applied sciences

MDPI

Article

Revisiting NIZK-Based Technique for Chosen-Ciphertext Security: Security Analysis and Corrected Proofs

Youngkyung Lee [1], Dong Hoon Lee [1] and Jong Hwan Park [2,*]

[1] Graduate School of Information Security, Korea University, Seoul 02841, Korea; dudrudve@korea.ac.kr (Y.L.); donghlee@korea.ac.kr (D.H.L.)
[2] Department of Computer Science, Sangmyung University, Seoul 03016, Korea
* Correspondence: jhpark@smu.ac.kr

Abstract: Non-interactive zero-knowledge (NIZK) proofs for chosen-ciphertext security are generally considered to give an impractical construction. An interesting recent work by Seo, Abdalla, Lee, and Park (Information Sciences, July 2019) proposed an efficient semi-generic conversion method for achieving chosen-ciphertext security based on NIZK proofs in the random oracle model. The recent work by Seo et al. demonstrated that the semi-generic conversion method transforms a one-way (OW)-secure key encapsulation mechanism (KEM) into a chosen-ciphertext secure KEM while preserving tight security reduction. This paper shows that the security analysis of the semi-generic conversion method has a flaw, which comes from the OW security condition of the underlying KEM. Without changing the conversion method, this paper presents a revised security proof under the changed conditions that (1) the underlying KEM must be chosen-plaintext secure in terms of indistinguishability and (2) an NIZK proof derived from the underlying KEM via the Fiat–Shamir transform must have the properties of zero-knowledge and simulation soundness. This work extended the security proof strategy to the case of identity-based KEM (IBKEM) and also revise the security proof for IBKEM of previous method by Seo et al. Finally, this work gives a corrected security proof by applying the new proofs to several existing (IB)KEMs.

Keywords: NIZK; chosen-ciphertext security; tight security reduction; random oracle model

Citation: Lee, Y.; Lee, D.H.; Park, J.H. Revisiting NIZK-Based Technique for Chosen-Ciphertext Security: Security Analysis and Corrected Proofs. *Appl. Sci.* **2021**, *11*, 3367. https://doi.org/10.3390/app11083367

Academic Editor: Arcangelo Castiglione

Received: 11 March 2021
Accepted: 5 April 2021
Published: 8 April 2021

1. Introduction

Non-interactive zero-knowledge (NIZK) proofs [1–3] are considered as some of the most fundamental and versatile cryptographic primitives [4,5]. One usage of NIZK is to construct public-key encryption schemes secure against chosen-ciphertext attacks (denoted as "CCA-security") based on the Naor–Yung double encryption paradigm [6–8]. As building blocks, this approach uses any public-key encryption scheme secure against chosen-plaintext attacks (denoted as "CPA security") and any NIZK proof system for all of $\mathcal{NP}$ [9,10]. However, the Naor–Yung paradigm has been perceived as a feasibility result for the existence of CCA-secure encryption schemes based on general cryptographic assumptions, leading to impractical constructions [11].

Interestingly, a recent work by Seo et al. [12] proposed a new semi-generic approach for constructing a CCA-secure (and practical) key encapsulation mechanism (KEM) based on NIZK proof systems derived from the Fiat–Shamir (FS) transform [13]. As building blocks, their technique uses a one-way (OW)-secure KEM and an FS-derived NIZK proof system to prove the relationship (such as equality or linearity) among discrete logarithms. In particular, the term "semi-generic" comes from the fact that the underlying OW-secure KEM should satisfy an additional property called "NIZK-compatibility", meaning that the pair {ciphertext, key} can be an NIZK statement when randomness for KEM is used as a witness for NIZK. Seo et al. [12] demonstrated that their approach can transform an OW-secure (and NIZK-compatible) KEM into a CCA-secure KEM in the random oracle model without security loss.

1.1. Organization

In Section 1.2, we briefly review the semi-generic conversion method from Seo et al. and explain the flaw in the security proof of the method. In Section 1.3, we present the concept of corrected proofs. In Section 2, we present the details of the interactive proof systems for proving the equality and linearity of discrete logarithms, and we describe how FS-derived NIZK proof systems can achieve the properties of zero-knowledge and simulation soundness. In Section 3, we introduced the conversion method for CCA-secure KEM and present our corrected proofs using the hybrid argument of indistinguishability-based framework. As in [12], we can extend our strategy to the case of identity-based KEM (IBKEM); thus, a newly corrected security proof for the conversion of IBKEM is given in Section 4. In Section 5, we apply our security result to several (IB)KEMs used in [12] and provide new corrected theorems for each scheme. Finally, we conclude this paper in Section 6.

1.2. Flaw in Security Proof in Previous Research Literature

The new conversion in [12] works correctly, but we have identified a flaw in its security analysis. For easier explanation, consider the ElGamal KEM as an underlying scheme. Let $\mathbb{G}$ be a group of prime order p, and let (g^w, h^w) be a pair {ciphertext, key} for group elements $g, h \in \mathbb{G}$, and a random $w \in \mathbb{Z}_p$. It is easy to see that the pair can be an NIZK statement $\mathsf{stmt} = (g^w, h^w)$ for equality of discrete logarithms when the randomness w is the witness (i.e., NIZK-compatible). Let com be a commitment (g^r, h^r) for a random $r \in \mathbb{Z}_p$. The Fiat–Shamir transform [13] gives the NIZK proof $\pi = (c, s)$, where $s = r + cw \in \mathbb{Z}_p$ and $c = H_1(\mathsf{stmt}, \mathsf{com})$. Assuming that the hash function H_1 is modeled as a random oracle, the NIZK proof system is proven to be honest verifier zero-knowledge and sound. The concept behind the transformation proposed in [12] is to see the original ElGamal KEM as a designated verifier proof system by setting $h = g^x$ for a secret x (only known to the verifier) so that the element $A_1 = g^w$ is sufficient for the designated verifier NIZK statement. In the transformed variant of the ElGamal KEM, the resulting ciphertext consists of $(A_1, \pi = (c, s))$, and the decapsulation algorithm (as a designated verifier) first recovers $A_2 = h^w$ by computing $(g^w)^x$ and then computes $\mathsf{com} = (g^s A_1^{-c}, h^s A_2^{-c})$ along with $\mathsf{stmt} = (g^w, h^w)$ and π. It also verifies that π is a valid proof for the recovered statement stmt. If π is valid, the decapsulation algorithm computes the final KEM key as $\mathsf{key} = H_2(\mathsf{stmt}, \mathsf{com}, \pi)$ for another hash function H_2.

Seo et al. [12] demonstrated that the above variant of the ElGamal KEM is tightly CCA-secure based on the *computational* Diffie–Hellman (CDH) assumption. From a theoretical viewpoint, the first challenge in their security analysis is how to handle decapsulation queries issued by an adversary without knowing the secret key x. Let $\mathsf{CT} = (A_1, \pi = (c, s))$ be a queried ciphertext, and let H_1, H_2 be modeled as random oracles. A reduction algorithm (simulator) finds a tuple in the H_1 query table such that $c = H_1(A_1, \star, \star, \star)$, say $c = H_1(A_1, A_2, B_1, B_2)$, and checks that π is the valid NIZK proof for the statement (A_1, A_2) and the commitment (B_1, B_2). If so, the simulator outputs the key $\mathsf{key} = H_2(A_1, A_2, B_1, B_2, \pi)$ from the H_2 query table. The decapsulation works correctly because of the soundness of the underlying NIZK proof system, meaning that the adversary is forced to generate the proof π for the well-formed ciphertext such that $\log_g A_1 = \log_h A_2$. The second challenge in their security analysis is to find the correct solution h^a to a given CDH instance (g, g^a, h) without security loss. The strategy in [12] is as follows: by using the zero-knowledge property, the simulator first generates a simulated proof $\pi^* = (c^*, s^*)$ with respect to the element $A_1^* = g^a$ for some unknown $a \in \mathbb{Z}_p$ and implicitly sets $c^* = H_1(A_1^*, \circ, B_1^*, \diamond)$ and $\mathsf{key}^* = H_2(A_1^*, \circ, B_1^*, \diamond, \pi^*)$, where "$\circ$" and "$\diamond$" indicate that the simulator does not know the entries. After the challenge ciphertext $\mathsf{CT}^* = (A_1^*, \pi^* = (c^*, s^*))$ and the challenge key key^* are given to the adversary, the simulator waits for the adversary to query the hash inputs $(A_1^*, \star, B_1^*, \star, \pi^*)$ for H_2. Seo et al.'s assertion [12] is that *because of the soundness of the NIZK proof system, there exists one query* $(A_1^*, A_2, B_1^*, B_2, \pi^*)$ *(with overwhelming probability) such that* π^* *is valid among the queried*

inputs $\{(A_1^*, \star, B_1^*, \star, \pi^*)\}$, *in which case the entry* A_2 *is the solution to the given CDH instance*. Thus, if their assertion was true, the adversary who succeeds in breaking the CCA-security of the variant must make the query, including the CDH solution, to H_2; otherwise, the adversary has no information on the challenge key.

However, we show that their assertion is not true for the following reasons. Let $\mathsf{CT}^* = (A_1^*, \pi^* = (c^*, s^*))$ and key^* be the challenge ciphertext and key, respectively. First, the adversary generates the value B_1^* by computing $g^{s^*}(A_1^*)^{-c^*}$, which is the same as the value that the simulator calculated. Next, the adversary can generate the values A_2 and B_2 as follows: pick a random $R \in \mathbb{Z}_p$, and set $A_2 = h^R$ and $B_2 = h^{s^*}(A_2)^{-c^*}$. Finally, the adversary issues $(A_1^*, A_2, B_1^*, B_2, \pi^*)$ to the H_2 query. We can see that $\pi^* = (c^*, s^*)$ is valid with respect to the *false* statement (A_1^*, A_2) and the relevant commitment (B_1^*, B_2); furthermore, polynomially-many such simulated queries can be made after CT^* and key^* are given to the adversary. If the adversary issues the correct query such that $A_2 = h^a$ among the polynomially-many simulated queries, the simulator cannot specify which query relates to the correct statement $(A_1^* = g^a, A_2 = h^a)$ associated with the CDH solution, unless it is given an oracle to solve the decisional Diffie–Hellman (DDH) problem. Moreover, the implicitly predetermined challenge key key^* will be mapped to a certain wrong H_2 query input $(A_1^*, A_2', B_1^*, B_2', \pi^*)$ with "high" probability. Clearly, such inconsistent mapping will allow the adversary to distinguish between a real attack and a simulation.

1.3. Concept of Corrected Proofs

Our strategy to revise Seo et al.'s proofs [12] is to change the security condition of the underlying KEM from OW to CPA while keeping their conversion method intact. In our corrected proofs, the underlying KEM must be CPA-secure in terms of indistinguishability, meaning (roughly) the following: given (CT^*, K^*) as a challenge, it is infeasible to determine whether the challenge key K^* was correct or false with respect to the challenge ciphertext CT^*. Now, the simulator attacking the CPA security of the underlying KEM is given (CT^*, K^*) and wants to determine whether K^* is correct or false using the adversary against CCA-security. Let us assume that the underlying KEM is still NIZK-compatible, as defined in [12]. Given (CT^*, K^*), the simulator proceeds as follows: it sets the pair as the NIZK statement, $\mathsf{stmt}^* = (\mathsf{CT}^*, K^*)$, and generates a simulated proof π^* for stmt^* using the zero-knowledge property. Then, it computes the challenge key $\mathsf{key}^* = H_2(\mathsf{stmt}^*, \mathsf{com}^*, \pi^*)$, where com^* is the relevant commitment for stmt^*. Then, it gives the challenge ciphertext (CT^*, π^*) and the challenge key key^* to the adversary. If K^* is the correct key corresponding to CT^*, then π^* is the NIZK proof for the correct statement (CT^*, K^*). Otherwise, π^* is the NIZK proof for the false statement (CT^*, K^*). Now that the simulator knows the candidate value K^* of the KEM key, it does distinguish whether K^* is correct or false by checking whether $(\mathsf{stmt}^*, \mathsf{com}^*, \pi^*)$ is issued to H_2 queries. Note that K^* is not given to the adversary. If K^* (including stmt^*) appears in H_2 queries, it is possible to ensure (with overwhelming probability) that K^* is the correct key corresponding to CT^*; otherwise, K^* is a false key. Furthermore, as long as the input $(\mathsf{stmt}^*, \mathsf{com}^*, \pi^*)$ is not queried to H_2 (modeled as a random oracle), the adversary obtains no information on the challenge key key^*. It follows that success in breaking the CCA-security of the transformed KEM straightforwardly leads to success in breaking the CPA security of the underlying KEM without causing security loss. When applying our proof strategy to the above variant of the ElGamal KEM, it can be proved to be tightly CCA-secure under the DDH assumption.

In our corrected proofs, decapsulation queries are handled by the simulator in the same way as in [12]. However, the NIZK property required for answering decapsulation queries is (one-time) *simulation soundness* (SS) rather than soundness. This is because π^* can be the *simulated* proof for the correct or false statement (CT^*, K^*) (given to the simulator) depending on K^*. Furthermore, once (CT^*, π^*) is given to the adversary as a challenge, it should be difficult for the adversary to generate a pair of a new *false* statement and its relevant proof that are correctly verified. In particular, a new false statement includes

any modification of $\mathsf{stmt}^* = (\mathsf{CT}^*, K^*)$. Unlike in Seo et al.'s proofs [12], the simulator in ours knows all hash input values regarding stmt^* and com^* explicitly; thus, the security flaw mentioned above does not occur. Rather, making hash queries about stmt^* by the adversary could increase the probability that the simulator will succeed in breaking the CPA security of the given KEM.

To show that the underlying NIZK proof system satisfies the simulation soundness property, we use the result of Faust et al. [14], which showed that (roughly) an FS-derived NIZK proof system Π^{FS} is simulation-sound in the random oracle model if a canonical three-round interactive proof system Π (The transcript of Π consists of {commitment, challenge, and response}.) has the properties of completeness, soundness, and unique response. In fact, along with the soundness, the unique response property gives the effect of making the resulting NIZK proof π strongly unforgeable [15] when viewing π as a (one-time) signature. As a building block, the conversion by Seo et al. [12] uses an FS-derived NIZK proof system to prove the equality or linearity of discrete logarithms, and it is easy to show that a three-round interactive proof system for equality or linearity satisfies the property of unique response as well as soundness. Thus, by adapting the result in [14], all the underlying (FS-derived) NIZK proof systems in our corrected proofs have the simulation soundness property in the random oracle model.

