




Access Control Encryption from Group Encryption

Xiuhua Wang, Harry W. H. Wong, and Sherman S. M. Chow^(✉) 

Department of Information Engineering, The Chinese University of Hong Kong,
Shatin, N.T., Hong Kong
xhwang@link.cuhk.edu.hk, {whwong, sherman}@ie.cuhk.edu.hk

Abstract. Access control encryption (ACE) enforces both read and write permissions. It kills off any unpermitted subliminal message channel via the help of a sanitizer who knows neither of the plaintext, its sender and receivers, nor the access control policy. This work aims to solve the open problem left by the seminal work of Damgård *et al.* (TCC 2016), namely, “to construct practically interesting ACE from noisy, post-quantum assumptions such as LWE.” We start with revisiting group encryption (GE), which allows anyone to encrypt to a certified group member, whom remains anonymous unless the opening authority decided to reveal him/her. We propose: 1) the notion of sanitizable GE (SGE), with specific changes for non-interactive proof, 2) the notion of traceable ACE (tACE), which helps damage control by tracing after-the-fact if some secret were leaked unluckily, 3) a generic construction of (t)ACE for equality policy (ACE-EP) from SGE, 4) a generic construction of ACE for general policy from ACE-EP, 5) a lattice-based instantiation of SGE, which comes with 6) a simple mechanism for checking that the randomness of ciphertexts can span the randomness space.

Keywords: Access control encryption · Group encryption · Lattice-based encryption · Learning with error · Post-quantum security · Chosen-ciphertext security · Sanitization · Traceability

1 Introduction

Sensitive data should not propagate arbitrarily without restriction; encryption techniques can enforce access control over the read but not write permissions. Meanwhile, enforcing control over who can write to whom is equally important. Consider a CEO who worries about leaking any strategic plan to arbitrary staff (*e.g.*, interns), say, via a malware-infected program s/he used for processing the related sensitive data. Note that digital signatures do not help since the recipient of the sensitive data can ignore any verification. Even worse, the data can be sent via a subliminal means, *e.g.*, embedding it as the randomness of a ciphertext.

Sherman S. M. Chow is supported by General Research Fund (Project Numbers: CUHK 14210217 and CUHK 14209918) from Research Grant Council, Hong Kong.

© Springer Nature Switzerland AG 2021

K. Sako and N. O. Tippenhauer (Eds.): ACNS 2021, LNCS 12726, pp. 417–441, 2021.

https://doi.org/10.1007/978-3-030-78372-3_16

It seems necessary to have a *sanitizer* to “monitor” the traffic for enforcing access control, especially over the write permissions. Ideally, the sanitization process should be “blindfolded,” *i.e.*, without the need to know who the sender is, who the recipient is, and what the access control policy is. Such an idea is formalized by Damgård, Haagh, and Orlandi [12] as access control encryption (ACE).

1.1 Designs from Two Ends of a Spectrum, and Open Problems

Sanitizing a ciphertext blindfolded is not an easy task. Damgård *et al.* [12] proposed two constructions. They first started with ACE for a single user (1-ACE) from standard (*e.g.*, decisional Diffie-Hellman) assumptions. To make it a fully-fledged ACE scheme, *i.e.*, supporting the *general policy* $P : \{0, 1\}^\ell \times \{0, 1\}^\ell \rightarrow \{0, 1\}$ which sender $ID_s \in \{0, 1\}^\ell$ can write to receiver $ID_r \in \{0, 1\}^\ell$ if and only if $P(ID_s, ID_r) = 1$, it runs 2^ℓ parallel copies of 1-ACE, making both the master public key and the ciphertext $O(2^\ell)$ -long. This is not only for hiding the intended reader but also for a uniform treatment in sanitization without knowing who the writer is. They also proposed a construction that offers $\text{poly}(\ell)$ efficiency, yet, it relies on a sanitizable variant of general-purpose functional encryption (FE) [6]. While FE for limited functionality (mostly inner-product) can be efficient, general-purpose FE is much more powerful and less efficient. Damgård *et al.* instantiated it with indistinguishability obfuscation.

Follow-up works mostly fall into two extremes: using practically-inefficient techniques to construct a regular ACE scheme, or practically-efficient techniques to construct an ACE scheme with limited functionality. Kim and Wu [18] built an ACE scheme from FE for randomized functionality (rFE) [2] and predicate-encryption (PE). Sanitization uses an FE key to create a PE ciphertext. Although the FE scheme can be instantiated by the LWE (learning-with-error) assumption, expressing the encryption algorithm of PE as a circuit is not that efficient. For the second paradigm, Fuchsbauer *et al.* [13] proposed a generic construction and a pairing-based construction for *equality policy* (ACE-EP), *i.e.*, the receiver is the sender. They also proposed to use many ACE-EP instances for interval membership policy, which is useful, albeit still not general.

A concurrent work by Wang and Chow [26] does not fall into the above two categories. In some sense, their generic construction can be considered as a “dual” of our proposed approach here. However, most of its building blocks, specifically structure-preserving signatures and broadcast encryption, are more “pairing-friendly,” meaning that lattice-based instantiations are still limited now.

One of the open problems left by Damgård *et al.* in their original work [12] is as follows: “*to construct practically interesting ACE from noisy, post-quantum assumptions such as LWE*” and they commented that “*the challenge here is that it always seems possible for a malicious sender to encrypt with just enough noise that any further manipulation by the sanitizer makes the decryption fail.*”

Tan *et al.* [25] use Gentry–Sahai–Waters fully-homomorphic encryption [14] to instantiate 1-ACE. Their 2^ℓ -extension still suffers from $O(2^\ell)$ ciphertext size. The scheme of Kim and Wu [18] still relies on a general-purpose rFE scheme for arbitrary functions (albeit it can be LWE-based). Both fail to close the above

open problem. Furthermore, both require the sanitizer to have a *private sanitization key*. Removing this requirement is also left as an open problem by Kim and Wu [18]. In this work, we ask ourselves a bigger question: “*Can we achieve the best of both worlds, i.e., using practically interesting lattice-based building blocks to build a general-policy ACE scheme, supporting keyless sanitization?*”

1.2 Viewing ACE Through the Lens of Group Encryption (GE)

Recurrent research activities in the cryptography community include identifying similarities and differences between primitives and connecting them if possible (e.g., [11]). Our starting point is group encryption (GE), introduced by Kiyas, Tsionis, and Yung [17]. GE is like public-key encryption (PKE). Anyone can encrypt to a certified group member. GE shares one basic feature of ACE, which is hiding who can decrypt a given ciphertext. In normal circumstances, this group member remains anonymous. When needed, an opening authority can reveal him/her. These features make GE an attractive primitive for privacy-preserving applications [17], e.g., filtering encrypted traffic or “oblivious retriever storage systems” [10]. There are a few existing GE schemes [3, 8, 21]. Notably, Libert *et al.* [20] proposed a lattice-based scheme (to be adapted by this paper).

However, GE falls short as ACE in many regards, notably the writing permission control: 1) Anyone can encrypt (no policy enforcement). 2) It does not feature a sanitization algorithm that randomizes a ciphertext (still without the need to know who can decrypt). It also falls short in terms of the reading permission control: 3) It encrypts to a single reader (not for the general policy).

The first two features can be added generically. Recall that an encryptor in GE first retrieves the public key of the intended receiver and its certificate issued by the group manager (GM) as a signature. The ciphertext contains a zero-knowledge proof of the certificate. By viewing the decryptor in GE also as the encryptor, we get ACE-EP. Sanitization, roughly, can be done by randomizing the ciphertext based on this hidden public key “accordingly” (which turns out to be tricky, see below). Indeed, these tricks are just rediscovery of the generic ACE-EP construction of Fuchsbauer *et al.* [13], who also mentioned, “A similar concept had previously been introduced in [15]¹.” We do not claim any novelty of extending GE with sanitizability [15]. What we deem important is that revisiting this conceptual connection allows us to borrow the existing results in lattice-based GE to solve our problems in ACE, forming the starting point of this work.

1.3 Sanitizable Group Encryption for Sanitization in ACE

We first describe what sanitizable group encryption (SGE) is and how its sanitizability is defined. Similar to how sanitizable PKE [13] (SPKE, see Sect. 3.3)

¹ Izabachene *et al.* [15] proposed mediated traceable anonymous encryption that pre-dates ACE. The mediator is essentially the sanitizer here. Like GE, it is a PKE scheme, and hence the missing feature is the enforcement of who can write. Their scheme design shares conceptual similarity with the 1-ACE scheme [12].

extends PKE, SGE extends GE with a *San* algorithm sanitizing ciphertexts. *San* essentially randomizes its input ciphertext without knowing the public key of its intended receipt. We expect *sanitizability*, which requires sanitized versions of an *adversarially generated* ciphertext and honest encryption of a random message remain computationally (instead of statistically [12]) indistinguishable. This definition is a variant of the subliminal-channel freeness of mediated traceable anonymous encryption [15] and similar to the no-write rule requirement of ACE.

With the working mechanisms of ACE-EP and SGE as outlined above, this paper starts with a generic construction of ACE-EP from any SGE. As a by-product, we obtain traceable ACE, with traceability analogous to the anonymity revocation of SGE, which traces the information (leakage) flow.

1.4 Meaningful Chosen-Ciphertext Security Under Sanitizability

It would be nice if there is a generic upgrade from any GE to SGE. However, the development of GE emphasizes security against chosen-ciphertext attacks (CCA) [3, 8, 20, 21]. A CCA-secure GE is unsanitizable by definition. For example, the lattice-based GE construction of Libert *et al.* [20] (the scheme we will modify) uses the transformation of Canetti, Halevi, and Katz (CHK) [7] to achieve CCA security. Encryption starts by picking a one-time signature key. The verification key is attached to the *label* of an underlying encryption that is secure against chosen-plaintext attacks (CPA-secure), such that no one can modify the label. The whole ciphertext is then signed by this one-time key, intuitively providing the integrity needed by CCA security. However, the label for tightly coupling the signature key with the ciphertext forms a convenient channel for a malicious writer that the sanitizer cannot easily randomize/sanitize.

This illustrates why most ACE literature did not consider CCA security. Notably, Badertscher, Matt, and Maurer [4] formulate a meaningful CCA security notion for ACE that protects the integrity of unsanitized ciphertexts. Consider a non-CCA-secure scheme and an attacker without any write permission. By capturing *only one* ciphertext before it reached the sanitizer, the attacker might be able to maul it to encrypt an arbitrary message and write to whomever the original creator is authorized to. Their CCA notion prevents such attacks.

Similarly, our SGE notion aims for such a flavor of CCA security. Following the generic GE construction of El Aïmani and Joye [3], we propose a generic SGE construction from a CPA-secure, key private, and sanitizable PKE scheme that still features a “compatible” public ciphertext validity check. Roughly, similar to the trick of Badertscher *et al.*, the ciphertext produced by our SGE scheme allows the sanitizer to easily check (for CCA security) and drop the “validity tag.” Any potential subliminal channel formed by this tag will be completely killed off, while the remaining parts can be sanitized.

1.5 Challenges in Sanitization

The property we stipulated above, namely, “*rerandomizability without knowing the underlying public key*,” turns out to be non-trivial to achieve in lattices.

Although rerandomization can be done by applying similar tricks of the existing ACE-EP construction via additive homomorphism, there is a mismatch in the threat models. In normal PKE usage, the encryptor has no intention to use imperfect randomness, while it is completely the opposite case for ACE, in which a malicious encryptor is motivated to establish a subliminal channel. To the best of our knowledge, no existing lattice-based PKE scheme is proven sanitizable.

To prevent the encryptor from cheating, *i.e.*, crafting a ciphertext c such that even an honest rerandomization of c will not result in a perfectly rerandomized ciphertext, we propose an efficient technique to detect such kind of adversarial behavior. At a high level, the vectors of randomness are required to be linearly independent to span the whole randomness space, so the sanitizer can use it to fully rerandomize a ciphertext. To filter out the randomness that fails to span the whole space, we leverage the lemma for rank relation of matrix multiplication for a ciphertext component formed by a multiplication between the public key and the randomness. This structure is not readily available, and we need to adapt an existing LWE-based scheme (see Footnote 4). The underlying sanitization technique requires a dedicated analysis, which may own independent interest.

1.6 Efficient ACE for General Policy from ACE for Equality Policy

Finally, we propose a generic upgrade extending an ACE for equality policy to a general-policy scheme for 2^ℓ users. Our crucial observation is to strategically manage the credentials, which does not require 2^ℓ -repetition of an underlying ACE scheme [12]. We still set the “legitimate decryptor” as the sender itself as in ACE for equality policy. Instead of granting the decryption key to the sender, we grant the decryption key to all the users that this particular sender can encrypt to. In this way, we obtain an ACE scheme for general policy, featuring constant-size ciphertexts, but at the cost of a decryption key that can be as long as the maximum number of senders a particular user can receive messages from.

