IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 March 2025, accepted 10 May 2025, date of publication 20 May 2025, date of current version 28 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3571714

RESEARCH ARTICLE

An Efficient and Generic Construction of Public Key Encryption With Equality Test Under the Random Oracle Model

SEONGBONG CHOI, CHANHYEOK PARK, AND HYUNG TAE LEE
School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea

Corresponding author: Hyung Tae Lee (hyungtaelee@cau.ac.kr)

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) Grant funded by the Korea Government (MSIT) under Grant RS-2023-00229400 (Development of User Authentication and Privacy Preserving Technology for a Secure Metaverse Environment, 75%); in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant NRF-2022R1A4A5034130 (Connected Mobility Basic Research Laboratory, 25%); and in part by the Chung-Ang University Research Scholarship Grants, in 2022.

ABSTRACT Public key encryption with equality test (PKE-ET) facilitates authorized entities in distinguishing if two ciphertexts involve the identical underlying message. This functionality has driven its adoption across various applications, such as secure data management in cloud environments, encrypted spam filtering, and keyword searches in encrypted databases. However, existing generic PKE-ET constructions often rely on cryptographic primitives that require strong security assumptions or additional complex functionalities, leading to inefficiencies. This paper proposes an improved generic construction for PKE-ET under the random oracle model. The proposed method leverages only fundamental cryptographic building blocks, relying exclusively on a standard public key encryption (PKE) scheme along with cryptographic hash functions, without requiring additional complex primitives. More concretely, the proposed construction leverages a PKE scheme that ensures one-wayness against chosen plaintext attacks (OW-CPA), differing from earlier generic frameworks that necessitate stronger security guarantees. We demonstrate that the proposed construction satisfies one-wayness against adaptively chosen ciphertext attacks (OW-CCA2) for a Type-1 adversary–who possesses equality test trapdoors–and indistinguishability against adaptively chosen ciphertext attacks (IND-CCA2) for a Type-2 adversary–who lacks such trapdoors–assuming that the exploited PKE scheme achieves OW-CPA security. Furthermore, we present three concrete instantiations of our generic framework using well-established PKE schemes: The hashed ElGamal encryption, the RSA encryption, and the BIKE cryptosystem. These instantiations offer notable advantages over existing PKE-ET designs. The first two constructions enhance efficiency in comparison to prior schemes, while the third stands out as the first code-based PKE-ET instantiation, relying solely on code-based security assumptions.

INDEX TERMS Equality test, public key encryption, generic construction, random oracle model, discrete logarithm-based instantiation, RSA-based instantiation, code-based instantiation.

I. INTRODUCTION

Public key encryption with equality test (PKE-ET), initially formulated by Yang et al. [1], is a cryptographic scheme that facilitates a designated tester in distinguishing if two ciphertexts encrypt the identical message, even if distinct public keys were used for encryption. This capability has made PKE-ET schemes highly relevant in various real-world applications, including secure email spam filtering, encrypted database searches, and personal health record management in cloud-based systems.

Given its broad applicability, numerous PKE-ET constructions and enhancements have been proposed [2], [3], [4], [5], [6], [7], [8], [9], aiming to improve functionality,

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Masucci.

© 2025 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

efficiency, and security. Among these, generic PKE-ET constructions have been of particular interest, because they provide a flexible framework that enables instantiations based on diverse cryptographic primitives. Several generic constructions [10], [11], [12] have been proposed, which provide insights into key design features and pave the way for more efficient PKE-ET schemes.

Despite these advancements, existing generic PKE-ET frameworks often rely on cryptographic primitives that impose strong security requirements or necessitate additional functionalities, making them computationally expensive. For example, generic PKE-ET approaches were proposed within the random oracle model [10], [11] using public key encryption (PKE) schemes that ensure indistinguishability against adaptively chosen ciphertext attacks (IND-CCA2), which is one of the most stringent security notions for PKE. On the other hand, Lee et al. [12] developed a generic approach in the standard model that builds upon a hierarchical identity-based encryption (HIBE) satisfying indistinguishability against selective identity and adaptively chosen ciphertext attacks (IND-sID-CCA2), along with strongly unforgeable signature schemes.

Consequently, instantiations derived from these existing generic constructions tend to be less efficient than concrete PKE-ET schemes designed directly without relying on generic constructions. Therefore, exploring alternative approaches that leverage more fundamental cryptographic primitives is crucial to obtaining a more efficient instantiation from generic approaches.

A. OUR RESULTS

We propose a novel and efficient generic construction for PKE-ET schemes within the random oracle model. Our proposed design employs a PKE scheme that guarantees only one-wayness against chosen plaintext attacks (OW-CPA), which is weaker than indistinguishability against chosen plaintext atttacks (IND-CPA) and IND-CCA2. We formally prove that, under this assumption, our generic scheme satisfies one-wayness against adaptively chosen ciphertext attacks (OW-CCA2) for a Type-1 adversary who possesses equality-test trapdoors, while ensuring IND-CCA2 security for a Type-2 adversary who lacks such trapdoors.

Furthermore, we provide concrete instantiations of our generic construction using three well-known PKE schemes:

- Hashed ElGamal-based instantiation: It achieves the best computational efficiency in encryption, decryption and test, and reduces ciphertext size by at least 11.1% compared to existing discrete logarithm-based PKE-ET schemes [2], [10], [11] under Tang's all-or-nothing (AoN) model [2].
- RSA-based instantiation: It reduces a ciphertext size by approximately 42% compared to the existing RSA-based PEK-ET scheme, proposed by Zhu et al. [13], without increasing computational or communication costs.
- BIKE-based instantiation: It represents the first known code-based PKE-ET construction, relying solely on code-based security assumptions.

B. OUR DESIGN STRATEGY

This study adopts Tang's all-or-nothing (AoN) model [2] as the underlying framework for our PKE-ET system. In this model, the sender passes a ciphetext to the recipient by encrypting a message with the public key of the recipient, and the recipient stores the ciphertext on a server. When needed, the recipient can provide an equality-test trapdoor to a tester, who then checks if two ciphertexts involve the identical message. A key feature of this model is that the tester cannot learn the plaintext content but can only confirm ciphertext equality. (See Section II-B for further details of Tang's AoN model.)

Our approach to designing a secure and efficient generic construction focuses on streamlining the encryption structure while maintaining fundamental security properties. The initial (insecure) PKE-ET schemes typically represent ciphertexts in the following form:

$$C_1 = \mathsf{PKE.E}(pk_1, M) \text{ and } C_2 = \mathsf{PKE.E}(pk_2, \mathcal{H}(M))$$

where $\mathcal{H}$ is a cryptographic hash function, $\mathsf{PKE.E}$ denotes an encryption algorithm of the employed PKE scheme PKE, and pk_1, pk_2 are the associated public keys. In these constructions, decryption is performed by applying the corresponding secret keys sk_1, sk_2 to C_1 and C_2, respectively, allowing the retrieval of message M. The decryption process further verifies whether the hash of the decrypted value from C_1 matches the value obtained from decrypting C_2. Moreover, the equality test algorithm determines if two ciphertexts encrypt the identical message by decrypting the C_2 component using the secret key sk_2, which functions as a trapdoor for conducting the equality test.

However, a ciphertext containing only C_1 and C_2 does not ensure IND-CCA2 security for a Type-2 adversary, who lacks a trapdoor associated with the target user. In the security game for IND-CCA2 security, the adversary is presented with two challenge messages, M_0, M_1, and receives the corresponding ciphertext. To compromise IND-CCA2 security, the adversary can generate C_2' by computing $C_2' = \mathsf{PKE.E}(pk_2, \mathcal{H}(M_0))$. Since the underlying encryption scheme PKE employs randomness for security, the newly generated ciphertext component C_2' will differ from C_2 with overwhelming probability. The adversary subsequently invokes the decryption query on the altered ciphertext (C_1, C_2'). Based on the message in the challenge ciphertext, if C_1 corresponds to the encryption of M_0, the decryption oracle returns M_0; otherwise, it outputs $\perp$. By analyzing the oracle's response, the adversary can successfully distinguish between two candidate messages, thereby breaking IND-CCA2 security.

To avoid such an attack, Lin et al. [10] developed a generic approach that introduces additional randomness into the encryption process. (Throughout the paper, we refer to

Lin et al.'s construction as LSQ18.) Their approach extends the original encryption scheme by incorporating randomly selected points into both the message and its hash digest. Furthermore, an additional ciphertext component C_3 is introduced, which is calculated as the hash digest of the concatenation of C_1, C_2, and a constant term derived from a degree-2 polynomial f. This polynomial is defined such that it passes through a designated public parameter point and two random points utilized in generating C_1 and C_2. Specifically, in LSQ18, a ciphertext encrypting a message M consists of three components, C_1, C_2, C_3, defined as follows:

$$\begin{aligned} C_1 &= \mathsf{PKE.E}(pk_1, M\|x_1\|y_1), \\ C_2 &= \mathsf{PKE.E}(pk_2, \mathcal{H}(M)\|x_2\|y_2), \\ C_3 &= \mathcal{H}'(C_1, C_2, f(0)) \end{aligned}$$

where $\mathcal{H}$ and $\mathcal{H}'$ are cryptographic hash functions, and f is a degree-2 polynomial that interpolates through three specific points, $P_1 = (\kappa_1, \kappa_2)$, $P_2 = (x_1, y_1)$, and $P_3 = (x_2, y_2)$. Here, the values x_1, x_2, y_1, y_2 are picked from $\mathbb{Z}_p^*$ randomly during the encryption process, where p is a sufficiently large prime. The parameters κ_1, κ_2 are picked from $\mathbb{Z}_p^*$ randomly during the setup phase. This construction ensures IND-CCA2 security by making any ciphertext modification infeasible without explicit knowledge of the random values associated with points P_2 and P_3. However, this improvement in security results in a larger ciphertext size, since both C_1 and C_2 need to accommodate additional randomness, ultimately diminishing the scheme's overall efficiency.

As an alternative approach, Lee et al. [11] designed a generic construction by leveraging a cryptographic property known as randomness extractability. (Throughout the paper, we denote Lee et al.'s construction from [11] as LLSW19.) This property enables the decryption algorithm to retrieve the randomness originally used during encryption. In their scheme, a ciphertext for a message M comprises three components, C_1, C_2, C_3, and is generated as follows:

$$\begin{aligned} C_1 &= \mathsf{PKE.E}(pk_1, M; r_1), \\ C_2 &= \mathsf{PKE.E}(pk_2, \mathcal{H}(M); r_2), \\ C_3 &= \mathcal{H}'(C_1, C_2, r_1, r_2) \end{aligned}$$

where r_1 and r_2 denote the randomness values used in the encryption of C_1 and C_2, respectively, and $\mathcal{H}, \mathcal{H}'$ are cryptographic hash functions. A distinguishing feature of the above approach is that the decryption algorithm can reconstruct r_1 and r_2 using randomness extractability of the underlying PKE scheme. This capability enables the decryptor to authenticate ciphertexts and detect unauthorized modifications. Since generating a valid ciphertext modification necessitates precise knowledge of r_1 and r_2, adversaries are effectively prevented from tampering with ciphertexts, thereby ensuring IND-CCA2 security. It is noteworthy that PKE schemes obtained by applying Fujisaki-Okamoto (FO) transformation are IND-CCA2 secure and naturally satisfy randomness extractability, making them well-suited for instantiation within Lee et al.'s generic framework. However, this approach incurs additional computational costs due to its reliance on a strongly secure PKE scheme, which reduces overall efficiency.

Both LSQ18 and LLSW19 mandate an IND-CCA2 secure PKE scheme because the exploited encryption mechanism is responsible for securing not only the message M but also its hash digest $\mathcal{H}(M)$. As a result, the encryption scheme must be robust enough to prevent adversaries from extracting any information about the plaintext or its hashed counterpart. This requirement makes IND-CCA2 security an inherent necessity for the PKE scheme exploited in these generic constructions.

To avoid the reliance on highly secure PKE schemes, our generic construction encrypts random values r_1 and r_2 instead of directly encrypting the message and its hash value. This approach utilizes the underlying PKE scheme to ensure security, while reducing the need for strong cryptographic assumptions. In our construction, a ciphertext corresponding to a message M consists of four components, C_1, C_2, C_3, and C_4, obtained by computing

$$\begin{aligned} &C_1 = \mathsf{PKE.E}(pk_1, r_1), \quad C_2 = \mathsf{PKE.E}(pk_2, r_2), \\ &C_3 = M \oplus \mathcal{H}'(r_1, r_2), \quad C_4 = \mathcal{H}(M) \oplus \mathcal{H}''(r_2, C_1, C_2, C_3) \end{aligned}$$

where $\mathcal{H}, \mathcal{H}', \mathcal{H}''$ denote cryptographic hash functions, and r_1, r_2 are randomly chosen from the message space of the exploited PKE construction. This design makes unauthorized ciphertext manipulation infeasible, as an adversary attempting to modify the challenge ciphertext in a meaningful way must determine both r_1 and r_2. As a result, our construction necessitates only OW-CPA security for the exploited PKE scheme, rather than relying on a stronger security assumption. Although our ciphertext structure consists of four components–compared to the three-component designs in LSQ18 and LLSW19, our concrete instantiations demonstrate that utilizing a PKE scheme with weaker security requirements leads to greater overall efficiency. A comprehensive efficiency analysis is presented in Section I-A and Section V.

C. RELATED WORKS

Since the first PKE-ET proposal by Yang et al. [1], a wide range of PKE-ET schemes and their extensions have been proposed. This subsection presents a concise overview of the most relevant works, with a particular emphasis on PKE-ET construction techniques.

