Group-Based Source-Destination Verifiable Encryption with Blacklist Checking

Zhongyuan Yao(✉), Yi Mu, and Guomin Yang

School of Computing and Information Technology, Institute of Cybersecurity and Cryptology, University of Wollongong, Keiraville, Wollongong 2522, Australia
zy454@uowmail.edu.au

Abstract. We consider user conditional privacy preservation in the context of public key encryption. Unlike the full privacy preservation, our conditional one ensures that the message sender's as well as the intended receiver's privacy are well preserved while their legitimation can still be verified; besides, the actual sender of an encrypted message can only be identified by the intended receiver. Furthermore, considering the practical scenario where the communication channels between some senders and receivers are controlled with a blacklist (BL), we address the issue how a message sender proves the legitimation of the communication channel with its intended communicator according to the BL. Previous works only partially solve the former problem and there exists no solution addressing the two aforementioned problems simultaneously. In this paper, we present an encryption scheme which keeps not only the transmitted message confidential but also the user's conditional privacy preserved. Besides, given the BL, our scheme also empowers the message sender the capability to give a proof of the legitimation of the communication channel with its communication partner without leaking their identities. In other words, only message senders form unblocked communication channels are able to produce such a proof. We provide the security models for our scheme and prove its security under the random oracle model.

Keywords: Public key encryption · Blacklist checking
Conditional privacy preservation · Source-destination verifiability

1 Introduction

Background. The security concerns of the public key encryption are mainly on the secrecy of the encrypted data. Some well studied security models, such as indistinguishably and non-malleability [8,13,20], are examples catering for different security requirements of the encrypted data. However, since encryption schemes are deployed in various hostile environments, the user privacy preservation problem should also be considered seriously since the attackers may be more interested in the exact parties participated in the communication.

J. K. Liu and P. Samarati (Eds.): ISPEC 2017, LNCS 10701, pp. 186–203, 2017.
https://doi.org/10.1007/978-3-319-72359-4_11

In fact, the user privacy preservation problems have been the subject of formal studies in cryptographic literature, for example, the primitives ring signature [21] and group signature [6] are popular tools protecting a message sender's privacy while still keeping the user authenticated. In the area of public key encryption (PKE), since the sender privacy preservation is considered to be an inherent property, literature related to user privacy preservation are mainly about key-privacy [4], or anonymity, which are security notions for receiver privacy preservation. In this paper, we are particularly interested in the user conditional privacy preservation property in PKE, which is different from the conventional one. The conditional privacy preservation notion keeps not only the privacy of the message sender as well as its communicator well preserved but also their legitimation publicly verifiable; besides, it also requires that, given a ciphertext, its actual sender can only be discovered by its intended receiver.

Apart from that, we take one step further by considering a more complex but practical scenario (e.g., in e-mail or other network communication systems), where an authority (or gateway) is able to forbid communications between specific message senders and receivers by blocking their communication channels, and those blocked channels are published as a blacklist by the authority. Under such condition, the message sender should be empowered with the capability to prove the legitimation of the communication channel between itself and its communicator; meanwhile, message senders from the blacklist should never be able to forge such a proof. In addition, the proof should not leak any privacy-related information of either the message sender or its communication partner.

There exists a primitive which solves our former problem partially. An example is the ring signcryption [16] which keeps the transmitted message confidential and the legitimation of the message sender publicly verifiable, but it cannot maintain the privacy preservation property of the message receiver. To the best of our knowledge, there is no solution tackling the two aforementioned problem at the same time properly.

Our Contribution. In this paper, we first present a group-based source-destination verifiable encryption scheme with blacklist checking. Our solution utilizes the zero-knowledge proof of membership and also zero-knowledge of inequality technique to handle the two previously mentioned problems, respectively.

Considering the security requirements of our scheme, we define four security models, which capture the message confidentiality, the sender and receiver privacy preservation, and the soundness of the legitimation proof. We then give security proofs under our predefined models with the help of the random oracle.

Related Work. Among all the existed primitives, the most promising one related to our problems is the ring signcryption, which was first proposed by Huang et al. [16]. As it inherits properties from both the ring signature [21] and public key encryption, this primitive provides anonymity, authenticity of the sender along with the message confidentiality. Following the work in [9], this primitive also considers protecting the receiver's privacy in the multi-recipient setting. Although some ring signcryption schemes have been proven to be insecure, this primitive remains to be a potential candidate when dealing with

problems about maintaining message confidentiality and user privacy simultaneously. However, because of the inherent property of the ring-based construction, this primitive always considers the complete anonymous of the message sender rather than the user conditional privacy preservation.

The user conditional privacy preservation is a more practical and attractive research problem comparing to the complete privacy preservation. Many existing works have considered it. The work in [18] addresses the issue about anonymous authentication of messages with traceability between the on-board-units (OBUs) and roadside units (RSUs) in vehicular ad hoc networks (VANETs), this conditional privacy preservation protocol relies on the authority to trace the origin of the authenticated messages. Another similar authentication with conditional privacy example can be found in [15], where it considers not only user conditional privacy but forward user revocation in wireless networks. The work in [10] uses pseudonym techniques to construct conditional privacy preservation methods and to protect the privacy of users in the NFC electronic payment environment.

The receiver privacy preservation, or key-privacy, problem was first formalized by Bellare et al. in [4] and later extended in [1], according to their paper, the receiver's privacy means that an eavesdropper, even in possession of a given ciphertext and a list of public keys, can not tell which specific key is the one used to generate the given ciphertext. This is the reason why they call this property key-privacy or anonymity. The paper defines practical security models about the key-privacy. Although some classical encryption schemes, such as the El Gamal scheme [12] and the Cramer-Shoup scheme [7], have already provided such key-privacy property, encryption schemes with careless construction, such as the broadcast encryption [11], still cannot hold this requirement. In [19], Mohassel discusses the key-privacy problem in hybrid encryption scenario, it shows that the combination of an anonymous key encapsulation mechanism (KEM) and an anonymous data encapsulation mechanism (DEM) cannot make the resulted hybrid encryption still anonymous unless the KEM is also weakly robust [2]. After considering the relation between the robustness and collision-freeness [2] properties of the KEM, this paper finally gives non-keyed transformation to transfer a collision-free PKE into a robust PKE. Key-privacy requirement is always considered in multi-receiver settings where multiple intended receivers are conventionally included in the generated ciphertext for the benefit that they can be easily identified by the message receiver. The work [14,22] discuss key-privacy in multi-receiver encryption scheme and use extended receiver sets including users who are not the intended receivers to hide the real receiver set. The anonymous broadcast encryption in [3] is the first work considering receiver's privacy in broadcast encryption schemes, in that paper, a broadcast encryption scheme is constructed achieving anonymity and IND-CCA security against static adversaries from a key-private, IND-CCA secure PKE scheme, however, the technique in [3] is only analyzed in Random Oracle Model. Later, Libert et al. in [17] proposed an anonymous broadcast encryption scheme with adaptive security in the Standard Model.