1.7 Putting It Altogether

Our instantiation mostly uses the building blocks underlying the lattice-based GE scheme of Libert *et al.* [20], but with two major changes as explained above. We replace the underlying encryption scheme with a modified version of Regev’s LWE encryption [23], in which we build an efficient detection technique for confirming if its ciphertext “spans.” For CCA security, we use the Naor–Yung transformation [22, 24]. This leads to our lattice-based construction of SGE. With our generic transformation, we get a lattice-based ACE scheme for general policy, featuring keyless sanitization and constant-size ciphertexts. It provides a solution to two open problems: one from Damgård *et al.* [12] since it does not use *general-purpose* FE for circuits, and another from Kim and Wu [18] that asks for a general-policy ACE scheme with public sanitizer key (which rules out FE-based sanitization [12, 18]). It is also the second in the ACE literature that features CCA security. Like other LWE-based schemes, it is also post-quantum secure.

Organization. Section 2 recalls the definitions of ACE. Section 3 defines the SGE notion and presents our generic SGE construction. We upgrade it to ACE-EP and ultimately general-policy ACE in Sect. 4. Finally, Sect. 5 presents our SGE instantiation from lattices, which leads to our general-policy ACE.

2 Access Control Encryption with Keyless Sanitization

2.1 Definition

ACE is defined by the following probabilistic polynomial-time (PPT) algorithms.

- $\text{Setup}(1^\lambda, P) \rightarrow \text{pp}$: This algorithm takes the security parameter λ and a policy $P : \{0, 1\}^\ell \times \{0, 1\}^\ell \rightarrow \{0, 1\}$ as input. It outputs the public parameter pp , which includes the message space \mathcal{M} and two ciphertext spaces \mathcal{C} and \mathcal{C}' .
- $\text{MKGen}(\text{pp}) \rightarrow (\text{mpk}, \text{msk})$: This algorithm takes pp as input. It outputs a master public-secret key pair (mpk, msk) . We assume mpk is an implicit input for all algorithms below.
- $\text{EKGen}(\text{msk}, \text{ID}_i) \rightarrow \text{ek}_{\text{ID}_i}$: This algorithm takes the master secret key msk and an identity $\text{ID}_i \in \{0, 1\}^\ell$ as input. It outputs an encryption key ek_{ID_i} .
- $\text{DKGen}(\text{msk}, \text{ID}_j) \rightarrow \text{dk}_{\text{ID}_j}$: This algorithm takes the master secret key msk and an identity $\text{ID}_j \in \{0, 1\}^\ell$ as input. It outputs a decryption key dk_{ID_j} .
- $\text{Enc}(\text{ek}, M) \rightarrow c$: This algorithm takes an encryption key ek and a message M as input. It outputs a ciphertext $c \in \mathcal{C}$.
- $\text{San}(c) \rightarrow c'$: This algorithm transforms an incoming ciphertext $c \in \mathcal{C}$ into a sanitized ciphertext $c' \in \mathcal{C}' \cup \{\perp\}$. We only consider keyless sanitization.
- $\text{Dec}(\text{dk}, c') \rightarrow M$: The algorithm takes a decryption key dk and a ciphertext $c \in \mathcal{C}'$ as input. It outputs a message $M \in \mathcal{M} \cup \{\perp\}$.

For all $M \in \mathcal{M}$ and $\text{ID}_i, \text{ID}_j \in \{0, 1\}^\ell$ with $P(\text{ID}_i, \text{ID}_j) = 1$, an ACE scheme is correct if: $\Pr[\text{Dec}(\text{dk}_{\text{ID}_j}, \text{San}(\text{Enc}(\text{ek}_{\text{ID}_i}, M))) \neq M] \leq \text{negl}(\lambda)$ where $\text{pp} \leftarrow \text{Setup}(1^\lambda, P)$, $(\text{mpk}, \text{msk}) \leftarrow \text{MKGen}(\text{pp})$, $\text{ek}_{\text{ID}_i} \leftarrow \text{EKGen}(\text{msk}, \text{ID}_i)$, and $\text{dk}_{\text{ID}_j} \leftarrow \text{DKGen}(\text{msk}, \text{ID}_j)$. The probability space is over the coin flips of all the algorithms.

2.2 Security

ACE is for enforcing two access-control rules: the *no-read rule* and the *no-write rule*. Most existing works [12, 13, 18, 25] consider them under only CPA-based definitions, where the adversary is given access to the oracles for encryption, encryption-key generation, and decryption-key generation. Badertscher *et al.* [4] consider a CCA-based definition with a malicious insider who can maul an honestly-generated and unsanitized ACE ciphertext into a carrier for sending a message to a receiver that is forbidden by the policy otherwise. Even though ACE needs to assume an operational environment where ciphertexts must be routed through the sanitizer before reaching their final destination, it does not assume that no one can eavesdrop and maul them before reaching the sanitizer.

Instead of a typical decryption oracle, Badertscher *et al.* proposed an oracle that first sanitizes the ciphertext then decrypts it, *i.e.*, a sanitize-then-decrypt oracle. If an ACE scheme remains CCA-secure in this sense, no one can maul an unsanitized ciphertext. Note that the CCA protection does not extend to a sanitized ciphertext. Also, in practice, the sanitizer can sign on the sanitized ciphertexts and publish them on a public bulletin board for (anonymous) retrieval.

Let $\mathcal{ACE} = (\text{Setup}, \text{MKGen}, \text{EKGen}, \text{DKGen}, \text{Enc}, \text{San}, \text{Dec})$ be an ACE scheme for policy $P : \{0, 1\}^\ell \times \{0, 1\}^\ell \rightarrow \{0, 1\}$ over a message space \mathcal{M} . For a security parameter λ and a random bit b drawn from a fair coin flip, the general experiment $\text{Exp}_{\mathcal{ACE}, \mathcal{A}}(\lambda)$ for a PPT adversary \mathcal{A} starts with the challenger sampling $\text{pp} \leftarrow \text{Setup}(1^\lambda, P)$ and $(\text{mpk}, \text{msk}) \leftarrow \text{MKGen}(\text{pp})$. Then $\text{Exp}_{\mathcal{ACE}, \mathcal{A}}(\lambda)$ diverges into no-read rule experiment $\text{Exp}_{\mathcal{ACE}, \mathcal{A}}^{\text{NoRead}}(\lambda)$ or no-write experiment $\text{Exp}_{\mathcal{ACE}, \mathcal{A}}^{\text{NoWrite}}(\lambda)$ with a different challenge oracle and a different set of training oracles as below.

- $\mathcal{O}^{\text{Enc}}(M, \text{ID}_i) \rightarrow c$: On input $M \in \mathcal{M}$ and a sender identity $\text{ID}_i \in \{0, 1\}^\ell$, the encryption oracle outputs $c \leftarrow \text{Enc}(\text{EKGen}(\text{msk}, \text{ID}_i), M)$.
- $\mathcal{O}^{\text{SanEnc}}(M, \text{ID}_i) \rightarrow c'$: On input a message $M \in \mathcal{M}$ and a sender identity $\text{ID}_i \in \{0, 1\}^\ell$, it outputs $c' \leftarrow \text{San}(\text{Enc}(\text{EKGen}(\text{msk}, \text{ID}_i), M))$.
- $\mathcal{O}^{\text{EKGen}}(\text{ID}_i) \rightarrow \text{ek}_{\text{ID}_i}$: On input a sender identity $\text{ID}_i \in \{0, 1\}^\ell$, the encryption key generation oracle outputs $\text{ek}_{\text{ID}_i} \leftarrow \text{EKGen}(\text{msk}, \text{ID}_i)$.
- $\mathcal{O}^{\text{DKGen}}(\text{ID}_j) \rightarrow \text{dk}_{\text{ID}_j}$: On input a receiver identity $\text{ID}_j \in \{0, 1\}^\ell$, the decryption key generation oracle outputs $\text{dk}_{\text{ID}_j} \leftarrow \text{DKGen}(\text{msk}, \text{ID}_j)$.
- $\mathcal{O}^{\text{Dec}}(\text{ID}_j, c) \rightarrow M$: With a receiver identity $\text{ID}_j \in \{0, 1\}^\ell$ and an unsanitized ciphertext $c \in \mathcal{C}$, it outputs $M \leftarrow \text{Dec}(\text{DKGen}(\text{msk}, \text{ID}_j), \text{San}(c))$.
- $\mathcal{O}^{\text{NoRead}}((M_0, M_1), (\text{ID}_0, \text{ID}_1)) \rightarrow c_b$: This is the challenge oracle for the no-read experiment. On input a pair of messages $(M_0, M_1) \in \mathcal{M} \times \mathcal{M}$ and a pair of sender indices $(\text{ID}_0, \text{ID}_1) \in \{0, 1\}^\ell \times \{0, 1\}^\ell$, the challenger responds with $c_b \leftarrow \text{Enc}(\text{EKGen}(\text{msk}, \text{ID}_b), M_b)$.
- $\mathcal{O}^{\text{NoWrite}}(c, \text{ID}^*) \rightarrow c_b$: This is the challenge oracle for the no-write experiment. On input an unsanitized ciphertext $c \in \mathcal{C}$ and a sender identity $\text{ID}^* \in \{0, 1\}^\ell$, it sets $c_0^* \leftarrow c$. Then the challenger samples $M^* \leftarrow \mathcal{M}$, computes $c_1^* \leftarrow \text{Enc}(\text{EKGen}(\text{msk}, \text{ID}^*), M^*)$, and responds with $c_b \leftarrow \text{San}(c_b^*)$.

\mathcal{A} outputs a bit $b' \in \{0, 1\}$ as the output of the experiment at the end.

Definition 1 (No-Read Rule). *A wins the no-read game with \mathcal{O}^{Enc} , $\mathcal{O}^{\text{EKGen}}$, $\mathcal{O}^{\text{DKGen}}$, \mathcal{O}^{Dec} , and $\mathcal{O}^{\text{NoRead}}$ if $b' = b$, $|M_0| = |M_1|$, for all queries $\text{ID}_j \in \{0, 1\}^\ell$ that \mathcal{A} makes to the $\mathcal{O}^{\text{DKGen}}$, $P(\text{ID}_0, \text{ID}_j) = P(\text{ID}_1, \text{ID}_j) = 0$, and c_b has never been queried to \mathcal{O}^{Dec} . ACE satisfies the no-read rule if for all PPT \mathcal{A} , the advantage for \mathcal{A} to win the no-read game is $\text{Adv}^{\mathcal{A}} = \Pr[\mathcal{A} \text{ wins } \text{Exp}_{\mathcal{ACE}, \mathcal{A}}^{\text{NoRead}}(\lambda)] - \frac{1}{2} \leq \text{negl}(\lambda)$.*

The adversary can compromise the sanitizer under the above definition since the challenge ciphertext is not sanitized, and our ACE notion does not have any sanitizer key. The no-read rule ensures payload privacy, *i.e.*, no unintended receivers can learn anything about the message. It also guarantees (outsider) sender anonymity, which holds against any coalition of receivers that cannot decrypt the challenge ciphertext. This definition is weaker than requiring sender anonymity to hold even against an adversary who can decrypt the ciphertext.

Definition 2 (No-Write Rule). Given the oracles of $\mathcal{O}^{\text{SanEnc}}$, $\mathcal{O}^{\text{EKGen}}$, $\mathcal{O}^{\text{DKGen}}$, \mathcal{O}^{Dec} , and $\mathcal{O}^{\text{NoWrite}}$, \mathcal{A} wins the no-write game if $b' = b$, $\text{San}(c) \neq \perp$, and:

- The adversary \mathcal{A} makes at most one query² to the challenge oracle $\mathcal{O}^{\text{NoWrite}}$.
- For all identities $\text{ID}_i \in \{0, 1\}^\ell$ that \mathcal{A} submits to $\mathcal{O}^{\text{EKGen}}$ prior to its challenge and all identities $\text{ID}_j \in \{0, 1\}^\ell$ that \mathcal{A} submits to $\mathcal{O}^{\text{DKGen}}$, $\text{P}(\text{ID}_i, \text{ID}_j) = 0$.

We say that an ACE scheme satisfies the no-write rule if for all PPT adversary \mathcal{A} , the advantage of \mathcal{A} is $\text{Adv}^{\mathcal{A}} = \Pr[\mathcal{A} \text{ wins } \text{Exp}_{\text{ACE}, \mathcal{A}}^{\text{NoWrite}}(\lambda)] - \frac{1}{2} \leq \text{negl}(\lambda)$.

The no-write rule means that a sender can only encrypt to receivers permitted by the policy. Even an adversary can somehow embed in a ciphertext some subliminal information, it will be killed off after sanitization. Likewise, this property should hold even when multiple senders and receivers collude.

Sender Policy and Message Policy. The formulation of Kim and Wu [18] additionally considers “fine-grained sender policy” with the access control policy also governs the messages a sender can send. This policy is embedded in and authorized via the encryption key. They also suggested that an encryption key for multiple policies over the message can be supported in a straightforward manner by granting the sender multiple certified encryption keys.