1) PKE-ET

The framework of PKE-ET was initially developed by Yang et al. [1], along with a pairing-based instantiation that facilitates anyone in distinguishing if two ciphertexts contain the identical message. Since then, various PKE-ET schemes have been developed, focusing on improving efficiency or enhancing functionality. To strengthen access control over equality test, Tang [2] proposed the AoN PKE-ET model, restricting the ability to perform equality tests only to user-authorized testers. Subsequently, Ma et al. [5]

introduced a construction in which a delegated entity is exclusively permitted to execute equality tests through interactions with the ciphertext-storing server. Simultaneously, Huang et al. [3] presented a PKE-ET construction where users generate trapdoors, enabling testers to conduct equality tests either across all ciphertexts belonging to the user or on a specific ciphertext. However, this scheme was later found to have security vulnerabilities, which were subsequently addressed by Lee et al. [14]. In addition, another PKE-ET construction that introduced four distinct authorization levels for performing equality tests was designed by Ma et al. [4]. Recently, Susilo et al. [15] introduced PKE-MET, an extension of PKE-ET that supports equality tests across multiple ciphertexts, along with PKE-FMET, which enables more flexible multiple ciphertexts equality test.

With the development of various PKE-ET models, numerous cryptographic techniques have been explored for constructing secure and efficient schemes. These include discrete logarithm-based approaches [7], [16], RSA-based designs [13], lattice-based constructions [9], [17], [18], [19], and multivariate equation-based methods [20].

The equality test functionality has been extended beyond traditional PKE to various other cryptographic frameworks, including identity-based encryption [6], [8], [10], [21], [22], attribute-based encryption [23], [24], [25], certificateless encryption [26], [27], and signcryption [10], [28], [29].

2) GENERIC CONSTRUCTIONS FOR PKE-ET

This section explores generic PKE-ET constructions, which are directly relevant to our work. In 2016, Lee et al. [30] introduced a semi-generic scheme for PKE-ET under the random oracle model. (Throughout this paper, the semi-generic scheme by Lee et al. [30] is referred as LLSW16.) This approach incorporates a computational Diffie-Hellman (CDH) problem instance into both the message and its hash digest during encryption. Since this scheme relies not only on the security of the exploited PKE scheme but also on the computational intractability of the CDH problem, it is classified as semi-generic rather than fully generic. To remove the dependency on the CDH assumption, Lin et al. [10] presented an alternative generic approach within the random oracle model, substituting CDH instances with randomly selected points on a degree-2 polynomial. Around the same time, Lee et al. [11] introduced another generic PKE-ET scheme under the random oracle model, based on PKE schemes which support randomness extractability–a property that enables efficient recovery of encryption randomness during decryption. Moving beyond random oracle heuristics, Lee et al. [12] developed a generic approach in the standard model. (Throughout the paper, we refer to Lee et al.'s generic scheme in [12] as LLS+20.) However, this approach requires more sophisticated cryptographic building blocks, a 2-level IND-sID-CCA2 secure HIBE scheme along with a strongly unforgeable signature scheme, making it significantly more complex. Beyond PKE-ET, generic constructions have also been extended to various encryption paradigms, including identity-based setting [10], [12], [31], [32], attribute-based setting [33], and signcryption setting [10].

D. ORGANIZATION OF THE PAPER

Section II covers the essential primitives, including formal definitions of PKE-ET and cryptographic building blocks utilized in our generic construction. Section III proposes our improved generic PKE-ET scheme and provides a comparative analysis highlighting between our approach and prior generic constructions. Section IV provides an in-depth security analysis, offering a rigorous proof of the proposed scheme's security guarantees. We showcase three concrete instantiations of our generic construction using well-established PKE schemes and evaluate their efficiency against existing PKE-ET constructions in Section V. At last, Section VI summarizes our contributions and concludes with final remarks.

II. PRELIMINARIES

This section presents key properties of essential cryptographic components that serve as the foundation for our generic construction. We start by presenting the notation and terminology that will be utilized throughout the paper.

Notation: κ indicates the security parameter. The notation $a \xleftarrow{\$} A$ signifies that a is drawn uniformly at random from the set A. For an algorithm Alg, $\mathsf{Alg} \rightarrow a$ indicates that a is the output generated by Alg. The concatenation of two strings, s_1 and s_2, is represented as $s_1 \| s_2$. For two integers a and b, the notation $[a, b]$ represents the inclusive range of integers between a and b.

A. PUBLIC KEY ENCRYPTION AND ITS OW-CPA SECURITY

This subsection reviews formal definitions of a PKE scheme and its OW-CPA security.

Definition 1: A PKE scheme PKE *comprises three polynomial-time algorithms,* $(\mathsf{PKE.KG}, \mathsf{PKE.E}, \mathsf{PKE.D})$*, formulated as below:*

- $\mathsf{PKE.KG}(1^\kappa)$*: Given* κ*, it generates and returns a public key* pk *and a corresponding secret key* sk*.*
- $\mathsf{PKE.E}(pk, M; r)$*: Given* pk*, a message* M*, and a randomness value* r*, it outputs a ciphertext* C*. Unless explicitly required, the randomness* r *in* $\mathsf{PKE.E}(pk, M; r)$ *is omitted for brevity.*
- $\mathsf{PKE.D}(sk, C)$*: Given* sk *and* C*, this algorithm returns a recovered message* M'*.*

A PKE scheme PKE achieves *correctness* if it satisfies that

$$\Pr[\mathsf{PKE.D}(sk, \mathsf{PKE.E}(pk, M)) = M] = 1$$

for any given message M and κ where the key pair (pk, sk) is obtained by executing $\mathsf{PKE.KG}(1^\kappa) \rightarrow (pk, sk)$.

Definition 2 (OW-CPA Security of a PKE Scheme): A PKE scheme PKE *achieves OW-CPA security if for any probabilistic polynomial-time (PPT) adversary* $\mathcal{A}$*,* $\mathcal{A}$*'s advantage in the*

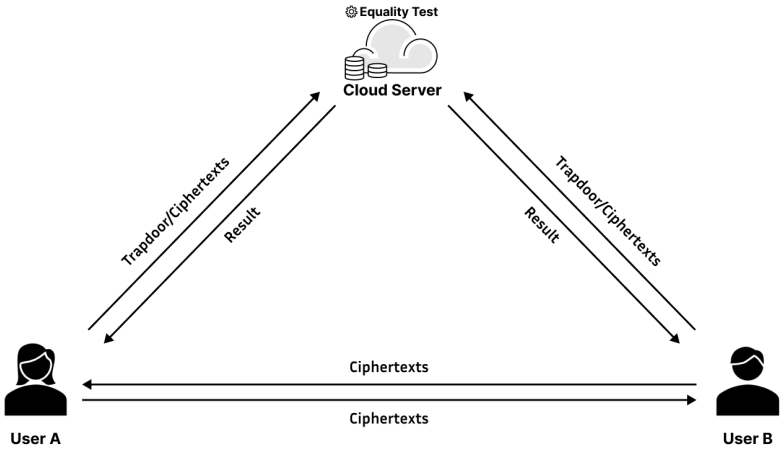


FIGURE 1. PKE-ET system model.

following security game against the challenger $\mathcal{C}$ is negligible in κ.

1) **Setup** : *$\mathcal{C}$ executes* $\mathsf{PKE.KG}(1^\kappa)$ *to get a public-secret key pair (pk, sk) and forwards pk to $\mathcal{A}$.*
2) **Challenge** : *$\mathcal{C}$ picks a message M^*, executes* $\mathsf{PKE.E}(pk, M^*) \rightarrow C^*$*, and forwards C^* to $\mathcal{A}$.*
3) **Guess** : *$\mathcal{A}$ outputs M'.*

$\mathcal{A}$'s advantage in breaking OW-CPA security is defined as

$$\mathsf{Adv}^{\mathrm{OW-CPA}}_{\mathcal{A},\mathrm{PKE}}(\kappa) = \Pr[M^* = M'].$$

B. DEFINITIONS OF PKE-ET

Throughout this paper, our system and security models for PKE-ET are based on Tang's AoN PKE-ET models [2].

1) SYSTEM MODEL FOR OUR PKE-ET

Figure 1 presents a conceptual representation of our PKE-ET system model. The model comprises two main types of entities: Users (which include senders and recipients) and testers. The sender encrypts a message with the recipient's public key and delivers the resulting ciphertext to the recipient. The recipient decrypts the ciphertext with their secret key and may store it on a server for later use. When an equality test is needed, the recipient generates an equality test trapdoor and grants it to a tester who is able to access the server that holds the recipient's encrypted data. The tester can distinguish if two ciphertexts under public keys associated with trapdoor-issuing recipients contain the identical message.

2) DEFINITIONS OF PKE-ET

Before formally defining PKE-ET, we first introduce the necessary notations for clarity. In our system, a unique index i is assigned to each user, where $i \in [1, N]$ and N indicates the total number of users. A user corresponding to index i is represented as U_i. The cryptographic elements associated with U_i are defined as follows: The public-secret key pair is represented as (pk_i, sk_i), the ciphertext encrypted under pk_i is represented as CT_i, and the equality test trapdoor is denoted as td_i.

Definition 3 (PKE-ET): A PKE-ET scheme $\mathsf{PKE\text{-}ET}$ *consists of six polynomial-time algorithms* $(\mathsf{S}, \mathsf{KG}, \mathsf{E}, \mathsf{D}, \mathsf{TD}, \mathsf{ET})$*, formulated as follows.*

- $\mathsf{S}(1^\kappa)$*: Given κ, it outputs a public parameter pp, which involves the message space $\mathcal{M}$ and the randomness space $\mathcal{R}$.*
- $\mathsf{KG}(pp)$*: Given pp, it generates and returns a pair of public and secret keys (pk, sk).*
- $\mathsf{E}(pk, M; r)$*: Given pk, a message $M \in \mathcal{M}$, and a random value $r \in \mathcal{R}$, it returns a ciphertext CT. The randomness r is omitted unless explicitly required.*
- $\mathsf{D}(sk, CT)$*: Given sk and CT, it recovers and returns the message M'.*
- $\mathsf{TD}(sk_i)$*: Given a user's secret key sk_i, it generates and returns the corresponding trapdoor td_i for user U_i.*
- $\mathsf{ET}(td_i, CT_i, td_j, CT_j)$*: Given two trapdoor-ciphertext pairs (td_i, CT_i) and (td_j, CT_j), it checks whether CT_i, CT_j encrypt the identical message. It returns* 1 *if they match and $\perp$ otherwise.*

A PKE-ET scheme $\mathsf{PKE\text{-}ET}$ achieves *correctness* if it satisfies the three properties below: For any $i, j \in [1, N]$ and any messages $M_i, M_j \in \mathcal{M}$, the scheme must ensure

1) Decryption correctness: $\Pr[\mathsf{D}(sk_i, CT_i) \rightarrow M_i] = 1$.
2) Equality test correctness: If the decrypted messages are identical and not invalid, then the equality test must return 1. That is,

$$\Pr[\mathsf{ET}(td_i, CT_i, td_j, CT_j) \rightarrow 1] = 1$$

if $\mathsf{D}(sk_i, CT_i) = \mathsf{D}(sk_j, CT_j) \neq \perp$.
3) Equality test soundness: If the decrypted messages are different, then the probability that the equality test erroneously outputs 1 must be negligibly small in the

security parameter κ. That is,

$$\Pr[\mathsf{ET}(td_i, CT_i, td_j, CT_j) \to 1]$$

must be negligibly small in κ whenever $\mathsf{D}(sk_i, CT_i) \neq \mathsf{D}(sk_j, CT_j)$.

These conditions hold under the following key generation and encryption processes: $\mathsf{S}(1^\kappa) \to pp$, $\mathsf{KG}(pp) \to (pk_i, sk_i)$, $\mathsf{KG}(pp) \to (pk_j, sk_j)$, $\mathsf{E}(pk_i, M_i) \to CT_i$, $\mathsf{TD}(sk_i) \to td_i$, $\mathsf{E}(pk_j, M_j) \to CT_j$, and $\mathsf{TD}(sk_j) \to td_j$.

The first condition guarantees the correctness of both encryption and decryption ensuring that ciphertexts are accurately decrypted to their original messages, as expected in any PKE scheme. The second and third conditions establish the correctness and reliability of authorization and equality test ensuring that equality tests yield accurate results.

3) ADVERSARY MODELS FOR OUR PKE-ET

Since an adversary may, in certain scenarios, obtain a trapdoor for performing equality tests, we categorize adversaries into two distinct types based on whether they have access to the target user's trapdoor.

- Type-1 adversary: This class of adversaries is assumed to have access to the trapdoor associated with all ciphertexts belonging to the target user. Since this class of adversaries can utilize the trapdoor for the challenge ciphertext to conduct equality tests, we focus on a one-wayness-based security notion in this case. The primary objective of a Type-1 adversary is to recover the challenge message from the challenge ciphertext.
- Type-2 adversary: This class of adversaries is assumed to have no access to any trapdoor associated with ciphertexts belonging to the target user. Since this class of adversaries aligns with those considered in traditional PKE security models, we adopt an indistinguishability-based security notion in this setting. The primary objective of a Type-2 adversary is to distinguish between two candidate messages within the challenge ciphertext.

First, we provide a formal definition of OW-CCA2 security in the context of a Type-1 adversary.

