($i > 0$) query, the adversary queries this oracle with an integer i. The challenger responds with the i-th user's public key upk_i but keeps the corresponding secret key usk_i. The challenger maintains a counter n to record the query times and updates $n = i$.
- Join Oracle. The adversary queries this oracle with upk_i which is an output of the UKeyGen. The challenger runs the Join procedure for (upk_i, usk_i). The transcript of this procedure and the corresponding group member public key mpk_i are sent to the adversary. The challenger updates the corresponding tracing list as the real scheme.
- Corruption Oracle. The adversary queries with mpk_i and obtains the corresponding secret key msk_i if mpk_i is in the group member public key list.
- Encryption Oracle. The adversary queries this oracle with (m, mpk_i), where m is a message in the message space and mpk_i is in the group member public key list. The challenger responds with the corresponding ciphertext c.
- Decryption Oracle. The adversary queries this oracle with a valid ciphertext for decryption. The challenger responds with the corresponding message.
- Trace Oracle. The adversary queries this oracle with a valid ciphertext. The challenger responds with the identity of the traced recipient and a proof to show that the ciphertext was sent to the traced recipient.

2.3 Security Definitions

We first consider the semantic security against chosen-chiphertext attacks. It states that the attacker cannot get any information of the encrypted message by accessing all the above oracles, including the decryption procedure provided

that the secret key of the intended recipient and the target ciphertext were not queried. More formally, it is defined by the following game between a challenger $\mathcal{CH}$ and an adversary $\mathcal{A}$.

Setup: $\mathcal{CH}$ runs ParaGen and GkeyGen algorithms to generate the system parameters π and the group public and secret key pair (gpk, gsk). (π, gpk) are sent to the attacker $\mathcal{A}$. $\mathcal{CH}$ also initializes a counter and three lists L_U, L_M, L_T to record the users, the group members, and the tracing trapdoors.

Phase 1: $\mathcal{A}$ can adaptively make all the oracles defined above.

Challenge: $\mathcal{A}$ chooses a tuple (m_0, m_1, mpk_i), where m_0, m_1 are in the message space and $mpk_i \in L_{pk}$ was never queried to the Corruption oracle. $\mathcal{CH}$ randomly selects a bit $b \in \{0, 1\}$ and outputs the challenge ciphertext $c^* = \mathsf{Encrypt}(\pi, mpk_i, m_b)$. $\mathcal{CH}$ sends c^* to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ may make another sequence of queries as in Phase 1 with a constraint that the Corruption oracle cannot be queried on mpk_i and c^* cannot be queried to the Decrypt oracle.

Output: Finally $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. $\mathcal{A}$ wins if $b' = b$. The advantage of $\mathcal{A}$ is defined as $|\Pr[b = b'] - \frac{1}{2}|$.

Definition 1. *We say that a GD scheme is semantically secure against chosen ciphertext attacks if no polynomially bounded adversary has non-negligible advantage in the above game.*

The anonymity states that the attacker cannot determine the intended recipient of a ciphertext with a probability non-negligible greater than the probability of a random guess, provided that the attacker did not query the Trace oracle or the receivers' secret keys to the Corruption oracle. Note that the attacker can choose the message to be encrypted. Formally, the anonymity is defined by the following game between a challenger $\mathcal{CH}$ and an adversary $\mathcal{A}$.

Setup: It is the same as the semantic security game.

Phase 1: $\mathcal{A}$ can adaptively make all the oracles defined above.

Challenge: $\mathcal{A}$ chooses a tuple $(m, mpk_{i_0}, mpk_{i_1})$, where $mpk_{i_0}, mpk_{i_1} \in L_{pk}$ were never queried to the Corruption oracle and m is in the message space. $\mathcal{CH}$ randomly selects a bit $b \in \{0, 1\}$ and outputs the challenge ciphertext $c^* = \mathsf{Encrypt}(\pi, mpk_{i_b}, m)$. $\mathcal{CH}$ sends c^* to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ may make another sequence of queries as in Phase 1 except that the Corruption oracle cannot be queried on mpk_{i_0}, mpk_{i_1} and c^* cannot be queried to the Trace oracle.

