Generic Construction of CCA-Secure PKE from Key-Insulated and Privacy-Preserving Signature Scheme with Publicly Derived Public Key

Ryo Mizuno[1] and Keita Emura*[1,2]

[1.]Kanazawa University, Japan.
[2.]AIST, Japan.

October 28, 2026

Abstract

To enhance the security of stealth addresses and protect user privacy, Liu et al. (EuroS&P 2019) proposed a Key-Insulated and Privacy-Preserving Signature Scheme with Publicly Derived Public Key (PDPKS). In this scheme, the payee generates a master public/secret key pair, and the payer derives a public key from the payee's master public key and binds the cryptocurrency asset to the resulting derived public key. The payee then verifies, using the master secret key, whether the derived public key has been correctly generated from the master public key. In this paper, we show that a public-key encryption (PKE) scheme secure against chosen-ciphertext attacks (CCA) can be generically constructed from PDPKS. Specifically, we utilize unlinkability, which hides the master public key from which a derived public key originates, to conceal plaintext information from ciphertexts. We also simulate the decryption oracle by leveraging the derived public key checking algorithm. In addition, to guarantee the correctness of the proposed PKE scheme, we rely on the consistency of PDPKS introduced by Emura (IACR CiC 2025). Given the motivation behind PDPKS, namely the construction of stealth addresses in a provably secure manner, unlinkability is regarded as the core security property of PDPKS. Our result shows that achieving this core security property essentially requires CCA-secure PKE, or cryptographic primitives of equivalent strength, and that such primitives are indispensable for constructing PDPKS.

1 Introduction

1.1 Research Background

Stealth addresses are designed to enhance transaction privacy on a blockchain by preventing third parties from linking transactions to the same recipient. In a typical stealth-address system, a sender derives a fresh public key for each transaction and associates a cryptocurrency asset with the derived public key, while the recipient can recognize that the derived public key corresponds to itself and recover the corresponding secret key. Using this secret key, the recipient can spend the associated cryptocurrency asset. To provide such functionality in a provably secure manner, Liu et al. [27] proposed a Key-Insulated and Privacy-Preserving Signature Scheme with Publicly Derived Public Key (PDPKS).[1] A flow of PDPKS is briefly described as follows. A payer derives a derived public

*Corresponding Author: k-emura@se.kanazawa-u.ac.jp

[1]The full version of [27] has been published in [28].

key dpk from a payee's master public key mpk by running the public key derivation algorithm $\mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk})$, where pp denotes the public parameter. Then, the payer binds a cryptocurrency asset to dpk. The payee verifies whether dpk is derived from its own master public key mpk using the master secret key msk by running the derived public key checking algorithm $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 0/1$. If the verification succeeds, i.e., the algorithm outputs 1, the payee computes the derived secret key dsk corresponding to dpk, and uses this secret key to generate a signature Σ on the transaction, thereby enabling the spending of the cryptocurrency asset associated with dpk. As a security property, unlinkability is defined to guarantee that no information of mpk is leaked from dpk. In addition, unforgeability is defined to guarantee that no third party, including the payer, can produce a valid signature Σ under dpk, and that only the payee can generate Σ.

As concrete PDPKS constructions, a pairing-based scheme [27] and a lattice-based scheme [26] have been proposed. In the pairing-based schemes [27], it was observed that, once the derived public key dpk is fixed, the process of using the master secret key msk to compute the corresponding derived secret key dsk is analogous to the key-generation mechanism in identity-based signatures (IBS). It was also observed that, unlike the usual signature verification procedure in IBS, the signature verification algorithm in PDPKS does not take mpk as input to provide unlinkability. Therefore, the pairing-based PDPKS scheme is constructed by focusing on the Barreto et al. IBS scheme [5], which has the master public key packing property (i.e., signature verification is possible without the master public key). In addition to IBS, their pairing-based PDPKS construction implicitly employs Non-Interactive Key Exchange (NIKE) [17]. The lattice-based PDPKS scheme [26] is constructed from the lattice basis delegation technique [3] and a key-private and CCA (Chosen Ciphertext Attack)-secure public-key encryption (PKE) scheme. In [15], as a weakened notion of unforgeability, outsider unforgeability was introduced where any third party other than the payer and the payee cannot forge a valid signature under a derived public key dpk. As a variant of outsider unforgeability, outsider strong unforgeability was introduced to consider strong existential unforgeability. Consistency was also introduced where it guarantees that $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 0$ if dpk is not derived from mpk. A generic construction of PDPKS, that provides outsider strong unforgeability and consistency (in addition to unlinkability), was proposed from key-private, CCA-secure, and strongly robust [1] PKE and strong existential unforgeable and S-CEO (strong conservative exclusive ownership) [13] secure signatures.

1.2 Our Motivation

Although concrete PDPKS schemes [26, 27] and the generic construction [15] have been proposed, no results are known regarding the feasibility of PDPKS itself. For example, all existing PDPKS constructions employ PKE (or cryptographic primitives equivalent to PKE (NIKE in [27])), yet it remains unknown whether the use of such primitives is inherently necessary for constructing PDPKS. As classical results, one-way functions are sufficient to construct signatures [34], and there is no black-box construction of PKE from one-way functions [21]. Since PDPKS is a variant of signatures, if PDPKS could be constructed from weaker cryptographic primitives than PKE, this could potentially lead to more efficient schemes and provide important insights for the design of PDPKS. Therefore, the aim of this paper is to answer the following question:

What cryptographic primitives are fundamentally required for constructing PDPKS?