2. Background

2.1. Notation

$\lambda \in \mathbb{N}$ is the security parameter. We say that a function $\nu : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every positive polynomial $\mathsf{poly}(\cdot)$, there exists an integer $N > 0$ such that for all $x > N$, it holds that $|\nu(x)| < \mathsf{poly}(x)^{-1}$. Given an algorithm A, we write $y \leftarrow A$ to denote that y is the output of A. If A is a probabilistic algorithm, then $y \xleftarrow{\$} A$ denotes that y is computed by A using fresh random coins. When A is a set, $a \xleftarrow{\$} A$ denotes that a is chosen uniformly over A. For $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. The above notation follows the notation from the work in [16].

2.2. Interactive Proof System

Let $\mathcal{L} = \{\mathsf{stmt} : \exists\ \mathsf{wit}\ s.t.\ (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathcal{L}}\}$ be an NP-language. We first review a three-move public coin proof system, where a prover wants to convince a verifier that a statement stmt belongs to $\mathcal{L}$ using a witness wit such that $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathcal{L}}$. Let a prover $\mathcal{P} = (\mathcal{P}_0, \mathcal{P}_1)$ and a verifier $\mathcal{V} = (\mathcal{V}_0, \mathcal{V}_1)$ be PPT algorithms that participate in the protocol. First, as inputs, $\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{stmt}, \mathsf{wit}; \rho)$ takes a statement stmt, a witness wit, and a random string ρ and then computes a commitment com and sends it to a verifier $\mathcal{V}$. Then, $\mathsf{chal} \xleftarrow{\$} \mathcal{V}_0(\mathsf{stmt}, \mathsf{com})$ inputs a statement stmt and commitment com, randomly chooses a challenge chal in the challenge space, and sends it to $\mathcal{P}$. Finally, as inputs, $\mathsf{resp} \leftarrow \mathcal{P}_1(\mathsf{com}, \mathsf{chal}, \mathsf{stmt}, \mathsf{wit}; \rho)$ takes com, chal, stmt, wit, and ρ; generates a response resp; and sends it to $\mathcal{V}$. Then, as inputs, $\{0, 1\} \leftarrow \mathcal{V}_1(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}, \mathsf{resp})$ takes stmt, com, chal, and resp, and then outputs 0 or 1. When $\mathcal{V}_1$ outputs 1, the verifier $\mathcal{V}$ is convinced that the statement stmt belongs to $\mathcal{L}$.

We now review the following properties.

2.2.1. Completeness

If $\mathsf{stmt} \in \mathcal{L}$, any proper execution of the protocol between $\mathcal{P}$ and $\mathcal{V}$ ends with the verifier accepting $\mathcal{P}$'s proof. That is, the following holds:

$$\Pr[1 \leftarrow \mathcal{V}_1(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}, \mathsf{resp}) : \mathsf{stmt} \in \mathcal{L} \wedge \mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{stmt}, \mathsf{wit}; \rho) \wedge \mathsf{chal} \xleftarrow{\$} \mathcal{V}_0(\mathsf{stmt}, \mathsf{com}) \wedge \mathsf{resp} \leftarrow \mathcal{P}_1(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}, \mathsf{wit}; \rho)] = 1.$$

2.2.2. Honest Verifier Zero-Knowledge (HVZK)

There exists an efficient algorithm $\mathcal{S}$ called a zero-knowledge simulator such that for any PPT distinguisher, $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$, and for any $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathcal{L}}$, it holds that

$$\begin{aligned}&\Pr[1 \leftarrow \mathcal{D}_1(\pi, \delta) : (\mathsf{stmt}, \mathsf{wit}, \delta) \leftarrow \mathcal{D}_0 \wedge \pi \leftarrow \langle \mathcal{P}(\mathsf{stmt}, \mathsf{wit}; \lambda), \mathcal{V}(\mathsf{stmt}; \lambda)\rangle] \\ &\approx \Pr[1 \leftarrow \mathcal{D}_1(\pi, \delta) : (\mathsf{stmt}, \mathsf{wit}, \delta) \leftarrow \mathcal{D}_0 \wedge \pi \leftarrow \mathcal{S}(\mathsf{stmt}, \lambda)],\end{aligned}$$

where $\pi \leftarrow \langle \mathcal{P}(\mathsf{stmt}, \mathsf{wit}; \lambda), \mathcal{V}(\mathsf{stmt}; \lambda)\rangle$ denotes the resulting transcript returned at the end of the interaction between $\mathcal{P}$ and $\mathcal{V}$ on common input stmt and private input wit.

2.2.3. Soundness

If $\mathsf{stmt} \notin \mathcal{L}$, any PPT adversary $\mathcal{P}^* = (\mathcal{P}_0^*, \mathcal{P}_1^*)$ is accepted only with negligible probability. That is, the following holds:

$$\begin{aligned}\Pr[\mathsf{resp} \leftarrow \mathcal{P}_1^*(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}; \rho) : \mathsf{stmt} \notin \mathcal{L} \wedge \mathsf{com} \leftarrow \mathcal{P}_0^*(\mathsf{stmt}; \rho) \\ \wedge\ \mathsf{chal} \xleftarrow{\$} \mathcal{V}_0(\mathsf{stmt}, \mathsf{com}) \wedge 1 \leftarrow \mathcal{V}_1(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}, \mathsf{resp})] \approx 0.\end{aligned}$$

2.2.4. Unique Response

An interactive proof system has the unique response if for any PPT adversary $\mathcal{A}$ and for any security parameter λ, it holds that

$$\begin{aligned}&\Pr[(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}, \mathsf{resp}, \mathsf{resp}') \leftarrow \mathcal{A}(\lambda) : \\ &\quad \mathcal{V}(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}, \mathsf{resp}) = \mathcal{V}(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}, \mathsf{resp}') = 1 \wedge \mathsf{resp} \neq \mathsf{resp}'] = 0.\end{aligned}$$

2.3. Protocol for Proving the Equality of Discrete Logarithms

We review the generalized interactive protocol for proving the equality of discrete logarithms [17–19]. Let $\mathbb{G}_1, \ldots, \mathbb{G}_n$ be groups of prime order p, and let $g_i \in \mathbb{G}_i$ be a generator for $i \in [n]$. Now, we consider the NP-language

$$\mathcal{L} = \{(g_1, \ldots, g_n, A_1, \ldots, A_n) : \exists\, x \in \mathbb{Z}_p \text{ s.t. } A_i = g_i^x \text{ for } i \in [n]\}.$$

The discrete logarithm $\mathsf{wit} = x$ is a witness for a statement: $\mathsf{stmt} = (g_1, \ldots, g_n, A_1, \ldots, A_n)$. Then, the interactive protocol $(\mathcal{P}, \mathcal{V})$ for proving the above language is as follows.

1. For a statement $\mathsf{stmt} = (g_1, \ldots, g_n, A_1, \ldots, A_n)$, the prover $\mathcal{P}$ selects a random exponent $r \in \mathbb{Z}_p$ and computes a commitment com as follows.
$$\mathsf{com} = (B_1, \ldots, B_n) = (g_1^r, \ldots, g_n^r).$$
2. The verifier $\mathcal{V}$ randomly selects a random challenge $\mathsf{chal} = c \in \mathbb{Z}_p$.
3. $\mathcal{P}$ computes a response $\mathsf{resp} = s = r + cd \in \mathbb{Z}_p$.
4. Given a proof $\pi = (\mathsf{com}, \mathsf{chal}, \mathsf{resp})$, $\mathcal{V}$ checks that
$$B_1 = g_1^s {A_1}^{-c}, \ \ldots, \ B_n = g_n^s {A_n}^{-c}.$$

It is well known that this protocol is a non-trivial three-round public coin interactive proof system satisfying completeness, HVZK, (statistical) soundness, and *unique response* properties.

2.4. Protocol for Proving the Linearity of Discrete Logarithms

We review the generalized interactive protocol for proving the linearity of discrete logarithms. Let $\mathbb{G}_1, \ldots, \mathbb{G}_{n+1}$ be groups of prime order p, and let $g_i \in \mathbb{G}_i$ be a generator for $i \in [n+1]$. Now, we consider the NP-language

$$\mathcal{L} = \{(g_1, \ldots, g_n, g_{n+1}, A_1, \ldots, A_n, A_{n+1}) : \\ \exists\, (x_1, \ldots, x_n) \in \mathbb{Z}_p^n \text{s.t. } A_i = g_i^{x_i} \text{ for } i \in [n],\ A_{n+1} = g_{n+1}^{x_1+\cdots+x_n}\}.$$

The discrete logarithm $\mathsf{wit} = (x_1, \ldots, x_n)$ is a witness for a statement $\mathsf{stmt} = (g_1, \ldots, g_n, g_{n+1}, A_1, \ldots, A_n, A_{n+1})$. Then, the interactive protocol $(\mathcal{P}, \mathcal{V})$ for proving the above language is as follows.

1. For a statement $\mathsf{stmt} = (g_1, \ldots, g_n, g_{n+1}, A_1, \ldots, A_n, A_{n+1})$, the prover $\mathcal{P}$ selects random exponents $(r_1, \ldots, r_n) \in \mathbb{Z}_p^n$ and computes a commitment com as follows.
$$\mathsf{com} = (B_1, \ldots, B_n, B_{n+1}) = (g_1^{r_1}, \ldots, g_n^{r_n}, g_{n+1}^{r_1+\cdots+r_n}).$$
2. The verifier $\mathcal{V}$ randomly selects a random challenge $\mathsf{chal} = c \in \mathbb{Z}_p$.
3. $\mathcal{P}$ computes a response $\mathsf{resp} = (s_1, \ldots, s_n)$, where $s_i = r_i + cx_i \in \mathbb{Z}_p$ for $i \in [n]$.
4. Given a proof $\pi = (\mathsf{com}, \mathsf{chal}, \mathsf{resp})$, $\mathcal{V}$ checks that
$$B_1 = g_1^{s_1} {A_1}^{-c}, \ldots, B_n = g_n^{s_n} {A_n}^{-c}, \text{ and } B_{n+1} = g_{n+1}^{s_1+\cdots+s_n} {A_{n+1}}^{-c}.$$

Similarly to the equality of discrete logarithms, this protocol is also a non-trivial three-round public coin interactive proof system satisfying completeness, HVZK, (statistical) soundness, and *unique response* properties.

2.5. NIZK in the Random Oracle Model

We can remove the interaction between $\mathcal{P}$ and $\mathcal{V}$ by replacing the challenge chal with a hash value $H(\mathsf{stmt}, \mathsf{com})$ computed by the prover, where H is a hash function modeled as a random oracle. By applying the Fiat–Shamir paradigm [13], we can transform an interactive protocol $(\mathcal{P}, \mathcal{V})$ into a non-interactive proof system $(\mathcal{P}^H, \mathcal{V}^H)$.

Syntax

Let $\mathcal{L} := \{\mathsf{stmt} : \exists\, \mathsf{wit} \text{ such that } R(\mathsf{stmt}, \mathsf{wit}) = 1\}$ is an NP-language defined by a binary relation R, and H is a hash function (modeled as a random oracle). A non-interactive proof system $(\mathcal{P}^H, \mathcal{V}^H)$ for $\mathcal{L}$ consists of two algorithms. The proving algorithm $\pi \xleftarrow{\$} \mathcal{P}^H(\mathsf{stmt}, \mathsf{wit})$ takes as input a statement stmt and a witness wit such that $R(\mathsf{stmt}, \mathsf{wit}) = 1$ and outputs a proof π. The verification algorithm $\mathcal{V}^H(\mathsf{stmt}, \pi) \in \{0, 1\}$ takes as input a statement stmt and a proof π and outputs 1 (true) or 0 (false).

We refer here to the zero-knowledge simulator $\mathcal{S}$ of a non-interactive zero-knowledge proof system [14] defined in the explicitly programmable random oracle model [20]. As a stateful algorithm, the simulator $\mathcal{S}$ can operate in two modes: $(h_i, st) \leftarrow \mathcal{S}(1, st, q_i)$ deals with answering random oracle queries such that $h_i = H(q_i)$, while $(\pi, st) \leftarrow \mathcal{S}(2, st, \mathsf{stmt})$ simulates the proof. Note that calls to $\mathcal{S}(1, \cdots)$ and $\mathcal{S}(2, \cdots)$ share the common state st, which is updated after each operation.

Definition 1 (Unbounded Non-Interactive Zero-Knowledge [14]). *Let $\mathcal{L}$ be an NP-language. Let $(\mathcal{S}_1, \mathcal{S}_2)$ denote the oracles such that $\mathcal{S}_1(q_i)$ returns the first output of $(h_i, st) \leftarrow \mathcal{S}(1, st, q_i)$ and $\mathcal{S}_2(\mathsf{stmt}, \mathsf{wit})$ returns the first output of $(\pi, st) \leftarrow \mathcal{S}(2, st, \mathsf{stmt})$ if $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathcal{L}}$. We say a protocol $(\mathcal{P}^H, \mathcal{V}^H)$ is an NIZK proof for language $\mathcal{L}$ in the random oracle model if there exists a PPT simulator $\mathcal{S}$ such that for all PPT distinguishers $\mathcal{D}$, the advantage ϵ_{ZK} such that*

$$\epsilon_{ZK}(\lambda) = \left| \Pr[1 \leftarrow \mathcal{D}^{H(\cdot), \mathcal{P}^H(\cdot)}(\lambda)] - \Pr[1 \leftarrow \mathcal{D}^{\mathcal{S}_1(\cdot), \mathcal{S}_2(\cdot)}(\lambda)] \right|$$

is negligible, where both $\mathcal{P}$ and $\mathcal{S}_2$ oracles output $\perp$ if $(\mathsf{stmt}, \mathsf{wit}) \neq \mathcal{R}_{\mathcal{L}}$.

Definition 2 (Unbounded Simulation Soundness [14])**.** *Let $\mathcal{L}$ be an NP-language. Consider a proof system $(\mathcal{P}^H, \mathcal{V}^H)$ for $\mathcal{L}$ with zero-knowledge simulator $\mathcal{S}$. Let $(\mathcal{S}_1, \mathcal{S}_2')$ denote the oracle such that $\mathcal{S}_1(q_i)$ returns the first output of $(h_i, st) \leftarrow \mathcal{S}(1, st, q_i)$ and $\mathcal{S}_2'$ returns the first output of $(\pi, st) \leftarrow \mathcal{S}(2, st, \mathsf{stmt})$. We say that $(\mathcal{P}^H, \mathcal{V}^H)$ is simulation-sound with respect to $\mathcal{S}$ in the random oracle model if for all PPT adversaries $\mathcal{A}$ the advantage ϵ_{SS} such that*

$$\epsilon_{\mathsf{SS}}(\lambda) = \Pr[(\mathsf{stmt}^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{S}_1(\cdot), \mathcal{S}_2'(\cdot)}(\lambda) : (\mathsf{stmt}^*, \pi^*) \neq \mathcal{T} \wedge \mathsf{stmt}^* \neq \mathcal{L} \wedge \mathcal{V}^{\mathcal{S}_1}(\mathsf{stmt}^*, \pi^*) = 1]$$

is negligible, where $\mathcal{T}$ is the list of pairs (stmt_i, π_i), i.e., (true or false) statements queried to and proofs returned by the simulator.