In this paper, we consider a variant definition that the message policy can be *ad hoc*, i.e., the sender can create ciphertexts encrypting different messages satisfying different relations to any legitimate receiver. This flexibility has (seemingly inherent) implications on privacy and the no-write rule since the sanitizer needs to know about the relation and cannot “sanitize” the relation.

2.3 Traceable ACE

To obtain traceable ACE (tACE), we equip the traceability feature via two algorithms below, and with Enc algorithm now takes, besides the user encryption key ek, also an input of opening-authority public key tpk for the tracing feature.

- $\text{TKGen}(\text{pp}) \rightarrow (\text{tpk}, \text{tsk})$: This algorithm takes as input the public parameter pp and outputs the tracer public/secret key pair (tpk, tsk).
- $\text{Trace}(\text{tsk}, c') \rightarrow \text{ID}$: This algorithm takes the input of the tracer secret key tsk and a sanitized ciphertext $c' \in \mathcal{C}'$. It outputs the sender identity ID of c' .

Tracing the sender can be desirable in the context of ACE since we can locate which user has his/her machine compromised that tried to leak information. One may consider an alternative formulation that traces the receiver.

Here, we only consider a primitive form of tracing that recovers a user identity ID associated with a ciphertext [15]. Akin to traceable signatures [16], one could consider using a user-specific trapdoor for ID to check [1] whether ID is associated with a ciphertext [21] or to trace [9] all ciphertexts associated with a specific ID.

² A standard hybrid argument shows that security against an adversary that makes a single challenge query implies security against one that makes multiple such queries.

Traceability Correctness and Soundness. For all $M \in \mathcal{M}$ and $ID_i \in \{0, 1\}^\ell$, $pp \leftarrow \text{Setup}(1^\lambda, P)$, $(\text{mpk}, \text{msk}) \leftarrow \text{MKGen}(pp)$, $(\text{tpk}, \text{tsk}) \leftarrow \text{TKGen}(pp)$, and $c \leftarrow \text{Enc}(\text{EKGen}(\text{msk}, ID_i), M)$, *traceability correctness* means $\Pr[\text{Trace}(\text{tsk}, \text{San}(c)) \neq ID_i] \leq \text{negl}(\lambda)$. The probabilities are taken over the randomness of all algorithms.

A tACE scheme has *traceability soundness* if, for any PPT adversary \mathcal{A} who queries to $\mathcal{O}^{\text{EKGen}}$ and outputs a ciphertext c , the advantage for \mathcal{A} to win, defined to be $\Pr[ID \notin \mathcal{Q}^{\text{EKGen}} | \text{Trace}(\text{tsk}, \text{San}(c)) = ID]$, is negligible, where $\mathcal{Q}^{\text{EKGen}}$ denotes the set of queries to $\mathcal{O}^{\text{EKGen}}$, *i.e.*, c should not trace to an uncompromised user.

3 Sanitizable Group Encryption

3.1 Syntax of Sanitizable Group Encryption

A sanitizable group encryption scheme consists of the following algorithms.

- $\text{Setup}(1^\lambda) \rightarrow pp$: On input a security parameter λ , this probabilistic algorithm outputs the public parameter pp as an implicit input of what follows.
- $(G_r, \text{Sample}_{\mathcal{R}})$: On input of λ , G_r generates the key pair $(pk_{\mathcal{R}}, sk_{\mathcal{R}})$ of the relation \mathcal{R} concerning a message M one might want to prove about. $sk_{\mathcal{R}}$ can be empty if \mathcal{R} is publicly sampleable. We assume there is a PPT algorithm that can check if $(pk_{\mathcal{R}}, sk_{\mathcal{R}})$ is a valid output of G_r . On input of $(pk_{\mathcal{R}}, sk_{\mathcal{R}})$, $\text{Sample}_{\mathcal{R}}$ produces (x, M) where x is an instance and M is a witness for \mathcal{R} .
- $\text{KeyGen}_E(pp) \rightarrow (pk_E, sk_E)$: This algorithm outputs the key pair (pk_E, sk_E) of the entity E in the system. E can either be the group manager GM, the opening authority OA, or a group member u identified by ID.
- $\text{Join}(sk_{GM}, pk_{GM}, pk_{ID}) \rightarrow \text{cert}_{pk_{ID}}$: This algorithm outputs a certificate $\text{cert}_{pk_{ID}}$ on public key pk_{ID} and stores $(ID, pk_{ID}, \text{cert}_{pk_{ID}})$ in a directory db.
- $\text{Vfcert}(pk_{GM}, pk_{ID}, \text{cert}_{pk_{ID}})$: It verifies the validity of $\text{cert}_{pk_{ID}}$ for pk_{ID} .
- $\text{Enc}(pk_{GM}, pk_{OA}, pk_{ID}, \text{cert}_{pk_{ID}}, (pk_{\mathcal{R}}, x,) M) \rightarrow c$: On input the respective public key pk_{GM} , pk_{OA} , and pk_{ID} of GM, OA, and a group member certified by $\text{cert}_{pk_{ID}}$, and optionally a relation $pk_{\mathcal{R}}$ with a public value x , it returns a ciphertext c of the plaintext M , which $(x, M) \in \mathcal{R}$ is supposed to hold.
- $\text{Vf}(pk_{GM}, pk_{OA}, (pk_{\mathcal{R}}, x,) c) \rightarrow \{0, 1\}$: It outputs 1 if c is valid; 0 otherwise.
- $\text{San}(c) \rightarrow c'$: On input a valid ciphertext c , this algorithm outputs its sanitization c' (or a rejection symbol \perp).
- $\text{Dec}(sk_{ID}, c') \rightarrow M$: On input the private key sk_{ID} and a sanitized ciphertext c' , this algorithm decrypts c' and outputs the message M (or \perp).
- $\text{Open}(sk_{OA}, c') \rightarrow pk_{ID}$: On input the private key sk_{OA} of OA and a sanitized ciphertext c' , this algorithm recovers from c' the public key pk_{ID} .

Similar to the application scenario of ACE, we consider SGE ciphertexts to be (verified and) sanitized before reaching the final destination. This explains why our Dec and Open algorithms only work on sanitized ciphertexts. One might consider an alternative definition that they also work on unsanitized ciphertexts.

Our SGE formulation is kept as non-interactive as possible. Instead of having an explicit Prove algorithm/protocol in the prior GE formulation, the ciphertext produced by Enc contains a non-interactive proof. Existing schemes can indeed be formulated in this setting, some at the cost of using Fiat–Shamir heuristics. Also, the Join protocol is reduced to a pair of algorithms that the GM uses the algorithm Join to sign on a given public key for generating a certificate on it³, which the user can then run Vfcert to verify its validity. This helps to simplify the security definitions. We remark that the security definition for existing non-interactive GE schemes [8, 21] still separates the proof from the ciphertext.

Correctness. We require for an SGE scheme, the correctness game Corr defined in Fig. 1 returns 1 with overwhelming probability.

```

Experiment ExpCorr(λ)
-----
1 : pp ← Setup(1λ); (pkR, skR) ← Gr(1λ); (x, M) ← SampleR(pkR, skR);
2 : (pkGM, skGM) ← KeyGenGM(pp); (pkOA, skOA) ← KeyGenOA(pp);
3 : certpkID ← Join(skGM, pkGM, pkID);
4 : c ← Enc(pkGM, pkOA, pkID, certpkID, (pkR, x, ) M); c' ← San(c);
5 : cond1 ← Vf(pkGM, pkOA, (pkR, x, ) c);
6 : cond2 ← (pkID = Open(skOA, c'));
7 : cond3 ← (M = Dec(skID, c'));
8 : Return (cond1 = cond2 = cond3 = 1).
    
```

Fig. 1. Experiment for the correctness of SGE

3.2 Security Model of Sanitizable Group Encryption

In the following, we assume the adversary \mathcal{A} is stateful. By maintaining the state information state, \mathcal{A} becomes aware of at which stage it is.

Message Indistinguishability (IND). An SGE scheme meets the IND-CCA notion if the success probability of any PPT adversary \mathcal{A} to distinguish among encryptions of a chosen message and of a random message is at most negligibly better (in parameter λ) than $\frac{1}{2}$ in the experiment Ind in Fig. 2a, where the oracles are defined as below.

- $\mathcal{O}_{\mathcal{L}}^{\text{Join}^*}()$ is a stateful oracle that simulates executions for honest users who request to join the group. It maintains as state information an initially empty list \mathcal{L} . For its i -th invocation, the simulator executes $(pk_{ID_i}, sk_{ID_i}) \leftarrow$

³ The public key can be proven valid by an external mechanism (e.g., via any proof-of-possession mechanism over its secret key). Our final goal is to reduce ACE to GE. In ACE, each public key is generated by a trusted key generator, and hence it suffices.

- $\text{KeyGen}_u(\text{pp})$, sends it to the adversary, which responds with $\text{cert}_{\text{pk}_{\text{ID}_i}}$. The output $(\text{pk}_{\text{ID}_i}, \text{sk}_{\text{ID}_i}, \text{cert}_{\text{pk}_{\text{ID}_i}})$ of the user is stored in \mathcal{L} if the $\text{Join}(\cdot)$ -executing \mathcal{A} provides a valid certificate $\text{cert}_{\text{pk}_{\text{ID}_i}}$.
- $\mathcal{O}_{-(\text{ID}^*, c_b)}^{\text{Dec}}$ (ID, c_i) is a stateless decryption oracle. On input a ciphertext c_i , it runs $M' \leftarrow \text{Dec}(\text{sk}_{\text{ID}}, \text{San}(c_i))$ and returns M' if $(\text{ID}, c_i) \neq (\text{ID}^*, c_b)$.
 - $\mathcal{O}_b^{\text{RoR}}$ $(\text{pk}_{\text{ID}}, \text{pk}_{\mathcal{R}}, x, M)$ is a real-or-random challenge oracle that is only queried once. For a bit b , it samples a random plaintext M_0 uniformly from \mathcal{M} , and sets $M_1 = M$. It returns $c_b \leftarrow \text{Enc}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}}, (\text{pk}_{\mathcal{R}}, x,) M_b)$.

Experiment $\text{Exp}_{\mathcal{A}}^{\text{Ind}}(\lambda)$	Experiment $\text{Exp}_{\mathcal{A}}^{\text{Ano}}(\lambda)$
1 : $\text{pp} \leftarrow \text{Setup}(1^\lambda)$;	1 : $\text{pp} \leftarrow \text{Setup}(1^\lambda)$;
2 : $(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{state}) \leftarrow \mathcal{A}(\text{pp})$;	2 : $(\text{pk}_{\text{OA}}, \text{sk}_{\text{OA}}) \leftarrow \text{KeyGen}_{\text{OA}}(\text{pp})$;
3 : $(\text{pk}_{\text{ID}^*}, \text{cert}_{\text{pk}_{\text{ID}^*}}, \text{pk}_{\mathcal{R}}, x, M, \text{state})$ $\leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{L}}^{\text{Join}^*}(\cdot), \mathcal{O}_{-(\text{ID}^*, c_b)}^{\text{Dec}}(\cdot, \cdot)}(\text{state})$;	3 : $(\text{pk}_{\text{GM}}, \text{state}) \leftarrow \mathcal{A}(\text{pp}, \text{pk}_{\text{OA}})$;
4 : If $((\text{pk}_{\text{ID}^*}, \cdot, \text{cert}_{\text{pk}_{\text{ID}^*}}) \notin \mathcal{L}) \vee$ $((x, M) \notin \mathcal{R})$ then return 0;	4 : $(\{\text{pk}_{\text{ID}_d}, \text{cert}_{\text{pk}_{\text{ID}_d}}\}_{d \in \{0,1\}}, \text{pk}_{\mathcal{R}}, x, M,$ $\text{state}) \leftarrow \mathcal{A}^{\mathcal{O}_{-c_b}^{\text{Open}}(\text{sk}_{\text{OA}}, \cdot)}(\text{state})$;
5 : $b \leftarrow \{0, 1\}$; $c_b \leftarrow \mathcal{O}_b^{\text{RoR}}(\text{pk}_{\text{ID}^*}, \text{pk}_{\mathcal{R}}, x, M)$;	5 : If $(\text{Vfcert}(\text{pk}_{\text{GM}}, \text{pk}_{\text{ID}_0}, \text{cert}_{\text{pk}_{\text{ID}_0}}) = 0) \vee$ $(\text{Vfcert}(\text{pk}_{\text{GM}}, \text{pk}_{\text{ID}_1}, \text{cert}_{\text{pk}_{\text{ID}_1}}) = 0) \vee$ $((x, M) \notin \mathcal{R})$ then return 0;
6 : $b^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{L}}^{\text{Join}^*}(\cdot), \mathcal{O}_{-(\text{ID}^*, c_b)}^{\text{Dec}}(\cdot, \cdot)}(c_b, \text{state})$;	6 : $\text{ch} \leftarrow (\{\text{pk}_{\text{ID}_d}, \text{cert}_{\text{pk}_{\text{ID}_d}}\}_{d \in \{0,1\}}, \text{pk}_{\mathcal{R}}, x)$;
7 : If $(b = b^*)$ then return 1 else return 0.	7 : $b \leftarrow \{0, 1\}$; $c_b \leftarrow \mathcal{O}_b^{\text{Ano}}(\text{pk}_{\text{GM}}, \text{ch}, M)$;
(a)	8 : $b^* \leftarrow \mathcal{A}^{\mathcal{O}_{-c_b}^{\text{Open}}(\text{sk}_{\text{OA}}, \cdot)}(c_b, \text{state})$;
	9 : If $(b = b^*)$ then return 1 else return 0.
	(b)

Fig. 2. Experiments for (2a) IND-CCA and (2b) ANO-CCA notions of SGE

Anonymity. The formal definition of anonymity against chosen-ciphertext attacks (ANO-CCA) is as follows. The notion is met if the success probability of any PPT adversary \mathcal{A} is at most negligibly better than $\frac{1}{2}$. We introduce the following oracles and the game Ano in Fig. 2b.