Output: Finally $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. $\mathcal{A}$ wins if $b' = b$. The advantage of $\mathcal{A}$ is defined as $|\Pr[b = b'] - \frac{1}{2}|$.

Definition 2. *We say that a GD scheme is anonymous if no polynomially bounded adversary has a non-negligible advantage in the above game.*

A GD scheme should allow to revoke the identity of the recipient's identity in the case of dispute. The traceability of a GD scheme is defined by the following game between a challenger $\mathcal{CH}$ and an adversary $\mathcal{A}$.

Setup: It is the same as the semantic security game.
Probe Phase: $\mathcal{A}$ can adaptively make queries to all the oracles defined above.
Output: $\mathcal{A}$ outputs a valid ciphertext c^*. $\mathcal{A}$ wins if $\mathcal{CH}$ cannot output the identity of the recipient in the member list or provide a valid zero-knowledge proof of the traced identity accepted by an honesty verifier.

Definition 3. *We say that a group decryption scheme is traceable if no polynomially bounded adversary has non-negligible probability to win the above game.*

3 Preliminaries

3.1 Bilinear Pairings

The implementation of our scheme requires the bilinear pairing techniques [18]. Let PGen be an algorithm that, on input a security parameter 1^λ, outputs a tuple $\Upsilon = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g_1, g_2, e)$, where $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$ have the same prime order p. $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ is an efficient bilinear map if it holds that $e(g_1, g_2) \neq 1$ and for all $h_1 \in \mathbb{G}_1, h_2 \in \mathbb{G}_2$ and $u, v \in \mathbb{Z}$, $e(h_1^u, h_2^v) = e(h_1, h_2)^{uv}$.

There are three types of pairing groups [10]: (1) $\mathbb{G}_2 = \mathbb{G}_1$. (2) $\mathbb{G}_2 \neq \mathbb{G}_1$ in which there is an *efficient distortion map* $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ but there is no *efficient* distortion map $\varphi : \mathbb{G}_1 \to \mathbb{G}_2$, where the distortion map satisfies $\psi(g_2^u) = \psi(g_2)^u \in \mathbb{G}_1$ for any $u \in Z_p$. (3) $\mathbb{G}_2 \neq \mathbb{G}_1$ but there is no *efficient* distortion map $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ or $\varphi : \mathbb{G}_1 \to \mathbb{G}_2$.

3.2 Computational Assumptions

Suppose that $\Upsilon = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g_1, g_2, e) \leftarrow \mathtt{PGen}(1^\lambda)$ are pairing groups, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ are public. Our proposals are based on the following assumptions about pairing groups. We recall that these assumptions have been used by previous works in the literature [1,2,4,15].

Assumption 1 (Inverse of Bilinear Pairing (IBP) Assumption). *Given a random value $A \in \mathbb{G}_2$, for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, the probability to compute $X \in \mathbb{G}_1$ satisfying $e(X, g_2) = A$ is negligible in λ.*

The IBP assumption is weaker than the co-CDH assumption [4]: Given g_2, $g_2^a \in \mathbb{G}_2$ and $h \in \mathbb{G}_1$,compute $h^a \in \mathbb{G}_1$. An adversary $\mathcal{A}$ breaking the IBP assumption can be efficiently converted into an adversary $\mathcal{B}$ to break the co-CDH assumption. The transformation is trivial: Given a co-CDH challenge (g_1, g_2, g_1^u, g_2^v), $\mathcal{B}$ computes $A = e(g_1^u, g_2^v) = e(g_1, g_2)^{uv}$ and queries $\mathcal{A}$ with (A, g_1, g_2). $\mathcal{B}$ straightforward uses $\mathcal{A}$'s reply $X = g_1^{uv}$ to answer the co-CDH challenge. Similarly, if $\mathbb{G}_1 = \mathbb{G}_2$, the IBP assumption is implied by the classical CDH assumption in the case $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$.