Theorem 1 (Fiat–Shamir NIZKs [14])**.** *Consider a three-round HVZK interactive proof system $(\mathcal{P}, \mathcal{V})$ for a language $\mathcal{L} \in$ NP with completeness. Let H be a function with range equal to the space of the verifier's coins. In the random oracle model, the proof system $(\mathcal{P}^H, \mathcal{V}^H)$ derived from $(\mathcal{P}, \mathcal{V})$ by applying the Fiat–Shamir transform is unbounded non-interactive zero-knowledge.*

Theorem 2 (Simulation Soundness of Fiat–Shamir NIZKs [14])**.** *Consider a three-round HVZK interactive proof system $(\mathcal{P}, \mathcal{V})$ for a language $\mathcal{L} \in$ NP with completeness, soundness, and unique response. In the random oracle model, the proof system $(\mathcal{P}^H, \mathcal{V}^H)$ derived from $(\mathcal{P}, \mathcal{V})$ via the Fiat–Shamir transform is simulation-sound NIZK with respect to its canonical simulator $\mathcal{S}$.*

By applying the above theorems to the previous interactive protocols for proving the *equality* and *linearity* of discrete logarithms, we can obtain NIZK proof systems for the equality and linearity of discrete logarithms, which both have the properties of zero-knowledge and simulation soundness. Note that the two previous interactive protocols satisfy the completeness, HVZK, (statistical) soundness, and unique response properties.

To measure the concrete bounds of the advantages ϵ_{ZK} and ϵ_{SS} with respect to the two interactive protocols, we follow the proofs in [14] (Theorems 1 and 2 therein). First, ϵ_{ZK} of the NIZK systems is bounded by the probability that the "good" NIZK simulator $\mathcal{S}$ fails and aborts during the simulation. Note that the NIZK simulator $\mathcal{S}$ fails when a collision of the random oracle H happens in the process of updating the hash table $\mathcal{T}_H$. Because the probability that a collision happens is at most q_H^2/p, where q_H is the maximum number of H queries and p is the group order including the size of the challenge space, we see that $\epsilon_{\mathsf{ZK}} \leq q_H^2/p$. Second, ϵ_{SS} of the NIZK scheme is bounded by $q_H \cdot \epsilon_{\mathsf{snd}} + q_H^2/p$, where ϵ_{snd} denotes the bound of advantage that the adversary violates the *soundness* of the proof system $(\mathcal{P}, \mathcal{V})$. The *soundness* bound ϵ_{snd} in the two interactive protocols is statistically the same as the inverse of the size of the challenge space, $1/p$, so we see that $\epsilon_{\mathsf{SS}} \leq (q_H + q_H^2)/p$.

3. Conversion Method for CCA-Secure KEM

3.1. KEM

3.1.1. Syntax

We follow the syntax of KEM from the work in [16]. A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KEM.Setup}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ consists of three algorithms: The setup algorithm $(\mathsf{PK}, \mathsf{SK}) \xleftarrow{\$} \mathsf{KEM.Setup}(\lambda)$ takes as input the security parameter λ and outputs a key pair $(\mathsf{PK}, \mathsf{SK})$. The encapsulation algorithm $(\mathsf{K}, \mathsf{CT}) \xleftarrow{\$} \mathsf{KEM.Encap}(\mathsf{PK})$ generates a key K and a ciphertext CT from input PK. The decapsulation algorithm $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{PK}, \mathsf{SK}, \mathsf{CT})$ takes as input a public key PK, a private key SK, and a ciphertext CT and outputs a key K.

3.1.2. Security Model of KEM

Here, we define the IND-{CPA,CCA} security of KEM by referring to [12]. First, a security experiment of IND-CPA played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is described as follows.

Experiment $\mathbf{Exp}^{\mathsf{KEM},\mathcal{A}}_{\mathsf{IND\text{-}CPA},b}(\lambda)$
$(\mathsf{PK},\mathsf{SK}) \xleftarrow{\$} \mathsf{KEM.Setup}(\lambda)$;
$\mathsf{K}^*_0 \xleftarrow{\$} \mathcal{K}$; $(\mathsf{K}^*_1,\mathsf{CT}^*) \xleftarrow{\$} \mathsf{KEM.Encap}(\mathsf{PK})$;
$b' \leftarrow \mathcal{A}\,(\mathsf{PK},\,\mathsf{CT}^*,\,\mathsf{K}^*_b)$;
Output b'.

The advantage of $\mathcal{A}$ for breaking the IND-CPA security of KEM is defined as $\mathbf{Adv}^{\mathsf{KEM},\mathcal{A}}_{\mathsf{IND\text{-}CPA}} = \left|\Pr\left[1 \leftarrow \mathbf{Exp}^{\mathsf{KEM},\mathcal{A}}_{\mathsf{IND\text{-}CPA},1}(\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Exp}^{\mathsf{KEM},\mathcal{A}}_{\mathsf{IND\text{-}CPA},0}(\lambda)\right]\right|$.

Definition 3. *The KEM scheme is (t,ϵ)-IND-CPA-secure if for any polynomial time adversary $\mathcal{A}$ that runs in at most time t, we have $\mathbf{Adv}^{KEM,\mathcal{A}}_{IND\text{-}CPA} < \epsilon$.*

Now, the security experiment of IND-CCA, where the additional decapsulation oracle is given to an adversary $\mathcal{A}$, is described as follows.

Experiment $\mathbf{Exp}^{\mathsf{KEM},\mathcal{A}}_{\mathsf{IND\text{-}CCA},b}(\lambda)$
$(\mathsf{PK},\mathsf{SK}) \xleftarrow{\$} \mathsf{KEM.Setup}(\lambda)$;
$\mathsf{K}^*_0 \xleftarrow{\$} \mathcal{K}$; $(\mathsf{K}^*_1,\mathsf{CT}^*) \xleftarrow{\$} \mathsf{KEM.Encap}(\mathsf{PK})$;
$b' \leftarrow \mathcal{A}^{\mathsf{DecO}(\cdot)}\,(\mathsf{PK},\,\mathsf{CT}^*,\,\mathsf{K}^*_b)$;
Output b'.

The oracle $\mathsf{DecO}(\cdot)$ takes as input CT and returns a key $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{PK},\,\mathsf{SK},\,\mathsf{CT})$ with the condition that $\mathcal{A}$ cannot query the challenge ciphertext CT^*. The advantage of $\mathcal{A}$ for breaking the IND-CCA-security of KEM is defined as $\mathbf{Adv}^{\mathsf{KEM},\mathcal{A}}_{\mathsf{IND\text{-}CCA}} = \left|\Pr\left[1 \leftarrow \mathbf{Exp}^{\mathsf{KEM},\mathcal{A}}_{\mathsf{IND\text{-}CCA},1}(\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Exp}^{\mathsf{KEM},\mathcal{A}}_{\mathsf{IND\text{-}CCA},0}(\lambda)\right]\right|$.

Definition 4. *The KEM scheme is (t,ϵ,q_d)-IND-CCA-secure if for any polynomial time adversary $\mathcal{A}$ that runs in time at most t and issues at most q_d decapsulation queries, then we have $\mathbf{Adv}^{KEM,\mathcal{A}}_{IND\text{-}CCA} < \epsilon$.*

3.2. Conversion Method

The conversion method is for a special case of KEM which is compatible with the NIZK schemes. We suppose that the KEM.Encap algorithm takes the public key pk and randomness x as inputs and returns a key k and a ciphertext ct (e.g., $(\mathsf{k},\mathsf{ct}) \leftarrow \mathsf{KEM.Encap}(\mathsf{pk};x)$). Then, we say that a KEM is NIZK-*compatible* if a tuple $(\mathsf{pk},\mathsf{ct},\mathsf{k})$ can be parsed as the statement for proving a relation for discrete logarithms using the witness x. For example, the ElGamal KEM and the linear KEM [21] are NIZK-compatible [12].

Let KEM= $(\mathsf{KEM.Setup},\mathsf{KEM.Encap},\mathsf{KEM.Decap})$ be an IND-CPA-secure KEM and NIZK-compatible. Using the KEM and NIZK scheme, the generic construction of IND-CCA-secure $\mathsf{KEM}' = (\mathsf{KEM.Setup}',\,\mathsf{KEM.Encap}',\,\mathsf{KEM.Decap}')$ by Seo et al. [12] is described as follows.

KEM.Gen$'(\lambda)$: Given security parameter λ, the setup algorithm proceeds as follows.

1. Generate sk and pk by running $\mathsf{KEM.Setup}(\lambda)$.
2. Choose $H_1 : \{0,1\}^* \to \{0,1\}^k$ and $H_2 : \{0,1\}^* \to \{0,1\}^\ell$ for $k, l \in \mathbb{Z}^+$.
3. Output $\mathsf{SK} = \mathsf{sk}$ and $\mathsf{PK} = (H_1, H_2, \mathsf{pk})$.

KEM.Encap$'(\mathsf{PK})$: Given a public key $\mathsf{PK} = (H_1, H_2, \mathsf{pk})$, the Encap algorithm proceeds as follows.

1. Choose the random coins x and compute $(\mathsf{ct},\mathsf{k}) \leftarrow \mathsf{KEM.Encap}\,(\mathsf{pk};x)$.

2. Set $\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}, \mathsf{k})$ and $\mathsf{wit} = x$.
3. Compute $\pi \leftarrow \mathcal{P}^{H_1}(\mathsf{stmt}, \mathsf{wit})$.
   (a) Choose the random value r and compute com.
   (b) Compute $c = H_1(\mathsf{stmt}, \mathsf{com})$ and the relevant s.
   (c) Output the proof $\pi = (s, c)$.
4. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$.
5. Output K and $\mathsf{CT} = (\mathsf{ct}, \pi)$.

KEM.Decap$'(\mathsf{PK}, \mathsf{SK}, \mathsf{CT})$: Given a public key $\mathsf{PK} = (H_1, H_2, \mathsf{pk})$, a ciphertext $\mathsf{CT} = (\mathsf{ct}, \pi = (s, c))$, and a private $\mathsf{SK} = \mathsf{sk}$, the Decap algorithm proceeds as follows.

1. Compute $\mathsf{k} \leftarrow \mathsf{KEM.Decap}(\mathsf{sk}, \mathsf{ct})$.
2. Set $\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}, \mathsf{k})$.
3. If $1 \leftarrow \mathcal{V}^{H_1}(\mathsf{stmt}, \pi)$, go on. Otherwise, abort.
   (a) Deterministically compute com.
   (b) If $c = H_1(\mathsf{stmt}, \mathsf{com})$, output 1. Otherwise, output 0.
4. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$ and output K.

3.3. Security Proof

Theorem 3. *Let H_1 and H_2 be modeled as random oracles with ϵ_{CR}-collision resistance. Suppose that KEM is (t', ϵ')-IND-CPA-secure and NIZK-compatible with ϵ_{ZK}-zero-knowledge and ϵ_{SS}-simulation soundness. Then, the resulting KEM$'$ is (t, ϵ, q_d)-IND-CCA-secure, where*

$$\epsilon \leq 2 \cdot (\epsilon_{CR} + \epsilon_{ZK} + q_d \cdot \epsilon_{SS} + \epsilon') + 1/|\mathcal{K}|, \quad t' \approx t + \mathcal{O}(q_d t_v).$$

Here, t_v is the required time for verification in NIZK, and $\mathcal{K}$ is the key space of KEM.

Proof. We consider a sequence of hybrid games Game 0, ..., Game 9. Game 0 is the actual IND-CCA security game, where an adversary is given a key from the encapsulation algorithm, and Game 9 is the actual game, where an adversary is given a key randomly chosen from a key space for the encapsulation query. Let Win_i denote the event that $\mathcal{A}$ wins in the Game i. Table 1 summarizes the games described below and the properties used to prove indistinguishability between consecutive games.

Game 0. This is the actual IND-CCA security game, which is executed with $b = 1$. Thus, the challenger always returns $(\mathsf{CT}^*, \mathsf{K}^*) = \mathsf{KEM.Encap}(\mathsf{PK})$ for the challenge.

Game 1. Let Coll be the event that a collision of the hash functions H_1 or H_2 occurs. This game is identical to Game 0 except that it aborts when the event Coll occurs during the simulation. Due to the collision resistance of the hash functions, we have

$$|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]| \leq \epsilon_{\mathsf{CR}}.$$

Game 2. This game is identical to Game 1 except that the challenge ciphertext $\mathsf{CT}^* = (\mathsf{ct}^*, \pi^*)$ is computed differently. Instead of using the real proving algorithm of NIZK, the challenger generates the challenge ciphertext using the simulator of NIZK (i.e., $\mathsf{CT}^* = (\mathsf{ct}^*, \pi^*_{\mathsf{sim}})$). Due to the non-interactive zero-knowledge property of NIZK, we have

$$|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq \epsilon_{\mathsf{ZK}}.$$

Game 3. This game is identical to Game 2 except that the decapsulation query is operated differently. In the previous game, for a given ciphertext $\mathsf{CT} = (\mathsf{ct}, \pi = (s, c))$, the challenger computes $\mathsf{k} \leftarrow \mathsf{KEM.Decap}(\mathsf{sk}, \mathsf{ct})$ using sk and returns $\mathsf{K} = H_2\big(\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}, \mathsf{k}), \mathsf{com} = (\mathsf{com}^{\mathsf{ct}}, \mathsf{com}^{\mathsf{k}})\big)$ only if $1 \leftarrow \mathcal{V}^{H_1}(\mathsf{stmt}, \pi)$ holds. In this game, the challenger retrieves a tuple $(\mathsf{stmt}, \mathsf{com}, c)$ in the H_1 hash table such that $c = H_1\big(\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}, \star), \mathsf{com} =$

$(\mathsf{com}_{\mathsf{ct}}, \star)\big)$. If the statement stmt, commitment com, challenge c, and response s are verified (i.e., $1 \leftarrow \mathcal{V}(\mathsf{stmt}, \mathsf{com}, c, s)$), then it returns $\mathsf{K} = H_2(\mathsf{stmt}, \mathsf{com}, \pi)$.