Paper Organization. The rest of our paper is organized as follows: Sect. 2 presents some notations and preliminaries. In Sect. 3, we give the formal definition of our group-based source-destination verifiable encryption scheme with blacklist checking and also define four security models in this section for the purpose of proving the security of our scheme. Our concrete construction of the scheme is presented in detail in Sect. 4. In Sect. 5, we prove the security of our scheme under the previously defined models respectively. In Sect. 6, we give the conclusion of our paper.

2 Notations and Preliminaries

Notations. We give notations which are used through the whole paper. Let 1^k be a binary string with length k while k is also called the security parameter. Let $\{r_i\}$ denote a set while r_i is one of its elements. When $\mathcal{PK}$ represents a set, then $|\mathcal{PK}|$ denotes the number of elements in this set, however, if a is an integer, then $|a|$ denotes the length of the binary representation of that integer. Let $\mathbb{G}$ be a multiplicative group of prime order q, then $x \xleftarrow{\mathrm{R}} G$ means the element x is randomly chosen from $\mathbb{G}$, while $X \in \mathbb{G}^l$ denotes that X is a tuple with l elements while each of them is chosen from $\mathbb{G}$. We use $\wedge$ to represent "AND" logic and $\vee$ to represent "OR" logic.

Decisional Diffie–Hellman Assumption (DDH). Let $\mathbb{G}_1$ be a multiplicative group of large prime order q with generator g. The DDH assumption for $\mathbb{G}_1$ holds if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, the following probability is negligibly close to $\frac{1}{2}$.

$$\Pr[a, b \leftarrow \mathbb{Z}_p; C_0 = g^{ab}; C_1 \leftarrow \mathbb{G}_1; d \leftarrow \{0,1\} : \mathcal{A}(g^a, g^b, C_d) = d]$$

Discrete Log Problem (DLP). The DLP in $\mathbb{G}_1$ is defined as follows: given a generator g of $\mathbb{G}_1$, a random element $C \in \mathbb{G}_1$ as input, output a $x \in \mathbb{Z}_p$ such that $g^x = C$. The DLP assumption holds in $\mathbb{G}_1$ if for any PPT adversary $\mathcal{A}$, the following probability is negligible.

$$\Pr[C \leftarrow \mathbb{G}_1; g^x = C : \mathcal{A}(g, C) = x]$$

3 Definitions and Security Models

3.1 Definition of the GSVEBC

There are three parties, the message sender, verifier and receiver respectively, involved in a group-based source-destination verifiable encryption scheme with blacklist checking (GSVEBC). In this scheme, the authority can publicly publish a set of sender receiver pairs as the blacklist denoted by BL, and each item of the BL is a block rule to forbid the communication between the sender and receiver specified in that item. The message sender creates and sends encrypted messages called GSVEBC ciphertexts to the receiver. It is the verifier which

verifies whether a given GSVEBC ciphertext comes from a given legitimated sender set and goes to a given legitimated receiver set without knowing the exact sender and receiver of that ciphertext. Besides, according to the blacklist BL, the verifier can also check whether the communication channel between the sender and receiver of a given GSVEBC ciphertext is blocked without learning any privacy information about them. The intended receiver of a GSVEBC ciphertext is the only party who recovers the original message as well the actual sender of that ciphertext. We give a definition of our GSVEBC scheme as follows;

Definition 1 (GSVEBC). *A group-based source-destination verifiable encryption scheme with blacklist checking (GSVEBC) scheme consists of the following polynomial time algorithms.*

- $\mathsf{Setup}(1^k)$: *Taking* 1^k *as input, this algorithm outputs the public parameter pp. For the ease of description, we assume the BL is included in pp, and each time when BL is changed by the authority, the pp should be changed accordingly.*
- $\mathsf{KeyGen}(pp)$: *For each user, this algorithm, on input pp, outputs a public key pair* (pk, sk)*. In order to make the notation more clear, let* (pk_s, sk_s) *denote a sender's key pair and* (pk_r, sk_r) *be a receiver's key pair.*
- $\mathsf{Enc}(pp, m, sk_s, pk_r, \mathcal{PK}_S, \mathcal{PK}_R)$: *This PPT algorithm can be executed by every message sender. Given a message* m*, the public parameter pp, two users' public key sets* $\mathcal{PK}_S, \mathcal{PK}_R$*, the message sender's private key* sk_s *and the receiver's public key* pk_r*, this algorithm outputs a GSVEBC ciphertext* C*.*
- $\mathsf{Ver}(pp, C)$: *The verification algorithm is deterministic. Taking pp and a given GSVEBC ciphertext* C *as inputs, that algorithm would first check whether the ciphertext comes from a given legitimated sender set and is sent to a given legitimated receiver set. Note that the given legitimated sender and receiver set should be included in the ciphertext* C*. After that, this algorithm can also check whether the communication between the sender and receiver of that given ciphertext* C *is permitted according to the blacklist BL included in pp. This algorithm returns a symbol of true if and only if all the above checks are successfully complete, otherwise, it returns a symbol of false. For privacy consideration, this algorithm is executed without the knowledge of the exact sender and receiver of the ciphertext* C*.*
- $\mathsf{Dec}(pp, C, sk_r)$: *The decryption algorithm* Dec *is deterministic and executed by the intended receiver. When a receiver gets* C*, he would first execute the previous verification algorithm* Ver*, if* Ver *returns a symbol of false, he just drops this message. Otherwise, the receiver executes* Dec*, which takes* pp, C *and the receiver's private key* sk_R *as inputs, and recovers the original message* m *as well the actual sender of* C*.*

Definition 2 (Security Model towards Message Confidentiality). *Setting the security parameter as* k*, then given our scheme* GSVEBC = (Setup, KeyGen, Enc, Ver, Dec)*, a polynomial* $n(\cdot)$*, a PPT (polynomial probabilistic time) adversary* $\mathcal{A}$ *and a simulator* $\mathcal{S}$*, we consider the following game between a simulator* $\mathcal{S}$ *and an adversary* $\mathcal{A}$ *capturing the message confidentiality property of our scheme:*