- $\mathcal{O}_{-c_b}^{\text{Open}}(\text{sk}_{\text{OA}}, \cdot)$ returns $\text{Open}(\text{sk}_{\text{OA}}, \text{San}(c))$ on input of a ciphertext $c \neq c_b$,
- $\mathcal{O}_b^{\text{Ano}}(\text{pk}_{\text{GM}}, \{\text{pk}_{\text{ID}_d}, \text{cert}_{\text{pk}_{\text{ID}_d}}\}_{d \in \{0,1\}}, \text{pk}_{\mathcal{R}}, x, M)$ is a challenge oracle that is only queried once. It returns $c_b \leftarrow \text{Enc}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_{\text{ID}_b}, \text{cert}_{\text{pk}_b}, (\text{pk}_{\mathcal{R}}, x,) M)$.

Soundness. In a soundness attack, \mathcal{A} creates adaptively the intended group of receivers communicating with the genuine GM. \mathcal{A} is successful if it can output a ciphertext c and a chosen $\text{pk}_{\mathcal{R}}$ such that (1) c is not in the valid ciphertext space denoted by $\mathcal{C}^{\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{db}, \text{pk}_{\mathcal{R}}, x} = \{\text{Enc}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}}, (\text{pk}_{\mathcal{R}}, x,) M) :$

$\{(x, M) \in \mathcal{R}\} \wedge (\text{pk}_{\text{ID}} \in \text{db}) \wedge \text{Vfcert}(\text{pk}_{\text{GM}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}}) = 1\}$, and (2) opening c results in a public key that does not belong to any group member.

An SGE scheme is sound if, for any PPT \mathcal{A} , the experiment $\text{Exp}_{\mathcal{A}}^{\text{Sound}}(\lambda)$ outputs 1 with negligible probability. We introduce the following oracle and the game Sound in Fig. 3a.

- $\mathcal{O}_{\text{db}}^{\text{Join}}(\text{sk}_{\text{GM}}, \text{pk}_{\text{GM}}, \cdot)$ is a stateful oracle that simulates GM and maintains db storing each registered public key pk_{ID} along with its certificate $\text{cert}_{\text{pk}_{\text{ID}}}$.

Experiment $\text{Exp}_{\mathcal{A}}^{\text{Sound}}(\lambda)$	Experiment $\text{Exp}_{\mathcal{A}}^{\text{wSan}}(\lambda)$
1 : $\text{pp} \leftarrow \text{Setup}(1^\lambda)$; $(\text{pk}_{\text{GM}}, \text{sk}_{\text{GM}}) \leftarrow \text{KeyGen}_{\text{GM}}(\text{pp})$; $(\text{pk}_{\text{OA}}, \text{sk}_{\text{OA}}) \leftarrow \text{KeyGen}_{\text{OA}}(\text{pp})$;	1 : $\text{pp} \leftarrow \text{Setup}(1^\lambda)$; 2 : $(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{state}) \leftarrow \mathcal{A}(\text{pp})$; 3 : $(\text{pk}_{\text{ID}^*}, \text{cert}_{\text{pk}_{\text{ID}^*}}, \text{pk}_{\mathcal{R}}, x, c, \text{state})$ $\leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{L}}^{\text{Join}^*}(\cdot), \mathcal{O}_{\neg(\text{ID}^*, c)}^{\text{Dec}}(\cdot, \cdot)}(\text{state})$;
2 : $(\text{pk}_{\mathcal{R}}, x, c)$ $\leftarrow \mathcal{A}^{\mathcal{O}_{\text{db}}^{\text{Join}}(\text{sk}_{\text{GM}}, \text{pk}_{\text{GM}}, \cdot)}(\text{pp}, \text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{sk}_{\text{OA}})$;	4 : If $(\text{pk}_{\text{ID}^*}, \cdot, \text{cert}_{\text{pk}_{\text{ID}^*}}) \notin \mathcal{L}$ then return 0;
3 : If $\text{Vf}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, (\text{pk}_{\mathcal{R}}, x), c) = 0$ then return 0;	5 : $b \leftarrow \{0, 1\}$; $c_b \leftarrow \mathcal{O}^{\text{wSan}}(\text{pk}_{\mathcal{R}}, \text{sk}_{\mathcal{R}}, c)$;
4 : $\text{cond}_1 = (\text{Open}(\text{sk}_{\text{OA}}, \text{San}(c)) \notin \text{db})$;	6 : $b^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{L}}^{\text{Join}^*}(\cdot), \mathcal{O}_{\neg(\text{ID}^*, c)}^{\text{Dec}}(\cdot, \cdot)}(c_b, \text{state})$;
5 : $\text{cond}_2 = (c \notin \mathcal{C}^{\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{db}, \text{pk}_{\mathcal{R}}, x})$;	7 : If $(b = b^*)$ then return 1 else return 0.
6 : Return $(\text{cond}_1 \vee \text{cond}_2)$.	(b)
(a)	

Fig. 3. Experiments for the (3a) Soundness and (3b) Sanitizability of SGE

Sanitizability. Sanitizability requires that sanitization of two ciphertexts, one given by the adversary and the other randomly picked from the ciphertext space, cannot be distinguished as long as the adversary has no decryption key that decrypts any one of the ciphertexts. An SGE scheme is sanitizable if, for any PPT \mathcal{A} , the experiment $\text{Exp}_{\mathcal{A}}^{\text{wSan}}(\lambda)$ outputs 1 with negligible probability. With the two oracles $\mathcal{O}_{\mathcal{L}}^{\text{Join}^*}(\cdot)$ and $\mathcal{O}_{\neg(\text{ID}^*, c)}^{\text{Dec}}(\text{ID}, c_i)$ introduced in Fig. 2a, we introduce an additional oracle below and the game wSan in Fig. 3b.

- $\mathcal{O}^{\text{wSan}}(\text{pk}_{\mathcal{R}}, \text{sk}_{\mathcal{R}}, c) \rightarrow c_b$ is a real-or-random challenge oracle that is only queried once. It aborts if $\text{Vf}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_{\mathcal{R}}, x, c) = 0$. For a bit b , it first sets $c_b^* \leftarrow c$ and runs $(x, M^*) \leftarrow \text{Sample}_{\mathcal{R}}(\text{pk}_{\mathcal{R}}, \text{sk}_{\mathcal{R}})$ to sample M^* uniformly from \mathcal{M} under the constraint that $(x, M^*) \in \mathcal{R}$. It then computes $c_1^* \leftarrow \text{Enc}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}}, (\text{pk}_{\mathcal{R}}, x), M^*)$ and returns $c_b \leftarrow \text{San}(c_b^*)$.

3.3 Sanitizable Public-Key Encryption

Let $\mathcal{E} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{San}, \text{Dec})$ be a key-private sanitizable PKE scheme. We omit to repeat the standard definitions of correctness, key privacy, and CPA

security here [13]. Sanitizability requires any adversary generating two pairs of message and randomness $(M_0, r_0), (M_1, r_1)$ cannot distinguish the random bit b when given a sanitized ciphertext $\text{San}(\text{Enc}(\text{pk}, M_b; r_b))$, where $\text{Enc}(\text{pk}, M; r)$ refers to using r as its internal randomness.

3.4 Generic SGE Construction

We construct SGE by adapting the generic GE construction of El Aimani and Joye [3]. In a nutshell, the membership certificate is a signature. To create a GE ciphertext, the message and the public key are encrypted in two ciphertexts, and the validity of the certificate and well-formedness of the GE ciphertext are proven by non-interactive zero-knowledge (NIZK) proof. To achieve CCA security, they use tag-based public-key encryption with label and employ the CHK transform. A one-time signature verification key is put as the label, which is not sanitizable.

Instead of the CHK transform, we use the Naor–Yung technique [22, 24] to upgrade from CPA- to CCA-security. At a high level, two “component” ciphertexts, both encrypting the same message, are proven to be so via non-malleable NIZK. To simulate the decryption oracle, the reduction knows the decryption key for one of the two PKE instances, and hence decryption is trivial. The challenge ciphertext can be simulated via simulation soundness of NIZK, which ensures that the adversary has no advantage even if the simulated NIZK for the challenge query is for a wrong statement, and can easily be achieved via, *e.g.*, Fiat–Shamir heuristic. With this approach, we can achieve a CCA-security definition akin to that of ACE we defined in Sect. 2.2, following the prior definition [4]. Namely, the sanitizer first checks the well-formedness of the ciphertexts, drops the proof and the redundant ciphertext, and then performs rerandomization.

Let $\Sigma = (\text{Gen}, \text{Sign}, \text{Vf})$ be a signature scheme that is existentially unforgeable against chosen-message attacks (EUF-CMA). Let h be a collision-resistant hash function from the public-key space to the message space of \mathcal{E} . With an NIZK proof system, an SGE scheme is constructed as follows.

- $\text{Setup}(1^\lambda) \rightarrow \text{pp}$: This algorithm runs the setup algorithms (if any) for the building blocks and outputs all the public parameters as pp . Let \mathcal{R} be a relation with a key pair $(\text{pk}_{\mathcal{R}}, \text{sk}_{\mathcal{R}})$ for sampling pairs $(x, M) \in \mathcal{R}$.
- $\text{KeyGen}_{\text{GM}}(\text{pp}) \rightarrow (\text{pk}_{\text{GM}}, \text{sk}_{\text{GM}})$: This key generation algorithm for the group manager outputs $(\text{pk}_{\text{GM}}, \text{sk}_{\text{GM}})$, which is set to be $(\Sigma.\text{pk}, \Sigma.\text{sk}) \leftarrow \Sigma.\text{Gen}(1^\lambda)$.
- $\text{KeyGen}_{\text{OA}}(\text{pp}) \rightarrow (\text{pk}_{\text{OA}}, \text{sk}_{\text{OA}})$: It runs $\mathcal{E}.\text{KeyGen}(1^\lambda)$, which output the pairs $(\mathcal{E}.\text{pk}_{\text{OA}}, \mathcal{E}.\text{sk}_{\text{OA}})$ and $(\mathcal{E}.\text{pk}_{\text{OA}}^*, \mathcal{E}.\text{sk}_{\text{OA}}^*)$. It returns $((\mathcal{E}.\text{pk}_{\text{OA}}, \mathcal{E}.\text{pk}_{\text{OA}}^*), \mathcal{E}.\text{sk}_{\text{OA}})$.
- $\text{KeyGen}_{\text{u}}(\text{pp}) \rightarrow (\text{pk}_{\text{ID}}, \text{sk}_{\text{ID}})$: It runs $\mathcal{E}.\text{KeyGen}(1^\lambda)$ twice. Let the outputs be $(\mathcal{E}.\text{pk}_{\text{ID}}, \mathcal{E}.\text{sk}_{\text{ID}})$ and $(\mathcal{E}.\text{pk}_{\text{ID}}^*, \mathcal{E}.\text{sk}_{\text{ID}}^*)$. It outputs $((\mathcal{E}.\text{pk}_{\text{ID}}, \mathcal{E}.\text{pk}_{\text{ID}}^*), \mathcal{E}.\text{sk}_{\text{ID}})$.
- $\text{Join}(\text{sk}_{\text{GM}}, \text{pk}_{\text{GM}}, \text{pk}_{\text{ID}}) \rightarrow \text{cert}_{\text{pk}_{\text{ID}}}$: It runs $\text{cert}_{\text{pk}_{\text{ID}}} \leftarrow \Sigma.\text{Sign}(\text{sk}_{\text{GM}}, \text{pk}_{\text{ID}})$ and returns $\text{cert}_{\text{pk}_{\text{ID}}}$ to user ID. GM also stores $(\text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}})$ in db.
- $\text{Vfcert}(\text{pk}_{\text{GM}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}})$: It outputs $\Sigma.\text{Vf}(\text{pk}_{\text{GM}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}})$.
- $\text{Enc}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}}, (\text{pk}_{\mathcal{R}}, x,) M) \rightarrow c$: It firstly generates $c_M \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{ID}}, M)$, $c_M^* \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{ID}}^*, M)$, $c_{\text{OA}} \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{OA}}, h(\text{pk}_{\text{ID}}))$, and $c_{\text{OA}}^* \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{OA}}^*, h(\text{pk}_{\text{ID}}))$, and a non-malleable NIZK proof π for:

$$\begin{aligned}
(x, M) &\in \mathcal{R}, & \Sigma.\text{Vf}(\text{pk}_{\text{GM}}, \text{cert}_{\text{pk}_{\text{ID}}}, \text{pk}_{\text{ID}}) &= 1, \\
c_M &\leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{ID}}, M), & c_M^* &\leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{ID}}^*, M), \\
c_{\text{OA}} &\leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{OA}}, h(\text{pk}_{\text{ID}})), & c_{\text{OA}}^* &\leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{OA}}^*, h(\text{pk}_{\text{ID}}))
\end{aligned}$$

with statement $(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_{\mathcal{R}}, x)$ and witness $(M, \text{coins}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}})$, and coins denotes the randomness used in all invocations of $\mathcal{E}.\text{Enc}()$ above. It outputs $c = (c_M, c_M^*, c_{\text{OA}}, c_{\text{OA}}^*, \pi)$. Note that $(\text{pk}_{\mathcal{R}}, x)$ can be optional.