Note that an adversary cannot distinguish Game 3 from Game 2 unless it queries a ciphertext $\tilde{\mathsf{CT}} = (\tilde{\mathsf{ct}}, \tilde{\pi} = (\tilde{s}, \tilde{c}))$ such that proof $\tilde{\pi}$ is verified with a retrieved tuple $(\tilde{\mathsf{stmt}}, \tilde{\mathsf{com}}, \tilde{c})$ in H_1 (i.e., $1 \leftarrow \mathcal{V}(\tilde{\mathsf{stmt}}, \tilde{\mathsf{com}}, \tilde{c}, \tilde{s})$) and $\tilde{\mathsf{stmt}}$ is a false statement. In other words, the adversary must forge a proof $\tilde{\pi}$ for a false statement $\tilde{\mathsf{stmt}}$ to distinguish the games. Then, by the simulation soundness of NIZK with respect to q_d decapsulation queries, we have

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| \leq q_d \cdot \epsilon_{\mathsf{SS}}.$$

Game 4. This game is identical to Game 3 except that the challenge ciphertext is computed differently. In the challenge phase, the challenger randomly selects k' instead of getting k^* from $(\mathsf{ct}^*, \mathsf{k}^*) \leftarrow \mathsf{KEM.Encap}(\mathsf{pk}; x)$. The other procedures are the same as in Game 3. Finally, it outputs $(\mathsf{CT}^*, \mathsf{K}^*)$. We can show that a distinguisher between Game 4 and Game 3 implies an adversary that breaks the IND-CPA security of the underlying KEM scheme. For the sake of simplicity, we prove this later. Then, we have

$$|\Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_3]| \leq \epsilon^{\mathsf{KEM}}_{\mathsf{IND\text{-}CPA}}.$$

Game 5. This game is identical to Game 4 except that it is executed with $b = 0$. In other words, the challenger always returns $\mathsf{K}^* \xleftarrow{\$} \{0,1\}^\ell$ instead of computing $\mathsf{K}^* = H_2(\mathsf{stmt} = (\mathsf{ct}^*, \mathsf{k}'), \mathsf{com} = (\mathsf{com}^{\mathsf{ct}^*}_{\mathsf{sim}}, \mathsf{com}^{\mathsf{k}'}_{\mathsf{sim}}), \pi^*_{\mathsf{sim}} = (s, c))$. The adversary cannot distinguish Game 5 from Game 4 unless it queries the tuple $\big(\mathsf{stmt} = (\mathsf{ct}^*, \mathsf{k}'), \mathsf{com} = (\mathsf{com}^{\mathsf{ct}^*}_{\mathsf{sim}}, \mathsf{com}^{\mathsf{k}'}_{\mathsf{sim}}), \pi^*_{\mathsf{sim}} = (s, c)\big)$ to the H_2 oracle. Note that k' is a random key from $\mathcal{A}$'s view because ct^* is independent of k'. Therefore, the probability that the adversary queries the tuple to the H_2 oracle is at most $1/|\mathcal{K}|$. Then, we have

$$|\Pr[\mathsf{Win}_5] - \Pr[\mathsf{Win}_4]| \leq 1/|\mathcal{K}|.$$

Game 6. This game is identical to Game 5 except that the challenge ciphertext is computed differently. In the challenge phase, the challenger uses back k^* from $(\mathsf{ct}^*, \mathsf{k}^*) \leftarrow \mathsf{KEM.Encap}(\mathsf{pk}; x)$. The other procedures are the same as in Game 5. Finally, it outputs $(\mathsf{CT}^*, \mathsf{K}^*)$. Like the case between Game 4 and Game 3, we can show that a distinguisher between Game 6 and Game 5 implies an adversary that breaks the IND-CPA security of the underlying KEM scheme. For the sake of simplicity, we prove this later. Then, we have

$$|\Pr[\mathsf{Win}_6] - \Pr[\mathsf{Win}_5]| \leq \epsilon^{\mathsf{KEM}}_{\mathsf{IND\text{-}CPA}}.$$

Game 7. This game is identical to Game 6 except that the decapsulation query is operated differently. In this game, the challenger changes back the operation of the decapsulation query to normal. For a given ciphertext $\mathsf{CT} = (\mathsf{ct}, \pi = (s, c))$, the challenger computes $\mathsf{k} \leftarrow \mathsf{KEM.Decap}(\mathsf{sk}, \mathsf{ct})$ using sk and returns $\mathsf{K} = H_2\big(\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}, \mathsf{k}), \mathsf{com} = (\mathsf{com}^{\mathsf{ct}}, \mathsf{com}^{\mathsf{k}})\big)$ only if $1 \leftarrow \mathcal{V}^{H_1}(\mathsf{stmt}, \pi)$ holds. Like the case between Game 3 and Game 2, an adversary cannot distinguish Game 7 from Game 6 unless it submits a ciphertext with a forged proof for a false statement. Then, by the simulation soundness of NIZK with respect to q_d decapsulation queries, we have

$$|\Pr[\mathsf{Win}_7] - \Pr[\mathsf{Win}_6]| \leq q_d \cdot \epsilon_{\mathsf{SS}}.$$

Game 8. This game is identical to Game 7 except that the challenge ciphertext is computed differently. The challenger uses the real proving algorithm for NIZK instead of using simulated proofs. Due to the zero-knowledge property of NIZK, we have

$$|\Pr[\mathsf{Win}_8] - \Pr[\mathsf{Win}_7]| \leq \epsilon_{\mathsf{ZK}}.$$

Game 9. This game is identical to Game 8 but it does not abort when the event Coll occurs during the simulation. Due to the collision resistance of the hash functions, we have

$$|\Pr[\mathsf{Win}_9] - \Pr[\mathsf{Win}_8]| \leq \epsilon_{\mathsf{CR}}.$$

Note that Game 9 is identical to the IND-CCA security game executed with $b = 0$. Then, we have

$$|\Pr[\mathsf{Win}_0] - \Pr[\mathsf{Win}_9]| \leq 2 \cdot (\epsilon_{\mathsf{CR}} + \epsilon_{\mathsf{ZK}} + q_d \cdot \epsilon_{\mathsf{SS}} + \epsilon^{\mathsf{KEM}}_{\mathsf{IND\text{-}CPA}}) + 1/|\mathcal{K}|.$$

Lemma 1. *If there exists a polynomial time distinguisher $\mathcal{D}$ between Game 3 and Game 4, then there exists a polynomial time adversary $\mathcal{A}$ that breaks the IND-CPA security of KEM.*

Proof. Let $\mathcal{D}$ be a distinguisher between Game 3 and Game 4. Then, we show how to construct an $\mathcal{A}$ that wins the IND-CPA security game of KEM using $\mathcal{D}$ with the same advantage.

$\mathcal{A}$ receives a public key pk as input. Then, $\mathcal{A}$ runs $\mathcal{D}$ by simulating Game 3 with changes as follows.

- In the setup phase, $\mathcal{A}$ sends $\mathsf{PK} = (H_1, H_2, \mathsf{pk})$ to $\mathcal{D}$.
- In the challenge phase, the challenge ciphertext and key $(\mathsf{ct}^*, \mathsf{k}^*_b)$ are given to $\mathcal{A}$. Then, $\mathcal{A}$ sets $\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*_b)$ and generates a simulated proof π^*_{sim} on the statement stmt. $\mathcal{A}$ sets $\mathsf{K}^* = H_2(\mathsf{stmt}, \mathsf{com}_{\mathsf{sim}}, \pi^*_{\mathsf{sim}})$, where $\mathsf{com}_{\mathsf{sim}}$ is the deterministically computed commitment from $(\mathsf{stmt}, \pi^*_{\mathsf{sim}})$. $\mathcal{A}$ sends $(\mathsf{CT}^* = (\mathsf{ct}^*, \pi^*_{\mathsf{sim}}), \mathsf{K}^*)$ to $\mathcal{D}$.

In the guess phase, $\mathcal{D}$ outputs its guess. If the guess is "Game 3", then $\mathcal{A}$ outputs 1; otherwise, it output 0.

Note that if $b = 1$, i.e., $\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*_b)$ is a true statement or $\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*_b) \in \mathcal{R}_{\mathcal{L}}$, the above simulation is identical to Game 3 from $\mathcal{D}$'s viewpoint. Otherwise, the above simulation is identical to Game 4. Therefore, we have

$$|\Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_3]| \leq \epsilon_{\mathcal{D}} = \epsilon^{\mathsf{KEM}}_{\mathsf{IND\text{-}CPA}}.$$

□

Lemma 2. *If there exists a polynomial time distinguisher $\mathcal{D}$ between Games 5 and 6, then there exists a polynomial time adversary $\mathcal{A}$ that breaks the IND-CPA security of KEM.*

Proof. Let $\mathcal{D}$ be a distinguisher between Games 5 and 6. Then, we show how to construct an $\mathcal{A}$ that wins the IND-CPA security game of KEM using $\mathcal{D}$ with the same advantage.

$\mathcal{A}$ receives a public key pk as input. Then, $\mathcal{A}$ runs $\mathcal{D}$ by simulating Game 5 with changes as follows.

- In the setup phase, $\mathcal{A}$ sends $\mathsf{PK} = (H_1, H_2, \mathsf{pk})$ to $\mathcal{D}$.
- In the challenge phase, the challenge ciphertext and key $(\mathsf{ct}^*, \mathsf{k}^*_b)$ are given to $\mathcal{A}$. Then, $\mathcal{A}$ sets $\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*_b)$ and generates a simulated proof π^*_{sim} on the statement stmt.

  Moreover, $\mathcal{A}$ randomly chooses $\mathsf{K}^* \xleftarrow{\$} \{0,1\}^\ell$ and sends $(\mathsf{CT}^* = (\mathsf{ct}^*, \pi^*_{\mathsf{sim}}), \mathsf{K}^*)$ to $\mathcal{D}$.

In the guess phase, $\mathcal{D}$ outputs its guess. If the guess is "Game 6", then $\mathcal{A}$ outputs 1; otherwise, it output 0.

Note that if $b = 1$, i.e., $\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}_b^*)$ is a true statement or $\mathsf{stmt} = (\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}_b^*) \in \mathcal{R}_{\mathcal{L}}$, the above simulation is identical to Game 6 from $\mathcal{D}$'s viewpoint. Otherwise, the above simulation is identical to Game 5. Therefore, we have

$$|\Pr[\mathsf{Win}_6] - \Pr[\mathsf{Win}_5]| \leq \epsilon_{\mathcal{D}} = \epsilon^{\mathsf{KEM}}_{\mathsf{IND\text{-}CPA}}.$$

□

This completes the proof of the theorem. □

Table 1. Values or operations of K^*, CT^*, SK, and DecO($\cdot$) in each game and properties ensuring the indistinguishability of consecutive games. RO + SS denotes that the simulator retrieves k from a random oracle table and checks the validity of the ciphertext based on the SS property.

Game	K^*	stmt^*	CT^*	SK	DecO($\cdot$)	Property
0	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi^*_{\mathsf{real}})$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi^*_{\mathsf{real}}$	sk	KEM.Decap($\cdot$)	-
1	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi^*_{\mathsf{real}})$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi^*_{\mathsf{real}}$	sk	KEM.Decap($\cdot$)	CR
2	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi^*_{\mathsf{sim}})$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi^*_{\mathsf{sim}}$	sk	KEM.Decap($\cdot$)	ZK
3	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi^*_{\mathsf{sim}})$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi^*_{\mathsf{sim}}$	-	RO + SS	SS
4	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi^*_{\mathsf{sim}})$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}')$	$\mathsf{ct}^*, \pi^*_{\mathsf{sim}}$	-	RO + SS	IND-CPA
5	$\mathsf{K} \xleftarrow{\$} \{0,1\}^{\ell}$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}')$	$\mathsf{ct}^*, \pi^*_{\mathsf{sim}}$	-	RO + SS	$1/\lvert\mathcal{K}\rvert$
6	$\mathsf{K} \xleftarrow{\$} \{0,1\}^{\ell}$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi^*_{\mathsf{sim}}$	-	RO + SS	IND-CPA
7	$\mathsf{K} \xleftarrow{\$} \{0,1\}^{\ell}$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi^*_{\mathsf{sim}}$	sk	KEM.Decap($\cdot$)	SS
8	$\mathsf{K} \xleftarrow{\$} \{0,1\}^{\ell}$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi^*_{\mathsf{real}}$	sk	KEM.Decap($\cdot$)	ZK
9	$\mathsf{K} \xleftarrow{\$} \{0,1\}^{\ell}$	$(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi^*_{\mathsf{real}}$	sk	KEM.Decap($\cdot$)	CR

4. Conversion Method for CCA-Secure IBKEM

4.1. IBKEM

4.1.1. Syntax

An ID-based key encapsulation mechanism IBKEM consists of four algorithms (IBKEM.Setup, IBKEM.KeyGen, IBKEM.Encap, and IBKEM.Decap). The setup algorithm $(\mathsf{PP}, \mathsf{MSK}) \xleftarrow{\$} \mathsf{IBKEM.Setup}(\lambda)$ takes as input the security parameter λ and outputs public parameters PP and a master secret key MSK. The key generation algorithm $\mathsf{SK}_{\mathsf{ID}} \xleftarrow{\$} \mathsf{IBKEM.KeyGen}(\mathsf{PP}, \mathsf{MSK}, \mathsf{ID})$ takes as input public parameters PP, the master secret key MSK, and an identity ID, and outputs an identity secret key $\mathsf{SK}_{\mathsf{ID}}$. The encapsulation algorithm $(\mathsf{K}, \mathsf{CT}) \xleftarrow{\$} \mathsf{IBKEM.Encap}\ (\mathsf{PP}, \mathsf{ID})$ takes as input PP and ID, and generates a key K and a ciphertext CT. The decapsulation algorithm $\mathsf{K} \leftarrow \mathsf{IBKEM.Decap}(\mathsf{PP}, \mathsf{CT}, \mathsf{SK}_{\mathsf{ID}})$ takes as input public parameters PP, a ciphertext CT, and an identity secret key $\mathsf{SK}_{\mathsf{ID}}$, and then outputs a key K.

4.1.2. Security Model of IBKEM

Here, we define the IND-ID-{CPA,CCA} security of IBKEM by referring to [12]. First, the security experiment of IND-ID-CPA played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is described as follows.