- *Setup phase: At the setup phase, the* Setup *algorithm of the scheme, which takes* 1^k *as input, is first run by* $\mathcal{S}$ *to produce the system parameter pp. Given a polynomial* $n(\cdot)$*,* $\mathcal{S}$ *runs* KeyGen*, with pp as input,* $n(k)$ *times. After all executions are properly finished,* $\mathcal{S}$ *gets a public key set* $\mathcal{PK}$*, a private key set* $\mathcal{SK}$*, where* $|\mathcal{PK}| = |\mathcal{SK}| = n(k)$*. The adversary* $\mathcal{A}$ *is given pp and* $\mathcal{PK}$*.*
- *Corruption phase: In order to enable* $\mathcal{A}$ *to do encryption itself,* $\mathcal{A}$ *is permitted to corrupt users with public keys from the set* $\mathcal{PK}$*. Namely,* $\mathcal{A}$ *can get the secret key of a user after submitting the corresponding public key to* $\mathcal{S}$ *as the query message in this phase. Let* $\mathcal{UPK}$ *denote the collection of all uncorrupted users.*
- *Decryption phase 1:* $\mathcal{A}$ *can also ask decryption queries adaptively to* $\mathcal{S}$*. That is, when* $\mathcal{A}$ *provides* $\mathcal{S}$ *a valid ciphertext,* $\mathcal{S}$ *needs to return the corresponding plaintext of that ciphertext to* $\mathcal{A}$*.*
- *Challenge phase:* $\mathcal{A}$ *chooses two messages* m_0, m_1 *from* $\mathcal{M}$*, two public keys* pk_s*,* pk_r *from* $\mathcal{UPK}$ *as the sender and receiver's public key respectively, two subsets* $\mathcal{PK}_S, \mathcal{PK}_R$ *from* $\mathcal{UPK}$ *such that* $pk_s \in \mathcal{PK}_S, pk_r \in \mathcal{PK}_R, |\mathcal{PK}_S| \geq 2, |\mathcal{PK}_R| \geq 2$*, and then sends them to the simulator. Upon receiving those information,* $\mathcal{S}$ *randomly chooses a bit b from* $\{0,1\}$ *and encrypts* m_b *using the encryption algorithm of our scheme, which takes* $m_b, sk_s, pk_r, \mathcal{PK}_S, \mathcal{PK}_R$ *and pp as inputs. After that, the generated ciphertext is given to* $\mathcal{A}$ *as the challenge ciphertext.*
- *Decryption phase 2: After receiving the challenge ciphertext,* $\mathcal{A}$ *can still query the decryption oracle adaptively with the only restriction that the queried ciphertext must be different from the challenge one.*
- *Guess phase: At the end of the game,* $\mathcal{A}$ *outputs the guess* $b^{'}$ *from* $\{0,1\}$ *about b. If* $b^{'} = b$*, then* $\mathcal{A}$ *succeeds in the game, otherwise* $\mathcal{A}$ *fails.*

Remark: $\mathcal{A}$ is allowed to ask hash queries under the random oracle model. According to the defined model, let $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{IND\text{-}CCA}}$ denote the probability that $\mathcal{A}$ wins the above game over random guess, then $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{IND\text{-}CCA}} = \left|\Pr\left[b' = b\right] - \frac{1}{2}\right|$.

Definition 3 (Security Model towards Sender Privacy Preservation). *Setting the security parameter as k, then given our scheme* GSVEBC = (Setup, KeyGen, Enc, Ver, Dec)*, a polynomial* $n(\cdot)$*, a PPT (polynomial probabilistic time) adversary* $\mathcal{A}$ *and a simulator* $\mathcal{S}$*, let's consider the following game, which captures the sender privacy property, played by* $\mathcal{A}$ *and* $\mathcal{S}$*:*

- *Setup phase: At the setup phase, the* Setup *algorithm of the scheme, which takes* 1^k *as input, is first run by* $\mathcal{S}$ *to produce the system public parameter pp. Given a polynomial* $n(\cdot)$*, the simulator runs* KeyGen*, with pp as input,* $n(k)$ *times. After all executions are properly finished,* $\mathcal{S}$ *gets a public key set* $\mathcal{PK}$*, a private key set* $\mathcal{SK}$*, where* $|\mathcal{PK}| = |\mathcal{SK}| = n(k)$*. The adversary* $\mathcal{A}$ *is given pp and* $\mathcal{PK}$*.*
- *Corruption phase: In order to enable* $\mathcal{A}$ *to do encryption itself,* $\mathcal{A}$ *is permitted to corrupt users with public keys from the set* $\mathcal{PK}$*. Namely,* $\mathcal{A}$ *can get the secret key of a user after submitting the corresponding public key to* $\mathcal{S}$ *as the query message in this phase. Let* $\mathcal{UPK}$ *denote the collection of all uncorrupted users.*

- *Sender extraction phase 1: When $\mathcal{A}$ makes such kind of query, he submits a ciphertext to $\mathcal{S}$, then he gets the public key of the original encryptor of that ciphertext from $\mathcal{S}$ when it is valid, otherwise, he gets nothing.*
- *Challenge phase: $\mathcal{A}$ chooses one message m from $\mathcal{M}$, pk_r from $\mathcal{UPK}$ as the receiver's public key and two subsets $\mathcal{PK}_S, \mathcal{PK}_R$ from $\mathcal{UPK}$ such that $pk_r \in \mathcal{PK}_R, |\mathcal{PK}_S| \geq 2, |\mathcal{PK}_R| \geq 2$, then sends them to $\mathcal{S}$. $\mathcal{S}$ randomly chooses a public key pk_s from the chosen subset $\mathcal{PK}_S$, and encrypts m by taking pk_s, $sk_s, pk_r, \mathcal{PK}_S, \mathcal{PK}_R$ and pp as inputs. The corresponding ciphertext is given to $\mathcal{A}$ as challenge ciphertext.*
- *Sender extraction phase 2: After receiving the challenge ciphertext, $\mathcal{A}$ can still ask sender extraction queries with the only constraint that the queried ciphertext must not be identical to the challenge one. The simulator behaves the same as in the sender extraction phase 1 in this phase.*
- *Guess phase: At the end of the game, $\mathcal{A}$ outputs his guess pk'_s about the public key of the actual sender of the challenge ciphertext from the chosen subset $\mathcal{PK}_S$. If $pk'_s = pk_s$, then $\mathcal{A}$ succeeds in the game, otherwise $\mathcal{A}$ fails.*

Remark: Under the random oracle model, $\mathcal{A}$ is allowed to ask hash queries. According to the defined model, let $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Sender\text{-}Anonymity}}$ denote the probability that $\mathcal{A}$ wins the above game over random guess, then $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Sender\text{-}Anonymity}} = \left|\Pr\left[pk'_s = pk_s\right] - \frac{1}{|\mathcal{PK}_S|}\right|$, where $|\mathcal{PK}_S|$ represents the size of the subset $\mathcal{PK}_S$.