1.3 Our Contribution

In this paper, we show that a PKE scheme secure against Chosen-Ciphertext Attacks (CCA) can be generically constructed from PDPKS.

- We utilize unlinkability, which hides the master public key from which a derived public key originates, to hide plaintext information from ciphertexts.
- We simulate the decryption oracle by using the derived public key checking algorithm $\mathsf{DpkCheck}$.

Given the motivation behind PDPKS, namely the construction of stealth addresses in a provably secure manner, unlinkability is regarded as the core security property of PDPKS. Our result shows that:

> *Achieving this core security property essentially requires CCA-secure PKE (or cryptographic primitives equivalent to it), which are indispensable for constructing PDPKS.*

Technical Overview. In the proposed construction, the public parameter pp and two master public keys mpk_0 and mpk_1 are set as the PKE public key pk, and the corresponding master secret keys msk_0 and msk_1 are set as the PKE decryption key dk. A ciphertext c for a one-bit plaintext $m \in \{0, 1\}$ is defined as

$$c := \mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_m)$$

For decryption, basically, if

$$\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_m, \mathsf{msk}_m, \mathsf{dpk}) = 1$$

then, the decryption algorithm outputs m. Due to unlinkability, no information leaks regarding whether dpk was derived from mpk_0 or mpk_1. Furthermore, the $\mathsf{DpkCheck}$ oracle defined in unlinkability is used to simulate the decryption oracle in the CCA security model. We remark that the $\mathsf{DpkDerive}$ algorithm is probabilistic, even though a stealth address might be intended for one-time use. The reason is that if the algorithm is deterministic, then anyone could break unlinkability by deriving keys from both mpk_0 and mpk_1 respectively and comparing the results to the challenge derived public key.

On Correctness. It should be noted that if

$$\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 1 \text{ and } \mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}) = 1$$

hold simultaneously, then the decryption result becomes uncertain. To explicitly capture this ambiguity, we introduce a special symbol $\mathsf{Uncertain}$ as a decryption output. One may consider replacing $\mathsf{Uncertain}$ with the ordinary error symbol $\perp$. However, doing so would conflate two fundamentally different situations. The output $\perp$ indicates that a ciphertext is not associated with any valid plaintext, whereas $\mathsf{Uncertain}$ indicates that a ciphertext is simultaneously associated with multiple plaintexts. The latter situation corresponds precisely to a violation of consistency of PDPKS. Our primary motivation is to clarify the feasibility of PDPKS and, in particular, the respective roles of unlinkability and consistency. Therefore, we explicitly distinguish $\mathsf{Uncertain}$ from $\perp$ here, although later we identify $\mathsf{Uncertain}$ with $\perp$ when applying the Myers-Shelat transformation [31] to obtain a multi-bit plaintext space. We introduce Correctness 1 and Correctness 2. Correctness 1 guarantees correct decryption of honestly generated ciphertexts, while Correctness 2 guarantees that no probabilistic polynomial-time (PPT) adversary can generate a ciphertext whose decryption result

is Uncertain. We show that if the underlying PDPKS scheme satisfies consistency, then Correctness 2 holds. Although consistency was introduced as a security notion in [15], it was also shown in [15] that earlier schemes [26–28] satisfy consistency as well. Thus, consistency should be regarded as a natural security property for PDPKS.

While our main result concerns unlinkability, consistency also plays an essential role in our construction. In particular, consistency guarantees that a ciphertext is associated with a unique plaintext. Without consistency, a trivial PDPKS scheme can be constructed by letting DpkCheck always output 1 and generating all values at random. Such a scheme trivially satisfies unlinkability but provides no meaningful functionality. The symbol Uncertain explicitly captures this phenomenon and enables us to formally relate consistency of PDPKS to correctness of the resulting PKE scheme.

On Multi-bit Plaintext Space. Myers and Shelat [31] showed that a CCA-secure PKE scheme with a one-bit plaintext space can be transformed in a black-box manner into a CCA-secure PKE scheme supporting a multi-bit plaintext space. Whether the transformation of Myers and Shelat can be directly applied to our setting requires additional consideration because our decryption algorithm may output the symbol Uncertain. Correctness 2 guarantees that no PPT adversary can generate a ciphertext whose decryption result is Uncertain. However, there remains a gap in directly applying the Myers-Shelat transformation, since a PPT adversary interacting with a decryption oracle may output a ciphertext whose decryption result is Uncertain. To apply the Myers-Shelat transformation, we define a modified decryption algorithm $\mathsf{PKE.Dec}'$ that replaces the output Uncertain with $\perp$. By Lemma 1, the resulting PKE scheme remains CCA secure. Therefore, the modified scheme is a standard one-bit CCA-secure PKE scheme to which the result of Myers and Shelat applies. Hence, a multi-bit CCA-secure PKE scheme can be constructed from PDPKS.