Experiment $\mathbf{Exp}^{\mathsf{IBKEM},\mathcal{A}}_{\mathsf{IND\text{-}ID\text{-}CPA},b}(\lambda)$

$(\mathsf{PP}, \mathsf{MSK}) \xleftarrow{\$} \mathsf{IBKEM.Setup}(\lambda)$;
$\mathsf{ID}^* \leftarrow \mathcal{A}^{\mathsf{KeyGenO}(\cdot)}\ (\mathsf{PP})$;
$\mathsf{K}_0^* \xleftarrow{\$} \mathcal{K}$; $(\mathsf{K}_1^*, \mathsf{CT}^*) \xleftarrow{\$} \mathsf{IBKEM.Encap}(\mathsf{PP}, \mathsf{ID}^*)$;
$b' \leftarrow \mathcal{A}^{\mathsf{KeyGenO}(\cdot)}\ (\mathsf{PP}, \mathsf{CT}^*, \mathsf{K}_b^*)$;
Output b'.

The oracle KeyGen$(\cdot)$ takes as input an identity ID and returns an identity secret key $\mathsf{SK}_{\mathsf{ID}} \xleftarrow{\$}$ IBKEM.KeyGen $(\mathsf{MSK}, \mathsf{ID})$ with the condition that $\mathcal{A}$ is not able to query the target identity ID^*. Then, the advantage of $\mathcal{A}$ for breaking the IND-ID-CPA security of IBKEM is defined as $\mathbf{Adv}^{\mathsf{IBKEM},\mathcal{A}}_{\mathsf{IND\text{-}ID\text{-}CPA}} = \left| \Pr\left[1 \leftarrow \mathbf{Exp}^{\mathsf{IBKEM},\mathcal{A}}_{\mathsf{IND\text{-}ID\text{-}CPA},1}(\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Exp}^{\mathsf{IBKEM},\mathcal{A}}_{\mathsf{IND\text{-}ID\text{-}CPA},0}(\lambda)\right] \right|$.

Definition 5. *The IBKEM scheme is (t, ϵ, q_{id})-IND-ID-CPA-secure if for any polynomial time adversary $\mathcal{A}$ that runs in time at most t and issues at most q_{id} key generation queries, then we have $\mathbf{Adv}^{IBKEM,\mathcal{A}}_{IND\text{-}ID\text{-}CPA} < \epsilon$.*

Now, the security experiment of IND-ID-CCA, where the additional decapsulation oracle is given to an adversary $\mathcal{A}$, is described as follows.

Experiment $\mathbf{Exp}^{\mathsf{IBKEM},\mathcal{A}}_{\mathsf{IND\text{-}ID\text{-}CCA},b}(\lambda)$
$(\mathsf{PP}, \mathsf{MSK}) \xleftarrow{\$} \mathsf{IBKEM.Setup}(\lambda)$;
$\mathsf{ID}^* \leftarrow \mathcal{A}^{\mathsf{KeyGenO}(\cdot),\mathsf{DecO}(\cdot)}(\mathsf{PP})$;
$\mathsf{K}^*_0 \xleftarrow{\$} \mathcal{K}$; $(\mathsf{K}^*_1, \mathsf{CT}^*) \xleftarrow{\$} \mathsf{IBKEM.Encap}(\mathsf{PP}, \mathsf{ID}^*)$;
$b' \leftarrow \mathcal{A}^{\mathsf{KeyGenO}(\cdot),\mathsf{DecO}(\cdot)}(\mathsf{PP}, \mathsf{CT}^*, \mathsf{K}^*_b)$;
Output b'.

The oracle KeyGen$(\cdot)$ is the same as that in the IND-ID-CPA security game. The additional oracle DecO$(\cdot)$ takes as input ID and CT and returns a key K $\leftarrow$ IBKEM.Decap (PP, CT, $\mathsf{SK}_{\mathsf{ID}}$), where $\mathsf{SK}_{\mathsf{ID}} \xleftarrow{\$}$ IBKEM.KeyGen$(\mathsf{PP}, \mathsf{MSK}, \mathsf{ID})$. The restriction on the oracle DecO$(\cdot)$ is that $\mathcal{A}$ is not able to query the pair $(\mathsf{ID}^*, \mathsf{CT}^*)$. The advantage of $\mathcal{A}$ for breaking the IND-ID-CCA-security of IBKEM is defined as $\mathbf{Adv}^{\mathsf{IBKEM},\mathcal{A}}_{\mathsf{IND\text{-}ID\text{-}CCA}} = \left| \Pr\left[1 \leftarrow \mathbf{Exp}^{\mathsf{IBKEM},\mathcal{A}}_{\mathsf{IND\text{-}ID\text{-}CCA},1}(\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Exp}^{\mathsf{IBKEM},\mathcal{A}}_{\mathsf{IND\text{-}ID\text{-}CCA},0}(\lambda)\right] \right|$.

Definition 6. *The IBKEM scheme is $(t, \epsilon, q_{id}, q_d)$-IND-ID-CCA-secure if for any polynomial time adversary $\mathcal{A}$ that runs in time at most t and issues at most q_{id} key generation queries and q_d decapsulation queries, then we have $\mathbf{Adv}^{IBKEM,\mathcal{A}}_{IND\text{-}ID\text{-}CCA} < \epsilon$.*

4.2. Conversion Method

Similar to the previous KEM, this conversion method is for a special case of IBKEM which is compatible with the NIZK schemes. We suppose that the IBKEM.Encap algorithm takes public parameters pp, an identity ID, and random coins x as inputs and returns a key k and a ciphertext ct $\left(\text{e.g., } (\mathsf{k}, \mathsf{ct}) \leftarrow \mathsf{IBKEM.Encap}(\mathsf{pp}, \mathsf{ID}; x)\right)$. Then, we say that an IBKEM scheme is NIZK-*compatible* if a tuple $(\mathsf{pp}, \mathsf{ID}, \mathsf{ct}, \mathsf{k})$ can be parsed as the statement for proving a relation for discrete logarithms using the witness x. For instance, the Boneh–Franklin IBKEM and Boneh–Boyen IBKEM are NIZK-compatible [12].

Let IBKEM = (IBKEM.Setup, IBKEM.KeyGen, IBKEM.Encap, IBKEM.Decap) be IND-ID-CPA-secure and NIZK-compatible. Using the IBKEM and NIZK scheme, the generic construction of the CCA-secure IBKEM′ = (IBKEM.Setup′, IBKEM.KeyGen′, IBKEM.Encap′, IBKEM.Decap′) by Seo et al. is described as follows.

IBKEM.Setup′(λ): Given security parameter λ, the setup algorithm proceeds as follows.

1. Generate msk and pp by running IBKEM.Setup(λ).
2. Choose $H_1 : \{0,1\}^* \to \{0,1\}^k$ and $H_2 : \{0,1\}^* \to \{0,1\}^\ell$ for $k, l \in \mathbb{Z}^+$.
3. Output MSK = msk and PP = (pp, H_1, H_2).

IBKEM.KeyGen′(PP, MSK, ID): Given public parameters PP = (pp, H_1, H_2), a master secret key MSK, and an identity ID, the KeyGen algorithm proceeds as follows.

1. Compute $\mathsf{sk}_{\mathsf{ID}} \xleftarrow{\$}$ IBKEM.KeyGen$(\mathsf{pp}, \mathsf{msk}, \mathsf{ID})$.
2. Set $\mathsf{SK}_{\mathsf{ID}} = \mathsf{sk}_{\mathsf{ID}}$ and output $\mathsf{SK}_{\mathsf{ID}}$.

IBKEM.Encap$'(\mathsf{PP}, \mathsf{ID})$: Given public parameters $\mathsf{PP} = (\mathsf{pp}, H_1, H_2)$ and an identity ID, the Encap algorithm proceeds as follows.

1. Choose the randomness x and compute $(\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{IBKEM.Encap}\ (\mathsf{pp}, \mathsf{ID}; x)$.
2. Set $\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}, \mathsf{ct}, \mathsf{k})$ and $\mathsf{wit} = x$.
3. Run $\pi \leftarrow \mathcal{P}^{H_1}(\mathsf{stmt}, \mathsf{wit})$.
   (a) Choose the randomness r and compute com.
   (b) Compute $c = H_1(\mathsf{stmt}, \mathsf{com})$ and the relevant s.
   (c) Output the proof $\pi = (s, c)$.
4. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$.
5. Output K and $\mathsf{CT} = (\mathsf{ct}, \pi)$.

IBKEM.Decap$'(\mathsf{PP}, \mathsf{CT}, \mathsf{SK}_{\mathsf{ID}})$: Given a public key $\mathsf{PP} = (\mathsf{pp}, H_1, H_2)$, a ciphertext $\mathsf{CT} = (\mathsf{ct}, \pi = (s, c))$, and a private $\mathsf{SK}_{\mathsf{ID}} = \mathsf{sk}_{\mathsf{ID}}$, the Decap algorithm proceeds as follows.

1. Compute $\mathsf{k} \leftarrow \mathsf{IBKEM.Decap}(\mathsf{pp}, \mathsf{ct}, \mathsf{sk}_{\mathsf{ID}})$.
2. Set $\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}, \mathsf{ct}, \mathsf{k})$.
3. If $1 \leftarrow \mathcal{V}^{H_1}(\mathsf{stmt}, \pi)$, go on. Otherwise, abort.
   (a) Deterministically compute com.
   (b) If $c = H_1(\mathsf{stmt}, \mathsf{com})$, output 1. Otherwise, output 0.
4. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$ and output K.

4.3. Security Proof

Theorem 4. *Let H_1 and H_2 be modeled as random oracles with ϵ_{CR}-collision resistance. Suppose that IBKEM is (t', ϵ', q'_{id})-IND-ID-CPA-secure and NIZK-compatible with ϵ_{ZK}-zero-knowledge and ϵ_{SS}-simulation soundness. Then, the resulting IBKEM′ is $(t, \epsilon, q_{id}, q_d)$-IND-ID-CCA-secure, where*

$$\epsilon \leq 2 \cdot (\epsilon_{CR} + \epsilon_{ZK} + q_d \cdot \epsilon_{SS} + \epsilon') + 1/|\mathcal{K}|, \quad t' \approx t + \mathcal{O}(q_d t_v).$$

Here, t_v is the required time for verification in the NIZK and $\mathcal{K}$ is the key space of IBKEM.

Proof. We consider a sequence of hybrid games Game 0,..., Game 9. Game 0 is the actual IND-ID-CCA security game, where an adversary is given a key from the encapsulation algorithm for the encapsulation query, and Game 9 is the actual IND-ID-CCA game, where an adversary is given a key randomly chosen from a key space for the encapsulation query. Let Win_i denote the event that $\mathcal{A}$ wins in Game i. Table 2 summarizes the games described below and the properties used to prove indistinguishability between consecutive games.

Game 0. This is the IND-ID-CCA security game executed with $b = 1$. Thus, the challenger always returns $(\mathsf{CT}^*, \mathsf{K}^*) = \mathsf{IBKEM.Encap}'(\mathsf{PP}, \mathsf{ID}^*)$ for the challenge.

Game 1. Let Coll be the event that a collision of the hash functions H_1 or H_2 occurs. This game is identical to Game 0 except that it aborts when the event Coll occurs during the simulation. Due to the collision resistance of the hash functions, we have

$$|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]| \leq \epsilon_{\mathsf{CR}}.$$

Game 2. This game is identical to Game 1 except that the challenge ciphertext is computed differently. Instead of using the real proving algorithm of NIZK, the challenger generates challenge ciphertext using the simulator of NIZK (i.e., $\mathsf{CT}^* = (\mathsf{ct}^*, \pi^*_{\mathsf{sim}})$). Due to the non-interactive zero-knowledge property of NIZK, we have

$$|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq \epsilon_{\mathsf{ZK}}.$$

Game 3. This game is identical to Game 2 except that the decapsulation query is operated differently. In the previous game, for a given pair $(\mathsf{CT} = (\mathsf{ct}, \pi = (s, c)), \mathsf{ID})$, the challenger runs $\mathsf{SK}_{\mathsf{ID}} \xleftarrow{\$} \mathsf{IBKEM.KeyGen}'\ (\mathsf{PP}, \mathsf{MSK}, \mathsf{ID})$, computes $\mathsf{K} \xleftarrow{\$} \mathsf{IBKEM.Decap}'(\mathsf{PP}, \mathsf{CT}, \mathsf{SK}_{\mathsf{ID}})$,

and returns K. In this game, if $\mathsf{ID} = \mathsf{ID}^*$, the challenger retrieves a tuple $(\mathsf{stmt}, \mathsf{com}, c)$ in the H_1 hash table such that $c = H_1\big(\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}, \star), \mathsf{com} = (\mathsf{com}_{\mathsf{ct}}, \star)\big)$; then, it returns $\mathsf{K} = H_2(\mathsf{stmt}, \mathsf{com}, \pi)$ only when these statement stmt, commitment com, challenge c, and response s are verified (i.e., $1 \leftarrow \mathcal{V}(\mathsf{stmt}, \mathsf{com}, c, s)$). Otherwise, if $\mathsf{ID} \neq \mathsf{ID}^*$, it generates $\mathsf{SK}_{\mathsf{ID}}$ using MSK and decapsulates the ciphertext CT using $\mathsf{SK}_{\mathsf{ID}}$, as in the previous game.