Definition 4 (Security Model towards Receiver Privacy Preservation). *Setting the security parameter as k, then given our scheme GSVEBC = (Setup, KeyGen, Enc, Ver, Dec), a polynomial $n(\cdot)$, a PPT (polynomial probabilistic time) adversary $\mathcal{A}$ and a PPT simulator $\mathcal{S}$, let's consider the following game, which captures the receiver privacy property, played by $\mathcal{A}$ and $\mathcal{S}$:*

- *Setup phase: At the setup phase, the Setup algorithm of the scheme, which takes 1^k as input, is first run by $\mathcal{S}$ to produce the public parameter pp. Given a polynomial $n(\cdot)$, the simulator runs KeyGen, with pp as input, $n(k)$ times. After all executions are properly finished, $\mathcal{S}$ gets a public key set $\mathcal{PK}$, a private key set $\mathcal{SK}$, where $|\mathcal{PK}| = |\mathcal{SK}| = n(k)$. The adversary $\mathcal{A}$ is given pp and $\mathcal{PK}$.*
- *Corruption phase: In order to enable $\mathcal{A}$ to do encryption itself, $\mathcal{A}$ is permitted to corrupt users with public keys from the set $\mathcal{PK}$. Namely, $\mathcal{A}$ can get the secret key of a user after submitting the corresponding public key to $\mathcal{S}$ as the query message in this phase. Let $\mathcal{UPK}$ denote the collection of all uncorrupted users.*
- *Receiver extraction phase 1: In this phase, when $\mathcal{A}$ submits a ciphertext to $\mathcal{S}$, $\mathcal{S}$ needs to send back the public key of the actual receiver of that ciphertext to $\mathcal{A}$ as response when it is valid. Otherwise, $\mathcal{A}$ gets nothing.*
- *Challenge Phase: In the phase, $\mathcal{A}$ randomly chooses a message m from $\mathcal{M}$, pk_s from $\mathcal{UPK}$ as the sender's public key, two public key pk_0, pk_1 and two public key sets $\mathcal{PK}_S$, $\mathcal{PK}_R$ from $\mathcal{UPK}$ such that $pk_s \in \mathcal{PK}_S, pk_0, pk_1 \in \mathcal{PK}_R, |\mathcal{PK}_S| \geq 2, |\mathcal{PK}_R| \geq 2$. $\mathcal{A}$ then sends those information to $\mathcal{S}$. $\mathcal{S}$ randomly chooses $pk_c \in \{pk_0, pk_1\}$ as the receiver's public key and encrypts message m using algorithm Enc, which takes $m, sk_s, pk_s, pk_c, \mathcal{PK}_S, \mathcal{PK}_R$ and pp*

as inputs. $\mathcal{S}$ sends the generated ciphertext as response and challenge ciphertext to $\mathcal{A}$.

- *Receiver extraction phase 2: After the challenge phase, $\mathcal{A}$ can still ask $\mathcal{S}$ to extract the public key of the receiver of a valid ciphertext for him adaptively, the only restriction is that $\mathcal{A}$ cannot use the challenge ciphertext as a queried message in this phase.*
- *Guess phase: At the end of the game, $\mathcal{A}$ would make a guess c' about the public key pk_c of the receiver of the challenge ciphertext from the subset $\mathcal{PK}_R$. If $c' = c$, then $\mathcal{A}$ succeeds in the game, otherwise $\mathcal{A}$ fails.*

Remark: $\mathcal{A}$ is allowed to ask hash queries under the random oracle model. According to the defined model, let $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Receiver\text{-}Anonymity}}$ denote the probability that $\mathcal{A}$ wins the above game over random guess, then $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Receiver\text{-}Anonymity}} = \left|\Pr\left[c = c'\right] - \frac{1}{2}\right|$.

Definition 5 (Security Model towards Soundness of Legitimation Proof). *Setting the security parameter as k, then given our scheme GSVEBC = (Setup, KeyGen, Enc, Ver, Dec), a polynomial $n(\cdot)$, a PPT (polynomial probabilistic time) adversary $\mathcal{A}$ and a PPT simulator $\mathcal{S}$, let's consider the following game, which captures the user impersonation resistance property, played by $\mathcal{A}$ and $\mathcal{S}$:*

- *Setup phase: At the setup phase, the Setup algorithm of the scheme, which takes 1^k as input, is first run by $\mathcal{S}$ to produce the public parameter pp, here the blacklist BL is also generated by $\mathcal{S}$ and included in pp. Given a polynomial $n(\cdot)$, the simulator runs KeyGen, with pp as input, $n(k)$ times. After all executions are properly finished, $\mathcal{S}$ gets a public key set $\mathcal{PK}$, a private key set $\mathcal{SK}$, where $|\mathcal{PK}| = |\mathcal{SK}| = n(k)$. The adversary $\mathcal{A}$ is given pp and $\mathcal{PK}$.*
- *Corruption phase: In order to enable $\mathcal{A}$ to do encryption itself, $\mathcal{A}$ is permitted to corrupt users with public keys from the set $\mathcal{PK}$. Namely, $\mathcal{A}$ can get the secret key of a user after submitting the corresponding public key to $\mathcal{S}$ as the query message in this phase. Let $\mathcal{UPK}$ denote the collection of all uncorrupted users.*
- *Decryption phase: $\mathcal{A}$ can also ask decryption queries adaptively to $\mathcal{S}$. That is, when $\mathcal{A}$ provides $\mathcal{S}$ a valid ciphertext, $\mathcal{S}$ needs to return the corresponding plaintext of that ciphertext to $\mathcal{A}$.*
- *Forge phase: In this phase, $\mathcal{A}$ chooses a message $m \in \mathcal{M}$, one sender-receiver pair g^{S_i}, g^{R_j} from BL randomly as the message sender and intended receiver's public key respectively, two user sets $\mathcal{PK}_S, \mathcal{PK}_R \in \mathcal{UPK}$ as its corresponding message sender and receiver set. After that, $\mathcal{A}$ tries to produce a valid ciphertext CT. $\mathcal{A}$ sends $(m, g^{S_i}, g^{R_j}, \mathcal{PK}_S, \mathcal{PK}_R, CT)$ to $\mathcal{S}$. $\mathcal{S}$ outputs 1 if and only if $\mathsf{Dec}(R_j, CT) = (g^{S_i}, m) \bigwedge \mathsf{Ver}(pp, CT) = 1$. Otherwise, $\mathcal{S}$ outputs 0.*

Remark: $\mathcal{A}$ is allowed to ask hash queries under the random oracle model. According to the defined model, let $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Soundness}}$ denote the probability that $\mathcal{A}$ wins the above game, then $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Soundness}} = \left|\Pr[\mathsf{Dec}(R_j, CT) = (g^{S_i}, m) \bigwedge \mathsf{Ver}(pp, CT) = 1]\right|$.

4 Our Concrete Construction

For the ease of description, we first give a group-based source-destination verifiable encryption scheme without blacklist checking capability, then we extend this scheme to the one with full functionalities.