Definition 4 (OW-CCA2 Security for a Type-1 Adversary): A PKE-ET scheme PKE-ET *achieves OW-CCA2 security for any Type-1 adversary if $\mathcal{A}_1$'s success probability in the security game below against the challenger $\mathcal{C}$ is negligible in κ for any PPT adversary $\mathcal{A}_1$: Let t be the index of the target user in the game.*

1) **Setup**: *$\mathcal{C}$ executes $\mathsf{S}(1^\kappa) \to pp$ and generates key pairs for all users by running $\mathsf{KG}(pp) \to (pk_i, sk_i)$ for $i \in [1, N]$. $\mathcal{C}$ then provides $\mathcal{A}_1$ with the set of public keys $\{pk_i\}_{i\in[1,N]}$.*
2) **Phase 1**: *$\mathcal{A}_1$ may invoke the following queries polynomially many times in any sequence*:
   - *$\mathcal{O}^{\mathsf{KG}}$: Given an index $i \neq t$ corresponding to a user U_i, it returns the secret key sk_i.*
   - *$\mathcal{O}^{\mathsf{TD}}$: Given an index i for a user U_i, it computes $\mathsf{TD}(sk_i) \to td_i$ and returns the trapdoor td_i.*
   - *$\mathcal{O}^{\mathsf{D}}$: Given a pair of user index and ciphertext (i, CT_i), it computes $\mathsf{D}(sk_i, CT_i) \to M_i$ and returns the decrypted message M_i.*
3) **Challenge**: *$\mathcal{C}$ picks a challenge message M^* and generates the challenge cipehrtext by computing $\mathsf{E}(pk_t, M^*) \to CT^*$. $\mathcal{C}$ provides $\mathcal{A}_1$ with CT^*.*
4) **Phase 2**: *This follows the same process as* **Phase 1**, *with the following constraints*:
   - *$\mathcal{A}_1$ cannot query t to $\mathcal{O}^{\mathsf{KG}}$.*
   - *$\mathcal{A}_1$ cannot submit CT^* with index t to $\mathcal{O}^{\mathsf{D}}$.*
5) **Guess**: *$\mathcal{A}_1$ returns M'.*

$\mathcal{A}_1$'s success probability in the game above is given by

$$\mathsf{Adv}^{\mathrm{OW-CCA2}}_{\mathcal{A}_1,\mathrm{PKE-ET}}(\kappa) = \Pr[M^* = M'].$$

We emphasize the necessity of restricting the message space in PKE-ET schemes to prevent a trivial attack by a Type-1 adversary. Since Type-1 adversaries have access to the trapdoor, allowing them to conduct equality tests on all ciphertexts belonging to the target user, they can effectively retrieve the challenge message within the challenge ciphertext by encrypting all possible messages and comparing them via equality tests. To avoid this type of attacks, we assume throughout this paper that the message space has an exponentially large size relative to the security parameter. Furthermore, we impose the requirement that the min-entropy of the message distribution be sufficiently large–specifically, exceeding κ–ensuring that exhaustive message enumeration remains computationally infeasible.

Next, we formally define IND-CCA2 security in the context of a Type-2 adversary.

Definition 5 (IND-CCA2 Security for a Type-2 adversary): A PKE-ET scheme PKE-ET *achieves IND-CCA2 security for any Type-2 adversary if $\mathcal{A}_2$'s advantage in the security game below against the challenger $\mathcal{C}$ is negligible in κ for any PPT adversary $\mathcal{A}_2$: Let t be the index of the target user in the game.*

1) **Setup** : *This follows the same process as the* **Setup** *phase in the security game for OW-CCA2 security defined in Definition 4.*
2) **Phase 1** : *$\mathcal{A}_2$ can request queries in the same manner as in* **Phase 1** *of the security game for OW-CCA2 security, with one additional restriction: $\mathcal{A}_2$ cannot query t to $\mathcal{O}^{\mathsf{TD}}$, in addition to being restricted from querying $\mathcal{O}^{\mathsf{KG}}$.*
3) **Challenge** : *$\mathcal{A}_2$ selects and sends two messages $M_0, M_1 \in \mathcal{M}$ to $\mathcal{C}$. $\mathcal{C}$ randomly picks $b \in \{0, 1\}$, encrypts M_b by executing $\mathsf{E}(pk_t, M_b) \to CT^*_b$, and forwards CT^*_b to $\mathcal{A}_2$.*
4) **Phase 2** : *This mirrors* **Phase 1**, *with the additional restriction that $\mathcal{A}_2$ cannot submit CT^*_b with index t to $\mathcal{O}^{\mathsf{D}}$.*
5) **Guess** : *$\mathcal{A}_2$ returns b'.*

TABLE 1. Main algorithms of our proposed generic PKE-ET construction.

Encryption $\mathsf{E}(pk, M)$	Decryption $\mathsf{D}(sk, CT)$	Test $\mathsf{ET}(td_i, CT_j, td_j, CT_j)$
$r_1, r_2 \xleftarrow{\$} \mathcal{M}_1$	$\mathsf{PKE.D}(sk_1, C_1) \rightarrow r'_1$	$\mathsf{PKE.D}(td_i, C_{i,2}) \rightarrow r'_{i,2}$
$\mathsf{PKE.E}(pk_1, r_1) \rightarrow C_1$	$\mathsf{PKE.D}(sk_2, C_2) \rightarrow r'_2$	$\mathsf{PKE.D}(td_j, C_{j,2}) \rightarrow r'_{j,2}$
$\mathsf{PKE.E}(pk_2, r_2) \rightarrow C_2$	$M' = C_3 \oplus \mathcal{H}_1(r'_1, r'_2)$	$L = C_{i,4} \oplus \mathcal{H}_3(r'_{i,2}, C_{i,1}, C_{i,2}, C_{i,3})$
$C_3 = M \oplus \mathcal{H}_1(r_1, r_2)$	Return M if $C_4 = \mathcal{H}_2(M') \oplus \mathcal{H}_3(r'_2, C_1, C_2, C_3)$	$R = C_{j,4} \oplus \mathcal{H}_3(r'_{j,2}, C_{j,1}, C_{j,2}, C_{j,3})$
$C_4 = \mathcal{H}_2(M) \oplus \mathcal{H}_3(r_2, C_1, C_2, C_3)$	$\perp$ otherwise	Return 1 if $L = R$
Return $CT = (C_1, C_2, C_3, C_4)$		0 otherwise

PKE is the underlying OW-CPA secure PKE scheme consisting of key generation PKE.KG, encryption PKE.E, and decryption PKE.D. $\mathcal{M}_1$ is the message space for PKE. $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$ are cryptographic hash functions.

$\mathcal{A}_2$'s advantage in distinguishing the encrypted message in the above game is given by

$$\mathsf{Adv}^{\mathrm{IND-CCA2}}_{\mathcal{A}_2,\mathrm{PKE-ET}}(\kappa) = \left|\Pr[b = b'] - \frac{1}{2}\right|.$$

III. OUR IMPROVED GENERIC CONSTRUCTION

We now provide a novel generic construction for PKE-ET schemes. Next, we examine its correctness and conduct a comparative analysis against existing (semi-)generic constructions for PKE-ET. A comprehensive examination of the security properties of the proposed construction will be presented in the subsequent section.

A. DESCRIPTION OF OUR IMPROVED GENERIC CONSTRUCTION

We first provide a formal description of our proposed generic construction below. The main algorithms for encryption, decryption, test in our scheme are also given in Table 1.

Our PKE-ET scheme is composed of six polynomial-time algorithms, S, KG, E, D, TD, and ET.

- $\mathsf{S}(1^\kappa)$: Given κ,
  1) Generate a PKE scheme $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.E}, \mathsf{PKE.D})$.
  2) Generate three cryptographic hash functions:
     – $\mathcal{H}_1 : \mathcal{M}_1 \times \mathcal{M}_1 \rightarrow \mathcal{M}_2$,
     – $\mathcal{H}_2 : \mathcal{M}_2 \rightarrow \{0,1\}^{2\kappa}$, and
     – $\mathcal{H}_3 : \{0,1\}^* \rightarrow \{0,1\}^{2\kappa}$,
     where $\mathcal{M}_1$, $\mathcal{M}_2$ represent the message spaces of PKE, PKE-ET, respectively.
  3) Output a public parameter $pp = (\kappa, \mathsf{PKE}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{M}_1, \mathcal{M}_2)$.
- $\mathsf{KG}(pp)$: On input pp,
  1) Run $\mathsf{PKE.KG}(1^\kappa) \rightarrow (pk_1, sk_1)$ and $\mathsf{PKE.KG}(1^\kappa) \rightarrow (pk_2, sk_2)$.
  2) Output a public key $pk = (pk_1, pk_2)$ and a secret key $sk = (sk_1, sk_2)$.
- $\mathsf{E}(pk, M)$: On input $pk = (pk_1, pk_2)$ and a message $M \in \mathcal{M}_2$,
  1) Sample two random variables $r_1, r_2 \xleftarrow{\$} \mathcal{M}_1$.
  2) Run $\mathsf{PKE.E}(pk_1, r_1) \rightarrow C_1$ and $\mathsf{PKE.E}(pk_2, r_2) \rightarrow C_2$.
  3) Compute $C_3 = M \oplus \mathcal{H}_1(r_1, r_2)$ and $C_4 = \mathcal{H}_2(M) \oplus \mathcal{H}_3(r_2, C_1, C_2, C_3)$.
  4) Output $CT = (C_1, C_2, C_3, C_4)$.
- $\mathsf{D}(sk, C)$: Given $sk = (sk_1, sk_2)$ and $CT = (C_1, C_2, C_3, C_4)$,
  1) Run $\mathsf{PKE.D}(sk_1, C_1) \rightarrow r'_1$ and $\mathsf{PKE.D}(sk_2, C_2) \rightarrow r'_2$.
  2) Compute $M' = C_3 \oplus \mathcal{H}_1(r'_1, r'_2)$.
  3) Check whether $C_4 = \mathcal{H}_2(M') \oplus \mathcal{H}_3(r'_2, C_1, C_2, C_3)$. If the condition is satisfied, output M'; otherwise, return $\perp$.
- $\mathsf{TD}(sk_i)$: Given $sk_i = (sk_{i,1}, sk_{i,2})$ for U_i, return a trapdoor $td_i = sk_{i,2}$.
- $\mathsf{ET}(td_i, CT_i, td_j, CT_j)$: Given a pair of td_i and $CT_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})$ for user U_i and a pair of td_j and $CT_j = (C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4})$ for user U_j,
  1) Execute $\mathsf{PKE.D}(td_i, C_{i,2}) \rightarrow r'_{i,2}$ and $\mathsf{PKE.D}(td_j, C_{j,2}) \rightarrow r'_{j,2}$.
  2) Compute $L = C_{i,4} \oplus \mathcal{H}_3(r'_{i,2}, C_{i,1}, C_{i,2}, C_{i,3})$ and $R = C_{j,4} \oplus \mathcal{H}_3(r'_{j,2}, C_{j,1}, C_{j,2}, C_{j,3})$.
  3) Check whether $L = R$. Output 1 if the condition is satisfied. Otherwise, output 0.

1) CORRECTNESS OF OUR GENERIC CONSTRUCTION

We examine the correctness of our proposed generic scheme.

Theorem 1: The generic PKE-ET construction PKE-ET, described in Section III-A, ensures correctness provided that the exploited encryption scheme PKE is correct and the exploited $\mathcal{H}_2$ is collision-resistant.

Proof: We first check the correctness of the decryption process. Let $CT = (C_1, C_2, C_3, C_4)$ be the ciphertext corresponding to the message M, generated by executing $\mathsf{E}(pk, M)$. Then, it takes the form of

$$C_1 = \mathsf{PKE.E}(pk_1, r_1), \quad C_2 = \mathsf{PKE.E}(pk_2, r_2)$$
$$C_3 = M \oplus \mathcal{H}_1(r_1, r_2), \quad C_4 = \mathcal{H}_2(M) \oplus \mathcal{H}_3(r_2, C_1, C_2, C_3),$$

where $r_1, r_2 \xleftarrow{\$} \mathcal{M}_1$ in the encryption process. In Step 1 of our decryption algorithm D, we obtain $r'_1 (= r_1)$ and $r'_2 (= r_2)$

TABLE 2. Characteristics comparison among all existing (semi-)generic schemes for PKE-ET.

	LLS+20 [12]	LLSW16 [30]	LSQ18 [10]	LLSW19 [11]	Ours
Type	G	SG	G	G	G
Security Model	SM	ROM	ROM	ROM	ROM
Underlying Primitives	- HIBE (IND-sID-CCA2) - HASH (OW&CR) - Signature (One-time SUF)	- PKE (IND-CCA2) - HASH (OW&CR)	- PKE (IND-CCA2) - HASH (OW&CR) - Lagrange interpolation	- PKE (IND-CCA2 & RE) - HASH (OW&CR)	- PKE (OW-CPA) - HASH (OW&CR)
Auxiliary Assumptions		CDH			

Legends: G: generic, SG: semi-generic, SM: standard model, ROM: random oracle model, RE: randomness extractability, HASH: hash function, OW: one-wayness, CR: collision-resistance, SUF: strong unforgeability

by running $\mathsf{PKE.D}(sk_1, C_1) \rightarrow r'_1$ and $\mathsf{PKE.D}(sk_2, C_2) \rightarrow r'_2$, respectively. Then, we obtain

$$\begin{aligned} M' &= C_3 \oplus \mathcal{H}_1(r'_1, r'_2) \\ &= M \oplus \mathcal{H}_1(r_1, r_2) \oplus \mathcal{H}_1(r'_1, r'_2) = M. \end{aligned}$$

Thus, we have correct $r_2 (= r'_2)$ and $M (= M')$, and it holds $C_4 = \mathcal{H}_2(M') \oplus \mathcal{H}_3(r'_2, C_1, C_2, C_3)$ as well. Hence, our encryption and decryption algorithms perform correctly.

We next demonstrate that our TD and ET algorithms function correctly. Let CT_i and CT_j be ciphertexts generated by running $\mathsf{E}(pk_i, M_i)$ and $\mathsf{E}(pk_j, M_j)$, respectively. Then, $CT_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})$ and $CT_j = (C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4})$ are structured as

$$\begin{aligned} C_{i,1} &= \mathsf{PKE.E}(pk_{i,1}, r_{i,1}), \\ C_{i,2} &= \mathsf{PKE.E}(pk_{i,2}, r_{i,2}), \\ C_{i,3} &= M_i \oplus \mathcal{H}_1(r_{i,1}, r_{i,2}), \\ C_{i,4} &= \mathcal{H}_2(M_i) \oplus \mathcal{H}_3(r_{i,2}, C_{i,1}, C_{i,2}, C_{i,3}) \end{aligned}$$

and

$$\begin{aligned} C_{j,1} &= \mathsf{PKE.E}(pk_{j,1}, r_{j,1}), \\ C_{j,2} &= \mathsf{PKE.E}(pk_{j,2}, r_{j,2}), \\ C_{j,3} &= M_j \oplus \mathcal{H}_1(r_{j,1}, r_{j,2}), \\ C_{j,4} &= \mathcal{H}_2(M_j) \oplus \mathcal{H}_3(r_{j,2}, C_{j,1}, C_{j,2}, C_{j,3}), \end{aligned}$$

respectively, where $r_{i,1}, r_{i,2}, r_{j,1}, r_{j,2} \xleftarrow{\$} \mathcal{M}_1$ in the encryption process. Let td_i and td_j be outcomes of $\mathsf{TD}(sk_i)$ and $\mathsf{TD}(sk_j)$ (i.e., $td_i = sk_{i,2}$ and $td_j = sk_{j,2}$), respectively. In the ET algorithm, we first run $\mathsf{PKE.D}(td_i, C_{i,2}) \rightarrow r'_{i,2} (= r_{i,2})$ and $\mathsf{PKE.D}(td_j, C_{j,2}) \rightarrow r'_{j,2} (= r_{j,2})$.