Differences from the Proceedings Version. An extended abstract appeared at the 28th International Conference on Information and Communications Security (ICICS 2026) [29]. In the proceedings version, in Correctness 2, we considered the case that a ciphertext is honestly generated. In this full version, we modified Correctness 2 such that an adversary outputs a ciphertext whose decryption result is Uncertain. This modification better captures the role of consistency, since consistency of PDPKS is defined in a setting where an adversary outputs a derived public key. We also revisited whether the transformation of Myers and Shelat can be applied. To bridge this gap in applying the Myers-Shelat transformation, we identify Uncertain with $\perp$. In our PKE construction, only a derived key dpk is utilized, and the signature component Σ of PDPKS is not used. In this full version, we additionally discuss what kinds of cryptographic primitives could be constructed depending on whether a payer (an encryptor in our PKE construction) can generate Σ or not. We believe that our discussion provides new insights into the feasibility of PDPKS. See Section 4.

1.4 Related Work

PDPKS can be regarded as a type of advanced signature. Similarly, group signatures [12] are known as another kind of advanced signature. Feasibility of group signatures has been extensively studied. For example, in [2], it is known that PKE can be constructed from a group signature that satisfies full anonymity, which guarantees anonymity even for holders of the signing key. Conversely, in the case of selfless anonymity, which guarantees anonymity only for those who do not possess the signing key, it has been shown that group signatures can be constructed from one-way functions and NIZKs [10], and moreover, that group signatures can even be constructed without NIZKs [24]. Furthermore, in [16], it is known that PKE with non-interactive opening

(PKENO) [14] can be constructed from group signatures that satisfy opening soundness [36]. It is also known that Group Signatures with Message-Dependent Opening (GS-MDO) [35], which is an extension of group signatures, imply identity-based encryption (IBE) [35] and timed-release encryption (TRE) [22]. PDPKS is a relatively new cryptographic primitive compared to group signatures. Investigating its feasibility is expected to provide new insights into PDPKS, such as alternative constructions or new security models, and this forms the motivation for our work in this paper. We believe that our results provide an important first step toward assessing the feasibility of PDPKS.

Yoshida et al. [38] proposed a generic construction of matchmaking PKE (PK-ME), which is a public-key variant of matchmaking encryption [4], from public key authenticated encryption with keyword search (PAEKS) [20,25]. They pointed out that, in the construction of IBE from public key encryption with keyword search (PEKS) by Boneh et al. [8], the resulting IBE scheme may fail to satisfy correctness if the underlying PEKS scheme does not provide consistency. Specifically, when decrypting a ciphertext for a keyword $m \in \{0,1\}$ (corresponding to plaintexts 0 and 1, respectively) using both the trapdoor for keyword 0 and the trapdoor for keyword 1, the decryption result becomes uncertain if the search algorithm returns a match under both trapdoors. To address this issue, Yoshida et al. introduced the symbol $\mathsf{Uncertain}$ as a possible decryption output. They showed that, if the underlying PAEKS scheme provides consistency, then the probability that decryption outputs $\mathsf{Uncertain}$ is negligible. Following this approach, we also introduce $\mathsf{Uncertain}$ as a possible output of the decryption algorithm in our PKE construction.

2 Preliminaries

In this section, we define PKE and PDPKS.

2.1 PKE

Let $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE scheme, defined as follows. Typically, the output of the decryption algorithm is either a plaintext m or an error symbol $\perp$; however, in this paper, we additionally introduce a special symbol, $\mathsf{Uncertain}$, to indicate that the decryption result is not uniquely determined.

Definition 1 (Syntax of PKE)**.**

$\mathsf{PKE.KeyGen}$**:** *The key generation algorithm takes as input a security parameter $k \in \mathbb{N}$ and outputs a public key pk and a decryption key dk.*

$\mathsf{PKE.Enc}$**:** *The encryption algorithm takes as input a public key pk and a plaintext $m \in \mathsf{MSpace_{PKE}}$, and outputs a ciphertext c where $\mathsf{MSpace_{PKE}}$ denotes the plaintext space.*

$\mathsf{PKE.Dec}$**:** *The decryption algorithm takes as input a public key pk, a decryption key dk, and a ciphertext c, and outputs either a plaintext m, the error symbol $\perp$, or $\mathsf{Uncertain}$.*

Next, we define correctness. Basically, correctness guarantees that if the key pair is honestly generated and the ciphertext is obtained by encrypting a plaintext m, then decrypting the ciphertext yields m, i.e., $\mathsf{PKE.Dec}(\mathsf{pk}, \mathsf{dk}, \mathsf{PKE.Enc}(\mathsf{pk}, m)) = m$ holds where $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{PKE.KeyGen}(1^k)$ and $m \in \mathsf{MSpace_{PKE}}$. Due to the introduction of $\mathsf{Uncertain}$, we distinguish between two notions of correctness. Correctness 1 guarantees that, when all values are generated according to the prescribed algorithms, the decryption result is the original plaintext m. Correctness 2 guarantees

that, for any PPT adversary, it is infeasible to produce a ciphertext whose decryption result is $\mathsf{Uncertain}$. We change "$\mathcal{A}$ chooses a message $m \in \mathsf{MSpace}_{\mathsf{PKE}}$ and sends it to $\mathcal{C}$. $\mathcal{C}$ computes $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m)$." in the proceedings version to "$\mathcal{A}$ outputs c." in this full version.