4.1 A Simple Construction Without Blacklist Checking

Setting the security parameter as k, our scheme works as follows;

- $\mathsf{Setup}(1^k)$: On input 1^k, it produces a cyclic group $\mathbb{G}$ of large prime order q with generator g, where $\mathbb{G}$ is a subgroup of $\mathbb{Z}_p^*$ and $q|p-1$. This algorithm also outputs a description of the message space $\mathcal{M} = \{0,1\}^q$ and a ciphertext space $\mathcal{C}$. $\mathbb{G}, q, g, \mathcal{M}, \mathcal{C}$ are considered as the system parameter pp and default inputs to all the following algorithms. pp also includes three collision resistance hash functions: $H_1 : \{0,1\}^q \times \mathbb{G}^3 \to \mathbb{Z}_q, H_2 : \mathbb{G} \to \{0,1\}^q, H_3 : \{0,1\}^* \to Z_q$.
- $\mathsf{KeyGen}(\cdot)$: For one user, U_i for example, he randomly chooses $x_i \in \mathbb{Z}_q$ as his private key and computes $y_i = g^{x_i} \in \mathbb{G}$ as his corresponding public key. Assuming the public key set $\mathcal{PK} = \{\dots, y_i, \dots\}$ contains all users' public key and is also published publicly.
- $\mathsf{Enc}(m, sk_s, pk_r, \mathcal{PK}_S, \mathcal{PK}_R)$: When a sender, U_i, wants to send a message to a receiver, U_j, for the purpose of illustrating our scheme more clear, let S_i, R_i denote the sender U_i and receiver U_j's secret key sk_s, sk_r respectively, accordingly, the sender and receiver's public key should be $pk_s = g^{S_i}$ and $pk_r = g^{R_j}$. Given a message $m \in \mathcal{M}$, the sender encrypts m as follows;

$$
\begin{aligned}
&r_1 \xleftarrow{R} \mathbb{Z}_q, C_1 = g^{r_1}, C_2 = g^{S_i r_1}, C_3 = g^{R_j r_1},\\
&r_2 = H_1(m, g^{S_i}, g^{R_j}, g^{S_i R_j r_1}),\\
&C_4 = g^{S_i r_2}, C_5 = g^{S_i} g^{S_i R_j r_2}, C_6 = m \oplus H_2(g^{S_i R_j \cdot (r_1 + r_2)}).
\end{aligned}
$$

  After that, the sender chooses a subgroup $\mathcal{PK}_S \subset \mathcal{PK}$, which includes the sender's public key g^{S_i}, and then proves its legitimation in that group. Here, we utilize the zero-knowledge proof technique to deal with the group membership issue. That is, the sender needs to give a proof like:

$$
pf(S_i : \log_g g^{S_i} = \log_{C_1} C_2 = \log_{g^{r_1}} (g^{S_i})^{r_1} \wedge g^{S_i} \in \mathcal{PK}_S).
$$

  To do such a proof, the sender does as follows;

  - For each public key $g^{x_l} \in \mathcal{PK}_S$ except g^{S_i}, the sender chooses challenge and response c_l, z_l randomly from $\mathbb{Z}_q$ respectively, then it computes two commitments
$$
\alpha_l = g^{z_l} (g^{x_l})^{c_l}, \beta_l = (C_1)^{z_l} (C_2)^{c_l}.
$$
  - For the sender's own pubic key g^{S_i}, it chooses $w_i \in \mathbb{Z}_q$ and sets the commitments as
$$
\alpha_i = g^{w_i}, \beta_i = (C_1)^{w_i}.
$$

Let $\{\alpha\}$ denote commitments set $\{\dots \alpha_l \dots \alpha_i \dots\}$ and $\{\beta\}$ commitments set $\{\dots \beta_l \dots \beta_i \dots\}$, where $|\alpha| = |\beta| = |\mathcal{PK}_S|$. The sender computes its challenge and response as:

$$h = H_3(\{\alpha\}, \{\beta\}, C_1, C_2, C_3, C_4, C_5, C_6),$$
$$c_i = h - \sum_{g^{x_l} \in \mathcal{PK}_S} c_l, z_i = w_i - c_i S_i.$$

- The sender sets the challenges set as $\{c\} = \{\dots c_i \dots c_l \dots\}$ the responses set as $\{z\} = \{\dots z_i \dots z_l \dots\}$, and value this two sets $\{c\}, \{z\}$ as the proof value.

The sender needs still to prove to the verifier that the generated ciphertext is sent to a legitimated receiver. To do this, the sender chooses a receiver subset $\mathcal{PK}_R \subset \mathcal{PK}$, which includes the receiver's public key, and gives a proof like:

$$pf(r_1 : \log_g C_1 = \log_{g^{R_j}} C_3 \wedge g^{R_j} \in \mathcal{PK}_R),$$

the sender generates the proof as follows;

- For each public key $g^{x_t} \in \mathcal{PK}_R$ except the intended receiver's public key g^{R_j}, the sender chooses challenge and response $\widehat{c}_t, \widehat{z}_t$ randomly from $\mathbb{Z}_q$ respectively, then it computes the commitments

$$\widehat{\alpha}_t = g^{\widehat{z}_t}(C_1)^{\widehat{c}_t}, \widehat{\beta}_t = (g^{x_t})^{\widehat{z}_t}(C_3)^{\widehat{c}_t}.$$

- For the intended receiver's pubic key g^{R_j}, it chooses $\widehat{w}_j \in \mathbb{Z}_q$ and sets the commitments as

$$\widehat{\alpha}_j = g^{\widehat{w}_j}, \widehat{\beta}_j = (g^{R_j})^{\widehat{w}_j}.$$

Let $\widehat{\{\alpha\}}$ denote commitments set $\{\dots \widehat{\alpha}_t \dots \widehat{\alpha}_j \dots\}$ and $\widehat{\{\beta\}}$ $\{\dots \widehat{\beta}_t \dots \widehat{\beta}_j \dots\}$ respectively, where $|\widehat{\{\alpha\}}| = |\widehat{\{\beta\}}| = |\mathcal{PK}_R|$. The sender computes its challenge and response as:

$$\widehat{h} = H_3(\widehat{\{\alpha\}}, \widehat{\{\beta\}}, C_1, C_2, C_3, C_4, C_5, C_6),$$
$$\widehat{c}_j = \widehat{h} - \sum_{g^{x_t} \in \mathcal{PK}_R} \widehat{c}_t, \widehat{z}_j = \widehat{w}_j - \widehat{c}_j r_1$$

- The sender sets the challenges set as $\widehat{\{c\}} = \{\dots \widehat{c}_j \dots \widehat{c}_t \dots\}$ the responses set as $\widehat{\{z\}} = \{\dots \widehat{z}_j \dots \widehat{z}_t \dots\}$, and value this two sets $\widehat{\{c\}}, \widehat{\{z\}}$ as the proof value.

After the two proofs are generated, the final ciphertext should be $CT = (C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \widehat{\{c\}}, \widehat{\{z\}})$.

– $\mathsf{Ver}(CT)$: Every user can act as the verifier. Upon receiving a given ciphertext like the above format $CT = (C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \{\widehat{c}\}, \{\widehat{z}\})$, a verifier does the following steps to verify the validity of the ciphertext:

- For the ciphertext components $(C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\})$, the verifier recomputes

$$\alpha_l^{'} = g^{z_l}(g^{x_l})^{c_l}, \beta_l^{'} = (C_1)^{z_l}(C_2)^{c_l} \text{ for each } g^{x_l} \in \mathcal{PK}_S$$

and gets two sets $\{\alpha^{'}\} = \{\ldots \alpha_l^{'} \ldots\}, \{\beta^{'}\} = \{\ldots \beta_l^{'} \ldots\}$, then it checks whether the equation

$$H_3(\{\alpha^{'}\}, \{\beta^{'}\}, C_1, C_2, C_3, C_4, C_5, C_6) = \sum_{c_l \in \{c\}} c_l$$

holds. If no, it returns a symbol of false and drops this ciphertext, otherwise it continues to the next step.