Next, we compute

$$\begin{aligned} L &= C_{i,4} \oplus \mathcal{H}_3(r'_{i,2}, C_{i,1}, C_{i,2}, C_{i,3}) \\ &= \mathcal{H}_2(M_i) \oplus \mathcal{H}_3(r_{i,2}, C_{i,1}, C_{i,2}, C_{i,3}) \\ &\quad \oplus \mathcal{H}_3(r'_{i,2}, C_{i,1}, C_{i,2}, C_{i,3}) \\ &= \mathcal{H}_2(M_i) \end{aligned}$$

and

$$\begin{aligned} R &= C_{j,4} \oplus \mathcal{H}_3(r'_{j,2}, C_{j,1}, C_{j,2}, C_{j,3}) \\ &= \mathcal{H}_2(M_j) \oplus \mathcal{H}_3(r_{j,2}, C_{j,1}, C_{j,2}, C_{j,3}) \\ &\quad \oplus \mathcal{H}_3(r'_{j,2}, C_{j,1}, C_{j,2}, C_{j,3}) \\ &= \mathcal{H}_2(M_j). \end{aligned}$$

Consequently, assuming that $\mathcal{H}_2$ is a collision-resistant hash function, the ET algorithm returns 1 correctly when $M_i = M_j$ and 0 with overwhelming probability when $M_i \neq M_j$. Thus, the TD and ET algorithms, which facilitate equality tests, function correctly as intended. ■

B. CHARACTERISTICS COMPARISON OF OUR CONSTRUCTION AND EXISTING (SEMI-)GENERIC SCHEMES

We compare our proposed construction with other existing (semi-)generic PKE-ET approaches introduced in prior works [10], [11], [12], [30]. Table 2 highlights the key features of LLS+20 [12], LLSW16 [30], LSQ18 [10], LLSW19 [11], and our proposed scheme. All constructions, except for LLS+20 [12], operate under the random oracle model, whereas LLS+20 is designed in the standard model.

Each scheme encrypts both the message and its hash digest but employs distinct techniques to achieve CCA2 security. Specifically, LLSW16 [30] utilizes an IND-CCA2 secure PKE scheme and appends parts of a CDH instance to the message and its hash digest during encryption. Consequently, its security relies on the computational intractability of the CDH problem. LSQ18 [10] also relies on an IND-CCA2 secure PKE scheme but replaces CDH instances with random points on a line, which are determined by the message and its hash digest. LSQ18 further employs Lagrange interpolation to establish the relation between the embedded points and the encrypted message. LLSW19 [11] eliminates the need to append additional auxiliary information. However, it requires the underlying PKE scheme to support both IND-CCA2 security and randomness extractability–a property ensuring

that the randomness used in encryption can be recovered during decryption. Conversely, LLS+20 [12], which operates in the standard model, employs advanced cryptographic primitives that provide stronger security guarantees. More precisely, it relies on an IND-sID-CCA2 secure HIBE scheme along with a strongly unforgeable signature scheme. Finally, our proposed construction takes a different approach by leveraging a PKE scheme with weaker security assumptions, particularly an OW-CPA secure PKE scheme. This design choice opens up new possibilities for constructing more efficient PKE-ET instantiations, potentially outperforming existing approaches in terms of both computational and communicational efficiency.

IV. SECURITY ANALYSIS

We now provide a rigorous security analysis for our proposed generic construction. Specifically, we establish that the construction introduced in Section III-A achieves OW-CCA2 security for any Type-1 adversary–who has access to trapdoors for the target user–under the random oracle model, provided that the underlying PKE scheme satisfies OW-CPA security. Additionally, we prove that our proposed construction attains IND-CCA2 security for any Type-2 adversary, who lacks access to trapdoors for the target user, under the same security assumption.

A. OW-CCA2 SECURITY OF OUR PROPOSED GENERIC CONSTRUCTION FOR ANY TYPE-1 ADVERSARY

We begin by analyzing OW-CCA2 security against a Type-1 adversary, who has access to the trapdoor associated with the target user.

Theorem 2: [OW-CCA2 Security of Our Generic Construction] Our generic PKE-ET scheme PKE-ET, described in Section III-A, achieves OW-CCA2 security for any Type-1 adversary in the random oracle model, provided that the exploited PKE scheme PKE satisfies OW-CPA security. Specifically, suppose an adversary $\mathcal{A}_1$ can compromise OW-CCA2 security of PKE-ET with an advantage of $\varepsilon_{\mathcal{A}_1}(\kappa)$. Then, we can design an algorithm $\mathcal{B}_1$ that successfully compromises OW-CPA security of the exploited PKE scheme with an advantage of at least $\frac{1}{q_{\mathcal{H}_1}}\varepsilon_{\mathcal{A}_1}(\kappa)$ by leveraging interaction with $\mathcal{A}_1$, where $q_{\mathcal{H}_1}$ represents the number of hash queries made for $\mathcal{H}_1$.

Proof: We design an algorithm $\mathcal{B}_1$ that compromises OW-CPA security of the exploited PKE scheme PKE by interacting with $\mathcal{A}_1$, which is a Type-1 adversary capable of breaking OW-CCA2 security of our proposed construction with an advantage of $\varepsilon_{\mathcal{A}_1}(\kappa)$. Let $\mathcal{C}$ represent the challenger in the OW-CPA security game for breaking OW-CPA security of the exploited PKE scheme PKE.

Setup: $\mathcal{C}$ executes $\mathsf{PKE.KG}(1^\kappa) \rightarrow (pk^*, sk^*)$ and passes pk^* to $\mathcal{B}_1$. Upon receiving pk^*, $\mathcal{B}_1$ runs $\mathsf{PKE.KG}(1^\kappa) \rightarrow (pk', sk')$ and sets $pk_t = (pk_{t,1}, pk_{t,2}) = (pk^*, pk')$ and $sk_t = (sk_{t,1}, sk_{t,2}) = (\varnothing, sk')$ for the target user U_t, where $\varnothing$ denotes an empty string. Then, for $i \in [1, N]\backslash\{t\}$, $\mathcal{B}_1$ executes $\mathsf{KG}(1^\kappa) \rightarrow (pk_i, sk_i)$. Finally, $\mathcal{B}_1$ sends $\{pk_i\}_{i\in[1,N]}$ to $\mathcal{A}_1$.

Phase 1: $\mathcal{A}_1$ is permitted to invoke the following queries a polynomial number of times in any order.

- $\mathcal{O}^{\mathcal{H}_1}$ **queries:** The table composed of tuples $\langle r_{i,1}, r_{i,2}, H_{i,1}\rangle$ is denoted by H1Table, which is first assigned an empty set. When a query to $\mathcal{O}^{\mathcal{H}_1}$ is made for $(r_{i,1}, r_{i,2})$, $\mathcal{B}_1$ reacts as follows.
  - If the pair of $r_{i,1}$ and $r_{i,2}$ already exists in H1Table, $\mathcal{B}_1$ returns the corresponding value $H_{i,1} \in \mathcal{M}_2$.
  - Otherwise, $\mathcal{B}_1$ samples $H_{i,1}$ from $\mathcal{M}_2$ uniformly at random and stores a tuple $\langle r_{i,1}, r_{i,2}, H_{i,1}\rangle$ at H1Table. Then, $\mathcal{B}_1$ returns $H_{i,1}$ to $\mathcal{A}_1$.
- $\mathcal{O}^{\mathcal{H}_2}$ **queries:** The table composed of tuples $\langle M_i, H_{i,2}\rangle$ is denoted by H2Table, which is first assigned an empty set. When a query to $\mathcal{O}^{\mathcal{H}_2}$ is made for M_i, $\mathcal{B}_1$ reacts as follows.
  - If M_i already exists in H2Table, $\mathcal{B}_1$ returns the corresponding value $H_{i,2} \in \{0,1\}^{2\kappa}$.
  - Otherwise, $\mathcal{B}_1$ samples $H_{i,2} \in \{0,1\}^{2\kappa}$ uniformly at random and stores a tuple $\langle M_i, H_{i,2}\rangle$ at H2Table. Then, $\mathcal{B}_1$ returns $H_{i,2}$ to $\mathcal{A}_1$.
- $\mathcal{O}^{\mathcal{H}_3}$ **queries:** The table composed of tuples $\langle r_{i,2}, C_{i,1}, C_{i,2}, C_{i,3}, H_{i,3}\rangle$ is denoted by H3Table, which is set to an empty set initially. When a query to $\mathcal{O}^{\mathcal{H}_3}$ is made for $(r_{i,2}, C_{i,1}, C_{i,2}, C_{i,3})$, $\mathcal{B}_1$ reacts as stated below.
  - If a tuple of $(r_{i,2}, C_{i,1}, C_{i,2}, C_{i,3})$ already exists in H3Table, $\mathcal{B}_1$ returns the corresponding $H_{i,3} \in \{0,1\}^{2\kappa}$.
  - Otherwise, $\mathcal{B}_1$ samples $H_{i,3} \in \{0,1\}^{2\kappa}$ uniformly at random and stores a tuple $\langle r_{i,2}, C_{i,1}, C_{i,2}, C_{i,3}, H_{i,3}\rangle$ at H3Table. Then, $\mathcal{B}_1$ returns $H_{i,3}$ to $\mathcal{A}_1$.
- $\mathcal{O}^{\mathsf{KG}}$ **queries:** Upon receiving an index $i \neq t$ for user U_i, $\mathcal{B}_1$ returns sk_i to $\mathcal{A}_1$.
- $\mathcal{O}^{\mathsf{Dec}}$ **queries:** Upon receiving a decryption query on a pair of user index and ciphertext, $(i, CT_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}))$, it performs as follows.
  - If $i = t$, $\mathcal{B}_1$ first executes $\mathsf{PKE.D}(sk', C_{i,2}) \rightarrow r_{i,2}$ and performs as stated below:
    1) Find a tuple $\langle r_{i,1}, r_{i,2}, H_{i,1}\rangle$ corresponding to $r_{i,2}$ in H1Table.
    2) Compute $M_i = C_{i,3} \oplus H_{i,1}$.
    3) Check whether $C_{i,4} = \mathcal{H}_2(M_i) \oplus \mathcal{H}_3(r_{i,2}, C_{i,1}, C_{i,2}, C_{i,3})$.
    4) If the verification in Step 3 is satisfied, return M_i to $\mathcal{A}_1$. Otherwise, repeat the process from Step 1, iterating over the remaining tuples.
  - Otherwise, if $i \neq t$, $\mathcal{B}_1$ computes M_i by executing $\mathsf{D}(sk_i, CT_i)$ and outputs the decrypted message M_i to $\mathcal{A}_1$.

- $\mathcal{O}^{\mathsf{Auth}}$ **queries:** After receiving an authorization query on index i for user U_i, $\mathcal{B}_1$ returns $td_i = sk_{i,2}$ to $\mathcal{A}_1$.

Challenge: $\mathcal{C}$ samples a challenge message $M^* \xleftarrow{\$} \mathcal{M}_1$, executes $\mathsf{PKE.E}(pk^*, M^*) \rightarrow C^*$, and forwards C^* to $\mathcal{B}_1$ in the OW-CPA security game of PKE. After receiving C^*, $\mathcal{B}_1$ constructs $CT^* = (C_1^*, C_2^*, C_3^*, C_4^*)$ for the OW-CCA2 security game of PKE-ET by computing

$$\begin{aligned} C_1^* &= C^*, \\ C_2^* &= \mathsf{PKE.E}(pk', r_2), \\ C_3^* &= M_{ch} \oplus R, \text{ and} \\ C_4^* &= \mathcal{H}_2(M_{ch}) \oplus H_3(r_2, C_1^*, C_2^*, C_3^*), \end{aligned}$$

where $r_2 \xleftarrow{\$} \mathcal{M}_1$ and $M_{ch}, R \xleftarrow{\$} \mathcal{M}_2$.

Phase 2: $\mathcal{A}_1$ can invoke queries in the same manner as **Phase 1**, except that decryption queries involving the pair (t, CT^*) are not permitted.

Guess: $\mathcal{A}_1$ returns $M' \in \mathcal{M}_2$. Depending on M',

- If $M_{ch} = M'$, $\mathcal{B}_1$ selects a tuple $\langle r_{i,1}, r_{i,2}, H_{i,1}\rangle$ from H1Table such that $r_{i,2} = r_2$, where r_2 is sampled in the challenge phase. Then, $\mathcal{B}_1$ returns the corresponding $r_{i,1}$ to $\mathcal{C}$.
- Otherwise, i.e., if $M_{ch} \neq M'$, $\mathcal{B}_1$ uniformly samples a random element from $\mathcal{M}_1$ and returns it.