The IBP assumption is an analog of the RSA assumption in the pairing group settings. We will use a strong version of the IBP assumption which can be viewed as an analog of the strong RSA assumption in the pairing group settings. This assumption holds only in the SXDH (symmetric external Diffie-Hellman) pairing groups (Type 3).

Assumption 2 (Strong Inverse of Bilinear Pairing (SIBP) Assumption). *Let Υ be pairing groups of Type 3. Given a random value $h_2 \in \mathbb{G}_2$, for any PPT adversary $\mathcal{A}$, the probability to compute a pair $(X, Y) \in \mathbb{G}_1^2$ satisfying $e(X, g_2) = e(Y, h_2)$ is negligible in λ.*

In pairing groups of type 3, the conventional DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$. Hence, such pairing groups are also called SXDH pairing groups [2]. In [2], Ateniese *et al.* exploited such pairing groups to built their practical group signatures without random oracles.

Assumption 3 (Symmetric External Diffie-Hellman (SXDH) Assumption). *Let Υ be pairing groups of Type 3. The SXDH assumption states that the standard DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

The LRSW assumption is a discrete-logarithm assumption originally introduced by Lysyanskaya *et al.* [15] and used in many subsequent works. Let $\mathbb{G}$ be a cyclic group with generator g and of order $|\mathbb{G}|$. Let g^x and g^y be given. Furthermore, assume that an oracle can be called that answers a query s by a triple(a, a^{sy}, a^{x+sxy}), where $a{=}g^z$ is a random group element of $\mathbb{G}$. Let this oracle be called for $s_1, s_2, \ldots$. Then, the problem is to generate a quadruple $(t, b, b^{ty}, b^{x+txy})$, where $t \notin \{0, s_1, s_2, \ldots\}$. Recently, a stronger form of the LRSW assumption, called Strong LRSW, was introduced by Ateniese *et al.* [1]. Strong LRSW only holds in SXDH pairing groups.

Assumption 4 (Strong LRSW Assumption). *For SXDH pairing groups $\Upsilon = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g_1, g_2, e) \leftarrow \mathtt{PGen}(1^\lambda)$, Let $X, Y \in \mathbb{G}_2$ be chosen at random, and $O_{X,Y}(\cdot)$ be an oracle that takes as input a value $v \in \mathbb{Z}_p^*$, and outputs an LRSW-tuple (a, a^x, a^{y+vxy}) for a random $a \in \mathbb{G}_1$. Then for any PPT adversary $\mathcal{A}^{(\cdot)}$ and all $u \in \mathbb{Z}_p^*$,*

$$\Pr\left[\begin{array}{l|l} x \leftarrow \mathbb{Z}_p & (a_1, a_2, a_3, a_4, a_5) \leftarrow \mathcal{A}^{O_{X,Y}}(\cdot)(g_1, g_2, X, Y) \\ y \leftarrow \mathbb{Z}_p & \wedge a_1 \in \mathbb{G}_1 \wedge a_2 = a_1^u \wedge a_3 = a_1^x \\ X = g_2^x & \wedge a_4 = a_1^{ux} \wedge a_5 = a_1^{y+uxy} \wedge u \notin \mathbb{Q} \\ Y = g_2^y & \end{array}\right] \leq \frac{1}{poly(\lambda)}$$

where $\mathbb{Q}$ is the set of queries $\mathcal{A}$ makes to $O_{X,Y}(\cdot)$.

4 Proposed Group Decryption Scheme

We propose a GD scheme following the definition. Currently and independently, Kiayias *et al.* [11] presented a primitive called *group encryption* to achieve the same security goals as ours. We refer to this primitive as group decryption to stress the anonymity on the receiver's side. Our implementation requires a number of knowledge signatures in pairing groups which can be found in the full version of the paper [16].