Note that an adversary cannot distinguish Game 3 from Game 2 unless it queries a ciphertext $\tilde{\mathsf{CT}} = (\tilde{\mathsf{ct}}, \tilde{\pi} = (\tilde{s}, \tilde{c}))$ such that proof $\tilde{\pi}$ is verified with a retrieved tuple $(\tilde{\mathsf{stmt}}, \tilde{\mathsf{com}}, \tilde{c})$ in H_1 (i.e., $1 \leftarrow \mathcal{V}(\tilde{\mathsf{stmt}}, \tilde{\mathsf{com}}, \tilde{c}, \tilde{s})$) and $\tilde{\mathsf{stmt}}$ is a false statement. In other words, the adversary must forge a proof $\tilde{\pi}$ for a false statement $\tilde{\mathsf{stmt}}$ to distinguish the games. Then, by the simulation soundness of NIZK with respect to q_d decapsulation queries, we have

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| \leq q_d \cdot \epsilon_{\mathsf{SS}}.$$

Game 4. This game is identical to Game 3 except that the challenge ciphertexts are computed differently. In the challenge phase, the challenger randomly selects k' instead of computing $\mathsf{k}^* \leftarrow \mathsf{IBKEM.Encap}(\mathsf{pp}, \mathsf{ID}^*; x)$. The other procedures are the same as in Game 3. Finally, it outputs $(\mathsf{CT}^*, \mathsf{K}^*)$. We can show that a distinguisher between Game 4 and Game 3 implies an adversary that breaks the IND-ID-CPA security of the IBKEM scheme. For the sake of simplicity, we prove this later. Then, we have

$$|\Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_3]| \leq \epsilon^{\mathsf{IBKEM}}_{\mathsf{IND\text{-}ID\text{-}CPA}}.$$

Game 5. This game is identical to Game 4 except that it is executed with $b = 0$. In other words, the challenger always returns $\mathsf{K}^* \xleftarrow{\$} \{0,1\}^\ell$ instead of computing $\mathsf{K}^* = H_2(\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}'), \mathsf{com} = (\mathsf{com}^{\mathsf{ct}^*}_{\mathsf{sim}}, \mathsf{com}^{\mathsf{k}'}_{\mathsf{sim}}), \pi^*_{\mathsf{sim}} = (s, c))$. $\mathcal{A}$ cannot distinguish Game 5 from Game 4 unless $\mathcal{A}$ queries the tuple $\big(\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}'), \mathsf{com} = (\mathsf{com}^{\mathsf{ct}^*}_{\mathsf{sim}}, \mathsf{com}^{\mathsf{k}'}_{\mathsf{sim}}), \pi^*_{\mathsf{sim}} = (s, c)\big)$ to the H_2 oracle. Note that k' is random from $\mathcal{A}$'s viewpoint because ct^* is independent of k'. Therefore, the probability that $\mathcal{A}$ queries the tuple to the H_2 oracle is $1/|\mathcal{K}|$. Then, we have

$$|\Pr[\mathsf{Win}_5] - \Pr[\mathsf{Win}_4]| \leq 1/|\mathcal{K}|.$$

Game 6. This game is identical to Game 5 except that the challenge ciphertexts are computed differently. In the challenge phase, the challenger uses back $\mathsf{k}^* \leftarrow \mathsf{IBKEM.Encap}(\mathsf{pp}, \mathsf{ID}^*; x)$. The other procedures are the same as in Game 5. Finally, it outputs $(\mathsf{CT}^*, \mathsf{K}^*)$. Like the case between Game 4 and Game 3, we can show that a distinguisher between Game 6 and Game 5 implies an adversary that breaks the IND-ID-CPA security of the IBKEM scheme. For the sake of simplicity, we prove this later. Then, we have

$$|\Pr[\mathsf{Win}_6] - \Pr[\mathsf{Win}_5]| \leq \epsilon^{\mathsf{IBKEM}}_{\mathsf{IND\text{-}ID\text{-}CPA}}.$$

Game 7. This game is identical to Game 6 except that the decapsulation query is operated differently. In this game, the challenger changes the operation of decapsulation query back to normal. For a given pair $(\mathsf{CT}, \mathsf{ID})$, the challenger generates $\mathsf{SK}_{\mathsf{ID}}$ using MSK and decapsulates the ciphertext CT using $\mathsf{SK}_{\mathsf{ID}}$. Like the case between Game 3 and Game 2, $\mathcal{A}$ cannot distinguish Game 7 from Game 6 unless it submits a ciphertext with a forged proof for a false statement. Then, by the simulation soundness of NIZK with respect to q_d decapsulation queries, we have

$$|\Pr[\mathsf{Win}_7] - \Pr[\mathsf{Win}_6]| \leq q_d \cdot \epsilon_{\mathsf{SS}}.$$

Game 8. This game is identical to Game 7 except that the challenge ciphertext is computed differently. The challenger uses the real proving algorithm for NIZK instead of using simulated proofs. Due to the zero-knowledge property of NIZK, we have

$$|\Pr[\mathsf{Win}_8] - \Pr[\mathsf{Win}_7]| \leq \epsilon_{\mathsf{ZK}}.$$

Game 9. This game is identical to Game 8 but it does not abort when the event Coll occurs during the simulation. Due to the collision resistance of the hash functions, we have

$$|\Pr[\mathsf{Win}_9] - \Pr[\mathsf{Win}_8]| \leq \epsilon_{\mathsf{CR}}.$$

Note that Game 9 is identical to the IND-ID-CCA security game executed with $b = 0$. Then, we have

$$|\Pr[\mathsf{Win}_0] - \Pr[\mathsf{Win}_9]| \leq 2 \cdot (\epsilon_{\mathsf{CR}} + \epsilon_{\mathsf{ZK}} + q_d \cdot \epsilon_{\mathsf{SS}} + \epsilon^{\mathsf{IBKEM}}_{\mathsf{IND\text{-}ID\text{-}CPA}}) + 1/|\mathcal{K}^{\mathsf{KEM}}|.$$

Lemma 3. *If there exists a polynomial time distinguisher $\mathcal{D}$ between Game 3 and Game 4, then there exists a polynomial time adversary $\mathcal{A}$ that breaks the IND-ID-CPA security of IBKEM.*

Proof. Let $\mathcal{D}$ be a distinguisher between Game 3 and Game 4. Then, we show how to construct $\mathcal{A}$ so that it wins the IND-ID-CPA security game of IBKEM using $\mathcal{D}$ with the same advantage.

$\mathcal{A}$ receives as input a public parameter pp. Then, $\mathcal{A}$ runs $\mathcal{D}$ by simulating Game 3 with changes as follows.

- In the setup phase, $\mathcal{A}$ sends $\mathsf{PP} = (\mathsf{pp}, H_1, H_2)$ to $\mathcal{D}$.
- When asked a decapsulation query (CT,ID), if $\mathsf{ID} \neq \mathsf{ID}^*$, $\mathcal{A}$ queries ID for key generation oracle of its challenger, decapsulates CT, and returns it to $\mathcal{D}$. Otherwise, if $\mathsf{ID} \neq \mathsf{ID}^*$, the procedure is the same as in Game 3.
- In the challenge phase, the distinguisher $\mathcal{D}$ sends ID^* to $\mathcal{A}$. Then, $\mathcal{A}$ sends ID^* to its challenger and receives the challenge ciphertext and key pair. Given the pair $(\mathsf{ct}^*, \mathsf{k}^*_b)$, $\mathcal{A}$ sets $\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*_b)$ and generates simulated proof π^*_{sim} on the statement stmt. Furthermore, $\mathcal{A}$ sets $\mathsf{K}^* = H_2(\mathsf{stmt}, \mathsf{com}_{\mathsf{sim}}, \pi^*_{\mathsf{sim}})$, where $\mathsf{com}_{\mathsf{sim}}$ is deterministically computed commitment from $(\mathsf{stmt}, \pi^*_{\mathsf{sim}})$, and sends $(\mathsf{CT}^* = (\mathsf{ct}^*, \pi^*_{\mathsf{sim}}), \mathsf{K}^*)$ to $\mathcal{D}$.

In the guess phase, $\mathcal{D}$ outputs its guess. If the guess is "Game 3", $\mathcal{A}$ outputs 1; otherwise, it output 0.

Note that if $b = 1$, i.e., $\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*_b)$ is a true statement or $\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*_b) \in \mathcal{R}_{\mathcal{L}}$, the above simulation is identical to Game 3 from $\mathcal{D}$'s viewpoint. Otherwise, the above simulation is identical to Game 4. Therefore, we have

$$|\Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_3]| \leq \epsilon_{\mathcal{D}} = \epsilon^{\mathsf{IBKEM}}_{\mathsf{IND\text{-}ID\text{-}CPA}}.$$

□

Lemma 4. *If there exists a polynomial time distinguisher $\mathcal{D}$ between Game 5 and Game 6, then there exists a polynomial time adversary $\mathcal{A}$ that breaks the IND-ID-CPA security of IBKEM.*

Proof. Let $\mathcal{D}$ be a distinguisher between Game 5 and Game 6. Then, we show how to construct $\mathcal{A}$ that wins the IND-ID-CPA security game of IBKEM using $\mathcal{D}$ with the same advantage.

$\mathcal{A}$ receives as input a public parameter pp. Then, $\mathcal{A}$ runs $\mathcal{D}$ by simulating Game 5 with changes as follows.

- In the setup phase, $\mathcal{A}$ sends $\mathsf{PP} = (\mathsf{pp}, H_1, H_2)$ to $\mathcal{D}$.
- When asked a decapsulation query (CT,ID), if $\mathsf{ID} \neq \mathsf{ID}^*$, $\mathcal{A}$ queries ID for key generation oracle of its challenger, decapsulates CT, and returns it to $\mathcal{D}$. Otherwise, if $\mathsf{ID} \neq \mathsf{ID}^*$, the procedure is the same as in Game 3.
- In the challenge phase, the distinguisher $\mathcal{D}$ sends ID^* to $\mathcal{A}$. Then, $\mathcal{A}$ sends ID^* to its challenger and receives the challenge ciphertext and key pair. Given the pair $(\mathsf{ct}^*, \mathsf{k}_b^*)$, $\mathcal{A}$ sets $\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}_b^*)$ and generates simulated proof π_{sim}^* on the statement stmt. Then, $\mathcal{A}$ randomly chooses $\mathsf{K}^* \xleftarrow{\$} \{0,1\}^\ell$ and sends $(\mathsf{CT}^* = (\mathsf{ct}^*, \pi_{\mathsf{sim}}^*), \mathsf{K}^*)$ to $\mathcal{D}$.

In the guess phase, $\mathcal{D}$ outputs its guess. If the guess is "Game 6", $\mathcal{A}$ outputs 1; otherwise, it output 0.

Note that if $b = 1$, i.e., $\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}_b^*)$ is a true statement or $\mathsf{stmt} = (\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}_b^*) \in \mathcal{R}_{\mathcal{L}}$, the above simulation is identical to Game 6 from $\mathcal{D}$'s viewpoint. Otherwise, the above simulation is identical to Game 5. Therefore, we have

$$|\Pr[\mathsf{Win}_6] - \Pr[\mathsf{Win}_5]| \leq \epsilon_{\mathcal{D}} = \epsilon_{\mathsf{IND\text{-}ID\text{-}CPA}}^{\mathsf{IBKEM}}.$$

□

This completes the proof of the theorem. □

Table 2. Values or operations of K^*, CT^*, SK, and $\mathsf{DecO}(\cdot)$ in each game and properties ensuring the indistinguishability of consecutive games. RO + SS denotes that the simulator retrieves k from a random oracle table and checks the validity of the ciphertext based on the SS property.

Game	K^*	stmt^*	CT^*	SK	$\mathsf{DecO}(\cdot)$	**Property**
0	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi_{\mathsf{real}}^*)$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi_{\mathsf{real}}^*$	sk	$\mathsf{IBKEM.Decap}(\cdot)$	-
1	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi_{\mathsf{real}}^*)$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi_{\mathsf{real}}^*$	sk	$\mathsf{IBKEM.Decap}(\cdot)$	CR
2	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi_{\mathsf{sim}}^*)$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi_{\mathsf{sim}}^*$	sk	$\mathsf{IBKEM.Decap}(\cdot)$	ZK
3	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi_{\mathsf{sim}}^*)$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi_{\mathsf{sim}}^*$	-	RO + SS	SS
4	$H_2(\mathsf{stmt}^*, \mathsf{com}, \pi_{\mathsf{sim}}^*)$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}')$	$\mathsf{ct}^*, \pi_{\mathsf{sim}}^*$	-	RO + SS	IND-ID-CPA
5	$\mathsf{K} \xleftarrow{\$} \{0,1\}^\ell$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}')$	$\mathsf{ct}^*, \pi_{\mathsf{sim}}^*$	-	RO + SS	$1/\lvert\mathcal{K}\rvert$
6	$\mathsf{K} \xleftarrow{\$} \{0,1\}^\ell$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi_{\mathsf{sim}}^*$	-	RO + SS	IND-ID-CPA
7	$\mathsf{K} \xleftarrow{\$} \{0,1\}^\ell$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi_{\mathsf{sim}}^*$	sk	$\mathsf{IBKEM.Decap}(\cdot)$	SS
8	$\mathsf{K} \xleftarrow{\$} \{0,1\}^\ell$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi_{\mathsf{real}}^*$	sk	$\mathsf{IBKEM.Decap}(\cdot)$	ZK
9	$\mathsf{K} \xleftarrow{\$} \{0,1\}^\ell$	$(\mathsf{pp}, \mathsf{ID}^*, \mathsf{ct}^*, \mathsf{k}^*)$	$\mathsf{ct}^*, \pi_{\mathsf{real}}^*$	sk	$\mathsf{IBKEM.Decap}(\cdot)$	CR

5. Application

In this section, we review the applications described by Seo et al. [12].

5.1. CCA-Secure ElGamal KEM

Let $\mathsf{GGen}^{\mathsf{DDH}}(\lambda)$ be a group generator generating a group $\mathcal{G} = (\mathbb{G}, p, g)$, where the DDH assumption holds. In the ElGamal KEM, a public key consists of two group elements $(g, h = g^x) \in \mathbb{G}^2$ with corresponding private key $x \in \mathbb{Z}_p$. The encapsulation is done by computing $\mathsf{ct} = g^w \in \mathbb{G}$ and $\mathsf{k} = h^w \in \mathbb{G}$, and decapsulation is done by computing $\mathsf{k} = (\mathsf{ct})^x \in \mathbb{G}$. Based on this scheme, the transformed IND-CCA-secure scheme is described as follows.

KEM.Gen(λ): For a security parameter λ, the setup algorithm proceeds as follows.

1. Generate $\mathcal{G} = (\mathbb{G}, p, g)$ by running $\mathsf{GGen}(\lambda)$.
2. Choose a random exponent $x \in \mathbb{Z}_p$ and set $h = g^x \in \mathbb{G}$.

3. Choose hash functions $H_1 : \{0,1\}^* \to \{0,1\}^k \subseteq \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^\ell$.
4. Output $\mathsf{SK} = x$ and $\mathsf{PK} = (\mathbb{G}, p, g, h, H_1, H_2)$.

KEM.Encap(PK): For a public key $\mathsf{PK} = (\mathbb{G}, p, g, h, H_1, H_2)$, the Encap algorithm proceeds as follows.

1. Choose the random exponent $w \in \mathbb{Z}_p$.
2. Set $\mathsf{stmt} = (g, h, g^w, h^w)$ and $\mathsf{wit} = w$.
3. Compute $\pi \leftarrow \mathcal{P}^{H_1}(\mathsf{stmt}, \mathsf{wit})$.
   (a) Choose the random exponent $r \in \mathbb{Z}_p$ and set $\mathsf{com} = (g^r, h^r) \in \mathbb{G}^2$.
   (b) Set $c = H_1(\mathsf{stmt}, \mathsf{com})$ and $s = r + wc \in \mathbb{Z}_p$.
   (c) Output the proof $\pi = (s, c)$.
4. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$.
5. Output K and $\mathsf{CT} = (g^w, \pi)$.