We first analyze the correctness of $\mathcal{B}_1$'s behavior by focusing on its responses to decryption queries. Verifying $\mathcal{B}_1$'s handling of decryption queries related to the target user is sufficient because for queries associated with other users, $\mathcal{B}_1$ can directly decrypt ciphertexts by executing the decryption algorithm $\mathsf{D}(sk_i, CT_i)$ utilizing the corresponding secret key sk_i. To handle decryption queries for the target user, once the pair of the user index and the ciphertext, $(t, CT_t = (C_{t,1}, C_{t,2}, C_{t,3}, C_{t,4}))$ is requested, $\mathcal{B}_1$ obtains $r_{t,2}$ by running $\mathsf{PKE.D}(sk_{t,2}, C_{t,2})$ where $sk_{t,2}$ was generated by $\mathcal{B}_1$ in the setup phase. Then, $\mathcal{B}_1$ searches for a tuple $\langle r_{j,1}, r_{j,2}, H_{j,1}\rangle$ in H1Table such that $r_{j,2} = r_{t,2}$. If $H_{t,1} = H_{j,1}$, then the correct message M_t can be obtained by computing

$$M_t = C_{t,3} \oplus H_{j,1} = M_t \oplus H_{t,1} \oplus H_{j,1} = M_t.$$

This message M_t is verified by checking whether

$$C_{t,4} = \mathcal{H}_2(M_t) \oplus \mathcal{H}_3(r_{t,2}, C_{t,1}, C_{t,2}, C_{t,3})$$

holds. If the verification fails, $\mathcal{B}_1$ selects other tuples $\langle r_{j,1}, r_{j,2}, H_{j,1}\rangle$ such that $r_{j,2} = r_{t,2}$ and continues the process to retrieve the correct message M_t. $H_{t,1}$, which was used to generate the ciphertext CT_t, should be stored in H1Table because the result of $\mathcal{H}_1$ can only be obtained through $\mathcal{O}^{\mathcal{H}_1}$ queries. The above confirms that $\mathcal{B}_1$'s behavior is correct.

Next, we calculate the advantage of $\mathcal{B}_1$. C_1^* is set to $C^* = \mathsf{PKE.E}(pk^*, M^*)$ in the challenge phase. Thus, if $\mathcal{A}_1$ successfully compromises OW-CCA2 security of our proposed construction, it implies the existence of a tuple $\langle r_{j,1}, r_{j,2}, H_{j,1}\rangle$ such that $r_{j,1} = M^*$. Consequently, $\mathcal{B}_1$ will defeat OW-CPA security of the exploited scheme PKE if $\mathcal{B}_1$ selects such an $r_{j,1}$ correctly. Let Break be the event in which $\mathcal{B}_1$ breaks OW-CPA security of the exploited scheme PKE. Then,

$$\begin{aligned} \mathsf{Adv}^{\mathrm{OW-CPA}}_{\mathcal{B}_1,\mathsf{PKE}}(\kappa) &= \Pr[\mathsf{Break}] \\ &= \Pr[\mathsf{Break}|M_{ch} = M']\Pr[M_{ch} = M'] \\ &\quad + \Pr[\mathsf{Break}|M_{ch} \neq M']\Pr[M_{ch} \neq M'] \\ &\geq \Pr[\mathsf{Break}|M_{ch} = M']\Pr[M_{ch} = M'] \\ &\geq \frac{1}{q_{\mathcal{H}_1}}\varepsilon_{\mathcal{A}_1}(\kappa) \end{aligned}$$

where $q_{\mathcal{H}_1}$ represents the number of queries requested to $\mathcal{O}^{\mathcal{H}_1}$. The last inequality holds because we assume that $\mathcal{B}_1$ selects the corresponding $r_{j,1}$ uniformly at random from H1Table. Consequently, if $\mathcal{A}_1$ can compromise OW-CCA2 security of our proposed generic construction with non-negligible probability, $\mathcal{B}_1$ can violate OW-CPA security of the underlying PKE scheme with non-negligible probability as well. Therefore, under assuming that the exploited PKE scheme PKE is OW-CPA secure, our proposed PKE-ET construction PKE-ET is proven to satisfy OW-CCA2 security for any Type-1 adversary. ■

B. IND-CCA2 SECURITY OF OUR PROPOSED CONSTRUCTION FOR ANY TYPE-2 ADVERSARY

Next, we investigate IND-CCA2 security for a Type-2 adversary who lacks a target user's trapdoor.

Theorem 3 (IND-CCA2 Security of Our Proposed Construction): Our PKE-ET scheme PKE-ET, as described in Section III-A, achieves IND-CCA2 security for any Type-2 adversary in the random oracle model, provided that the exploited PKE scheme PKE satisfies OW-CPA security. Specifically, suppose $\mathcal{A}_2$ can compromise IND-CCA2 security of PKE-ET with an advantage of $\varepsilon_{\mathcal{A}_2}(\kappa)$. Then, we can design an algorithm $\mathcal{B}_2$ that violates OW-CPA security of the exploited PKE scheme PKE with an advantage of at least $\frac{1}{q_{\mathcal{H}_1}}\varepsilon_{\mathcal{A}_2}(\kappa)$ by interacting with $\mathcal{A}_2$ where $q_{\mathcal{H}_1}$ indicates the number of hash queries made for $\mathcal{H}_1$.

Proof: We design an algorithm $\mathcal{B}_2$ that violates OW-CPA security of the exploited PKE scheme PKE by interacting with $\mathcal{A}_2$, which is a Type-2 adversary who compromises IND-CCA2 security of our generic scheme with an advantage of $\varepsilon_{\mathcal{A}_2}(\kappa)$. Let $\mathcal{C}$ represent the challenger in the subsequent security game for breaking OW-CPA security of PKE.

Setup: $\mathcal{C}$ runs $\mathsf{PKE.KG}(1^\kappa) \rightarrow (pk^*, sk^*)$ and forwards pk^* to $\mathcal{B}_2$. Upon receiving pk^*, $\mathcal{B}_2$ executes $\mathsf{PKE.KG}(1^\kappa) \rightarrow (pk', sk')$ and randomly selects β from $\{0, 1\}$ by flipping a fair coin. Then, $\mathcal{B}_2$ sets pk_t and sk_t as follows:

$$pk_t = \begin{cases} (pk^*, pk') & \text{if } \beta = 0 \\ (pk', pk^*) & \text{if } \beta = 1, \end{cases} \quad sk_t = \begin{cases} (\varnothing, sk') & \text{if } \beta = 0 \\ (sk', \varnothing) & \text{if } \beta = 1, \end{cases}$$

where $\varnothing$ denotes an empty string. Next, for $i \in [1, N]\backslash\{t\}$, $\mathcal{B}_2$ generates key pairs by executing $\mathsf{KG}(1^\kappa) \rightarrow (pk_i, sk_i)$. Finally, B_2 sends $\{pk_i\}_{i\in[1,N]}$ to $\mathcal{A}_2$.

Phase 1: $\mathcal{A}_2$ is allowed to invoke oracle queries in the same manner as described in **Phase 1** of Theorem 2, except for decryption and authorization queries. For these two queries, $\mathcal{B}_2$ handles them as follows:

- $\mathcal{O}^{\mathsf{Dec}}$ **queries:** Upon receiving a decryption query on $CT_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})$ with the user index i,
  - If $i = t$, $\mathcal{B}_2$ first executes $\mathsf{PKE.D}(sk', C_{i,2-\beta}) \rightarrow r_{i,2-\beta}$ and performs as follows:
    1) Search for a tuple $\langle r_{j,1}, r_{j,2}, H_{j,1}\rangle$ such that $r_{j,2-\beta} = r_{i,2-\beta}$ in $\mathsf{H1Table}$.
    2) Compute $M_i = C_{i,3} \oplus H_{j,1}$.
    3) Check whether $C_{i,4} = \mathcal{H}_2(M_i) \oplus \mathcal{H}_3(r_{j,2}, C_{i,1}, C_{i,2}, C_{i,3})$.
    4) If Step 3 passes, return M_i to $\mathcal{A}_2$. Otherwise, restart from Step 1 and repeat the process with the remaining tuples.
  - Otherwise, if $i \neq t$, $\mathcal{B}_2$ runs $\mathsf{D}(sk_i, CT_i) \rightarrow M_i$ and gives the decrypted message M_i to $\mathcal{A}_2$.
- $\mathcal{O}^{\mathsf{Auth}}$ **queries:** Upon receiving an authorization query on index i for user U_i, $\mathcal{B}_2$ returns the trapdoor $td_i = sk_{i,2}$ to $\mathcal{A}_2$. Querying this oracle with the index t is prohibited.

Challenge: $\mathcal{C}$ samples a challenge message $M^* \xleftarrow{\$} \mathcal{M}_1$, runs $\mathsf{PKE.E}(pk^*, M^*) \rightarrow C^*$, and sends C^* to $\mathcal{B}_2$. $\mathcal{A}_2$ chooses and submits two messages $M_0, M_1 \xleftarrow{\$} \mathcal{M}_2$ to $\mathcal{B}_2$. $\mathcal{B}_2$ passes $CT^*_{\beta,b} = (C^*_1, C^*_2, C^*_3, C^*_4)$ by computing

$$C^*_1 = \begin{cases} C^* & \text{if } \beta = 0 \\ \mathsf{PKE.E}(pk', r_1) & \text{if } \beta = 1, \end{cases}$$

$$C^*_2 = \begin{cases} \mathsf{PKE.E}(pk', r_2) & \text{if } \beta = 0 \\ C^* & \text{if } \beta = 1, \end{cases}$$

$$C^*_3 = M_b \oplus L,$$

$$C^*_4 = \mathcal{H}_2(M_b) \oplus R$$

where $r_1, r_2 \xleftarrow{\$} \mathcal{M}_1$, $L \xleftarrow{\$} \mathcal{M}_2$, $R \xleftarrow{\$} \{0,1\}^{2\kappa}$, and $b \xleftarrow{\$} \{0,1\}$ by $\mathcal{B}_2$.

Phase 2: $\mathcal{A}_2$ can invoke queries as **Phase 1**, with the exception that it is restricted from submitting $(t, CT^*_{\beta,b})$, as a decryption query.

Guess: $\mathcal{A}_2$ returns $b' \in \{0,1\}$. Depending on b',

- If $b = b'$, $\mathcal{B}_2$ searches for all tuples $\langle r_{i,1}, r_{i,2}, H_{i,1}\rangle$ from $\mathsf{H1Table}$ such that $r_{i,2-\beta} = r_{2-\beta}$, where $r_{2-\beta}$ is sampled in the challenge phase. Then, $\mathcal{B}_2$ selects a tuple randomly from the searched tuples and returns the corresponding $r_{i,1+\beta}$ to $\mathcal{C}$.
- Otherwise, if $b' \neq b$, $\mathcal{B}_2$ randomly picks a message from $\mathcal{M}_1$ and outputs it.

We now analyze the correctness of $\mathcal{B}_2$'s behavior. As in the proof of Theorem 2, it is sufficient to consider only the case of decryption queries related to the target user. When a decryption query is made for the target user's index t and ciphertext $CT_t = (C_{t,1}, C_{t,2}, C_{t,3}, C_{t,4})$, $\mathcal{B}_2$ retrieves $r_{t,2}$ if $\beta = 0$, and $r_{t,1}$ otherwise, by decrypting $C_{t,2-\beta} = \mathsf{PKE.E}(pk', r_{t,2-\beta})$. Then, $\mathcal{B}_2$ searches for a tuple $\langle r_{j,1}, r_{j,2}, H_{j,1}\rangle$ such that $r_{j,2-\beta} = r_{t,2-\beta}$ in $\mathsf{H1Table}$. If $H_{t,1} = H_{j,1}$, then the correct message M_i can be obtained by computing

$$M_t = C_{t,3} \oplus H_{j,1} = M_t \oplus H_{t,1} \oplus H_{j,1} = M_t.$$

This message is verified by checking whether $C_{t,4} = \mathcal{H}_2(M_i) \oplus \mathcal{H}_3(r_{j,2}, C_{t,1}, C_{t,2}, C_{t,3})$ holds. If the verification fails, $\mathcal{B}_2$ selects other tuples $\langle r_{j,1}, r_{j,2}, H_{j,1}\rangle$ such that $r_{j,2-\beta} = r_{t,2-\beta}$ in $\mathsf{H1Table}$ and continues the process to retrieve the correct message M_t. As in the proof of Theorem 2, $H_{t,1}$, which was used to generate the ciphertext CT_t, should be stored in $\mathsf{H1Table}$ because the result of $\mathcal{H}_1$ can only be obtained through $\mathcal{O}^{\mathcal{H}_1}$ queries. The above confirms that $\mathcal{B}_2$'s behavior is correct.

Next, we calculate the advantage of $\mathcal{B}_2$. $C_{1+\beta}$ is set to $C^* = \mathsf{PKE.E}(pk^*, M^*)$ in the challenge phase. Thus, if $\mathcal{A}_2$ successfully compromises IND-CCA2 security of the proposed construction, it implies the existence of a tuple $\langle r_{j,1}, r_{j,2}, H_{j,1}\rangle$ such that $r_{j,1+\beta} = M^*$. As a result, $\mathcal{B}_2$ will violate OW-CPA security of the exploited scheme PKE if it correctly identifies such an $r_{j,1+\beta} = M^*$. Let Break denote the event in which $\mathcal{B}_2$ breaks OW-CPA security of the exploited PKE scheme PKE. Then,

$$\begin{aligned} \mathsf{Adv}^{\mathrm{OW-CPA}}_{\mathcal{B}_2,\mathsf{PKE}}(\kappa) &= \Pr[\mathsf{Break}] \\ &= \Pr[\mathsf{Break}|b = b']\Pr[b = b'] \\ &\quad + \Pr[\mathsf{Break}|b \neq b']\Pr[b \neq b'] \\ &\geq \Pr[\mathsf{Break}|b = b']\Pr[b = b'] \\ &\geq \frac{1}{q_{\mathcal{H}_1}}\varepsilon_{\mathcal{A}_2}(\kappa) \end{aligned}$$

where $q_{\mathcal{H}_1}$ indicates the number of queries made for $\mathcal{O}^{\mathcal{H}_1}$. Thus, if $\mathcal{A}_2$ can compromise IND-CCA2 security of the proposed construction with non-negligible probability, $\mathcal{B}_2$ can violate OW-CPA security of the exploited PKE scheme PKE with non-negligible probability as well. Consequently, under the assumption that the exploited PKE scheme PKE is OW-CPA secure, our proposed generic PKE-ET construction $\mathsf{PKE\text{-}ET}$ is IND-CCA2 secure for any Type-2 adversary. ■

V. CONCRETE INSTANTIATIONS OF OUR GENERIC CONSTRUCTIONS

Now, we introduce three concrete instantiations of our generic approach, each based on an existing PKE scheme: The hashed ElGamal encryption scheme [34], the RSA encryption scheme [35], and the BIKE cryptosystem [36]. To our knowledge, the BIKE-based PKE-ET instantiation is the first construction that relies solely on code-based assumptions.