- For the ciphertext components $(C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_R, \widehat{\{c\}}, \widehat{\{z\}})$, the verifier further computes

$$\widehat{\alpha'}_t = g^{\widehat{z}_t}(C_1)^{\widehat{c}_t}, \widehat{\beta'}_t = (g^{x_t})^{\widehat{z}_t}(C_3)^{\widehat{c}_t} \text{ for each } g^{x_t} \in \mathcal{PK}_R.$$

Then it gets two sets $\widehat{\{\alpha'\}} = \{\ldots \widehat{\alpha'}_t \ldots \widehat{\alpha'}_j \ldots\}, \widehat{\{\beta'\}} = \{\ldots \widehat{\beta'}_t \ldots \widehat{\beta'}_j \ldots\}$. The verifier finally checks whether the equation

$$H_3(\widehat{\{\alpha'\}}, \widehat{\{\beta'\}}, C_1, C_2, C_3, C_4, C_5, C_6) = \sum_{\widehat{c}_t \in \widehat{\{c\}}} \widehat{c}_t$$

holds. If no, the verifier returns a symbol of false and drops this ciphertext, otherwise it returns a symbol of true and then relays this ciphertext to the receiver set.

– $\mathsf{Dec}(CT, R_x)$: This decryption algorithm are executed by all the possible receivers of a given ciphertext. When given a copy of the ciphertext $CT = (C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \widehat{\{c\}}, \widehat{\{z\}})$, all possible receivers in set $\mathcal{PK}_R$ do as following:

- They would first execute the verification algorithm Ver of our scheme as a subroutine. If Ver returns false, they drop CT and return a symbol of failure, otherwise they continue to the next step.
- Each user U_x in $\mathcal{PK}_R$ uses its secret key R_x to check whether equation $C_1^{R_x} = C_3$ holds. If not, it drops CT and returns a symbol of failure, otherwise, this user goes to the next step.
- For each of the users whose secret key satisfying the above equation, it first gets the possible public key, which is denoted by $g^{s^{'}}$, of the original sender of the given CT by computing

$$g^{s^{'}} = \frac{C_5}{{C_4}^{R_x}},$$

then it recover the encrypted message, denoted by m', as

$$m' = C_6 \oplus H_2((C_2 C_4)^{R_x}).$$

After getting $g^{s'}$ and m', it would check whether the equation

$$C_4 = g^{s' H_1(m', g^{s'}, g^{R_x}, {C_2}^{R_x})}$$

holds, if yes, this user outputs $g^{s'}$ as the public key of the actual message sender and m' as the original message. Otherwise, this user drops CT and returns a symbol of failure.

4.2 Our Concrete Construction with Blacklist Checking

Basing on the former scheme, We give another construction to empower our scheme with blacklist checking capability. Here, for simplicity, we assume the blacklist BL is publicly produced by the system authority. It contains numbers of block rule and each of which can be expressed as $< pk_s, pk_r >$, where the former is one specific sender's and the other is one specific receiver's public key respectively, such block rule is used to disable the communication from one message sender to one receiver. Our scheme assures that a verifier can check whether a given ciphertext should be rejected according to the BL.

By applying the technique of zero-knowledge proof of inequality of two discrete logarithms, which was proposed in [5], we find a way to extend our original scheme to a scheme with blacklist checking, which only add a set of proof values to the original one. Because those two schemes are pretty similar, we only give explicit description of the most different part between them.

Our public key encryption scheme with source-destination verifiability and block rules checking consists of the following polynomial time algorithms.

- $\mathsf{Setup}(1^k)$: This algorithm is similar to the previous scheme except that the public parameter pp should include the blacklist BL. Notice that pp is also considered as default input to all the following algorithms.
- $\mathsf{KeyGen}(\cdot)$: This algorithm is also identical to the aforementioned one.
- $\mathsf{Enc}(m, sk_s, pk_r, \mathcal{PK}_S, \mathcal{PK}_R)$: Apart from the encryption process of the encryption scheme of the previous scheme, here the sender also needs to generate a proof to convince the verifier that the generated ciphertext should not be blocked according to the blacklist. Assuming there is a blacklist in pp like follows;

$< . \ , \ . >$
$< g^S, g^R >$
$< . \ , \ . >$

Assuming there is one message sender with identity g^{S_i}, one ciphertext

$$CT = (C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \{\widehat{c}\}, \{\widehat{z}\})$$

which is generated by that sender and sent to a receiver with identity g^{R_j}, for each block rule, $< g^S, g^R >$ for example, in the blacklist, the message sender

needs to prove that CT does not come form a user with identity g^S or goes to a user with identity g^R. That is, according to our scheme, the message sender should produce a proof like

$$pf((S_i \vee r_1) : \log_{C_1} C_2 \neq \log_g g^S \vee \log_g C_1 \neq \log_{g^R} C_3)$$

for this rule.

According to the technique given in [5], the message sender produces such a proof pf for that ciphertext basing on the following different conditions of CT;

- When $\log_{C_1} C_2 = \log_g g^S$ and $\log_g C_1 \neq \log_{g^R} C_3$, that is $g^{S_i} = g^S$ and $g^{R_j} \neq g^R$:
  * For the case $g^{S_i} = g^S$, the message sender needs to simulate a proof like $pf((\gamma = S_i \cdot \delta, \delta) : St_0 = g^\gamma/(g^S)^\delta \neq 1 \vee St_1 = (g^{r_1})^\gamma/(g^{S_i r_1})^\delta = 1)$, where $\delta \in_R \mathbb{Z}_q$.
    That is, the message sender first chooses two statements $St_0 \in \mathbb{G}$ and $St_1 = 1 \in \mathbb{G}$, a challenge $CH \in \mathbb{Z}_q$ and two responses $e_0, e_1 \in \mathbb{Z}_q$ respectively, and sets the two commitments

$$COM_0 = {St_0}^{CH}(g)^{e_0}/(g^S)^{e_1},$$
$$COM_1 = {St_1}^{CH}(g^{r_1})^{e_0}/(g^{S_i r_1})^{e_1}$$

  * For the case $g^{R_j} \neq g^R$, the message sender gives a real proof like $pf((\widehat{\gamma} = r_1 \cdot \widehat{\delta}, \widehat{\delta}) : \widehat{St_0} = (g^R)^{\widehat{\gamma}}/(g^{R_j r_1})^{\widehat{\delta}} \neq 1 \vee \widehat{St_1} = (g)^{\widehat{\gamma}}/(g^{r_1})^{\widehat{\delta}} = 1)$, where $\widehat{\delta} \in_R \mathbb{Z}_q$.
    That is, the message sender first chooses two elements $\widehat{w_0}, \widehat{w_1} \in \mathbb{Z}_q$ and computes the two commitments

$$\widehat{COM_0} = (g^R)^{\widehat{w_0}}/(g^{R_j r_1})^{\widehat{w_1}}, \widehat{COM_1} = (g)^{\widehat{w_0}}/(g^{r_1})^{\widehat{w_1}}.$$