KEM.Decap(PK, CT, SK): For a public key $\mathsf{PK} = (\mathbb{G}, p, g, h, H_1, H_2)$, a ciphertext $\mathsf{CT} = (\mathsf{ct}, \pi = (s, c))$, and a private $\mathsf{SK} = x$, the Decap algorithm proceeds as follows.

1. Set $\mathsf{k} = (\mathsf{ct})^x$ and $\mathsf{stmt} = (g, h, \mathsf{ct}, \mathsf{k}) \in \mathbb{G}^4$.
2. If $1 \leftarrow \mathcal{V}^{H_1}(\mathsf{stmt}, \pi)$, go on. Otherwise, abort.
   (a) Set $\mathsf{com} = (g^s \cdot \mathsf{ct}^{-c}, h^s \cdot \mathsf{k}^{-c}) \in \mathbb{G}^2$.
   (b) If $c = H_1(\mathsf{stmt}, \mathsf{com})$, output 1. Otherwise, output 0.
3. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$ and output K.

Theorem 5. *Suppose that the $(t_{ddh}, \epsilon_{ddh})$ DDH assumption holds in $\mathcal{G}$, and H_1 and H_2 are random oracles. Then, the ElGamal KEM is (t, ϵ, q_d)-IND-CCA-secure, where*

$$\epsilon \leq \frac{\epsilon_{ddh}}{2} + \frac{q_{H_1}^2}{2^{k-1}} + \frac{q_{H_2}^2}{2^{\ell-1}} + \frac{2q_{H_1}^2 + 2q_d \cdot (q_{H_1} + q_{H_1}^2) + 1}{p}, t_{ddh} \approx t + \mathcal{O}(q_d t_e).$$

Here, $\{q_{H_1}, q_{H_2}\}$ are the numbers of $\{H_1, H_2\}$ queries, t_e is the required time for exponentiation in $\mathcal{G}$, and p is the group order.

Proof. We can obtain the proof by applying Theorem 3 to the facts that $\epsilon' \leq \epsilon_{ddh}$, $t_{ddh} \approx t'$, $\epsilon_{\mathsf{ZK}} \leq q_{H_1}^2/p$, and $\epsilon_{\mathsf{SS}} \leq (q_{H_1} + q_{H_1}^2)/p$. Moreover, we use the fact that t_v becomes similar to t_e in Theorem 3. □

5.2. CCA-Secure Linear KEM

Let $\mathsf{GGen}^{\mathsf{DLIN}}(\lambda)$ be a group generator generating a group $\mathcal{G} = (\mathbb{G}, p)$, where the decision linear assumption [21] holds. In the linear KEM, a public key consists of three group elements $(g_1, g_2, h) \in \mathbb{G}^3$ such that $h = g_1^{x_1} = g_2^{x_2}$ with corresponding private key $(x_1, x_2) \in \mathbb{Z}_p^2$. The encapsulation is done by computing $\mathsf{ct} = (g_1^{w_1}, g_2^{w_2}) \in \mathbb{G}$ and $\mathsf{k} = h^{w_1+w_2} \in \mathbb{G}$, and decapsulation is done by computing $\mathsf{k} = (g_1^{w_1})^{x_1}(g_2^{w_2})^{x_2} \in \mathbb{G}$. Based on this scheme, the transformed IND-CCA-secure scheme is described as follows.

KEM.Gen(λ): For security parameter λ, the setup algorithm proceeds as follows.

1. Generate $\mathcal{G} = (\mathbb{G}, p)$ by running $\mathsf{GGen}^{\mathsf{DLIN}}(\lambda)$.
2. Choose a random generator $g_1 \in \mathbb{G}$ and a random exponent $x_1 \in \mathbb{Z}_p$.
3. Choose a random exponent t and set $g_2 = g_1^t \in \mathbb{G}$ and $x_2 = x_1 t^{-1} \in \mathbb{Z}_p$ such that $g_1^{x_1} = g_2^{x_2} = h$ for some $h \in \mathbb{G}$.
4. Choose hash functions $H_1 : \{0,1\}^* \to \{0,1\}^k \subseteq \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^\ell$.
5. Output $\mathsf{SK} = (x_1, x_2)$ and $\mathsf{PK} = (\mathbb{G}, p, g_1, g_2, h, H_1, H_2)$.

KEM.Encap(PK): For a public key $\mathsf{PK} = (\mathbb{G}, p, g_1, g_2, h, H_1, H_2)$, the Encap algorithm proceeds as follows.

1. Choose random exponents $w_1, w_2 \in \mathbb{Z}_p$.
2. Set $\mathsf{stmt} = (g_1, g_2, h, g_1^{w_1}, g_2^{w_2}, h^{w_1+w_2})$ and $\mathsf{wit} = (w_1, w_2)$.

3. Compute $\pi \leftarrow \mathcal{P}^{H_1}(\mathsf{stmt}, \mathsf{wit})$.
   (a) Choose random exponents $r_1, r_2 \in \mathbb{Z}_p$.
   (b) Set $\mathsf{com} = (g_1^{r_1}, g_2^{r_2}, h^{r_1+r_2}) \in \mathbb{G}^3$.
   (c) Compute $c = H_1(\mathsf{stmt}, \mathsf{com})$ and $s_1 = r_1 + w_1 c$, $s_2 = r_2 + w_2 c \in \mathbb{Z}_p$.
   (d) Output the proof $\pi = (s_1, s_2, c)$.
4. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$.
5. Output K and $\mathsf{CT} = (g_1^{w_1}, g_2^{w_2}, \pi)$.

KEM.Decap(PK, CT, SK): For a public key $\mathsf{PK} = (\mathbb{G}, p, g_1, g_2, h, H_1, H_2)$, a ciphertext $\mathsf{CT} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi = (s_1, s_2, c))$, and a private $\mathsf{SK} = (x_1, x_2)$, the Decap algorithm proceeds as follows.

1. Compute $\mathsf{k} = \mathsf{ct}_1^{x_1} \mathsf{ct}_2^{x_2} \in \mathbb{G}$ and set $\mathsf{stmt} = (g_1, g_2, h, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{k})$.
2. If $1 \leftarrow \mathcal{V}^{H_1}(\mathsf{stmt}, \pi)$, go on. Otherwise, abort.
   (a) Set $\mathsf{com} = (g_1^{s_1} \cdot \mathsf{ct}_1^{-c}, g_2^{s_2} \cdot \mathsf{ct}_2^{-c}, h^{(s_1+s_2)} \cdot \mathsf{k}^{-c}) \in \mathbb{G}^3$.
   (b) If $c = H_1(\mathsf{stmt}, \mathsf{com})$, output 1. Otherwise, output 0.
3. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$ and output K.

Theorem 6. *Suppose that the $(t_{dlin}, \epsilon_{dlin})$ decision linear assumption holds in $\mathcal{G}$, and H_1 and H_2 are random oracles. Then, the linear KEM is (t, ϵ, q_d)-IND-CCA-secure, where*

$$\epsilon \leq \frac{\epsilon_{dlin}}{2} + \frac{q_{H_1}^2}{2^{k-1}} + \frac{q_{H_2}^2}{2^{\ell-1}} + \frac{2q_{H_1}^2 + 2q_d \cdot (q_{H_1} + q_{H_1}^2) + 1}{p}, t_{dlin} \approx t + \mathcal{O}(q_d t_e).$$

Here, $\{q_{H_1}, q_{H_2}\}$ are the numbers of $\{H_1, H_2\}$ queries, t_e is the required time for exponentiation in $\mathcal{G}$, and p is the group order.

Proof. We can obtain the proof by applying Theorem 3 to the facts that $\epsilon' \leq \epsilon_{dlin}$, $t_{dlin} \approx t'$, $\epsilon_{\mathsf{ZK}} \leq q_{H_1}^2/p$, and $\epsilon_{\mathsf{SS}} \leq (q_{H_1} + q_{H_1}^2)/p$. Moreover, we use the fact that t_v becomes similar to t_e in Theorem 3. □

5.3. CCA-Secure Boneh–Franklin IBKEM

Let $\mathsf{GGen}^{\mathsf{DBDH}}(\lambda)$ be a bilinear group generator generating a group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, where the DBDH assumption [22,23] holds. In the Boneh–Franklin IBKEM [22], public parameters consist of two group elements $(g, h) \in \mathbb{G}_2^2$ such that $h = g^x$ with corresponding master secret key $x \in \mathbb{Z}_p$. For an identity ID, the secret key is generated by computing $\mathsf{sk}_{\mathsf{ID}} = H(\mathsf{ID})^x \in \mathbb{G}_1$. The encapsulation is done by computing $\mathsf{ct} = g^w \in \mathbb{G}_2$ and $\mathsf{k} = e(H(\mathsf{ID}), h)^w \in \mathbb{G}_T$, and decapsulation is done by computing $\mathsf{k} = e(H(\mathsf{ID})^x, \mathsf{ct}) \in \mathbb{G}_T$. Based on this scheme, the transformed IND-CCA-secure scheme is described as follows.

IBKEM.Setup(λ): For security parameter λ, the setup algorithm proceeds as follows.

1. Generate $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathsf{GGen}^{\mathsf{DBDH}}(\lambda)$.
2. Select a random generator $g \in \mathbb{G}_2$.
3. Select a random exponent $x \in \mathbb{Z}_p$ and compute $h = g^x \in \mathbb{G}_2$.
4. Select hash functions $H : \{0,1\}^* \to \mathbb{G}_1$, $H_1 : \{0,1\}^* \to \{0,1\}^k \subseteq \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^\ell$.
5. Output $\mathsf{MSK} = x$ and $\mathsf{PP} = (\mathcal{G}, g, h, H, H_1, H_2)$.

IBKEM.KeyGen(PP, MSK, ID): For public parameters $\mathsf{PP} = (\mathcal{G}, g, h, H, H_1, H_2)$, a master secret key $\mathsf{MSK} = x$, and an identity ID, the KeyGen algorithm proceeds as follows.

1. Compute $\mathsf{SK}_{\mathsf{ID}} = H(\mathsf{ID})^x \in \mathbb{G}_1$.
2. Output $\mathsf{SK}_{\mathsf{ID}}$.

IBKEM.Encap(PP, ID): For public parameters $\mathsf{PP} = (\mathcal{G}, g, h, H, H_1, H_2)$ and an identity ID, the Encap algorithm proceeds as follows.

1. Select a random exponent $w \in \mathbb{Z}_p$.
2. Compute $Q_{\mathsf{ID}} = e(H(ID), h) \in \mathbb{G}_T$.
3. Set $\mathsf{stmt} = (g, h, \mathsf{ID}, g^w, Q_{\mathsf{ID}}^w)$ and $\mathsf{wit} = w$.
4. Compute $\pi \leftarrow \mathcal{P}^{H_1}(\mathsf{stmt}, \mathsf{wit})$.
   (a) Choose the random exponent $r \in \mathbb{Z}_p$.
   (b) Set $\mathsf{com} = (g^r, Q_{\mathsf{ID}}^r) \in \mathbb{G}_2 \times \mathbb{G}_T$.
   (c) Compute $c = H_1(\mathsf{stmt}, \mathsf{com})$ and $s = r + wc$.
   (d) Output the proof $\pi = (s, c)$.
5. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$.
6. Output K and $\mathsf{CT} = (g^w, \pi)$.

IBKEM.Decap(PP, CT, $\mathsf{SK}_{\mathsf{ID}}$): For public parameters $\mathsf{PP} = (\mathcal{G}, g, h, H, H_1, H_2)$, a ciphertext $\mathsf{CT} = (\mathsf{ct}, \pi = (s, c))$, and a private $\mathsf{SK}_{\mathsf{ID}}$, the Decap algorithm proceeds as follows.

1. Compute $\mathsf{k} = e(\mathsf{SK}_{\mathsf{ID}}, \mathsf{ct}) \in \mathbb{G}_T$.
2. Set $\mathsf{stmt} = (g, h, \mathsf{ID}, \mathsf{ct}, \mathsf{k})$.
3. If $1 \leftarrow \mathcal{V}^{H_1}(\mathsf{stmt}, \pi)$, go on. Otherwise, abort.
   (a) Compute $Q_{\mathsf{ID}} = e(H(\mathsf{ID}), h) \in \mathbb{G}_T$.
   (b) Set $\mathsf{com} = (g^s \cdot \mathsf{ct}^{-c}, Q_{\mathsf{ID}}^s \cdot \mathsf{k}^{-c}) \in \mathbb{G}_2 \times \mathbb{G}_T$.
   (c) If $c = H_1(\mathsf{stmt}, \mathsf{com})$, output 1. Otherwise, output 0.
4. Compute $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$ and output K.

Theorem 7. *Suppose that the $(t_{dbdh}, \epsilon_{dbdh})$ DBDH assumption holds in $\mathcal{G}$, and H_1 and H_2 are random oracles. Then, the Boneh–Franklin IBKEM is $(t, \epsilon, q_{id}, q_d)$-IND-ID-CCA-secure, where*

$$\epsilon \leq \frac{e(1+q_{id})\epsilon_{dbdh}}{2} + \frac{q_{H_1}^2}{2^{k-1}} + \frac{q_{H_2}^2}{2^{\ell-1}} + \frac{2q_H^2 + 2q_{H_1}^2 + 2q_d \cdot (q_{H_1} + q_{H_1}^2) + 1}{p}, t_{dbdh} \approx t + \mathcal{O}((q_{id} + q_H + q_d)t_e).$$

Here, $\{q_H, q_{H_1}, q_{H_2}\}$ are the numbers of $\{H, H_1, H_2\}$ queries, t_e is the required time for exponentiation in $\mathcal{G}$, and p is the group order.