- $\text{Vf}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, (\text{pk}_{\mathcal{R}}, x), c) \rightarrow \{0, 1\}$: For $c = (c_M, c_M^*, c_{\text{OA}}, c_{\text{OA}}^*, \pi)$, it verifies the NIZK proof π , and outputs 1 if the proof is accepted, 0 otherwise.
- $\text{San}(c) \rightarrow c'$: It calls Vf on c and returns \perp if it is invalid. It then parses c into $(c_M, c_M^*, c_{\text{OA}}, c_{\text{OA}}^*)$, and outputs $c' = (\mathcal{E}.\text{San}(c_M), \mathcal{E}.\text{San}(c_{\text{OA}}))$.
- $\text{Dec}(\text{sk}_{\text{ID}}, c') \rightarrow M$: Parsing $c' = (c'_M, c'_{\text{OA}})$, it returns $M \leftarrow \mathcal{E}.\text{Dec}(\text{sk}_{\text{ID}}, c'_M)$.
- $\text{Open}(\text{sk}_{\text{OA}}, c') \rightarrow \text{pk}_{\text{ID}}$: It parses $c' = (c'_M, c'_{\text{OA}})$, runs $h^* \leftarrow \mathcal{E}.\text{Dec}(\text{sk}_{\text{OA}}, c'_{\text{OA}})$, then looks up at db and outputs the public key pk_{ID} such that $h(\text{pk}_{\text{ID}}) = h^*$.

The following theorems assert the security of our generic construction.

Theorem 1. *Our SGE scheme satisfies IND-CCA security if \mathcal{E} is IND-CCA-secure, Σ is EUF-CMA-secure, and NIZK is zero-knowledge proof-of-knowledge.*

Theorem 2. *Our SGE scheme satisfies ANO-CCA anonymity if \mathcal{E} is ANO-CCA-anonymous, Σ is EUF-CMA-secure, and NIZK is zero-knowledge proof-of-knowledge.*

Theorem 3. *Our SGE scheme satisfies soundness if Σ is EUF-CMA-secure and NIZK is zero-knowledge proof-of-knowledge.*

Theorem 4. *Our SGE scheme satisfies sanitizability if Σ is EUF-CMA-secure, \mathcal{E} is key-private and sanitizable, and NIZK is zero-knowledge proof-of-knowledge.*

The proofs for the first three mostly follow those for the generic GE construction of El Aimani and Joye [3]. The proof for the sanitizability mostly follows that for the no-write rule of the generic ACE construction of Fuchsbauer *et al.* [13]. Their details are deferred to the full version.

4 ACE from Sanitizable Group Encryption

4.1 Our Generic Construction of (t)ACE-EP

Using an SGE scheme, we can construct a (t)ACE scheme for the equality policy, *i.e.*, $\text{P}(\text{ID}_i, \text{ID}_j) = 1$ iff $\text{ID}_i = \text{ID}_j$ as follows. **Setup** of (t)ACE includes **Setup** of SGE. $(\mathcal{G}_r, \text{Sample}_{\mathcal{R}})$ is optional. ACE, by default, does not expect the message as a witness of some relation. However, incorporating so means that we can also enforce what kind of messages (even) a legitimate sender can send (*cf.*, [18]).

The key generator takes the roles of GM of SGE. It generates keys for users in the system by calling KeyGen_{u} and **Join** to generate the public/secret key pair

and create a certificate on the public key sequentially. To support sender tracing, the key generator stores $(ID, pk_{ID}, cert_{pk_{ID}})$ in a directory db .

For access control, any sender of $(t)ACE$ should be a group member in SGE. The group member keeps the certificate $cert_{pk_{ID}}$ on pk_{ID} returned from GM privately as the encryption key for proving the write permission and keeps sk_{ID} as the decryption key for exercising the read permission. During encryption, the sender calls the algorithm Enc of SGE to generate a ciphertext consists of an NIZK proof for the following relation: (1) the anonymous decryptor is a group member, (2) the payload message is encrypted under the public key of that decryptor, and for $tACE$ (3) the hash of the public key of the decryptor is encrypted in a ciphertext attached which is decryptable by the secret tracing key. The ciphertext is sent to the sanitizer. If the sanitizer accepts the proof embedded inside the ciphertext, it sanitizes the ciphertext and broadcasts it. Finally, the receiver calls the algorithm Dec of SGE to decrypt. The tracer can call the algorithm $Open$ of SGE to search for the corresponding ID if the need arises.

Let $\mathcal{GE} = (\text{Setup}, (G_r, \text{Sample}_{\mathcal{R}}), \text{KeyGen}_{GM}, \text{KeyGen}_{OA}, \text{KeyGen}_u, \text{Join}, \text{Vfcert}, \text{Enc}, \text{Vf}, \text{San}, \text{Dec}, \text{Open})$ be an SGE scheme. Our $(t)ACE$ scheme for equality policy, or $(t)ACE-EP$, is constructed as follows.

- $\text{Setup}(1^\lambda, P) \rightarrow pp_{ACE}$: With security parameter λ and the policy P , this algorithm runs $pp \leftarrow \mathcal{GE}.\text{Setup}(1^\lambda)$ and returns $pp_{ACE} = pp$.
- $\text{MKGen}(pp_{ACE}) \rightarrow (mpk, msk)$: It runs $(pk_{GM}, sk_{GM}) \leftarrow \mathcal{GE}.\text{KeyGen}_{GM}(pp)$ and returns master public/secret key tuple as $(mpk, msk) = (pk_{GM}, sk_{GM})$.
- $\text{TKGen}(pp_{ACE}) \rightarrow (tpk, tsk)$: It runs $(pk_{OA}, sk_{OA}) \leftarrow \mathcal{GE}.\text{KeyGen}_{OA}(pp)$ and returns $(tpk, tsk) = (pk_{OA}, sk_{OA})$.
- $\text{EKGen}(msk, ID_i) \rightarrow ek_{ID_i}$: With the input of ID_i , it first calls $(pk_{ID_i}, sk_{ID_i}) \leftarrow \mathcal{GE}.\text{KeyGen}_u(pp)$ then $cert_{pk_{ID_i}} \leftarrow \mathcal{GE}.\text{Join}(sk_{GM}, pk_{GM}, pk_{ID_i})$ (and stores in a directory db $(ID, pk_{ID_i}, cert_{pk_{ID_i}})$ for tracing). Finally, $ek_{ID_i} = (pk_{ID_i}, cert_{pk_{ID_i}})$.
- $\text{DKGen}(msk, ID_j) \rightarrow dk_{ID_j}$: For a receiver with identity ID_j , this algorithm returns sk_{ID_j} that has been generated by $\text{EKGen}(msk, ID_i)$. In practice, the key generator can use a pseudorandom function output of ID_j as the randomness used by $\mathcal{GE}.\text{KeyGen}_u(pp)$ within $\text{EKGen}(msk, ID_i)$. It outputs $dk_{ID_j} = sk_{ID_j}$.
- $\text{Enc}(ek_{ID_i}, tpk, M) \rightarrow c$: Using an encryption key $ek_{ID_i} = (pk_{ID_i}, cert_{pk_{ID_i}})$, possibly with a tracer public key $tpk = pk_{OA}$ (in $tACE$), this algorithm encrypts a message M via $c \leftarrow \mathcal{GE}.\text{Enc}(pk_{GM}, pk_{OA}, pk_{ID_i}, cert_{pk_{ID_i}}, M)$, or $c \leftarrow \mathcal{GE}.\text{Enc}(pk_{GM}, pk_{OA}, pk_{ID}, cert_{pk_{ID}}, pk_{\mathcal{R}}, x, M)$ if the message policy with respect to $(pk_{\mathcal{R}}, x)$ where $(x, M) \in \mathcal{R}$ is also enforced.
- $\text{San}(c) \rightarrow c'$: Output $c' \leftarrow \mathcal{GE}.\text{San}(c)$ if $\text{Vf}(pk_{GM}, pk_{OA}, c)$ returns true; \perp otherwise. If the policy also mandates $(x, M) \in \mathcal{R}$, San takes additional inputs of $(pk_{\mathcal{R}}, x)$ and runs $\text{Vf}(pk_{GM}, pk_{OA}, pk_{\mathcal{R}}, x, c)$ instead.
- $\text{Dec}(dk_{ID_j}, c') \rightarrow M$: On input a ciphertext c' and secret key $dk_{ID_j} = sk_{ID_j}$, this algorithm runs $M \leftarrow \mathcal{GE}.\text{Dec}(sk_{ID_j}, c')$, which either returns M or \perp .
- $\text{Trace}(tsk, c') \rightarrow ID$: On input a ciphertext c' and the tracing secret key tsk , it runs $pk_{ID} \leftarrow \mathcal{GE}.\text{Open}(sk_{OA}, c')$ and returns ID by looking up stored db .

The correctness of this (t)ACE-EP scheme directly follows from the correctness of the SGE scheme \mathcal{GE} . Since the SGE does not require any sanitizer key, the sanitization of our construction is done without any sanitizer key as well.

The proofs for correctness and security are mostly straightforward since sanitizable group encryption and (traceable) access control encryption are almost equivalent, modulo to the terminologies. They are deferred to the full version.

4.2 Extension to General Policy

Similar to the scheme of Fuchsbauer *et al.* [13], this (t)ACE-EP construction can also be extended to support range policies and a disjunction clause over them, with both the ciphertext size and decryption key size being $\text{poly}(\ell)$.

Beyond the above policies, we show that (t)ACE-EP can be extended to support general policy. Our intuition is as follows. For a receiver ID_j , the system generates the decryption key sk_{ID_i} of the (t)ACE-EP scheme for each ID_i where $P(ID_i, ID_j) = 1$, *i.e.*, receiver ID_j holds a set of decryption keys $\{dk_{ID_i}\}_{P(ID_i, ID_j)=1}$.

Let $(t)\mathcal{ACE}_{\text{eq}} = (\text{Setup}, \text{MKGen}, (\text{TKGen}), \text{EKGen}, \text{DKGen}, \text{Enc}, \text{San}, \text{Dec}, (\text{Trace}))$ be an (t)ACE-EP scheme for $P_{\text{eq}}(ID_i, ID_j) = 1$ iff $ID_i = ID_j$. We construct out (t)ACE scheme for general policy by changing the DKGen and Dec algorithms (all other algorithms remain unchanged).

- $\text{DKGen}(\text{msk}, ID_j) \rightarrow dk_{ID_j}$: With the input of msk and an identity ID_j , for any identities ID_i with predicate $P(ID_i, ID_j) = 1$, this algorithm computes $dk_{ID_i} \leftarrow \mathcal{ACE}_{\text{eq}}.\text{DKGen}(\text{msk}, ID_i)$ and returns the set $\{dk_{ID_i}\}_{P(ID_i, ID_j)=1}$ as the decryption key for receiver ID_j .
- $\text{Dec}(dk, c) \rightarrow M$: With the input of a ciphertext c' and decryption key $dk_{ID_j} = \{dk_{ID_i}\}_{P(ID_i, ID_j)=1}$, this algorithm decrypts c' using each dk_{ID_i} . It outputs the message M if one of the decryptions succeeds, \perp otherwise.

Our method needs not to replicate the whole cryptosystem for 2^ℓ copies [12]. The ciphertext size of our ACE scheme for the general policy is the same as the underlying (t)ACE-EP scheme, which is $O(1)$. The encryption key size remains the same as the underlying, *i.e.*, $O(1)$ too. The decryption key size is bounded by the maximum number of senders any user can receive messages from (denoted by s_{max}). Theoretically speaking, this can still be as long as 2^ℓ when a particular user can receive from all other users. In practice, we can always heuristically assign a special identity to this kind of users to reduce the key size. Table 1 compares the size of the parameters of interests for our general-policy ACE instantiation (from ACE-EP) and the existing one (from 1-ACE [12, §3]).