– **ParaGen:** Let $\Upsilon = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, g_1, g_2, e) \leftarrow \mathtt{PGen}(1^\lambda)$ be pairing groups of Type 3. $\mathcal{H}(\cdot) : \{0,1\}^* \rightarrow \mathbb{Z}_p$ is a cryptographic hash function. Let h_2 be an independent generator of $\mathbb{G}_2$. The global parameter set is $\pi = \{\Upsilon, \mathcal{H}, h_2\}$.

This procedure is to generate the system parameters. It can be done by a trusted third-party or the group manager in a verifiable manner. This is possible as this procedures requires no secret inputs. That is, all the inputs to generate the system parameters can be published to enable any one to verify the execution of ParaGen.

- **GKeyGen:** This procedure is to generate the group key and run by the group manager. Randomly select x, y in $\mathbb{Z}_p^*$. Compute $X = g_2^x, Y = g_2^y$. The public and secret keys of the group manager are $gpk = (X, Y), gsk = (x, y)$.
- **UKeyGen:** This procedure is to generate the users' keys and run by each group user. Some of these users may join the group in the Join stage. Choose at random u in $\mathbb{Z}_p^*$. Compute $U = e(g_1, g_2)^u$. The public and secret keys of the user are $upk = U, usk = u$.
- **Join:** A user $\mathcal{U}$ can join a group and become a group member via the following protocol with the group manager $\mathcal{GM}$.

  1. $\mathcal{U}$ sends $E = g_1^u, T = g_2^u$ to $\mathcal{GM}$ via a (an Ind-CCA2) *confidential* channel and proves the knowledge of the decryption key: $\varrho = PK\{u|E = g_1^u\}$. Since T is used as the tracing trapdoor for the group manager, it cannot be obtained by anyone except $\mathcal{GM}$. Note that the DDH problem is easy in $\mathbb{G}$ and E has to be keep confidential to attackers to meet anonymity. Hence, a confidential channel is required to transfer E, T.
  2. This procedure is to generate the member certificate for legitimate members. $\mathcal{GM}$ checks the validity of ϱ and $e(E, g_2) = e(g_1, T) = U$. If any check fails or T has been in its local database, $\mathcal{GM}$ aborts the Join protocol. Else, $\mathcal{GM}$ blindly generates a CL$^+$-signature $S = (a_1, a_2, a_3, a_4, a_5)$ on the user's secret key u corresponding to the public key $U = e(g_1, g_2)^u$, where $a_1 = g_1^\gamma, a_2 = E^\gamma, a_3 = a_1^x, a_4 = a_2^x, a_5 = (a_1 a_4)^y$ for a randomly chosen value $\gamma \leftarrow \mathbb{Z}_p^*$. $\mathcal{GM}$ sends $S = (a_1, a_2, a_3, a_4, a_5)$ to $\mathcal{U}$ as its group certificate corresponding to U, where the secret key is hidden in the form $a_2 = a_1^u$. $\mathcal{GM}$ adds (T, U) to its local database.
  3. The registering user shows that he/she is the owner of the member certificate and complete his/her member public key in this step. The user checks that the group certificate $S = (a_1, a_2, a_3, a_4, a_5)$ is a valid CL$^+$-signature [1]: $e(a_1, X) = e(a_3, g_2), e(a_2, X) = e(a_4, g_2)$, $e(a_1 a_4, Y) = e(a_5, g_2)$. If any equation does not hold, the Join protocol fails. Else, the user computes a knowledge signature $\sigma = KS\{u, T|e(a_1, T) = e(a_2, g_2) \wedge e(g_1, T) = U \wedge a_1^u = a_2\}(gpk||upk||S)$ on a message containing the group public key, the user's own public key and the corresponding certificate. Here, a knowledge signature $\sigma = KS\{x|y = f(x)\}(m)$ denotes a signature σ of message m showing the knowledge of x such that $y = f(x)$. The user $\mathcal{U}$ who has become a group member obtains its member public key $mpk = \{S, U, \sigma\}$ and secret key $msk = u$.