Definition 2 (Correctness)**.** *We define a PKE scheme to be correct if it satisfies both Correctness 1 and Correctness 2.*

Correctness 1: *For all $k \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{PKE.KeyGen}(1^k)$, $m \in \mathsf{MSpace}_{\mathsf{PKE}}$, and $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m)$, it holds that $\mathsf{PKE.Dec}(\mathsf{pk}, \mathsf{dk}, c) = m$ with overwhelming probability in the security parameter k.*

Correctness 2: *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be a challenger. $\mathcal{C}$ runs $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{PKE.KeyGen}(1^k)$ and sends pk to $\mathcal{A}$. $\mathcal{A}$ outputs c. If $\mathsf{PKE.Dec}(\mathsf{pk}, \mathsf{dk}, c) = \mathsf{Uncertain}$, then $\mathcal{A}$ wins. If, for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{correct2}}_{\mathcal{A},\mathsf{PKE}}(k) := \Pr[\mathcal{A} \text{ wins}]$ is negligible in the security parameter k, then we define PKE to provide Correctness 2.*

Next, we introduce one-bit CCA security defined in [31]. Since $\mathsf{MSpace}_{\mathsf{PKE}} = \{0, 1\}$, the challenge plaintexts are 0 and 1 regardless of the $\mathcal{A}$'s choice. Thus, the challenger $\mathcal{C}$ generates the challenge ciphertext c^* in the setup phase and sends c^* to $\mathcal{A}$ together with pk.

Definition 3 (CCA Security)**.** Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be a challenger. $\mathcal{C}$ runs $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{PKE.KeyGen}(1^k)$. $\mathcal{C}$ randomly selects $b \leftarrow \{0, 1\}$, computes the challenge ciphertext $c^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, b)$, and sends (pk, c^*) to $\mathcal{A}$. $\mathcal{A}$ is allowed to issue decryption queries as follows.

Decryption Query: $\mathcal{A}$ sends a ciphertext $c \neq c^*$ to $\mathcal{C}$. $\mathcal{C}$ returns the result of $\mathsf{PKE.Dec}(\mathsf{pk}, \mathsf{dk}, c)$ to $\mathcal{A}$.

$\mathcal{A}$ outputs a guessing bit b' and $\mathcal{A}$ wins if $b = b'$. The advantage is defined as $\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{A},\mathsf{PKE}}(k) := |\Pr[b = b'] - 1/2|$. If, for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{A},\mathsf{PKE}}(k)$ is negligible, then the PKE scheme is said to be CCA secure.

If $\mathcal{A}$ is not allowed to issue a decryption query, then it is said to be CPA (Chosen-Plaintext Attack) secure.

2.2 PDPKS

Let $\mathsf{PDPKS} = (\mathsf{PDPKS.Setup}, \mathsf{MasterKeyGen}, \mathsf{DpkDerive}, \mathsf{DpkCheck}, \mathsf{DskDerive}, \mathsf{PDPKS.Sign}, \mathsf{PDPKS.Verify})$ be a PDPKS scheme defined as follows.

Definition 4 (Syntax of PDPKS)**.**

PDPKS.Setup: *The setup algorithm takes as input a security parameter $k \in \mathbb{N}$ and outputs public parameter pp. We assume that pp includes the message space $\mathsf{MSpace}_{\mathsf{PDPKS}}$.*

MasterKeyGen: *The master key generation algorithm takes pp as input and outputs a master public key mpk and a master secret key msk.*

DpkDerive: *The public key derivation algorithm takes pp and mpk as input and outputs a derived public key dpk.*

DpkCheck: *The derived public key verification algorithm takes pp, mpk, msk, and dpk as input, and outputs 1 (if dpk is derived from mpk) or 0 (if dpk is not derived from mpk).*

DskDerive: *The derived secret key generation algorithm takes* pp, mpk, msk, *and* dpk *as input. If* $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 0$, *it outputs* $\perp$. *Otherwise, it outputs the derived secret key* dsk.

PDPKS.Sign: *The signing algorithm takes as input* pp, dpk, dsk, *and* $M \in \mathsf{MSpace}_{\mathsf{PDPKS}}$, *and outputs a signature* Σ.

PDPKS.Verify: *The verification algorithm takes as input* pp, dpk, M, *and* Σ, *and outputs* 1 *if* Σ *is valid, and* 0 *otherwise.*

Next, we define correctness. It guarantees that if all values are honestly generated, then the DpkCheck algorithm and PDPKS.Verify algorithm outputs 1.

Definition 5 (Correctness)**.** *For all security parameter* $k \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^k)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, $\mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk})$, *and* $M \in \mathsf{MSpace}_{\mathsf{PDPKS}}$, *if* $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 1$ *and* $\mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}, M, \Sigma) = 1$ *hold with overwhelming probability in the security parameter* k, *then the* PDPKS *scheme is said to satisfy correctness. Here,* $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk})$ *and* $\Sigma \leftarrow \mathsf{PDPKS.Sign}(\mathsf{pp}, \mathsf{dpk}, \mathsf{dsk}, M)$.