Note that this scheme only achieves sender anonymity against outsiders. In other words, a legitimate decryptor can learn information about who the sender is. This matches the level of Kim–Wu ACE [18]. As argued [18], it suffices for all application scenarios originally envisioned [12].

Table 1. Comparison of key size and ciphertext (Ctxt.) sizes

Instantiations from different building blocks	Enc. Key	Dec. Key	Ctxt.	San. Key
General-policy ACE from 1-ACE [12, §3]	$O(2^\ell)$	$O(1)$	$O(2^\ell)$	$O(2^\ell)$
General-policy ACE from ACE-EP (This work)	$O(1)$	$O(s_{\max})$	$O(1)$	Nil

5 Lattice-Based Access Control Encryption

To achieve our final goal, we adapt Libert *et al.* [20]’s lattice-based GE scheme, which is not sanitizable due to a non-randomizable tag. We thus disassemble it and replace its encryption scheme with a new lattice-based SPKE scheme.

5.1 Lattice Background

We quickly review some preliminaries in lattice-based cryptography. We cite a special version of the leftover hash lemma [23], which argues the indistinguishability from a uniform distribution. For our scheme, we consider \mathbb{Z}_q^n as the Abelian group \mathbb{G} and $m = 2n \log q$ be the maximum number of samples to be summed up.

Theorem 5 ([23]). *Let \mathbb{G} be some finite Abelian group and let k be some integer. For any m elements $g_1, \dots, g_m \in \mathbb{G}$, consider the statistical distance between the uniform distribution on \mathbb{G} and the distribution given by the sum of a random subset of g_1, \dots, g_m . The expectation of this statistical distance over a uniform choice of $g_1, \dots, g_m \in \mathbb{G}$ is at most $\sqrt{|\mathbb{G}|/2^m}$. In particular, the probability that this statistical distance is more than $\sqrt[4]{|\mathbb{G}|/2^m}$ is at most $\sqrt[4]{|\mathbb{G}|/2^m}$.*

The decisional-LWE problem asks to distinguish samples from a perturbed linear system and random elements from the uniform distribution.

Definition 3 (Decisional Learning with Error [23]). *Let \mathbb{Z}_q be the ring of integers modulo a positive integer q , and \mathbb{Z}_q^n be the set of n -vectors over \mathbb{Z}_q . Given a probability distribution χ over \mathbb{Z} , a positive integer n , and a positive integer q of size dependent on n , the goal of learning with error $\text{LWE}_{q,\chi}$ is to distinguish between the sample (\mathbf{a}, \mathbf{b}) from distribution $A_{\mathbf{s},\chi}$, defined as $\mathbf{b} = \mathbf{a}\mathbf{s} + \mathbf{e}$ with $\mathbf{e} \leftarrow \chi$ for some uniform secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and (\mathbf{a}, \mathbf{u}) is sampled (via oracle accesses) from a uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ for randomly sampled \mathbf{a} .*

Definition 4 (Noise Sample Space [23]). *For $\alpha \in (0, 1)$ and a prime q , let Ψ_α denote the distribution over \mathbb{Z}_q of the random variable $\lfloor qX \rfloor \bmod q$, where X is a normal random variable with mean 0 and standard deviation $\alpha/2\sqrt{\pi}$.*

Specifically, if the noise added to the perturbed linear system being an amplified-then-quantized Gaussian noise (as mentioned in the above definition of noise sample space), one can apply the following theorem to reduce the hardness of LWE to existing lattice problems, which helps us to decide the parameters.

Theorem 6 (Regev’s Reduction [23]). *For $\alpha \in (0, 1)$ and prime q , if there exists an efficient, possibly quantum, algorithm for deciding the $(\mathbb{Z}_q, n, \Psi_\alpha)$ -LWE problem for $\alpha q > 2\sqrt{n}$, then there is an efficient quantum algorithm for approximating the shortest independent vector problem and the gap shortest vector problem, to within $\tilde{O}(n/\alpha)$ factors in ℓ_2 norm, in the worst case.*

5.2 Lattice-Based Sanitizable Encryption

We start with Regev’s encryption scheme based on the LWE problem [23]. It features ciphertext indistinguishability, meaning that an honestly generated ciphertext is indistinguishable from a random element in the ciphertext space, which implies key privacy. Sanitization relies on encryption of 0 as *randomizers*. When the randomizers form a basis for spanning the randomness space of Regev’s scheme, the ciphertext can be rerandomized by adding a random subset-sum of the randomizers. This requires additive homomorphism. However, the noise accumulates after homomorphic evaluations. We thus change the parameters of the scheme such that the evaluation correctness (decryption correctness of an evaluated ciphertext) holds for a bounded number of additions. Namely, we scale up the modulo size to increase the noise tolerance of decryption.

This sanitization technique assumes an honest encryptor to prepare linearly independent randomness components, which mismatches the threat model that randomness is adversarially picked. We address this by a specific structure of the randomizer that allows us to check the rank of the randomizer, which implies the rank of the underlying randomness used by the randomizer.

Denote matrix and vector by bold capital and small letter, respectively; our SPKE scheme (Setup, KeyGen, Enc, Dec, San) is as follows.

- Setup(1^λ) \rightarrow pp: Set $n = O(\lambda)$, prime $q = \tilde{O}(n) > 16m(m + 1)$, $m = 2n \log q$, $k = \text{poly}(n)$. The probability distribution χ is taken to be Ψ_α , with $\alpha = 1/(\sqrt{m}\omega(\sqrt{\log n}))$. Sample $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times m}$ and compute $r_{\bar{\mathbf{A}}} = \text{Rank}(\bar{\mathbf{A}})$. Output pp = $(q, n, m, k, \chi, \bar{\mathbf{A}}, r_{\bar{\mathbf{A}}})$.
- KeyGen(pp) \rightarrow (pk, sk): Sample $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow \chi^m$. Compute $\mathbf{b} = \bar{\mathbf{A}}^\top \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$. Output (pk, sk) = (\mathbf{b}, \mathbf{s}) .
- Enc(pk, \mathbf{m}) \rightarrow \mathbf{c} : Given $\mathbf{m} \in \{0, 1\}^k$, sample $\mathbf{R}_m \leftarrow \{0, 1\}^{k \times m}$ and linearly independent $\mathbf{R}_r \leftarrow \{0, 1\}^{m \times m}$. Set $\mathbf{c}_m = (\mathbf{R}_m \bar{\mathbf{A}}^\top, \mathbf{R}_m \mathbf{b} + \mathbf{m} \cdot \lfloor q/2 \rfloor) \in \mathbb{Z}_q^{(k+1) \times n}$ and $\mathbf{c}_r = (\mathbf{R}_r \bar{\mathbf{A}}^\top, \mathbf{R}_r \mathbf{b}) \in \mathbb{Z}_q^{(m+1) \times n}$. Output $\mathbf{c} = (\mathbf{c}_m, \mathbf{c}_r)$.
- Dec(sk, \mathbf{c}') \rightarrow \mathbf{m} : We suppose \mathbf{c}' has been sanitized by San. Parse $\mathbf{c}' = (\mathbf{c}_0, \mathbf{c}_1)$. Set $\mathbf{m}' = \mathbf{c}_1 - \mathbf{c}_0 \mathbf{s} \bmod q$. For each entry i of \mathbf{m}' , say \mathbf{m}'_i , set $\mathbf{m}_i = 0$ if \mathbf{m}'_i is closer to 0 than $\lfloor q/2 \rfloor$. Otherwise, set $\mathbf{m}_i = 1$. Output \mathbf{m} .
- San(\mathbf{c}) \rightarrow \mathbf{c}' : Parse $\mathbf{c} = ((\mathbf{c}_{m,0}, \mathbf{c}_{m,1}), (\mathbf{c}_{r,0}, \mathbf{c}_{r,1}))$. Check if $\text{Rank}(\mathbf{c}_{r,0}) = r_{\bar{\mathbf{A}}}$, output \perp if it is not. Otherwise, sample $\mathbf{R} \in \{0, \pm 1\}^{k \times m}$ and output $\mathbf{c}' = (\mathbf{c}_{m,0} + \mathbf{R} \mathbf{c}_{r,0}, \mathbf{c}_{m,1} + \mathbf{R} \mathbf{c}_{r,1}) \in \mathbb{Z}_q^{(k+1) \times n}$.

Correctness. The decryption correctness of sanitized ciphertext largely follows the original scheme [23] and by scaling up the modulo q by $m+1$ times to preserve

the correctness after the additions done by San. Decryption outputs

$$\begin{aligned}
 \mathbf{c}_1 - \mathbf{c}_0\mathbf{s} &= (\mathbf{c}_{\mathbf{m},1} + \mathbf{R}\mathbf{c}_{r,1}) - (\mathbf{c}_{\mathbf{m},0} + \mathbf{R}\mathbf{c}_{r,0})\mathbf{s} \\
 &= (\mathbf{R}_m\mathbf{b} + \mathbf{m} \cdot \lfloor q/2 \rfloor + \mathbf{R}\mathbf{R}_r\mathbf{b}) - (\mathbf{R}_m\bar{\mathbf{A}}^\top + \mathbf{R}\mathbf{R}_r\bar{\mathbf{A}}^\top)\mathbf{s} \\
 &= (\mathbf{R}_m + \mathbf{R}\mathbf{R}_r)(\mathbf{b} - \bar{\mathbf{A}}^\top\mathbf{s}) + \mathbf{m} \cdot \lfloor q/2 \rfloor \\
 &= (\mathbf{R}_m + \mathbf{R}\mathbf{R}_r)\mathbf{e} + \mathbf{m} \cdot \lfloor q/2 \rfloor.
 \end{aligned}$$

Since \mathbf{R}_m , \mathbf{R} and \mathbf{R}_r are binary matrices, the absolute value of entries in $\mathbf{R}'\mathbf{e}$ for $\mathbf{R}' = \mathbf{R}_m + \mathbf{R}\mathbf{R}_r$ is upper bounded by $(m + 1) \sum_{i=1}^m e_i$, where e_i is the i -th entry of vector \mathbf{e} . To recover \mathbf{m} , we need to show that the entry of $\mathbf{R}'\mathbf{e}$ is upper bounded by $q/16$; in other words, $\sum_{i=1}^m e_i$ is upper bounded by $q/16(m + 1)$.

By the definition of Ψ_α , $e_i = \lfloor qx_i \rfloor \bmod q$, where x_i 's are independent normal variables with mean 0 and variances α^2 . Note that $\sum_{i=1}^m e_i$ is at most $m/2 \leq q/32$ away from $\sum_{i=1}^m qx_i \bmod q$. It suffices to show that $|\sum_{i=1}^m qx_i \bmod q| \leq q/16(m + 1)$ with high probability. Since x_i 's are independent, $|\sum_{i=1}^m x_i \bmod q|$ is distributed as a normal variable with mean 0 and standard deviation $\sqrt{m} \cdot \alpha \leq 1/\omega(\sqrt{\log n})$. Thus, by the tail inequality on normal variables, the probability that the absolute value of the entry in $\mathbf{R}'\mathbf{e}$ greater than $q/16$ is negligible.

Security. The proof mostly follows the existing [23]. We sketch its two hybrids.

The first hybrid game shows that a “well-formed” public key is indistinguishable from a random element based on the decisional-LWE assumption (Definition 3). By this assumption, replacing the component of the public key \mathbf{b} (the LWE instance) with a random element \mathbf{u} is indistinguishable.

The second hybrid game shows that a ciphertext is statistically indistinguishable from a random element by the leftover hash lemma (Theorem 5). Since the ciphertext $(\mathbf{r}^\top \bar{\mathbf{A}}^\top, \mathbf{r}^\top \mathbf{u})$ is a random subset-sum of $(\bar{\mathbf{A}}^\top, \mathbf{u})$ as $\mathbf{r} \in \{0, 1\}^m$, it is statistically indistinguishable from a uniform distribution. It completes our argument for indistinguishability from random.

Sanitizability. Although checking for uniformly-sampled randomizer is difficult, one can check whether the randomizers are linearly independent (for randomness space being a vector) instead so that the randomizers always span the whole randomness space. Recall that one of the components of the ciphertext is $\mathbf{c}_{r,0} = \mathbf{R}_r \bar{\mathbf{A}}^\top$, we leverage the following lemma for the rank of matrix multiplication in linear algebra to check whether the randomness \mathbf{R}_r is linearly independent or not.⁴ Given m -dimensional square matrix \mathbf{R} and $(n \times m)$ -dimensional matrix \mathbf{A} , if \mathbf{R} is full rank, $\text{Rank}(\mathbf{R}\mathbf{A}^\top) = \text{Rank}(\mathbf{A}^\top)$. Hence, if

⁴ We remark that the dual version of our sanitizable encryption scheme has no such efficient machinery (which explains our choices). Although randomly adding randomizer to payload-part can still rerandomize its randomness, “linearly independence” is not well-defined (the randomness space is \mathbb{Z}_q^n but it needs $n \log q$ linearly independent vectors to rerandomize, where linear independence means none of the samples is a *subset-sum* of the other samples), and checking seems to have the same complexity as the NP-complete subset-sum problem. Also, the corresponding ciphertext component to be checked is perturbed by noise, which ruins the structure.