    The sender then computes a hash value

$$X = H_3(COM_0, COM_1, \widehat{COM_0}, \widehat{COM_1})$$

    and sets the challenge of this proof as $\widehat{CH} = X - CH$, the two responses should be

$$\widehat{e_0} = \widehat{w_0} - \widehat{CH} \cdot \widehat{\gamma}, \widehat{e_1} = \widehat{w_1} - \widehat{CH} \cdot \widehat{\delta}$$

    respectively.
  * After all the required values are properly computed, let pf denote the proof values for that block rule, then

$$pf = (St_0, St_1, CH, e_0, e_1, \widehat{St_0}, \widehat{St_1}, \widehat{CH}, \widehat{e_0}, \widehat{e_1}).$$

- If $\log_{C_1} C_2 \neq \log_g g^S$ and $\log_g C_1 = \log_{g^R} C_3$, that is $g^{S_i} \neq g^S$ and $g^{R_j} = g^R$:

* For the case $g^{R_j} = g^R$, the message sender needs to simulate a proof like $pf(\widehat{\gamma} = r_1 \cdot \widehat{\delta}, \widehat{\delta}) : \widehat{St_0} = (g^R)^{\widehat{\gamma}}/(g^{R_j r_1})^{\widehat{\delta}} \neq 1 \vee \widehat{St_1} = (g)^{\widehat{\gamma}}/(g^{r_1})^{\widehat{\delta}} = 1)$, where $\widehat{\delta} \in_R \mathbb{Z}_q$.
  That is, the message sender chooses two statements $\widehat{St_0} \in \mathbb{G}$ and $\widehat{St_1} = 1 \in \mathbb{G}$, a challenge $\widehat{CH} \in \mathbb{Z}_q$ and two responses $\widehat{e_0}, \widehat{e_1} \in \mathbb{Z}_q$ respectively, and sets the two commitments

$$\widehat{COM_0} = \widehat{St_0}^{\widehat{CH}}(g^R)^{\widehat{e_0}}/(g^{R_j r_1})^{\widehat{e_1}},$$
$$\widehat{COM_1} = \widehat{St_1}^{\widehat{CH}}(g)^{\widehat{e_0}}/(g^{r_1})^{\widehat{e_1}}.$$

* For the case $g^{S_i} \neq g^S$, the message sender gives a real proof like $pf((\gamma = S_i \cdot \delta, \delta) : St_0 = (g)^{\gamma}/(g^S)^{\delta} \neq 1 \vee St_1 = (g^{r_1})^{\gamma}/(g^{S_i r_1})^{\delta} = 1)$, where $\delta \in_R \mathbb{Z}_q$.
  That is, the message sender first chooses two elements $w_0, w_1 \in \mathbb{Z}_q$ and computes the two commitments

$$COM_0 = (g)^{w_0}/(g^S)^{w_1}, COM_1 = (g^{r_1})^{w_0}/(g^{S_i r_1})^{w_1}.$$

  The sender then computes a hash value

$$X = H_3(COM_0, COM_1, \widehat{COM_0}, \widehat{COM_1})$$

  and sets the challenge of this proof as $CH = X - \widehat{CH}$, the two responses should be

$$e_0 = w_0 - CH \cdot \gamma,\ e_1 = w_1 - CH \cdot \delta$$

  respectively.
* After all the required values are properly computed, let pf denote the proof values for that block rule, then

$$pf = (St_0, St_1, CH, e_0, e_1, \widehat{St_0}, \widehat{St_1}, \widehat{CH}, \widehat{e_0}, \widehat{e_1})$$

- If $\log_{C_1} C_2 \neq \log_g g^S$ and $\log_g C_1 \neq \log_{g^R} C_3$, that is $g^{S_i} \neq g^S$ and $g^{R_j} \neq g^R$:
  * For the case $g^{S_i} \neq g^S$, the message sender gives a real proof like $pf((\gamma = S_i \cdot \delta, \delta) : St_0 = (g)^{\gamma}/(g^S)^{\delta} \neq 1 \vee St_1 = (g^{r_1})^{\gamma}/(g^{S_i r_1})^{\delta} = 1)$, where $\delta \in_R \mathbb{Z}_q$.
    That is, the message sender first chooses two elements $w_0, w_1 \in \mathbb{Z}_q$ and computes the two commitments

$$COM_0 = (g)^{w_0}/(g^S)^{w_1},\ COM_1 = (g^{r_1})^{w_0}/(g^{S_i r_1})^{w_1}.$$

    The sender then chooses a challenge of this proof $CH \in \mathbb{Z}_q$, the two responses should be

$$e_0 = w_0 - CH \cdot \gamma,\ e_1 = w_1 - CH \cdot \delta$$

    respectively.

* For the case $g^{R_j} \neq g^R$, The message sender gives a real proof like $pf((\widehat{\gamma} = r_1 \cdot \widehat{\delta}, \widehat{\delta}) : \widehat{St_0} = (g^R)^{\widehat{\gamma}}/(g^{R_j r_1})^{\widehat{\delta}} \neq 1 \vee \widehat{St_1} = (g)^{\widehat{\gamma}}/(g^{r_1})^{\widehat{\delta}} = 1)$, $\widehat{\delta} \in_R \mathbb{Z}_q$.
That is, the message sender first chooses two elements $\widehat{w_0}, \widehat{w_1} \in \mathbb{Z}_q$ and computes the two commitments

$$\widehat{COM_0} = (g^R)^{\widehat{w_0}}/(g^{R_j r_1})^{\widehat{w_1}}, \widehat{COM_1} = (g)^{\widehat{w_0}}/(g^{r_1})^{\widehat{w_1}}.$$

The sender then computes a hash value

$$X = H_3(COM_0, COM_1, \widehat{COM_0}, \widehat{COM_1})$$

and sets the challenge of this proof as $\widehat{CH} = X - CH$, the two responses should be

$$\widehat{e_0} = \widehat{w_0} - \widehat{CH} \cdot \widehat{\gamma},\ \widehat{e_1} = \widehat{w_1} - \widehat{CH} \cdot \widehat{\delta}$$

respectively.

* After all the required values are properly computed, let pf denote the proof values for that block rule, then

$$pf = (St_0, St_1, CH, e_0, e_1, \widehat{St_0}, \widehat{St_1}, \widehat{CH}, \widehat{e_0}, \widehat{e_1}).$$

Assuming there are n rules in BL, the message sender needs to generate n proofs accordingly. Let $\{pf\}$ denote all those proofs, then the full ciphertext CT should be $(C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \widehat{\{c\}}, \widehat{\{z\}}, \{pf\})$.