- **Encryption:** Let a sender want to send a committed message $m \in G_1$ to a group member $\mathcal{U}$. It can verifiably send it to $\mathcal{U}$ without leaking the identity of $\mathcal{U}$ as follows.

1. Membership check: The sender verifies the validity of S and σ. If any check fails, the sender aborts. This procedure is to convince the sender that its intended receiver is a registered group member.
2. Message commitment: For $m \in \mathbb{G}_1$, commit the secret message m by randomly choosing $\delta \leftarrow \mathbb{G}_1$ and computing $c_0 = e(m, g_2)e(\delta, h_2)$.
3. Key Re-randomization: Randomly select $r \leftarrow \mathbb{Z}_p^*$ and randomize the group certificate of U by computing $c_1 = a_1^r$, $c_2 = a_2^r$, $c_3 = a_3^r$, $c_4 = a_4^r$, $c_5 = a_5^r$.
4. Message encryption: Randomly choose $s \leftarrow \mathbb{Z}_p^*$, compute $c_6 = a_1^s$, $c_7 = m^{-1}a_2^s$.
5. Encryption proof: Prove that (c_0, c_6, c_7) has been correctly generated by compute the knowledge signature $c_8 = KS\{M, s | e(c_7, g_2)c_0 = e(M, g_2)e(\delta, h_2) \wedge c_6 = a_1^s \wedge M = a_2^s\}(c_0||c_1||\cdots||c_7)$, which is equivalent to the following knowledge signature: $c_8' = KS\{m, s | c_0 = e(m, g_2)e(\delta, h_2) \wedge c_6 = a_1^s \wedge c_7 = m^{-1}a_2^s\}(c_0||c_1||\cdots||c_7)$. Output $c = (c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$ as the resulting ciphertext of message m.

– **Encryption Verification:** Any verifier can verify the validity of the ciphertext as follows:

1. Check that $e(c_1, X) = e(c_3, g_2), e(c_2, X) = e(c_4, g_2), e(c_1c_4, Y) = e(c_5, g_2)$.
2. Check that c_8 is a valid knowledge signature as defined.

If any check fails, the ciphertext is rejected. Else it is accepted.

– **Decryption:** The group member $\mathcal{U}$ with a secret key u decrypts a ciphertext c as follows:

1. Check that $c_2 = c_1^u$ to decide whether the ciphertext is intended to $\mathcal{U}$.
2. Check that validity of the CL^+-signature by the equalities $e(c_3, g_2) = e(c_1, X)$, $e(c_4, g_2) = e(c_2, X)$, $e(c_5, g_2) = e(c_1c_4, Y)$. This check validates that the intended recipient is a registered group member.
3. Check that c_8 is a valid knowledge signature to validate that the cipherxtext is computed from the committed message.

If any check fails, the group member $\mathcal{U}$ aborts the Decryption procedure. Else, it outputs message $m = c_6^u/c_7$.

– **Receiver Tracing:** The group manager can trace the recipient as follows. It checks whether there exists $(T, \mathcal{U})$ in its local database such that $e(c_1, T) = e(c_2, g_2)$. If so, the group manager outputs U. Else it outputs an error message. The group manager can prove to a verifier that the recipient has been correctly traced with the following zero-knowledge proof $PK\{T | e(c_1, T) = e(c_2, g_2) \wedge e(g_1, T) = U\}$.

The correctness of the scheme follows from a straightforward verification. For the security, we have the following claims. The proofs are given in the full version of the paper [16].