$\text{Rank}(\mathbf{c}_{r,0}) = \text{Rank}(\bar{\mathbf{A}}^\top)$, it implies that \mathbf{R}_r is full rank (linearly independent), and the corresponding randomizers can be used to span the whole randomness space. For example, sanitization changes the randomness from \mathbf{R}_m to $\mathbf{R}_m + \mathbf{R}\mathbf{R}_r$.

For the indistinguishability of sanitized ciphertexts from a random element in the space of sanitized ciphertexts because of the changes in randomness space (from binary to integer), an appropriate version of the leftover hash lemma can be used. In its proof [23, Sect. 5], selecting a subset-sum or an integer combination of the basis vector does not affect the argument.

Verifiable Encryption. We show the proof system [20] for the ciphertext’s well-formedness of our SPKE scheme. The ciphertext is transformed into a linear relation in the form of $\mathbf{P} \cdot \mathbf{x} = \mathbf{v}$, where witness \mathbf{x} has the same Hamming weight for 0 and 1. Consider the payload-part of a single-bit encryption $\mathbf{c}_m = (\mathbf{r}_m \bar{\mathbf{A}}^\top, \mathbf{r}_m \mathbf{b} + \mathbf{m} \cdot \lfloor q/2 \rfloor)$ as an example. One can trivially extend to the multi-bit version as in our scheme since the matrix witness can be formulated as a single vector by concatenating the columns of the matrix one-by-one. Also, the randomizer \mathbf{c}_r is an encryption of 0. By matrix arrangement (joining two relations via an “AND” relation), the well-formedness of the whole ciphertext is guaranteed. Consider the relation:

$$\mathcal{R} = \{((\mathbf{c}_{m,0}, \mathbf{c}_{m,1}, \bar{\mathbf{A}}), (\mathbf{r}_m, \mathbf{b})) : \mathbf{c}_{m,0} = \mathbf{r}_m \bar{\mathbf{A}}^\top \wedge \mathbf{c}_{m,1} = \mathbf{r}_m \mathbf{b} + \mathbf{m} \cdot \lfloor q/2 \rfloor \wedge \mathbf{r}_m \in \{0, 1\}^m \wedge \mathbf{m} \in \{0, 1\}\}.$$

With the techniques of Libert *et al.* [20], the quadratic relation $\mathbf{c}_{m,1} = \mathbf{r}_m \mathbf{b} + \mathbf{m} \cdot \lfloor q/2 \rfloor$ boils down to $(\mathbf{c}_{m,1} = \mathbf{qz} + \mathbf{m} \cdot \lfloor q/2 \rfloor) \wedge (\mathbf{z} = \text{expand}^\otimes(\mathbf{r}_m, \text{vdec}_{m,q}(\mathbf{b})))$, where expand^\otimes is a function that exhausts all possibilities of two binary vectors, one obtained from the binary decomposition function vdec , and $\mathbf{z} \in \{0, 1\}^{4m^2 \log q}$ such that $\mathbf{qz} = \mathbf{r}_m \mathbf{b}$. Hence, we have

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{0}^{n \times m} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{q} & \lfloor q/2 \rfloor & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{r}_m \\ \mathbf{r}_m^c \\ \mathbf{z} \\ \mathbf{m} \\ \mathbf{m}^c \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{c}_{m,0} \\ \mathbf{c}_{m,1} \end{bmatrix},$$

with $\mathbf{z} = \text{expand}^\otimes(\mathbf{r}_m, \text{vdec}_{m,q}(\mathbf{b}))$ as an additional part to be verified. Within \mathbf{x} , \mathbf{r}_m^c is a padding (complement) to make the concatenation of \mathbf{r}_m and \mathbf{r}_m^c having the same Hamming weight for 0 and 1. The term \mathbf{m}^c is for similar usage.

5.3 Lattice-Based Sanitizable Group Encryption

With our generic ACE construction from any SGE, it remains to instantiate our generic SGE construction. We mostly adopt the building blocks of Libert *et al.* [20], *i.e.*, the signature scheme $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Vf})$ based on the short-integer-solution assumption they used [5, 19] (its detailed description [20, Appendix A.1] is not repeated here) and their techniques in

zero-knowledge arguments for matrix-vector relations, but with the encryption scheme replaced by our SPKE scheme $\mathcal{E} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{San}, \text{Dec})$.

We omitted $(G_r, \text{Sample}_{\mathcal{R}})$ and the related inputs and steps of Enc below since they are independent of the cryptosystem. The encryptor can add the NIZK proof for the desired relation, *e.g.*, for inhomogeneous SIS [20], if needed.

– $\text{Setup}(1^\lambda) \rightarrow \text{pp}$:

1. Run $\mathcal{E}.\text{pp} \leftarrow \mathcal{E}.\text{Setup}(1^\lambda)$ and $\Sigma.\text{pp} \leftarrow \Sigma.\text{Setup}(1^\lambda)$.
2. Pick two random matrices $\mathbf{F}, \mathbf{F}^* \leftarrow \mathbb{Z}_q^{n \times m \log q}$, which will be used to hash a user public key from $\mathbb{Z}_q^{m \log q}$ to \mathbb{Z}_q^n .
3. Set matrix $\mathbf{H}_{n,q} \in \mathbb{Z}_q^{n \times \bar{m}}$ such that for any $\mathbf{x} \in \mathbb{Z}_q^n$, $\mathbf{x} = \mathbf{H}_{n,q} \cdot \text{vdec}_{n,q}(\mathbf{x})$, and $\text{vdec}_{n,q} : \mathbb{Z}_q^n \rightarrow \{0, 1\}^{\bar{m}}$ is an injective vector-decomposition function [20, Sect. 3.1].

Output $\text{pp} = (\mathcal{E}.\text{pp}, \Sigma.\text{pp}, \mathbf{F}, \mathbf{F}^*)$.

- $\text{KeyGen}_{\text{GM}}(\text{pp}) \rightarrow (\text{pk}_{\text{GM}}, \text{sk}_{\text{GM}})$: Output $(\text{pk}, \text{sk}) \leftarrow \Sigma.\text{KeyGen}(1^\lambda)$.
- $\text{KeyGen}_{\text{OA}}(\text{pp}) \rightarrow (\text{pk}_{\text{OA}}, \text{sk}_{\text{OA}})$:
1. Run $\mathcal{E}.\text{KeyGen}(1^\lambda)$ for twice to get $(\mathcal{E}.\text{pk}_{\text{OA}}, \mathcal{E}.\text{sk}_{\text{OA}})$ and $(\mathcal{E}.\text{pk}_{\text{OA}}^*, \mathcal{E}.\text{sk}_{\text{OA}}^*)$.
 2. Output $((\mathcal{E}.\text{pk}_{\text{OA}}, \mathcal{E}.\text{pk}_{\text{OA}}^*), \mathcal{E}.\text{sk}_{\text{OA}})$.
- $\text{KeyGen}_{\text{u}}(\text{pp}) \rightarrow (\text{pk}_{\text{ID}}, \text{sk}_{\text{ID}})$:
1. Run $\mathcal{E}.\text{KeyGen}(1^\lambda)$ for twice to get $(\mathcal{E}.\text{pk}_{\text{ID}}, \mathcal{E}.\text{sk}_{\text{ID}})$ and $(\mathcal{E}.\text{pk}_{\text{ID}}^*, \mathcal{E}.\text{sk}_{\text{ID}}^*)$.
 2. Output $((\mathcal{E}.\text{pk}_{\text{ID}}, \mathcal{E}.\text{pk}_{\text{ID}}^*), \mathcal{E}.\text{sk}_{\text{ID}})$.
- $\text{Join}(\text{sk}_{\text{GM}}, \text{pk}_{\text{GM}}, \text{pk}_{\text{ID}}) \rightarrow \text{cert}_{\text{pk}_{\text{ID}}}$:
1. Parse pk_{ID} as $(\mathcal{E}.\text{pk}_{\text{ID}}, \mathcal{E}.\text{pk}_{\text{ID}}^*)$ and compute a single hash value of them: $\mathbf{h}_{\text{ID}} = \mathbf{F} \cdot \text{vdec}_{m,q}(\mathcal{E}.\text{pk}_{\text{ID}}) + \mathbf{F}^* \cdot \text{vdec}_{m,q}(\mathcal{E}.\text{pk}_{\text{ID}}^*) \in \mathbb{Z}_q^n$.
 2. Output $\text{cert}_{\text{pk}_{\text{ID}}} \leftarrow \Sigma.\text{Sign}(\text{sk}_{\text{GM}}, \mathbf{h}_{\text{ID}})$ and store $(\text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}})$ in db.
- $\text{Vfcert}(\text{pk}_{\text{GM}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}})$: Output $\Sigma.\text{Vf}(\text{pk}_{\text{GM}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}})$.
- $\text{Enc}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_{\text{ID}}, \text{cert}_{\text{pk}_{\text{ID}}}, (\text{pk}_{\mathcal{R}}, x,) \mathbf{m}) \rightarrow \mathbf{c}$: To encrypt $\mathbf{m} \in \{0, 1\}^m$,
1. Parse pk_{OA} as $(\mathcal{E}.\text{pk}_{\text{OA}}, \mathcal{E}.\text{pk}_{\text{OA}}^*) = (\mathbf{b}_{\text{OA}}, \mathbf{b}_{\text{OA}}^*)$.
 2. Parse pk_{ID} as $(\mathcal{E}.\text{pk}_{\text{ID}}, \mathcal{E}.\text{pk}_{\text{ID}}^*) = (\mathbf{b}_{\text{ID}}, \mathbf{b}_{\text{ID}}^*)$.
 3. Compute the hash value $\mathbf{h}_{\text{ID}} = \mathbf{F} \cdot \text{vdec}_{m,q}(\mathcal{E}.\text{pk}_{\text{ID}}) + \mathbf{F}^* \cdot \text{vdec}_{m,q}(\mathcal{E}.\text{pk}_{\text{ID}}^*)$.
 4. Compute the ciphertexts $\mathbf{c}_{\mathbf{m}} \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{ID}}, \mathbf{m})$, $\mathbf{c}_{\mathbf{m}}^* \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{ID}}^*, \mathbf{m})$, $\mathbf{c}_{\text{OA}} \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{OA}}, \text{vdec}_{n,q}(\mathbf{h}_{\text{ID}}))$, and $\mathbf{c}_{\text{OA}}^* \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{OA}}^*, \text{vdec}_{n,q}(\mathbf{h}_{\text{ID}}))$.
5. Generate the non-interactive proof π with witnesses:
- for signature verification: $[\mathbf{d}_1^T \parallel \mathbf{d}_2^T \parallel \tau[1] \cdot \mathbf{d}_2^T \parallel \dots \parallel \tau[l] \cdot \mathbf{d}_2^T]^T$, \mathbf{b} , \mathbf{b}^* , \mathbf{r} ,
 - for vector decomposition: $\mathbf{w} = \text{vdec}_{n,q}(\mathbf{D}_0 \cdot \mathbf{r} + \mathbf{D}_1 \cdot \mathbf{h})$, $\mathbf{h} = \text{vdec}_{n,q}(\mathbf{h}_{\text{ID}})$,
 - for encryption of message: $\mathbf{R}_{\mathbf{m}}$, \mathbf{b} , \mathbf{m} , $\mathbf{R}_{\mathbf{m},r}$, $\mathbf{R}_{\mathbf{m}}^*$, \mathbf{b}^* , $\mathbf{R}_{\mathbf{m},r}^*$,
 - for encryption of hash of public key: \mathbf{R}_{OA} , \mathbf{h}_{ID} , $\mathbf{R}_{\text{OA},r}$, \mathbf{R}_{OA}^* , $\mathbf{R}_{\text{OA},r}^*$,
- in the following relations:
- $\Sigma.\text{Vf}(\text{pk}_{\text{GM}}, \text{cert}_{\text{pk}_{\text{ID}}}, \text{pk}_{\text{ID}}) = 1$ with $(\mathbf{A}, \mathbf{A}_0, \dots, \mathbf{A}_l, \mathbf{D}, \mathbf{D}_0, \mathbf{D}_1)$ from pk_{GM} and $(\tau, \mathbf{d}, \mathbf{r})$ from $\text{cert}_{\text{pk}_{\text{ID}}}$:

$$\mathbf{u} = [\mathbf{A} \parallel \mathbf{A}_0 \parallel \dots \parallel \mathbf{A}_l] \cdot [\mathbf{d}_1^T \parallel \mathbf{d}_2^T \parallel \tau[1] \mathbf{d}_2^T \parallel \dots \parallel \tau[l] \mathbf{d}_2^T]^T + (-\mathbf{D}) \cdot \mathbf{w} \bmod q,$$

$$\mathbf{0} = \mathbf{H}_{n,q} \cdot \mathbf{w} + (-\mathbf{D}_0) \cdot \mathbf{r} + (-\mathbf{D}_1) \cdot \mathbf{h} \bmod q$$

$$\mathbf{0} = \mathbf{H}_{m,q} \cdot \mathbf{h} + (-\mathbf{F}) \cdot \text{vdec}_{m,q}(\mathbf{b}) + (-\mathbf{F}^*) \cdot \text{vdec}_{m,q}(\mathbf{b}^*) \bmod q.$$