– $\mathsf{Ver}(CT)$: During the execution of this algorithm, a verifier would first do the same as what in the verification algorithm of the previous scheme. Besides, to check the block rules in BL, for each proof $(St_0, St_1, CH, e_0, e_1, \widehat{St_0}, \widehat{St_1}, \widehat{CH}, \widehat{e_0}, \widehat{e_1})$ in $\{pf\}$ and its corresponding rule $< g^S, g^R >$, the verifier computes

$$COM'_0 = {St_0}^{CH}(g)^{e_0}/(g^S)^{e_1}, COM'_1 = {St_1}^{CH}(g^{r_1})^{e_0}/(g^{S_i r_1})^{e_1},$$
$$\widehat{COM}'_0 = \widehat{St_0}^{\widehat{CH}}(g^R)^{\widehat{e_0}}/(g^{R_j r_1})^{\widehat{e_1}}, \widehat{COM}'_1 = \widehat{St_1}^{\widehat{CH}}(g)^{\widehat{e_0}}/(g^{r_1})^{\widehat{e_1}})$$

and then checks whether the equation

$$CH + \widehat{CH} = H_3(COM'_0 + COM'_1 + \widehat{COM}'_0 + \widehat{COM}'_1)$$

holds. If yes, the verifier turns to the next proof in the list $\{pf\}$, otherwise it drops this ciphertext. The verifier would relay the ciphertext if all the proofs in $\{pf\}$ and rules in BL are successfully checked.

– $\mathsf{Dec}(CT, R_j)$: This algorithm shares no difference from that in the previous scheme.

5 Security Proofs

Theorem 1. *Our scheme maintains message confidentiality under the previously defined message confidentiality model assuming the DDH problem is hard in* $\mathbb{G}$ *when hash functions* H_1, H_2, H_3 *are modeled as random oracles. Concretely, if there is an adversary* $\mathcal{A}$ *which can break our scheme with non-negligible probability* ϵ*, supposing* $\mathcal{A}$ *makes at most* $q_{H_1}, q_{H_2}, q_{H_3}$ *queries to the* H_1, H_2, H_3 *hash oracles respectively, and* q_D *queries to the decryption oracle, then we can construct another algorithm* $\mathcal{B}$ *that solves the DDH problem in* $\mathbb{G}$ *with advantage at least* $\frac{1}{n^2}(1 - \frac{q_D}{2^k})\epsilon$*, where* k *is the security parameter and* n *is a constant.*

Theorem 2. *Our proposed scheme holds sender privacy under the previously defined model assuming the DDH problem is hard in* $\mathbb{G}$ *where hash functions* H_1, H_2, H_3 *are modeled as random oracles. Concretely, if there exists such an adversary* $\mathcal{A}$ *which can break our scheme with non-negligible probability* ϵ*, supposing* $\mathcal{A}$ *makes at most* $q_{H_1}, q_{H_2}, q_{H_3}$ *queries to the* H_1, H_2, H_3 *hash oracles respectively, and* q_{se} *sender extraction queries, then we can construct another algorithm that solves the DDH problem in* $\mathbb{G}$ *with probability at least* $\frac{1}{n}(1 - \frac{q_{se}}{2^k})\epsilon$*.*

Theorem 3. *Our scheme holds receiver privacy under the predefined security model assuming the DDH problem is hard in* $\mathbb{G}$ *when hash functions* H_1, H_2, H_3 *are modeled as random oracles. That is, if there is an adversary* $\mathcal{A}$ *which can break our scheme with non-negligible probability* ϵ*, assuming* $\mathcal{A}$ *asks* $q_{H_1}, q_{H_2}, q_{H_3}$ *queries to* H_1, H_2, H_3 *respectively and* q_{re} *receiver extraction queries during the game, then we can construct another algorithm* $\mathcal{B}$ *which breaks the DDH problem with probability at least* $\frac{1}{2} \cdot \epsilon - \frac{1}{2^{k-1}}$*, where* k *is the security parameter.*

Because of the page limitation, here we only give the theorem. People can find the formal proof in the full version of this paper.

6 Conclusion

We considered the user conditional privacy preservation problem. With the blacklist scenario, we explained how a message sender proves the legitimation of the communication channel with its communication partner. To solve the aforementioned two problems, we proposed a group-based source-destination verifiable encryption scheme with blacklist checking. In order to discuss the security of our scheme, we further defined three security models to capture the message confidentiality, sender privacy preservation and receiver privacy preservation accordingly, and then gave three formal proofs under the predefined models with the help of the random oracle.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. J. Cryptol. **21**(3), 350–391 (2008)
2. Abdalla, M., Bellare, M., Neven, G.: Robust encryption. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 480–497. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_28
3. Barth, A., Boneh, D., Waters, B.: Privacy in encrypted content distribution using private broadcast encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 52–64. Springer, Heidelberg (2006). https://doi.org/10.1007/11889663_4
4. 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). https://doi.org/10.1007/3-540-45682-1_33
5. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_8
6. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_22
7. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055717
8. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, New Orleans, Louisiana, USA, 5–8 May 1991, pp. 542–552 (1991)
9. Duan, S., Cao, Z.: Efficient and provably secure multi-receiver identity-based signcryption. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 195–206. Springer, Heidelberg (2006). https://doi.org/10.1007/11780656_17
10. Eun, H., Lee, H., Oh, H.: Conditional privacy preserving security protocol for NFC applications. IEEE Trans. Consum. Electron. **59**(1), 153–160 (2013)
11. Gafni, E., Staddon, J., Yin, Y.L.: Efficient methods for integrating traceability and broadcast encryption. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 372–387. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_24
12. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theor. **31**(4), 469–472 (1985)
13. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
14. Harn, L., Chang, C., Wu, H.: An anonymous multi-receiver encryption based on RSA. Int. J. Netw. Secur. **15**(4), 307–312 (2013)
15. He, D., Bu, J., Chan, S., Chen, C.: Handauth: efficient handover authentication with conditional privacy for wireless networks. IEEE Trans. Comput. **62**(3), 616–622 (2013)
16. Huang, X., Susilo, W., Mu, Y., Zhang, F.: Identity-based ring signcryption schemes: cryptographic primitives for preserving privacy and authenticity in the ubiquitous world. In: 19th International Conference on Advanced Information Networking and Applications (AINA 2005), Taipei, Taiwan, 28–30 March 2005, pp. 649–654 (2005)

17. Libert, B., Paterson, K.G., Quaglia, E.A.: Anonymous broadcast encryption: adaptive security and efficient constructions in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 206–224. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_13
18. Lu, R., Lin, X., Zhu, H., Ho, P., Shen, X.: ECPP: efficient conditional privacy preservation protocol for secure vehicular communications. INFOCOM 2008, pp. 1229–1237 (2008)
19. Mohassel, P.: A closer look at anonymity and robustness in encryption schemes. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 501–518. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_29
20. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_35
21. 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). https://doi.org/10.1007/3-540-45682-1_32
22. Zhang, J., Mao, J.: An improved anonymous multi-receiver identity-based encryption scheme. Int. J. Commun. Syst. **28**(4), 645–658 (2015)