Next, we define consistency. Assuming that $(\mathsf{mpk}_0, \mathsf{msk}_0)$ and $(\mathsf{mpk}_1, \mathsf{msk}_1)$ are honestly generated, it guarantees that it is hard to find dpk^* such that $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}^*) = \mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}^*) = 1$. We note that the signing oracle is not employed in our PKE construction.

Definition 6 (Consistency)**.** Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be a challenger. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^k)$, $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, and $(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, and gives $(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1)$ to $\mathcal{A}$. $\mathcal{A}$ is allowed to issue the following queries.

Signing Query: $\mathcal{A}$ sends $M \in \mathsf{MSpace}_{\mathsf{PDPKS}}$ and an index $i \in \{0, 1\}$ to $\mathcal{C}$. $\mathcal{C}$ computes $\mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_i)$, $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{dpk})$, and $\Sigma \leftarrow \mathsf{PDPKS.Sign}(\mathsf{pp}, \mathsf{dpk}, \mathsf{dsk}, M)$, and then returns (dpk, Σ) to $\mathcal{A}$.

$\mathcal{A}$ outputs dpk^*. $\mathcal{A}$ wins if both $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}^*) = 1$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}^*) = 1$ hold. The advantage is defined as $\mathsf{Adv}^{\mathsf{consist}}_{\mathcal{A},\mathsf{PDPKS}}(k) = \Pr[\mathcal{A} \text{ wins}]$. If, for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{consist}}_{\mathcal{A},\mathsf{PDPKS}}(k)$ is negligible, then the PDPKS scheme is said to satisfy consistency.

Next, we define unlinkability. It guarantees that no information of the master public key mpk is leaked from a derived public key dpk. We note that the derived secret key oracle and the signing oracle are not employed in our PKE construction.

Definition 7 (Unlinkability)**.** Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be a challenger. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^k)$, $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, and $(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$. $\mathcal{C}$ selects $b \leftarrow \{0, 1\}$ and computes $\mathsf{dpk}^* \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_b)$, then gives $(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1, \mathsf{dpk}^*)$ to $\mathcal{A}$. $\mathcal{C}$ initializes $L_{\mathsf{dpk},0} := \emptyset$ and $L_{\mathsf{dpk},1} := \emptyset$. The adversary $\mathcal{A}$ is allowed to issue the following queries.

Derived Public Key Verification Query: $\mathcal{A}$ sends $\mathsf{dpk} \neq \mathsf{dpk}^*$ and an index $i \in \{0, 1\}$ to $\mathcal{C}$. $\mathcal{C}$ returns the result of $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{dpk})$. If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{dpk}) = 1$, then $\mathcal{C}$ updates $L_{\mathsf{dpk},i} \leftarrow L_{\mathsf{dpk},i} \cup \{\mathsf{dpk}\}$.

Derived Secret Key Query: $\mathcal{A}$ sends $\mathsf{dpk} \in L_{\mathsf{dpk},i}$ to $\mathcal{C}$. $\mathcal{C}$ returns $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{dpk})$.

Signing Query: $\mathcal{A}$ sends $M \in \mathsf{MSpace}_{\mathsf{PDPKS}}$ and $\mathsf{dpk} \in L_{\mathsf{dpk},i} \cup \{\mathsf{dpk}^*\}$ to $\mathcal{C}$. $\mathcal{C}$ returns $\Sigma \leftarrow \mathsf{PDPKS.Sign}(\mathsf{pp}, \mathsf{dpk}, \mathsf{dsk}, M)$, where $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{dpk})$.

Note that $\mathsf{dpk}^* \notin L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1}$. $\mathcal{A}$ outputs a guessing bit $b' \in \{0,1\}$ and wins if $b = b'$. The advantage is defined as $\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{A},\mathsf{PDPKS}}(k) := |\Pr[b = b'] - 1/2|$. If, for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{A},\mathsf{PDPKS}}(k)$ is negligible, then the PDPKS scheme is said to satisfy unlinkability.

3 Proposed Construction

3.1 Proposed PKE Construction

Here, we construct a PKE scheme $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ from a PDPKS scheme $\mathsf{PDPKS} = (\mathsf{PDPKS.Setup}, \mathsf{MasterKeyGen}, \mathsf{DpkDerive}, \mathsf{DpkCheck}, \mathsf{DskDerive}, \mathsf{PDPKS.Sign}, \mathsf{PDPKS.Verify})$ as follows. Note that in the proposed PKE construction, the $\mathsf{DskDerive}$, $\mathsf{PDPKS.Sign}$ and $\mathsf{PDPKS.Verify}$ algorithms are not used. We also set $\mathsf{MSpace}_{\mathsf{PKE}} = \{0,1\}$.

$\mathsf{PKE.KeyGen}(1^k)$**:** Run $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^k)$, and compute $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$ and $(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$. Output $\mathsf{pk} = (\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1)$ and $\mathsf{dk} = (\mathsf{msk}_0, \mathsf{msk}_1)$.

$\mathsf{PKE.Enc}(\mathsf{pk}, m)$**:** Let $\mathsf{pk} = (\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1)$. For $m \in \{0,1\}$, compute $\mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_m)$ and output $c = \mathsf{dpk}$.