- $\mathbf{c}_m \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{ID}}, \mathbf{m})$ and $\mathbf{c}_m^* \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{ID}}^*, \mathbf{m})$ with $(\mathbf{b}, \mathbf{b}^*)$ from public key pk_{ID} and $(\mathbf{R}_m, \mathbf{R}_{m,r}, \mathbf{R}_m^*, \mathbf{R}_{m,r}^*)$ as the randomness:

$$\begin{aligned} \mathbf{c}_{m,0} &= \mathbf{R}_m \bar{\mathbf{A}}^\top, & \mathbf{c}_{m,1} &= \mathbf{R}_m \mathbf{b} + \mathbf{m} \cdot \lfloor q/2 \rfloor, \\ \mathbf{c}_{m,r,0} &= \mathbf{R}_{m,r} \bar{\mathbf{A}}^\top, & \mathbf{c}_{m,r,1} &= \mathbf{R}_{m,r} \mathbf{b}, \\ \mathbf{c}_{m,0}^* &= \mathbf{R}_{\text{OA}}^* \bar{\mathbf{A}}^\top, & \mathbf{c}_{m,1}^* &= \mathbf{R}_m^* \mathbf{b}^* + \mathbf{m} \cdot \lfloor q/2 \rfloor, \\ \mathbf{c}_{m,r,0}^* &= \mathbf{R}_{m,r}^* \bar{\mathbf{A}}^\top, & \mathbf{c}_{m,r,1}^* &= \mathbf{R}_{m,r}^* \mathbf{b}^*. \end{aligned}$$

- $\mathbf{c}_{\text{OA}} \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{OA}}, \mathbf{h})$ and $\mathbf{c}_{\text{OA}}^* \leftarrow \mathcal{E}.\text{Enc}(\mathcal{E}.\text{pk}_{\text{OA}}^*, \mathbf{h})$ with $\mathbf{h} = \text{vdec}_{n,q}(\mathbf{h}_{\text{ID}})$, $(\mathbf{b}_{\text{OA}}, \mathbf{b}_{\text{OA}}^*)$ from pk_{OA} and $(\mathbf{R}_{\text{OA}}, \mathbf{R}_{\text{OA},r}, \mathbf{R}_{\text{OA}}^*, \mathbf{R}_{\text{OA},r}^*)$ as the randomness:

$$\begin{aligned} \mathbf{c}_{\text{OA},0} &= \mathbf{R}_{\text{OA}} \bar{\mathbf{A}}^\top, & \mathbf{c}_{\text{OA},1} &= \mathbf{R}_{\text{OA}} \mathbf{b}_{\text{OA}} + \mathbf{h} \cdot \lfloor q/2 \rfloor, \\ \mathbf{c}_{\text{OA},r,0} &= \mathbf{R}_{\text{OA},r} \bar{\mathbf{A}}^\top, & \mathbf{c}_{\text{OA},r,1} &= \mathbf{R}_{\text{OA},r} \mathbf{b}_{\text{OA}}, \\ \mathbf{c}_{\text{OA},0}^* &= \mathbf{R}_{\text{OA}}^* \bar{\mathbf{A}}^\top, & \mathbf{c}_{\text{OA},1}^* &= \mathbf{R}_{\text{OA}}^* \mathbf{b}_{\text{OA}}^* + \mathbf{h} \cdot \lfloor q/2 \rfloor, \\ \mathbf{c}_{\text{OA},r,0}^* &= \mathbf{R}_{\text{OA},r}^* \bar{\mathbf{A}}^\top, & \mathbf{c}_{\text{OA},r,1}^* &= \mathbf{R}_{\text{OA},r}^* \mathbf{b}_{\text{OA}}^*. \end{aligned}$$

Some witnesses are transformed to binary representation, which fits with the existing proof for the linear system [20] that uses binary witness.

6. Output the ciphertext $\mathbf{c} = (\mathbf{c}_m, \mathbf{c}_m^*, \mathbf{c}_{\text{OA}}, \mathbf{c}_{\text{OA}}^*, \pi)$.

- $\text{Vf}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_{\mathcal{R}}, x, \mathbf{c})$: Return the verification result of proof π against \mathbf{c} .
- $\text{San}(\mathbf{c}) \rightarrow \mathbf{c}'$: Call Vf over \mathbf{c} and output \perp if it is invalid; otherwise, parse $\mathbf{c} = (\mathbf{c}_m, \mathbf{c}_m^*, \mathbf{c}_{\text{OA}}, \mathbf{c}_{\text{OA}}^*, \pi)$ and output $(\mathcal{E}.\text{San}(\mathbf{c}_m), \mathcal{E}.\text{San}(\mathbf{c}_{\text{OA}}))$.
- $\text{Dec}(\text{sk}_{\text{ID}}, \mathbf{c}') \rightarrow \mathbf{m}$: Parse \mathbf{c}' as $(\mathbf{c}'_m, \mathbf{c}'_{\text{OA}})$ and output $\mathbf{m} \leftarrow \mathcal{E}.\text{Dec}(\text{sk}_{\text{ID}}, \mathbf{c}'_m)$.
- $\text{Open}(\text{sk}_{\text{OA}}, \mathbf{c}') \rightarrow \text{pk}_{\text{ID}}$: Parse \mathbf{c}' as $(\mathbf{c}'_m, \mathbf{c}'_{\text{OA}})$ and run $\mathbf{h} \leftarrow \mathcal{E}.\text{Dec}(\text{sk}_{\text{ID}}, \mathbf{c}'_{\text{OA}})$. Compute $\mathbf{h}^* = \mathbf{H}_{m,q} \cdot \mathbf{h}$ and search for the public key hashes to the value \mathbf{h}^* by $\mathbf{F} \cdot \text{vdec}_{m,q}(\cdot) + \mathbf{F}^* \cdot \text{vdec}_{m,q}(\cdot)$. Output the corresponding public key pk_{ID} ; or \perp if it is not found.

Optimization. Instantiating our generic construction as above is not optimized. Specifically, for the ciphertexts marked with *, *i.e.*, $\mathbf{c}_{m,r}^*$ and $\mathbf{c}_{\text{OA},r}^*$, their randomizer components are redundant because these ciphertexts are not sanitized at San but simply dropped instead. One can remove these randomizers from the ciphertexts, which also reduces the size of the witness.

Furthermore, the randomizers for encryption of \mathbf{m} for a user and $\text{vdec}_{n,q}(\mathbf{h}_{\text{ID}})$ for OA can be shared by using only single randomness \mathbf{R} as a witness instead of two $\mathbf{R}_{m,r}, \mathbf{R}_{\text{OA},r}$. Specifically, $\mathbf{c}_{m,r,0}$ and $\mathbf{c}_{\text{OA},r,0}$ can be shared, *i.e.*, $\mathbf{c}_{m,r}$ and $\mathbf{c}_{\text{OA},r}$ are changed into $(\mathbf{c}_r = \mathbf{R}_r \bar{\mathbf{A}}^\top, \mathbf{c}_{m,r} = \mathbf{R}_r \mathbf{b}, \mathbf{c}_{\text{OA},r} = \mathbf{R}_r \mathbf{b}_{\text{OA}})$, with San algorithm inputs $(\mathbf{c}_r, \mathbf{c}_{m,r})$ for \mathbf{c}_m and $(\mathbf{c}_r, \mathbf{c}_{\text{OA},r})$ for \mathbf{c}_{OA} as randomizers.

Concerns with Subliminal Channel over Error. One may concern that the error term may form a subliminal channel, as mentioned in the open problem of Damgård *et al.* [12]. We briefly explain how our construction prevents it. First, the public key is certified that the error term there cannot be changed. Second,

our encryption algorithm by itself does not need any other noises (beyond the involvement of the public key). Third, if the adversary tries to introduce an error term to the ciphertext, it will fail the proof verification.

Our sanitization mechanism critically relies on homomorphism. As argued before, a sanitized ciphertext of our construction, which is a homomorphically-evaluated HE ciphertext, remains a random element in the ciphertext space to any adversary without the decryption key. The noise analysis of lattice-based encryption schemes, *e.g.*, for breaking circuit privacy, is not applicable here since it requires the knowledge of the decryption key.

6 Concluding Remarks

We connect two seemingly related but different primitives, namely, access control encryption and group encryption. We borrowed the wisdom from the group encryption literature and proposed a new access control encryption scheme. Together with our sanitization technique for LWE-based encryption, we provide a candidate solution to the open problem left by Damgård in their seminal work in access control encryption, namely, a practically interesting access control encryption scheme from noisy, post-quantum assumptions, instead of using heavyweight tools such as indistinguishability obfuscation or fully homomorphic encryption.

While we slightly optimized our instantiation (compared to the existing lattice-based group encryption scheme), there is still room for improvement, especially for the delicate proof techniques. For practical efficiency, our suggestion is to use the latest access control encryption scheme of Wang and Chow [26], which comes with timing figures for a prototype implementation (and appears to be adaptive secure in the random oracle model, or selective secure in the common reference string model by replacing Fiat–Shamir proof with ZK non-interactive succinct argument of knowledge). A long-term research problem is to improve the efficiency of cryptosystems with resiliency to potential quantum computers.

References

1. Abe, M., Chow, S.S.M., Haralambiev, K., Ohkubo, M.: Double-trapdoor anonymous tags for traceable signatures. *Int. J. Inf. Sec.* **12**(1), 19–31 (2013)
2. Agrawal, S., Wu, D.J.: Functional encryption: deterministic to randomized functions from simple assumptions. In: Coron, J.-S., Nielsen, J.B. (eds.) *EUROCRYPT 2017*. LNCS, vol. 10211, pp. 30–61. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_2
3. El Aïmani, L., Joye, M.: Toward practical group encryption. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) *ACNS 2013*. LNCS, vol. 7954, pp. 237–252. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38980-1_15

4. Badertscher, C., Matt, C., Maurer, U.: Strengthening access control encryption. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 502–532. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_18
5. Böhl, F., Hofheinz, D., Jäger, T., Koch, J., Striecks, C.: Confined guessing: new signatures from standard assumptions. *J. Cryptol.* **28**(1), 176–208 (2014). <https://doi.org/10.1007/s00145-014-9183-z>
6. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
7. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_13
8. Cathalo, J., Libert, B., Yung, M.: Group encryption: non-interactive realization in the standard model. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 179–196. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_11
9. Chow, S.S.M.: Real traceable signatures. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 92–107. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05445-7_6
10. Chow, S.S.M., Feh, K., Lai, R.W.F., Malavolta, G.: Multi-client oblivious RAM with poly-logarithmic communication. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12492, pp. 160–190. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_6
11. Chow, S.S.M., Roth, V., Rieffel, E.G.: General certificateless encryption and timed-release encryption. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 126–143. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85855-3_9
12. Damgård, I., Haagh, H., Orlandi, C.: Access control encryption: enforcing information flow with cryptography. In: Hirt, M., Smith, A. (eds.) TCC 2016-B. LNCS, vol. 9986, pp. 547–576. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_21
13. Fuchsbauer, G., Gay, R., Kowalczyk, L., Orlandi, C.: Access control encryption for equality, comparison, and more. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 88–118. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_4
14. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
15. Izabachène, M., Pointcheval, D., Vergnaud, D.: Mediated traceable anonymous encryption. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 40–60. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14712-8_3
16. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_34
17. Kiayias, A., Tsiounis, Y., Yung, M.: Group encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 181–199. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76900-2_11

18. Kim, S., Wu, D.J.: Access control encryption for general policies from standard assumptions. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 471–501. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_17
19. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Signature schemes with efficient protocols and dynamic group signatures from lattice Assumptions. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 373–403. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_13
20. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 101–131. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_4
21. Libert, B., Yung, M., Joye, M., Peters, T.: Traceable group encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 592–610. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_34
22. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: ACM Symposium on Theory of Computing (STOC), pp. 427–437 (1990)
23. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Symposium on Theory of Computing (STOC), pp. 84–93 (2005)
24. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: ACM Symposium on Foundations of Computer Science (FOCS), pp. 543–553 (1999)
25. Tan, G., Zhang, R., Ma, H., Tao, Y.: Access control encryption based on LWE. In: ACM ASIA Public-Key Cryptography@AsiaCCS, pp. 43–50 (2017)
26. Wang, X., Chow, S.S.M.: Cross-domain access control encryption: arbitrary-policy, constant-size, efficient. In: IEEE Symposium on Security and Privacy (S&P), pp. 388–401 (2021)