A. INSTANTIATION BASED ON THE HASHED ELGAMAL ENCRYPTION

We begin by introducing a concrete instantiation of our generic construction using the hashed ElGamal encryption

scheme [34]. Prior to that, we offer a concise summary of the hashed ElGamal encryption scheme.

1) HASHED ELGAMAL ENCRYPTION SCHEME

It comprises four polynomial-time algorithms, $\mathsf{HEIG.S}$, $\mathsf{HEIG.KG}$, $\mathsf{HEIG.E}$, and $\mathsf{HEIG.D}$. These algorithms are described as follows:

- $\mathsf{HEIG.S}(1^\kappa)$: On input κ,
  1) Construct a cyclic group $\mathbb{G}$ with a prime order $q = q(\kappa)$.
  2) Pick a random generator g of $\mathbb{G}$.
  3) Generate a cryptographic hash function $\mathcal{H} : \mathbb{G} \to \{0,1\}^{2\kappa}$.
  4) Output $\mathsf{PP_{HEIG}} = \{\kappa, \mathbb{G}, \mathcal{H}, q, g\}$ as a public parameter.
- $\mathsf{HEIG.KG}(\mathsf{PP_{HEIG}})$: Given $\mathsf{PP_{HEIG}} = \{\kappa, \mathbb{G}, \mathcal{H}, q, g\}$,
  1) Pick an element $x \xleftarrow{\$} \mathbb{Z}_q^*$ randomly and compute $X = g^x \in \mathbb{G}$.
  2) Return a public key $pk = X$ and a secret key $sk = x$.
- $\mathsf{HEIG.E}(pk, M)$: Given $pk = X$ and a message $M \in \{0,1\}^{2\kappa}$,
  1) Pick an element $r \xleftarrow{\$} \mathbb{Z}_q^*$ randomly.
  2) Compute $C_1 = g^r$ and $C_2 = M \oplus \mathcal{H}(X^r)$.
  3) Output $C = (C_1, C_2)$.
- $\mathsf{HEIG.D}(sk, C)$: Given $sk = x$ and $C = (C_1, C_2)$, calculate $M' = C_2 \oplus \mathcal{H}(C_1^x)$ and return M'.

Under the CDH assumption within the random oracle model, the hashed ElGamal encryption scheme provides OW-CPA security and even achieves IND-CPA security.

2) INSTANTIATION BASED ON THE HASHED ELGAMAL ENCRYPTION SCHEME

We provide a specific instantiation of our generic PKE-ET scheme based on the hashed ElGamal encryption scheme. This instantiation comprises the following six polynomial-time algorithms, S, KG, E, D, TD, and ET:

- $\mathsf{S}(1^\kappa)$: On input κ,
  1) Run $\mathsf{HEIG.S}(1^\kappa) \to \mathsf{PP_{HEIG}} = \{\kappa, \mathbb{G}, \mathcal{H}, q, g\}$.
  2) Generate three cryptographic hash functions:
     - $\mathcal{H}_1 : \{0,1\}^{2\kappa} \times \{0,1\}^{2\kappa} \to \{0,1\}^{2\kappa}$,
     - $\mathcal{H}_2 : \{0,1\}^{2\kappa} \to \{0,1\}^{2\kappa}$, and
     - $\mathcal{H}_3 : \{0,1\}^* \to \{0,1\}^{2\kappa}$.
  3) Output a public parameter $\mathsf{PP_{PKE\text{-}ET}} = \{\mathsf{PP_{HEIG}}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\}$.
- $\mathsf{KG}(\mathsf{PP_{PKE\text{-}ET}})$: Given $\mathsf{PP_{PKE\text{-}ET}} = \{\mathsf{PP_{HEIG}}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\}$,
  1) Run $\mathsf{HEIG.KG}(\mathsf{PP_{HEIG}}) \to (pk_1, sk_1)$ and $\mathsf{HEIG.KG}(\mathsf{PP_{HEIG}}) \to (pk_2, sk_2)$.
  2) Output a public key $pk = (pk_1, pk_2)$ and a secret key $sk = (sk_1, sk_2)$.
- $\mathsf{E}(pk, M)$: Given $pk = (pk_1, pk_2)$ and a message $M \in \{0,1\}^{2\kappa}$,
  1) Sample two strings $r_1, r_2 \xleftarrow{\$} \{0,1\}^{2\kappa}$.
  2) Execute $\mathsf{HEIG.E}(pk_1, r_1) \to C_1$ and $\mathsf{HEIG.E}(pk_2, r_2) \to C_2$.
  3) Compute $C_3 = M \oplus \mathcal{H}_1(r_1, r_2)$.
  4) Compute $C_4 = \mathcal{H}_2(M) \oplus \mathcal{H}_3(r_2, C_1, C_2, C_3)$.
  5) Output $CT = (C_1, C_2, C_3, C_4)$.
- $\mathsf{D}(sk, CT)$: Given $sk = (sk_1, sk_2)$ and $CT = (C_1, C_2, C_3, C_4)$,
  1) Execute $\mathsf{HEIG.D}(sk_1, C_1) \to r_1'$ and $\mathsf{HEIG.D}(sk_2, C_2) \to r_2'$.
  2) Calculate $M' = C_3 \oplus \mathcal{H}_1(r_1', r_2')$.
  3) Check if $C_4 = \mathcal{H}_2(M') \oplus \mathcal{H}_3(r_2', C_1, C_2, C_3)$. If it holds, return M'. Otherwise, return $\perp$.
- $\mathsf{TD}(sk_i)$: On input $sk_i = (sk_{i,1}, sk_{i,2})$ of U_i, it returns a trapdoor $td_i = sk_{i,2}$.
- $\mathsf{ET}(td_i, CT_i, td_j, CT_j)$: Given a pair of td_i and $CT_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})$ for U_i and a pair of td_j and $CT_j = (C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4})$ for U_j,
  1) Run $\mathsf{HEIG.D}(td_i, C_{i,2}) \to r_{i,2}'$ and $\mathsf{HEIG.D}(td_j, C_{j,2}) \to r_{j,2}'$.
  2) Compute $L = C_{i,4} \oplus \mathcal{H}_3(r_{i,2}', C_{i,1}, C_{i,2}, C_{i,3})$ and $R = C_{j,4} \oplus \mathcal{H}_3(r_{j,2}', C_{j,1}, C_{j,2}, C_{j,3})$.
  3) Check whether $L = R$. If the verification fails, return $\perp$. Otherwise, return 1.

The correctness and security of the above instantiation are directly derived from those of our generic PKE-ET construction and the hashed ElGamal encryption scheme.

3) EFFICIENCY COMPARISON

We compare the efficiency of our hashed ElGamal encryption-based instantiation with other specific discrete logarithm (DL)-based PKE-ET constructions [2], [10], [11] under Tang's AoN PKE-ET model [2] within the random oracle model. Table 3 presents computational and communication costs of Tang's PKE-ET scheme [2], three hashed ElGamal encryption-based instantiations of $\mathsf{LSQ18}$ [10], $\mathsf{LLSW19}$ [11], and our generic construction proposed in this paper. The generic constructions in [10] and [11] require an IND-CCA2 secure PKE scheme. In order to achieve this, we apply the Fujisaki-Okamoto (FO) transformation [37] to the hashed ElGamal encryption scheme, incorporating AES encryption [38] as a symmetric encryption scheme within the FO transformation. For consistency, both the ciphertext size of AES encryption and the hash digest length were set to 2κ each. Additionally, we set the message space for all PKE-ET schemes in Table 3 to $\{0,1\}^{2\kappa}$. This ensures that the message space has an exponentially large size in κ and that the min-entropy of the message distribution is at least κ, thereby preventing brute-force attacks. Furthermore, we exclude the costs of XOR operations and hash functions

TABLE 3. Comparison of existing DL-based PKE-ET constructions under the AoN model (Unit for size: Bits).

Type		[2]	[10]	[11]	Ours
Cost	Encryption	5Exp	4Exp + 6SE	4Exp + 2SE	4Exp
	Decryption	2Exp	2Exp + 6SE	2Exp + 2SE	2Exp
	Test	4Exp	2Exp + 6SE	2Exp + 2SE	2Exp
Size	Public Key	$2\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	$2\|\mathbb{G}\|$
	Ciphertext	$4\|\mathbb{G}\| + \|\mathbb{Z}_q\| + 2\kappa$	$2\|\mathbb{G}\| + 18\kappa$	$2\|\mathbb{G}\| + 10\kappa$	$2\|\mathbb{G}\| + 8\kappa$
	Trapdoor	$\|\mathbb{Z}_q\|$	$\|\mathbb{Z}_q\|$	$\|\mathbb{Z}_q\|$	$\|\mathbb{Z}_q\|$

Legends: Exp: computational cost for exponentiation, SE: computational cost for symmetric key encryption/decryption for a 2κ-bit message, $|\mathbb{G}|$: the number of bits needed to encode an element in the base group $\mathbb{G}$, $|\mathbb{Z}_q|$: the number of bits required to represent an element in $\mathbb{Z}_q$ where q denotes the order of $\mathbb{G}$

from the efficiency comparison, as these operations are much cheaper than other computations, such as exponentiation in the underlying group.

In the hashed ElGamal scheme, the encryption and decryption algorithms require 2 exponentiations and 1 exponentiation, respectively. A ciphertext in the hashed ElGamal scheme is composed of a single element from the base group $\mathbb{G}$ and a hash digest of length 2κ bits. Hence, its total size is $|\mathbb{G}| + 2\kappa$ bits, where $|\mathbb{G}|$ indicates the number of bits needed to represent an element in the base group $\mathbb{G}$.

Our encryption and decryption algorithms each perform two hashed ElGamal encryptions and decryptions, respectively. Consequently, our encryption algorithm requires a total of four exponentiations, while decryption requires two. Our test algorithm takes sk_2 as a trapdoor and checks the equality of two results obtained by decrypting the second components C_2 of two input ciphertexts, each using the corresponding trapdoor. This process requires 2 exponentiations to decrypt C_2 of two input ciphertexts. Finally, a ciphertext consists of two hashed ElGamal ciphertexts and two 2κ-bit strings, resulting in a total size of $2|\mathbb{G}| + 8\kappa$ bits.

As aforementioned, the generic constructions in [10], [11] use a PKE scheme that is IND-CCA2 secure, which we instantiate using the hashed ElGamal encryption combined with the FO transformation. Consequently, their encryption algorithms incorporate AES encryptions for both the message and its hash value. Notably, the message size for AES encryption in [10] is three times larger than that in [11] when the base group order is set to a 2κ-bit prime. Thus, the encryption algorithms in [10] and [11] require 6 and 2 AES encryptions, respectively. Similarly, the decryption algorithms in [10] and [11] require 6 and 2 AES decryptions, respectively. Their ciphertexts are composed of AES encryptions, 2 hashed ElGamal ciphertexts, and a 2κ-bit hash digest. Consequently, the ciphertext size is $2|\mathbb{G}| + 18\kappa$ bits in the instantiation of [10] and is $2|\mathbb{G}| + 10\kappa$ bits in the instantiation of [11].

We also provide computational and communication costs of Tang's PKE-ET scheme in the third column of the table. Our instantiation reduces the encryption and test algorithms by one and two exponentiations, respectively, compared to Tang's scheme. The difference in ciphertext sizes between Tang's construction and our instantiation is $2|\mathbb{G}| + |\mathbb{Z}_q| - 6\kappa$ bits, which depends on the size of the base group. When instantiated over elliptic curve groups where $|\mathbb{G}| = 4\kappa$ and $|\mathbb{Z}_q| = 2\kappa$ without point compression, our ciphertext is 4κ bits shorter than Tang's.

Finally, we summarize the efficiency comparisons of our instantiation with those of Tang, LSQ18, and LLSW19. In terms of computational costs, our instantiation has the best performance across all algorithms compared to the other constructions. Regarding communication costs, while the public key and trapdoor sizes remain identical across all constructions, our ciphertext size is the smallest. For instance, when instantiated over elliptic curve groups providing 128-bit security, $|\mathbb{G}| = 512$ without point compression and $|\mathbb{Z}_q| = 256$. The ciphertext sizes of ours, Tang's, LSQ18, and LLSW19 constructions are 2048, 2688, 3328, and 2304 bits, respectively. This indicates that our instantiation reduces the ciphertext size by at least 11.1% compared to existing constructions.

B. INSTANTIATION BASED ON THE RSA ENCRYPTION

This subsection provides an instantiation of our generic scheme with the RSA encryption scheme [35]. We start by offering an overview of the original RSA encryption scheme.

1) RSA ENCRYPTION SCHEME

The original RSA encryption scheme comprises the following three polynomial-time algorithms.

- $\mathsf{RSA.KG}(1^\kappa)$: Given κ,
  1) Select two distinct random primes $p = p(\kappa)$ and $q = q(\kappa)$.
  2) Calculate their product $n = pq$ and evaluate Euler's totient function $\phi(n) = (p-1)(q-1)$.
  3) Pick a public exponent $e \xleftarrow{\$} \mathbb{Z}_n^*$ randomly, ensuring that it satisfies $\gcd(e, \phi(n)) = 1$.
  4) Determine the corresponding secret exponent d such that $d \cdot e \equiv 1 \pmod{\phi(n)}$.
  5) Output the public key $pk = (n, e)$ and the secret key $sk = d$.

- $\mathsf{RSA.E}(pk, M)$: Given $pk = (n, e)$ and a message $M \in \mathbb{Z}_n$, calculate and output $C = M^e \bmod n$.
- $\mathsf{RSA.D}(sk, C)$: Given $sk = d$ and C, return the message M' by computing $M' = C^d \bmod n$.

The original RSA encryption scheme is OW-CPA secure under the RSA assumption.