Proof. We can obtain the proof by applying Theorem 4 to the facts that $\epsilon' \leq e(1+q_{id})\epsilon_{dbdh}$, $t_{dbdh} \approx t' + \mathcal{O}((q_{id} + q_H)t_e)$, $\epsilon_{\mathsf{ZK}} \leq q_{H_1}^2/p$, and $\epsilon_{\mathsf{SS}} \leq (q_{H_1} + q_{H_1}^2)/p$. Moreover, we use the fact that t_v becomes similar to t_e in Theorem 4. □

5.4. CCA-Secure Boneh–Boyen IBKEM

Let $\mathsf{GGen}^{\mathsf{DBDH}}(\lambda)$ be a bilinear group generator generating a group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, where the DBDH assumption holds. In the Boneh–Boyen IBKEM [24], public parameters consist of four group elements $(g_1, y_1, y_2, \Lambda = e(g_1, g_2)^\alpha) \in \mathbb{G}_1^3 \times \mathbb{G}_T$ such that $y_1 = g_1^x$, $y_2 = g_1^y$ with corresponding master secret key $(g_2, x, y, \alpha) \in \mathbb{G}_2 \times \mathbb{Z}_p^3$. For an identity ID, the secret key is generated by computing $\mathsf{sk}_{\mathsf{ID}} = (d_1, d_2) = (g_2^{\alpha + (x + H(\mathsf{ID})y)t}, g_2^t) \in \mathbb{G}_2^2$. The encapsulation is done by computing $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2) = \left(g_1^w, (y_1 y_2^{H(\mathsf{ID})})^w\right) \in \mathbb{G}_1^2$ and $\mathsf{k} = \Lambda^w \in \mathbb{G}_T$, and decapsulation is done by computing $\mathsf{k} = e(\mathsf{ct}_1, d_1)/e(\mathsf{ct}_2, d_2) \in \mathbb{G}_T$. Based on this scheme, the transformed IND-CCA-secure scheme is described as follows.

IBKEM.Setup(λ): For security parameter λ, the setup algorithm proceeds as follows.

1. Generate $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathsf{GGen}^{\mathsf{DBDH}}(\lambda)$.
2. Select two random generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.
3. Select random exponents $x, y, \alpha \in \mathbb{Z}_p$.
4. Compute $y_1 = g_1^x$, $y_2 = g_1^y$, and $\Lambda = e(g_1, g_2)^\alpha$.

5. Select hash functions $H : \{0,1\}^* \to \mathbb{Z}_p$, $H_1 : \{0,1\}^* \to \{0,1\}^k \subseteq \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^\ell$.
6. Output $\mathsf{MSK} = (g_2, x, y, \alpha)$ and $\mathsf{PP} = (\mathcal{G}, g_1, y_1, y_2, \Lambda, H, H_1, H_2)$.

IBKEM.KeyGen$(\mathsf{PP}, \mathsf{MSK}, \mathsf{ID})$: For public parameters $\mathsf{PP} = (\mathcal{G}, g_1, y_1, y_2, \Lambda, H, H_1, H_2)$, a master secret key $\mathsf{MSK} = (g_2, x, y, \alpha)$, and an identity ID, the KeyGen algorithm proceeds as follows.

1. Select a random exponent $t \in \mathbb{Z}_p$.
2. Compute $\mathsf{SK}_{\mathsf{ID}} = (g_2^{\alpha+(x+H(\mathsf{ID})y)t}, g_2^t) \in \mathbb{G}_2^2$.
3. Output $\mathsf{SK}_{\mathsf{ID}}$.

IBKEM.Encap$(\mathsf{PP}, \mathsf{ID})$: For public parameters $\mathsf{PP} = (\mathcal{G}, g_1, y_1, y_2, \Lambda, H, H_1, H_2)$ and an identity ID, the Encap algorithm proceeds as follows.

1. Select a random exponent $w \in \mathbb{Z}_p$.
2. Compute $Q_{\mathsf{ID}} = y_1 y_2^{H(\mathsf{ID})} \in \mathbb{G}_1$.
3. Set $\mathsf{stmt} = (g_1, y_1, y_2, \Lambda, \mathsf{ID}, g_1^w, Q_{\mathsf{ID}}^w, \Lambda^w)$ and $\mathsf{wit} = w$.
4. Compute $\pi \leftarrow \mathcal{P}^{H_1}(\mathsf{stmt}, \mathsf{wit})$.
   (a) Select the random exponent $r \in \mathbb{Z}_p$.
   (b) Set $\mathsf{com} = (g_1^r, Q_{\mathsf{ID}}^r, \Lambda^r) \in \mathbb{G}_1^2 \times \mathbb{G}_T$.
   (c) Compute $c = H_1(\mathsf{stmt}, \mathsf{com})$ and $s = r + wc$.
   (d) Output the proof $\pi = (s, c)$.
5. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$.
6. Output K and $\mathsf{CT} = (g_1^w, Q_{\mathsf{ID}}^w, \pi)$.

IBKEM.Decap$(\mathsf{PP}, \mathsf{CT}, \mathsf{SK}_{\mathsf{ID}})$: For public parameters $\mathsf{PP} = (\mathcal{G}, g_1, y_1, y_2, \Lambda, H, H_1, H_2)$, a ciphertext $\mathsf{CT} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi = (s, c))$, and a private $\mathsf{SK}_{\mathsf{ID}} = (d_1, d_2)$, the Decap algorithm proceeds as follows.

1. Compute $\mathsf{k} = e(\mathsf{ct}_1, d_1)/e(\mathsf{ct}_2, d_2) \in \mathbb{G}_T$.
2. Set $\mathsf{stmt} = (g_1, y_1, y_2, \Lambda, \mathsf{ID}, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{k})$.
3. If $1 \leftarrow \mathcal{V}^{H_1}(\mathsf{stmt}, \pi)$, go on. Otherwise, abort.
   (a) Compute $Q_{\mathsf{ID}} = y_1 y_2^{H(\mathsf{ID})} \in \mathbb{G}_1$.
   (b) Set $\mathsf{com} = (g_1^s \cdot \mathsf{ct}_1^{-c}, Q_{\mathsf{ID}}^s \cdot \mathsf{ct}_2^{-c}, \Lambda^s \cdot \mathsf{k}^{-c}) \in \mathbb{G}_1^2 \times \mathbb{G}_T$.
   (c) If $c = H_1(\mathsf{stmt}, \mathsf{com})$, output 1. Otherwise, output 0.
4. Set $\mathsf{K} \leftarrow H_2(\mathsf{stmt}, \mathsf{com}, \pi)$ and output K.

Theorem 8. *Suppose that the $(t_{dbdh}, \epsilon_{dbdh})$ DBDH assumption holds in $\mathcal{G}$, and H_1 and H_2 are random oracles. Then, the Boneh–Boyen IBKEM is $(t, \epsilon, q_{id}, q_d)$-IND-ID-CCA-secure, where*

$$\epsilon \leq \frac{q_H \cdot \epsilon_{dbdh}}{2} + \frac{q_{H_1}^2}{2^{k-1}} + \frac{q_{H_2}^2}{2^{\ell-1}} + \frac{2q_H^2 + 2q_{H_1}^2 + 2q_d \cdot (q_{H_1} + q_{H_1}^2) + 1}{p}, t_{dbdh} \approx t + \mathcal{O}((q_{id} + q_H + q_d)t_e).$$

Here, $\{q_H, q_{H_1}, q_{H_2}\}$ are the numbers of $\{H, H_1, H_2\}$ queries, t_e is the required time for exponentiation in $\mathcal{G}$, and p is the group order.

Proof. We can obtain the proof by applying Theorem 4 to the facts that $\epsilon' \leq q_H \cdot \epsilon_{dbdh}$, $t_{dbdh} \approx t' + \mathcal{O}(q_{id} t_e)$, $\epsilon_{\mathsf{ZK}} \leq q_{H_1}^2/p$, and $\epsilon_{\mathsf{SS}} \leq (q_{H_1} + q_{H_1}^2)/p$. Moreover, we use the fact that t_v becomes similar to t_e in Theorem 4. □

6. Conclusions

This paper shows the security proof of the recent work for the CCA conversion method in [12] has a flaw and proposes the corrected proof for the method. The CCA conversion method [12] from OW-CPA-KEM to IND-CCA-KEM is a tightly secure conversion and is based on the Random Oracle model. Without changing the basic conversion method, this

paper proposes the new corrected security proof of the conversion method. If the security proof for the newly designed cryptography scheme is not correct, there may exist some vulnerability in the system using the scheme. Therefore, this report on the flaw of security proof of the conversion method [12] and the fixed security proof should be considered by developers that use this conversion method. With the revised CCA conversion method, one can obtain an IND-CCA-KEM scheme from an IND-CPA-KEM scheme, and an IND-ID-CCA-IBKEM scheme from an IND-ID-CPA-IBKEM scheme, respectively. This paper provides the revised applications of the CCA conversion method which are the practical encryption schemes and the ID-based encryption schemes. As a result, information systems such as software tools, image processing, and sound transmission [25–27] that require information security can use those applications of the CCA conversion method to enhance data privacy.

However, this work has the limitations that the new security proof assumes the stronger condition that an underlying CPA-KEM is secure in terms of IND instead of OW. As future work, we note that proving the security of the conversion method with a weaker assumption that an underlying CPA-KEM is secure in terms of OW is an interesting open question.

Author Contributions: Conceptualization, J.H.P.; Formal analysis, Y.L. and J.H.P.; methodology, Y.L. and J.H.P.; validation, D.H.L. and J.H.P.; writing—original draft preparation, Y.L. and J.H.P.; writing—review and editing, Y.L., D.H.L., and J.H.P.; supervision, J.H.P.; project administration, D.H.L.; All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean Government (MSIT) (No. 2016-6-00600, A Study on Functional Encryption: Construction, Security Analysis, and Implementation.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Blum, M.; Feldman, P.; Micali, S. Non-interactive zero-knowledge and its applications. In Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC'88), Chicago, IL, USA, 2–4 May 1988; ACM: New York, NY, USA, 1998; pp. 103–112.
2. Blum, M.; Santis, A.D.; Micali, S.; Persiano, G. Noninteractive zero-knowledge. *SIAM J. Comput.* **1991**, *20*, 1084–1118. [CrossRef]
3. Feige, U.; Lapidot, D.; Shamir, A. Multiple non-interactive zero knowledge proofs based on a single random string. In Proceedings of the 31st Annual Symposium on Foundations of Computer Science, St. Louis, MO, USA, 22–24 October 1990; pp. 308–317.
4. Ma, S.; Deng, Y.; He, D.; Zhang, J.; Xie, X. An efficient NIZK scheme for privacy-preserving transactions over account-model blockchain. *IEEE Trans. Depend. Secur. Comput.* **2021**, *18*, 641–651. [CrossRef]
5. Lin, C.; Luo, M.; Huang, X.; Choo, K.-K.R.; He, D. An Efficient Privacy-Preserving Credit Score System Based on Noninteractive Zero-Knowledge Proof. *IEEE Syst. J.* **2021**, 1–10. [CrossRef]
6. Naor, M.; Yung, M. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, STOC'90, Baltimore, MD, USA, 14–16 May 1990; Ortiz, H., Ed.; ACM: New York, NY, USA, 1990; pp. 427–437.
7. Dolev, D.; Dwork, C.; Naor, M. Nonmalleable cryptography. *SIAM J. Comput.* **2000**, *30*, 391–437. [CrossRef]
8. Sahai, A. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In Proceedings of the 40th Annual Symposium on Foundations of Computer Science, FOCS '99, New York, NY, USA, 17–18 October 1999; pp. 543–553.
9. Feige, U.; Lapidot, D.; Shamir, A. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.* **1999**, *29*, 1–28.
10. Rothblum, R.D.; Sealfon, A.; Sotiraki, K. Toward Non-interactive Zero-Knowledge Proofs for NP from LWE. *J. Cryptol.* **2021**, *34*, 3. [CrossRef]
11. Boneh, D.; Canetti, R.; Halevi, S.; Katz, J. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.* **2007**, *36*, 1301–1328. [CrossRef]
12. Seo, M.; Abdalla, M.; Lee, D.H.; Park, J.H. New technique for chosen-ciphertext security based on non-interactive zero-knowledge. *Inf. Sci.* **2019**, *490*, 18–35. [CrossRef]

13. Fiat, A.; Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques, CRYPTO' 86*; Lecture Notes in Computer Science; Odlyzko, A.M., Ed.; Springer: Berlin/Heidelberg, Germany, 1986; Volume 263, pp. 186–194.
14. Faust, S.; Kohlweiss, M.; Marson, G.A.; Venturi, D. On the non-malleability of the fiat-shamir transform. In *Progress in Cryptology—INDOCRYPT'12*; Lecture Notes in Computer Science; Galbraith, S., Nandi, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7668, pp. 60–79.
15. Abdalla, M.; Fouque, P.; Lyubashevsky, V.; Tibouchi, M. Tightly secure signatures from lossy identification schemes. *J. Cryptol.* **2016**, *29*, 597–631. [CrossRef]
16. Lee, Y.; Lee, D.H.; Park, J.H. Tightly CCA-secure encryption scheme in a multi-user setting with corruptions. *Des. Codes Cryptogr.* **2020**, *88*, 2433–2452. [CrossRef]
17. Camenisch, J.; Stadler, M. *Proof Systems for General Statements about Discrete Logarithms*; Technical Report 260; Institute for Theoretical Computer Science (ETH Zurich): Zürich, Switzerland, 1997; pp. 1–13.
18. Chaum, D.; Evertse, J.H.; van de Graaf, J. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology—EUROCRYPT' 87*; Lecture Notes in Computer Science; Chaum, D., Price, W.L., Eds.; Springer: Berlin/Heidelberg, Germany, 1987; Volume 304, pp. 127–141.
19. Goh, E.J.; Jarecki, S. A signature scheme as secure as the diffie-hellman problem. In *Advances in Cryptology—EUROCRYPT'03*; Lecture Notes in Computer Science; Biham, E., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2656, pp. 401–415.
20. Wee, H. Zero knowledge in the random oracle model, revisited. In *Advances in Cryptology—ASIACRYPT'09*; Lecture Notes in Computer Science; Matsui, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5912, pp. 417–434.
21. Boneh, D.; Boyen, X.; Shacham, H. Short group signatures. In *Advances in Cryptology—CRYPTO'04*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3152, pp. 41–55.
22. Boneh, D.; Franklin, M.K. Identity-based encryption from the weil pairing. In *Advances in Cryptology—CRYPTO'01*; Lecture Notes in Computer Science; Kilian, J., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2139, pp. 213–229.
23. Libert, B. New Secure Applications of Bilinear Maps in Cryptography. Ph.D. Thesis, Universitï Catholique de Louvain, Ottignies-Louvain-la-Neuve, Belgium, 2006.
24. Boneh, D.; Boyen, X. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*; Lecture Notes in Computer Science; Cachin, C., Camenisch, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3027, pp. 223–238.
25. Buraga, S.C.; Dospinescu, O. A knowledge-based pilot study on assessing the music influence. *Comput. Mater. Contin.* **2021**, *66*, 2857–2873. [CrossRef]
26. Shankar, K.; Elhoseny, M. Trust based cluster head election of secure message transmission in MANET using multi secure protocol with TDES. *J. Univers. Comput. Sci.* **2019**, *25*, 1221–1239.
27. Dospinescu, O.; Brodner, P. Integrated Applications with Laser Technology. *Informatică Economică* **2013**, *17*, 53–61. [CrossRef]