$\mathsf{PKE.Dec}(\mathsf{pk}, \mathsf{dk}, c)$**:** Let $\mathsf{pk} = (\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1)$, $\mathsf{dk} = (\mathsf{msk}_0, \mathsf{msk}_1)$, and $c = \mathsf{dpk}$.

- If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 0$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}) = 0$, then output $\perp$.
- If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 1$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}) = 0$, then output $m = 0$.
- If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 0$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}) = 1$, then output $m = 1$.
- If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 1$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}) = 1$, then output $\mathsf{Uncertain}$.

3.2 Security Proof

Here, we provide security proofs of the proposed PKE construction.

Correctness. We prove Correctness 1 and Correctness 2 of the proposed PKE construction by relying on the correctness and consistency of the underlying PDPKS scheme. If the PDPKS scheme is correct, then it is immediate that the proposed PKE construction provides Correctness 1. We further show that, if the PDPKS scheme satisfies consistency, then the proposed PKE construction provides Correctness 2.

Theorem 1. *If the underlying PDPKS scheme provides consistency, then the proposed construction provides Correctness 2.*

Proof. Let $\mathcal{C}$ be the challenger for the consistency of PDPKS, and $\mathcal{A}$ be an adversary of Correctness 2 of PKE. We construct a reduction algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the consistency of PDPKS as follows. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^k)$, $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, and $(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow$

$\mathsf{MasterKeyGen}(\mathsf{pp})$, and sends $(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1)$ to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{pk} := (\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1)$ and forwards pk to $\mathcal{A}$. $\mathcal{A}$ outputs c. Let $c := \mathsf{dpk}^*$. Since the decryption result of c is $\mathsf{Uncertain}$ due to the winning condition, it follows that $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}^*) = \mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}^*) = 1$ holds. $\mathcal{B}$ outputs dpk^*. For $m \in \{0, 1\}$, since the checking results are 1 under both $(\mathsf{mpk}_0, \mathsf{msk}_0)$ and $(\mathsf{mpk}_1, \mathsf{msk}_1)$, $\mathcal{B}$ breaks the consistency of PDPKS. □

CCA Security. Next, we prove that the proposed PKE construction is CCA secure if the underlying PDPKS scheme is unlinkable. Since no information of $b \in \{0, 1\}$ is revealed from $\mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_b)$, the proposed PKE construction is immediately CPA secure. To prove the CCA security, we simulate the decryption oracle using the derived public key verification oracle. Note that the derived secret key oracle and the signing oracle defined in the security model of unlinkability are not used in the proof.

Theorem 2. *If the underlying PDPKS scheme provides unlinkability, then the proposed construction is CCA secure.*

Proof. Let $\mathcal{C}$ be the challenger for unlinkability, and $\mathcal{A}$ be an adversary of CCA security. We construct a reduction algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break unlinkability as follows. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^k)$, $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, and $(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, selects $b \in \{0, 1\}$, computes $\mathsf{dpk}^* \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_b)$, and sends $(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1, \mathsf{dpk}^*)$ to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{pk} := (\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1)$ and $c^* := \mathsf{dpk}^*$, and sends (pk, c^*) to $\mathcal{A}$.

Decryption Query $\mathcal{A}$ sends $c \neq c^*$ to $\mathcal{B}$. $\mathcal{B}$ forwards $(c, 0)$ and $(c, 1)$ to $\mathcal{C}$ as derived public key verification queries, respectively, where 0 and 1 denote the indices specified in the derived public key verification query.

- If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, c) = 0$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, c) = 0$, then $\mathcal{B}$ returns $\perp$.
- If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, c) = 1$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, c) = 0$, then $\mathcal{B}$ returns $m = 0$.
- If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, c) = 0$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, c) = 1$, then $\mathcal{B}$ returns $m = 1$.
- If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, c) = 1$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, c) = 1$, then $\mathcal{B}$ returns $\mathsf{Uncertain}$.

Finally, $\mathcal{A}$ outputs a guessing bit $b' \in \{0, 1\}$. $\mathcal{B}$ outputs b', and breaks unlinkability with the same advantage of $\mathcal{A}$ that breaks CCA security. □

3.3 Applying the Myers-Shelat Transformation

We construct a PKE scheme $\mathsf{PKE}' = (\mathsf{PKE.KeyGen}', \mathsf{PKE.Enc}', \mathsf{PKE.Dec}')$ from $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ as follows.

$\mathsf{PKE.KeyGen}(1^k)$**:** Run $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{PKE.KeyGen}(1^k)$ and output $(\mathsf{pk}, \mathsf{dk})$.

$\mathsf{PKE.Enc}'(\mathsf{pk}, m)$**:** Run $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m)$ and output c.

$\mathsf{PKE.Dec}'(\mathsf{pk}, \mathsf{dk}, c)$**:** Run $m \leftarrow \mathsf{PKE.Dec}(\mathsf{pk}, \mathsf{dk}, c)$. If $m \in \{\perp, \mathsf{Uncertain}\}$, then output $\perp$. Otherwise, output m.