2) INSTANTIATION BASED ON THE RSA ENCRYPTION SCHEME

Next, we present an instantiation of our generic PKE-ET construction based on the RSA encryption scheme. This instantiation comprises the following six polynomial-time algorithms, S, KG, E, D, TD, and ET:

- $\mathsf{S}(1^\kappa)$: On input κ,
  1) Generate three cryptographic hash functions
     - $\mathcal{H}_1 : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^{2\kappa}$,
     - $\mathcal{H}_2 : \{0,1\}^{2\kappa} \to \{0,1\}^{2\kappa}$, and
     - $\mathcal{H}_3 : \{0,1\}^* \to \{0,1\}^{2\kappa}$.
  2) Return the public parameter $\mathsf{PP} = \{\kappa, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\}$.
- $\mathsf{KG}(\mathsf{PP})$: On input $\mathsf{PP} = \{\kappa, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\}$,
  1) Run $\mathsf{RSA.KG}(1^\kappa) \to (pk_1, sk_1)$ and $\mathsf{RSA.KG}(1^\kappa) \to (pk_2, sk_2)$.
  2) Output a public key $pk = (pk_1, pk_2)$ and a secret key $sk = (sk_1, sk_2)$.
- $\mathsf{E}(pk, M)$: Given $pk = (pk_1, pk_2)$ and a message $M \in \{0,1\}^{2\kappa}$,
  1) Sample two random integers $r_1 \xleftarrow{\$} \mathbb{Z}_{n_1}$ and $r_2 \xleftarrow{\$} \mathbb{Z}_{n_2}$, where n_1 and n_2 are the RSA moduli included in pk_1 and pk_2, respectively.
  2) Run $\mathsf{RSA.E}(pk_1, r_1) \to C_1$ and $\mathsf{RSA.E}(pk_2, r_2) \to C_2$.
  3) Compute $C_3 = M \oplus \mathcal{H}_1(r_1, r_2)$.
  4) Compute $C_4 = \mathcal{H}_2(M) \oplus \mathcal{H}_3(r_2, C_1, C_2, C_3)$.
  5) Output $CT = (C_1, C_2, C_3, C_4)$.
- $\mathsf{D}(sk, CT)$: Given $sk = (sk_1, sk_2)$ and $CT = (C_1, C_2, C_3, C_4)$,
  1) Run $\mathsf{RSA.D}(sk_1, C_1) \to r'_1$ and $\mathsf{RSA.D}(sk_2, C_2) \to r'_2$.
  2) Compute $M' = C_3 \oplus \mathcal{H}_1(r'_1, r'_2)$.
  3) Check if $C_4 = \mathcal{H}_2(M') \oplus \mathcal{H}_3(r'_2, C_1, C_2, C_3)$. If it holds, return M'. Otherwise, return $\perp$.
- $\mathsf{TD}(sk_i)$: On input $sk_i = (sk_{i,1}, sk_{i,2})$ for user U_i, it returns a trapdoor $td_i = sk_{i,2}$.
- $\mathsf{ET}(td_i, CT_i, td_j, CT_j)$: Given a pair of td_i and $CT_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})$ for user U_i and a pair of td_j and $CT_j = (C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4})$ for user U_j,
  1) Execute $\mathsf{RSA.D}(td_i, C_{i,2}) \to r'_{i,2}$ and $\mathsf{RSA.D}(td_j, C_{j,2}) \to r'_{j,2}$.
  2) Compute $L = C_{i,4} \oplus \mathcal{H}_3(r'_{i,2}, C_{i,1}, C_{i,2}, C_{i,3})$ and $R = C_{j,4} \oplus \mathcal{H}_3(r'_{j,2}, C_{j,1}, C_{j,2}, C_{j,3})$.
  3) Check whether $L = R$. If it holds, return 1. Otherwise, return $\perp$.

Similar to the hashed ElGamal-based instantiation in Section V-A, the correctness and security of this instantiation follow directly from those of our generic PKE-ET construction and the RSA encryption scheme.

3) EFFICIENCY COMPARISON

We evaluate the efficiency of our RSA-based instantiation in comparison with the RSA-based PKE-ET scheme introduced by Zhu et al. [13]. Table 4 presents the efficiency metrics for both Zhu et al.'s scheme and ours. To generate the data in the table, the message space size for all PKE-ET schemes was set to 2κ bits. For simplicity, we assume $|\mathbb{Z}_{\phi(n)}| = |\mathbb{Z}_n|$, where $|\mathbb{Z}_{\phi(n)}|$ and $|\mathbb{Z}_n|$ indicate the bit lengths to represent elements in $\mathbb{Z}_{\phi(n)}$ and $\mathbb{Z}_n$, respectively. (This assumption generally holds for an RSA modulus n.) Additionally, it is assumed that all RSA moduli used in the system have the same bit length. When analyzing the computational costs of these schemes, we omit the cost of XOR operations and hash functions, since their computational costs are insignificant in comparison to that of modular exponentiation.

TABLE 4. Comparison of the RSA-based PKE-ET constructions (Unit for size: Bits).

Type		[13]	Ours
Cost	Encryption	2Exp	2Exp
	Decryption	2Exp	2Exp
	Test	2Exp	2Exp
Size	Public Key	$4\lvert\mathbb{Z}_n\rvert$	$4\lvert\mathbb{Z}_n\rvert$
	Ciphertext	$5\lvert\mathbb{Z}_n\rvert + 4\kappa$	$2\lvert\mathbb{Z}_n\rvert + 4\kappa$
	Trapdoor	$\lvert\mathbb{Z}_n\rvert$	$\lvert\mathbb{Z}_n\rvert$

Legends: n: an RSA modulus, $|\mathbb{Z}_n|$: the bit length needed to represent an element in $\mathbb{Z}_n$

In the RSA encryption scheme, both the encryption and decryption processes require performing a single exponentiation in $\mathbb{Z}_n$. The ciphertext, public key, and secret key sizes are all equal to $|\mathbb{Z}_n|$. (One may consider using small public or secret keys to improve efficiency; however, in this subsection, we adhere strictly to the original description of the RSA scheme for simple and fair comparison.)

Our encryption algorithm performs 2 RSA encryptions to generate C_1 and C_2, which requires a total of two exponentiations. Similarly, our decryption algorithm performs 2 RSA decryptions to decrypt C_1 and C_2, which also requires 2 exponentiations. Our test algorithm takes sk_2 as a trapdoor and checks the equality of two results obtained by decrypting the second components C_2 of two input ciphertexts, each using the corresponding trapdoor. This process requires two exponentiations to decrypt C_2 of two input ciphertexts. Finally, a ciphertext in our RSA-based instantiation consists of two RSA ciphertexts and two 2κ-bit strings, resulting in a total size of $2|\mathbb{Z}_n| + 4\kappa$ bits.

In Zhu et al.'s scheme [13], the components C_1, C_2 are the same as those in our construction, while C_3 and C_4 are replaced by

$$C_3 = (M \| r_2) \oplus \mathcal{H}_1(r_1) \quad \text{and} \quad C_4 = \mathcal{H}_2(M, r_1),$$

respectively, and a new component,

$$C_5 = (x_1 \| y_1) \oplus \mathcal{H}_3(r_2, C_1, C_2, C_3, C_4)$$

is included, where $r_1, r_2, x_1 \xleftarrow{\$} \mathbb{Z}_n$, $y_1 = f(x_1)$ for a linear polynomial f that passes through the origin and a point $p = (\mathcal{H}_4(M), \mathcal{H}_5(M))$. Here, $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4$, and $\mathcal{H}_5$ are appropriate cryptographic hash functions. The above algorithm has almost the same computational costs as ours after excluding the costs of XOR computations and hash functions. However, the sizes of C_3 and C_5 are $2\kappa + |\mathbb{Z}_n|$ bits and $2|\mathbb{Z}_n|$ bits, respectively. Thus, the ciphertext size is a total of $5|\mathbb{Z}_n| + 4\kappa$ bits, which is $3|\mathbb{Z}_n|$ bits larger than the size of our ciphertext. For example, when N is a 3072-bit RSA modulus to achieve 128-bit security, the ciphertext size of Zhu et al.'s construction is 15872 bits, whereas, in our instantiation, it is 6656 bits–approximately 42 % of the size of Zhu et al.'s ciphertext.

In summary, our RSA-based instantiation and Zhu et al.'s RSA-based PKE-ET construction have nearly the same computational costs. The public key and trapdoor sizes are identical in both schemes. However, our instantiation significantly outperforms Zhu et al.'s in terms of ciphertext size.

C. INSTANTIATION BASED ON BIKE CRYPTOSYSTEM

This subsection introduces an instantiation of our generic scheme with the code-based encryption scheme BIKE [36], which was selected as a fourth-round submission in the NIST post-quantum cryptography standardization process. As far as we know, this BIKE-based instantiation represents the first code-based PKE-ET construction whose security relies solely on code-based assumptions. We begin by providing an overview of the OW-CPA secure variant of the BIKE cryptosystem.

1) BIKE CRYPTOSYSTEM

The BIKE cryptosystem comprises the following four polynomial-time algorithms, BIKE.S, BIKE.KG, BIKE.E, and BIKE.D:

- $\mathsf{BIKE.S}(1^\kappa)$: Given κ,
  1) Define parameters, $r = r(\kappa)$ (block size), $w = w(\kappa)$ (row weight) and $t = t(\kappa)$ (error weight).
  2) Construct a cyclic polynomial ring $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$.
  3) Define the secret key space $\mathcal{HW}_w = \{(h_1, h_2) \in \mathcal{R}^2 | \mathsf{wt}(h_i) = w/2,\ i \in [1, 2]\}$ where wt represents the Hamming weight function, which returns the number of nonzero components in a given vector.
  4) Define the error space $\mathcal{E}_t = \{(e_1, e_2) \in \mathcal{R}^2 | \mathsf{wt}(e_1) + \mathsf{wt}(e_2) = t\}$. Note that $\mathcal{E}_t$ serves as the message space for this BIKE encryption scheme.
  5) Construct the decoding algorithm Decode (s, h_1, h_2), which takes a syndrome $s \in \mathcal{R}$ and parity polynomials $(h_1, h_2) \in \mathcal{HW}_w$ as inputs, and outputs a sparse vector $(e_1, e_2) \in \mathcal{R}^2$ satisfying the equation $e_1 h_1 + e_2 h_2 = s$.
  6) Output the public parameter $\mathsf{PP_{BIKE}} = \{\kappa, \mathcal{R}, \mathcal{HW}_w, \mathcal{E}_t\}$ along with the decoding algorithm Decode.
- $\mathsf{BIKE.KG}(\mathsf{PP_{BIKE}})$: Given $\mathsf{PP_{BIKE}} = \{\kappa, \mathcal{R}, \mathcal{HW}_w, \mathcal{E}_t\}$,
  1) Sample sparse parity polynomials $(h_1, h_2) \xleftarrow{\$} \mathcal{HW}_w$.
  2) Compute the public polynomial $h = h_2 h_1^{-1}$.
  3) Output the public key $pk = h$ and the secret key $sk = (h_1, h_2)$.
- $\mathsf{BIKE.E}(pk, M)$: Given $pk = h$ and a message $M = (m_1, m_2) \in \mathcal{E}_t$, calculate the ciphertext $C = m_1 + m_2 h$ and output the resulting ciphertext C.
- $\mathsf{BIKE.D}(sk, C)$: Given $sk = (h_1, h_2)$ and C,
  1) Compute the syndrome $s = C \cdot h_1$.
  2) Apply the decoding algorithm $\mathsf{Decode}(s, h_1, h_2) \to M' = (m'_1, m'_2)$.
  3) Output the recovered message M'.

The BIKE construction, as described, is OW-CPA secure under the hardness of the quasi-cyclic syndrome decoding (QCSD) problem and the quasi-cyclic codeword finding (QCCF) problem.

2) INSTANTIATION BASED ON THE BIKE CRYPTOSYSTEM

We present an instantiation of our generic construction utilizing the BIKE cryptosystem. This instantiation comprises the following six polynomial-time algorithms, S, KG, E, D, TD, and ET:

- $\mathsf{S}(1^\kappa)$: Given κ,
  1) Obtain $\mathsf{PP_{BIKE}} = \{\kappa, \mathcal{R}, \mathcal{HW}_w, \mathcal{E}_t\}$ and a decode algorithm Decode by executing $\mathsf{BIKE.S}(1^\kappa)$.
  2) Generate three cryptographic hash functions:
     - $\mathcal{H}_1 : \mathcal{E}_t \times \mathcal{E}_t \to \{0, 1\}^{2\kappa}$,
     - $\mathcal{H}_2 : \{0, 1\}^{2\kappa} \to \{0, 1\}^{2\kappa}$, and
     - $\mathcal{H}_3 : \{0, 1\}^* \to \{0, 1\}^{2\kappa}$.
  3) Output a public parameter $\mathsf{PP_{PKE\text{-}ET}} = \{\mathsf{PP_{BIKE}}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\}$.
- $\mathsf{KG}(\mathsf{PP_{PKE\text{-}ET}})$: Given $\mathsf{PP_{PKE\text{-}ET}} = \{\mathsf{PP_{BIKE}}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\}$,
  1) Run $\mathsf{BIKE.KG}(\mathsf{PP_{BIKE}}) \to (pk_1, sk_1)$ and $\mathsf{BIKE.KG}(\mathsf{PP_{BIKE}}) \to (pk_2, sk_2)$.
  2) Output a public key $pk = (pk_1, pk_2)$ and a secret key $sk = (sk_1, sk_2)$.