Theorem 1. *The proposed GD scheme is semantically secure against chosen ciphertext attacks in the random oracle model under the DDH assumption and the Strong LRSW assumption in SXDH pairing groups.*

Theorem 2. *The proposed GD scheme is anonymous in the random oracle model under the DDH assumption and the Strong LRSW assumption in SXDH pairing groups.*

Theorem 3. *The proposed GD scheme is traceable in the random oracle model under the Strong LRSW assumption in SXDH pairing groups.*

Since we use the traditional Fiat-Shamir transformation to convert the required interactive zero-knowledge proofs into knowledge signatures, our security proofs rely on the random oracle model. However, one may note that a recent result [6] on the Fiat-Shamir transformation can be applied to our scheme to remove the random oracle. Following [6], the signer (i.e., the sender or the group manger in our scheme) has a primary public key matching primary secret key. Each time it wants to produce a knowledge signature, it generates a fresh pair of secondary public and secret keys and produces the signature as a function of these variables, the primary keys and the message. Verification requires not only the primary public key but also the secondary one associated to the message. Security requires that it be computationally infeasible to forge a relation between the primary public key and any secondary public key that was generated by the signer, even under a chosen-message attack.

5 Conclusion

In this paper, we formalized the notion of GD schemes. It allows a sender to verifiably encrypt a committed message intended to any group member. The recipient of the ciphertext remains anonymous. The group manager can verifiably open the identity of the recipient if necessary. We proposed the first GD scheme from pairing groups secure in the random oracle model, but it can improved to be secure in the standard model using a recent result [6]. Our scheme has *constant complexity* in both computation and the communication.

Disclaimer and Acknowledgments

This work was supported by the Australian ARC Discovery Grant DP0877123, and partially by the China NSF projects 60673071 and 60873268, the China NSF key project 90718006, the China 863 projects 2008AA01Z40, 2006AA01Z442 and 2007AA01Z411, the China 973 project 2007CB310801, and the Spanish Ministry of Science and Innovation through projects TSI2007-65406-C03-01 “E-AEGIS” and CONSOLIDER CSD2007-00004 “ARES”, and by the Government of Catalonia under grant 2005 SGR 00446. The authors being with the UNESCO Chair in Data Privacy are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization.

References

1. Ateniese, G., Camenisch, J., de Medeiros, B.: Untraceable RFID Tags via Insubvertible Encryption. In: Meadows, C. (ed.) ACM CCS 2005, pp. 92–101. ACM Press, New York (2005)
2. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical Group Signatures without Random Oracles (2005), http://eprint.iacr.org/
3. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
4. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
5. Bellare, M., Rogaway, P.: Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In: Ganesan, R., Sandhu, R. (eds.) ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)
6. Bellare, M., Shoup, S.: Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 201–216. Springer, Heidelberg (2007)
7. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
8. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
9. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
10. Galbraith, S.-D., Paterson, K.G., Smart, N.-P.: Pairings for Cryptographers (2006), http://eprint.iacr.org/
11. Kiayias, A., Tsiounis, Y., Yung, M.: Group encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 181–199. Springer, Heidelberg (2007)
12. Liu, J.K., Tsang, P.P., Wong, D.S.: Efficient verifiable ring encryption for ad hoc groups. In: Molva, R., Tsudik, G., Westhoff, D. (eds.) ESAS 2005. LNCS, vol. 3813, pp. 1–13. Springer, Heidelberg (2005)
13. Liu, J.K., Tsang, P.P., Wong, D.S., Zhu, R.W.: Universal custodian-hiding verifiable encryption for discrete logarithms. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 389–409. Springer, Heidelberg (2006)
14. Liu, J.K., Wei, V.K., Wong, D.S.: Custodian-hiding verifiable encryption. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 51–64. Springer, Heidelberg (2005)
15. Lysyanskaya, A., Rivest, R.-L., Sahai, A., Wolf, S.: Pseudonym Systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
16. Qin, B., Wu, Q., Susilo, W., Mu, Y.: Group Decryption (The full version) (2006), http://eprint.iacr.org/
17. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
18. Verheul, E.R.: Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 195–210. Springer, Heidelberg (2001)