Lemma 1. *If* PKE *is CCA secure,* PKE' *is also CCA secure.*

Proof. Let $\mathcal{C}$ be the challenger of CCA security for PKE, and $\mathcal{A}$ be the CCA adversary for PKE'. We construct an algorithm $\mathcal{B}$ that breaks CCA security of PKE as follows. $\mathcal{C}$ runs $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{PKE.KeyGen}(1^k)$, selects $b \in \{0,1\}$, computes $c^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, b)$, and sends (pk, c^*) to $\mathcal{B}$. $\mathcal{B}$ sends (pk, c^*) to $\mathcal{A}$. Upon a decryption query c sent from $\mathcal{A}$, $\mathcal{B}$ sends c to $\mathcal{C}$. If the response is $\mathsf{Uncertain}$, then $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ forwards the decryption result to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs b' and breaks the CCA security with the same advantage of $\mathcal{A}$. □

4 Discussion

4.1 Relation between Our Result and Previous Constructions

The lattice-based PDPKS scheme [26] and the generic construction [15] explicitly employ CCA-secure PKE as its building block. As mentioned above, the pairing-based PDPKS scheme [27] implicitly employs NIKE, and since PKE can be naturally derived from NIKE, their results do not contradict our result: PDPKS implies PKE. Next, we need to discuss whether the pairing-based PDPKS scheme implicitly requires a cryptographic primitive equivalent to CCA-secure PKE. The pairing-based PDPKS scheme is proved to be secure under q-strong Diffie-Hellman assumption and the Computational Diffie-Hellman (CDH) assumption. Since the CDH assumption is sufficient to construct CCA-secure PKE [11, 18], from the viewpoint of complexity assumptions, we conclude that their result does not contradict our result: PDPKS implies CCA-secure PKE. Related to this discussion, it is not clear whether cryptographic tools equivalent to CCA-secure PKE are required to construct IBS with the master public key packing property, although IBS can be generically constructed from one-way functions [6, 34]. A more detailed investigation is left for future work.

4.2 Employing the Signature Part

In our PKE construction, only dpk is utilized, and the signature component Σ of PDPKS is not used. From a security perspective, although we rely on unlinkability, we do not employ the derived secret key oracle or the signing oracle. That is, we show that even this weak form of unlinkability is sufficient to construct a CCA-secure PKE scheme. Moreover, our PKE construction does not rely on the other security property of PDPKS, namely (outsider strong) unforgeability. By leveraging the signature part Σ of PDPKS, the (outsider strong) unforgeability property, and the derived secret key and signing oracles in the unlinkability game, it may become possible to construct more advanced cryptographic primitives from PDPKS beyond CCA-secure PKE. In this section, we separately discuss what kinds of cryptographic primitives can be constructed depending on whether the payer, who corresponds to the encryptor in our PKE construction, can or cannot generate Σ.

Case of Unforgeability. Unforgeability [26, 27] requires that nobody, except the corresponding master secret key holder, a payee, can produce a valid signature under the derived public key. That is, a payer is not allowed to produce a signature Σ. We clarify that the payee can produce a signature after the payee observes dpk. Given this order, in our PKE construction, the decryptor who obtains a ciphertext dpk produces a signature on an arbitrary message M that can be verified under dpk as the verification key. More clearly, the decryptor checks $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 1$ or $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}) = 1$, and let $m \in \{0,1\}$ be the decryption result. Then, the decryptor computes $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}_m, \mathsf{msk}_m, \mathsf{dpk})$ and can produce a signature $\Sigma \leftarrow \mathsf{PDPKS.Sign}(\mathsf{pp}, \mathsf{dpk}, \mathsf{dsk}, M)$. Then, (M, Σ) is verified under dpk, i.e., $\mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}, M, \Sigma) = 1$. To provide unlinkability, the $\mathsf{PDPKS.Verify}$ algorithm does not take mpk_m as input.

We observe that Σ implicitly proves that the decryptor has correctly decrypted dpk and recognized the decryption result, without revealing the decryption result. This could be employed to construct a one-bit key sharing: a sender computes dpk from either mpk_0 or mpk_1 and sends dpk to a receiver. The receiver computes Σ (if either $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 1$ or $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}) = 1$) and sends (M, Σ) to the sender as a proof. Then, the sender and the receiver share $m \in \{0,1\}$ and the receiver can produce a proof. Here, M can be any string and selected according to the application. We note that Σ itself does not prove that the corresponding dsk is derived by either msk_0 or msk_1. In the above interactive protocol, the receiver rejects dpk if $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 0$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}) = 0$. Thus, $m \in \{0,1\}$ is guaranteed. To the best of our knowledge, we are unaware of any direct connection between the above interactive protocol and existing protocols. However, our observation indicates that PDPKS possesses the functionality of providing a form of proof of decryption without revealing the decryption result. Exploring this direction may be of interest for further demonstrating the feasibility of PDPKS, and we leave this as future work.

Case of Outsider Unforgeability. Outsider unforgeability [15] requires that nobody, except the payer and the payee, can produce a valid signature. That is, even if the payer can produce a signature Σ, it does not violate outsider unforgeability, and actually, the generic construction proposed in [15] allows the payer to produce a signature Σ since the payer first generates dsk (a signing key of the underlying signature scheme) and encrypts dsk using a public key of the payee.