- $\mathsf{E}(pk, M)$: Given $pk = (pk_1, pk_2)$ and a message $M \in \{0,1\}^{2\kappa}$,
  1) Sample random values $r_1, r_2 \xleftarrow{\$} \mathcal{E}_t$.
  2) Execute $\mathsf{BIKE.E}(pk_1, r_1) \rightarrow C_1$ and $\mathsf{BIKE.E}(pk_2, r_2) \rightarrow C_2$.
  3) Compute $C_3 = M \oplus \mathcal{H}_1(r_1, r_2)$.
  4) Compute $C_4 = \mathcal{H}_2(M) \oplus \mathcal{H}_3(r_2, C_1, C_2, C_3)$.
  5) Return $CT = (C_1, C_2, C_3, C_4)$.
- $\mathsf{D}(sk, CT)$: Given $sk = (sk_1, sk_2)$ and $CT = (C_1, C_2, C_3, C_4)$,
  1) Execute $\mathsf{BIKE.D}(sk_1, C_1) \rightarrow r'_1$ and $\mathsf{BIKE.D}(sk_2, C_2) \rightarrow r'_2$.
  2) Compute $M' = C_3 \oplus \mathcal{H}_1(r'_1, r'_2)$.
  3) Check if $C_4 = \mathcal{H}_2(M') \oplus \mathcal{H}_3(r'_2, C_1, C_2, C_3)$. If it holds, return M'. Otherwise, return $\perp$.
- $\mathsf{TD}(sk_i)$: On input $sk_i = (sk_{i,1}, sk_{i,2})$ for U_i, return a trapdoor $td_i = sk_{i,2}$.
- $\mathsf{ET}(td_i, CT_i, td_j, CT_j)$: Given a pair of td_i and $CT_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})$ for user U_i and a pair of td_j and $CT_j = (C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4})$ for user U_j,
  1) Execute $\mathsf{BIKE.D}(td_i, C_{i,2}) \rightarrow r'_{i,2}$ and $\mathsf{BIKE.D}(td_j, C_{j,2}) \rightarrow r'_{j,2}$.
  2) Compute $L = C_{i,4} \oplus \mathcal{H}_3(r'_{i,2}, C_{i,1}, C_{i,2}, C_{i,3})$ and $R = C_{j,4} \oplus \mathcal{H}_3(r'_{j,2}, C_{j,1}, C_{j,2}, C_{j,3})$.
  3) Check whether $L = R$. If it holds, return 1. Otherwise, return $\perp$.

Similar to the aforementioned instantiations, the correctness and security of the BIKE-based instantiation are directly derived from those of our generic scheme and the BIKE scheme.

3) EFFICIENCY ANALYSIS

The OW-CPA secure BIKE cryptosystem, as described at the beginning of this subsection, requires a single addition and a single multiplication in the underlying cyclic polynomial ring $\mathcal{R}$ for encryption and a single execution of the decoding algorithm Decode and a single multiplication in $\mathcal{R}$ for decryption. The public key, secret key, and ciphertext are composed of one, two, and one element(s) in $\mathcal{R}$, respectively.

Our BIKE-based instantiation performs two BIKE encryptions for encryption and two BIKE decryptions for decryption. Thus, it requires two additions and two multiplications in $\mathcal{R}$ for encryption, and two executions of the decoding algorithm Decode and two multiplications in $\mathcal{R}$ for decryption. For the test algorithm, it also performs two BIKE decryptions, so it requires two executions of the decoding algorithm and two multiplications in $\mathcal{R}$. The public key and the trapdoor in our instantiation consist of two elements and a single element in $\mathcal{R}$, respectively. The ciphertext in our instantiation comprises two elements in $\mathcal{R}$ and two hash digests of 2κ-bit each.

As aforementioned, to our knowledge, the BIKE-based instantiation of our generic construction is the first code-based PKE-ET construction whose security relies solely on code-based assumptions. While one may consider obtaining an instantiation from other generic constructions, such as those proposed by LSQ18 [10] and LLSW19 [11], these instantiations would require additional cryptographic primitives, such as the Fujisaki-Okamoto (FO) transformation combined with other symmetric-key encryption schemes. Consequently, the derived schemes would exhibit lower efficiency than our BIKE-based instantiation.

VI. CONCLUSION

We introduced an enhanced generic construction for PKE-ET within the random oracle model. Unlike previous generic constructions, which rely on stronger cryptographic primitives or more complex security assumptions, our proposed construction is designed using a PKE scheme that satisfies only OW-CPA security. Through a rigorous security analysis, we proved that if the exploited PKE scheme is OW-CPA secure, the proposed construction achieves OW-CCA2 security for any Type-1 adversary who possesses a trapdoor for the target user, while it achieves IND-CCA2 security for any Type-2 adversary who lacks access to a trapdoor. Furthermore, we presented three concrete instantiations of our generic constructions using existing PKE schemes. Both the hashed ElGamal-based and RSA-based instantiations demonstrate notable efficiency improvements over existing PKE-ET constructions. As far as we know, the BIKE-instantiation represents the first code-based PKE-ET construction whose security relies solely on code-based assumptions.

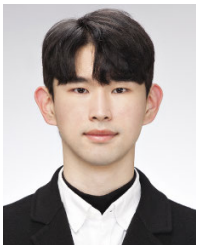
REFERENCES

[1] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong, "Probabilistic public key encryption with equality test," in *Proc. CT-RSA*. Cham, Switzerland: Springer, Jan. 2010, pp. 119–131.
[2] Q. Tang, "Public key encryption supporting plaintext equality test and user-specified authorization," *Secur. Commun. Netw.*, vol. 5, no. 12, pp. 1351–1362, Dec. 2012.
[3] K. Huang, R. Tso, Y.-C. Chen, S. M. M. Rahman, A. Almogren, and A. Alamri, "PKE-AET: Public key encryption with authorized equality test," *Comput. J.*, vol. 58, no. 10, pp. 2686–2697, Oct. 2015.
[4] S. Ma, Q. Huang, M. Zhang, and B. Yang, "Efficient public key encryption with equality test supporting flexible authorization," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 458–470, Mar. 2015.
[5] S. Ma, M. Zhang, Q. Huang, and B. Yang, "Public key encryption with delegated equality test in a multi-user setting," *Comput. J.*, vol. 58, no. 4, pp. 986–1002, Apr. 2015.
[6] S. Ma, "Identity-based encryption with outsourced equality test in cloud computing," *Inf. Sci.*, vol. 328, pp. 389–402, Jan. 2016.
[7] K. Zhang, J. Chen, H. T. Lee, H. Qian, and H. Wang, "Efficient public key encryption with equality test in the standard model," *Theor. Comput. Sci.*, vol. 755, pp. 65–80, Jan. 2019.
[8] W. Susilo, D. H. Duong, and H. Q. Le, "Efficient post-quantum identity-based encryption with equality test," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2020, pp. 633–640.
[9] D. H. Duong, P. S. Roy, W. Susilo, K. Fukushima, S. Kiyomoto, and A. Sipasseuth, "Chosen-ciphertext lattice-based public key encryption with equality test in standard model," *Theor. Comput. Sci.*, vol. 905, pp. 31–53, Feb. 2022.

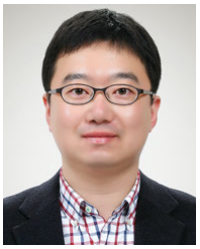
[10] X.-J. Lin, L. Sun, and H. Qu, "Generic construction of public key encryption, identity-based encryption and signcryption with equality test," *Inf. Sci.*, vol. 453, pp. 111–126, Jul. 2018.
[11] H. T. Lee, S. Ling, J. H. Seo, and H. Wang, "Public key encryption with equality test from generic assumptions in the random Oracle model," *Inf. Sci.*, vol. 500, pp. 15–33, Oct. 2019.
[12] H. T. Lee, S. Ling, J. H. Seo, H. Wang, and T.-Y. Youn, "Public key encryption with equality test in the standard model," *Inf. Sci.*, vol. 516, pp. 89–108, Apr. 2020.
[13] H. Zhu, D. Xie, H. Ahmad, and H. N. H. Abdullah, "New constructions of equality test scheme for cloud-assisted wireless sensor networks," *PLoS ONE*, vol. 16, no. 10, Oct. 2021, Art. no. e0258746.
[14] H. T. Lee, S. Ling, J. H. Seo, and H. Wang, "CCA2 attack and modification of Huang et al.'s public key encryption with authorized equality test," *Comput. J.*, vol. 59, no. 11, pp. 1689–1694, Nov. 2016.
[15] W. Susilo, F. Guo, Z. Zhao, and G. Wu, "PKE-MET: Public-key encryption with multi-ciphertext equality test in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 1476–1488, Apr. 2022.
[16] X.-J. Lin, L. Sun, H. Qu, and X. Zhang, "Public key encryption supporting equality test and flexible authorization without bilinear pairings," *Comput. Commun.*, vol. 170, pp. 190–199, Mar. 2021.
[17] D. H. Duong, K. Fukushima, S. Kiyomoto, P. S. Roy, and W. Susilo, "A lattice-based public key encryption with equality test in standard model," in *Proc. ACISP*. Cham, Switzerland: Springer, Jan. 2019, pp. 138–155.
[18] P. S. Roy, D. H. Duong, W. Susilo, A. Sipasseuth, K. Fukushima, and S. Kiyomoto, "Lattice-based public-key encryption with equality test supporting flexible authorization in standard model," *Theor. Comput. Sci.*, vol. 929, pp. 124–139, Sep. 2022.
[19] K. Xiao, X. Chen, J. Huang, H. Li, and Q. Huang, "A lattice-based public key encryption scheme with delegated equality test," *Comput. Standards Interfaces*, vol. 87, Jan. 2024, Art. no. 103758.
[20] X. Shen, L. Wang, H. Zhu, and Y. Liu, "A multivariate public key encryption scheme with equality test," *IEEE Access*, vol. 8, pp. 75463–75472, 2020.
[21] Q. Qu, B. Wang, L. Wang, Y. Wang, and Y. Yan, "More efficient tightly-secure lattice-based IBE with equality test," *Comput. Standards Interfaces*, vol. 86, Aug. 2023, Art. no. 103736.
[22] J. Lu, H. Li, J. Huang, S. Ma, M. H. A. Au, and Q. Huang, "An identity-based encryption with equality test scheme for healthcare social apps," *Comput. Standards Interfaces*, vol. 87, Jan. 2024, Art. no. 103759.
[23] Q. Wang, L. Peng, H. Xiong, J. Sun, and Z. Qin, "Ciphertext-policy attribute-based encryption with delegated equality test in cloud computing," *IEEE Access*, vol. 6, pp. 760–771, 2018.
[24] Y. Cui, Q. Huang, J. Huang, H. Li, and G. Yang, "Ciphertext-policy attribute-based encrypted data equality test and classification," *Comput. J.*, vol. 62, no. 8, pp. 1166–1177, Aug. 2019.
[25] Y. Wang, Y. Cui, Q. Huang, H. Li, J. Huang, and G. Yang, "Attribute-based equality test over encrypted data without random oracles," *IEEE Access*, vol. 8, pp. 32891–32903, 2020.
[26] H. Qu, Z. Yan, X.-J. Lin, Q. Zhang, and L. Sun, "Certificateless public key encryption with equality test," *Inf. Sci.*, vol. 462, pp. 76–92, Sep. 2018.
[27] R. Elhabob, M. Taha, H. Xiong, M. K. Khan, S. Kumari, and P. Chaudhary, "Pairing-free certificateless public key encryption with equality test for Internet of Vehicles," *Comput. Electr. Eng.*, vol. 116, May 2024, Art. no. 109140.
[28] H. Q. Le, D. H. Duong, P. S. Roy, W. Susilo, K. Fukushima, and S. Kiyomoto, "Lattice-based signcryption with equality test in standard model," *Comput. Standards Interfaces*, vol. 76, Jun. 2021, Art. no. 103515.
[29] Y. Hou, Y. Cao, H. Xiong, Y. Hu, and M. H. Eiza, "Heterogeneous signcryption scheme with group equality test for satellite-enabled IoVs," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10666–10681, Mar. 2024.
[30] H. T. Lee, S. Ling, J. H. Seo, and H. Wang, "Semi-generic construction of public key encryption and identity-based encryption with equality test," *Inf. Sci.*, vol. 373, pp. 419–440, Dec. 2016.
[31] K. Emura and A. Takayasu, "A generic construction of CCA-secure identity-based encryption with equality test against insider attacks," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 106, no. 3, pp. 193–202, 2023.
[32] K. Emura, "On the feasibility of identity-based encryption with equality test against insider attacks," in *Proc. ACISP*, vol. 14895. Cham, Switzerland: Springer, 2024, pp. 261–280.
[33] K. Asano, K. Emura, A. Takayasu, and Y. Watanabe, "A generic construction of CCA-secure attribute-based encryption with equality test," in *Proc. ProvSec*. Cham, Switzerland: Springer, Jan. 2022, pp. 3–19.
[34] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. TIT-31, no. 4, pp. 469–472, Jul. 1985.
[35] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 26, no. 1, pp. 120–126, Jan. 1978.
[36] N. Aragon, P. S. L. M. Barreto, S. Bettaieb, L. B. O. B. J.-C. Deneuville, P. Gaborit, S. Ghosh, S. Gueron, T. Gneysu, C. A. Melchor, R. Misoczki, E. Persichetti, J. Richter-Brockmann, N. Sendrier, J.-P. Tillich, V. Vasseur, and G. Zmor, "BIKE: Bit flipping key encapsulation," Nat. Inst. Standards Technol. (NIST), Gaithersburg, MD, USA, Tech. Rep., 2022. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions
[37] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," *J. Cryptol.*, vol. 26, no. 1, pp. 80–101, Jan. 2013.
[38] J. Daemen and V. Rijmen, "Rijndael for AES," in *Proc. 3rd Adv. Encryption Standard Candidate Conf.*, New York, NY, USA, Jan. 2000, pp. 343–348.



SEONGBONG CHOI received the B.Sc. degree in computer science and engineering from Jeonbuk National University, Republic of Korea, in 2020, and the M.S. degree in computer science and engineering from Chung-Ang University, Republic of Korea, in 2022, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. His research interest includes public key cryptography.



CHANHYEOK PARK received the B.S. and M.S. degrees in computer science and engineering from Chung-Ang University, Republic of Korea, in 2022 and 2024, respectively, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. His research interests include cryptography and information security.



HYUNG TAE LEE received the B.Sc., M.Sc., and Ph.D. degrees in mathematics from Seoul National University, Republic of Korea, in 2006, 2008, and 2013, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, Chung-Ang University, Republic of Korea. Prior to that, he was a Research Fellow with Nanyang Technological University, Singapore; and then an Assistant Professor with Jeonbuk National University, Republic of Korea. His research interests include computational number theory, cryptography, and information security.