At first glance, our PKE construction can be upgraded to provide public verifiability where the $\mathsf{PKE.Enc}$ algorithm computes $\mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_m)$ and also computes Σ on dpk (i.e., a signing message M is dpk), and sets $c = (\mathsf{dpk}, \Sigma)$. The $\mathsf{PKE.Dec}$ algorithm first checks whether $\mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}, \mathsf{dpk}, \Sigma) = 1$ and runs the decryption procedure using the $\mathsf{DpkCheck}$ algorithm. However, this observation is wrong because $\mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}, \mathsf{dpk}, \Sigma) = 1$ does not guarantee that dpk is derived from either mpk_0 or mpk_1, and thus the decryption procedure may output $\perp$. For example, an encryptor generates mpk and derives dpk from mpk. Then, $\mathsf{PDPKS.Verify}$ $(\mathsf{pp}, \mathsf{dpk}, \mathsf{dpk}, \Sigma) = 1$ holds but $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 0$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1,$ $\mathsf{dpk}) = 0$ hold. Thus, providing a signature Σ by an encryptor does not provide public verifiability against our PKE construction.

As another perspective, we point out that, in the generic construction proposed in [15], the payer and the payee share the same signing key of the underlying signature scheme. That is, both the payer and the payee are able to generate valid signatures, and it is indistinguishable whether a given signature was produced by the payer or by the payee. At the same time, the non-signing party can ascertain that the signature was generated by the other party. From this perspective, PDPKS with outsider unforgeability and payer–payee signer indistinguishability is closely related to designated verifier signatures (DVS) [23]. Since designated verifier signatures (DVS) and ring signatures are equivalent according to prior results [7, 9, 19, 33], such a PDPKS may also be related to ring signatures with two members. We note that multi-designated verifier signatures (MDVS) cannot be constructed from ring signatures in a black-box manner [37], and thus our observation is restricted to the two-member setting. We emphasize that payer-payee signer indistinguishability has not been required as a security property of PDPKS and has not been formalized so far. It may be interesting to investigate whether payer-payee signer indistinguishability is a desirable property and to further explore the relationship between PDPKS and DVS or ring signatures. We leave this as future work.

5 Conclusion

In this paper, we show that a CCA-secure PKE can be generically constructed from PDPKS, thereby clarifying that achieving unlinkability, the core security property of PDPKS, essentially requires CCA-secure PKE or an equivalent cryptographic primitive.

As extensions of PDPKS, Pu et al. [32] proposed Stealth Signatures, and Mongardini et al. [30] proposed Identity-Based Matchmaking Stealth Signatures (IB-MSS). Our result may be extended to explore the feasibility of them. A detailed investigation of these directions is left for future work.

Acknowledgment: This work was supported by the Telecommunications Advancement Foundation and JSPS KAKENHI Grant Number JP25H01106. This paper was edited for grammar using Copilot.

References

[1] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. *Journal of Cryptology*, 31(2):307–350, 2018.

[2] Michel Abdalla and Bogdan Warinschi. On the minimal assumptions of group signature schemes. In *ICICS*, pages 1–13, 2004.

[3] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[4] Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In *CRYPTO, Part II*, pages 701–731, 2019.

[5] Paulo S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *ASIACRYPT*, pages 515–532, 2005.

[6] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In *EUROCRYPT*, pages 268–286, 2004.

[7] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography*, pages 60–79, 2006.

[8] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[9] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. Post-quantum asynchronous deniable key exchange and the signal handshake. In *Public-Key Cryptography, Part II*, pages 3–34. Springer, 2022.

[10] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks*, pages 120–133, 2004.

[11] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In *EUROCRYPT*, pages 127–145, 2008.

[12] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[13] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In *IEEE Symposium on Security and Privacy*, pages 1696–1714. IEEE, 2021.

[14] Ivan Damgård, Dennis Hofheinz, Eike Kiltz, and Rune Thorbek. Public-key encryption with non-interactive opening. In *CT-RSA*, pages 239–255, 2008.

[15] Keita Emura. Key-insulated and privacy-preserving signature scheme with publicly derived public key, revisited: Consistency, outsider strong unforgeability, and generic construction. *IACR Communications in Cryptology*, 2(3):17, 2025.

[16] Keita Emura, Goichiro Hanaoka, Yusuke Sakai, and Jacob C. N. Schuldt. Group signature implies public-key encryption with non-interactive opening. *International Journal of Information Security*, 13(1):51–62, 2014.

[17] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In *Public-Key Cryptography*, pages 254–271, 2013.

[18] Goichiro Hanaoka and Kaoru Kurosawa. Efficient chosen ciphertext secure public key encryption under the Computational Diffie-Hellman assumption. In *ASIACRYPT*, pages 308–325, 2008.

[19] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for signal's handshake (X3DH): post-quantum, state leakage secure, and deniable. In *Public-Key Cryptography, Part II*, pages 410–440, 2021.

[20] Qiong Huang and Hongbo Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, 403:1–14, 2017.

[21] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *ACM STOC*, pages 44–61, 1989.

[22] Yuto Imura and Keita Emura. Group signatures with message-dependent opening directly imply timed-release encryption. In *CANS*, pages 251–271, 2025.

[23] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, pages 143–154, 1996.

[24] Shuichi Katsumata and Shota Yamada. Group signatures without NIZK: from lattices in the standard model. In *EUROCRYPT*, pages 312–344, 2019.

[25] Qinyi Li and Xavier Boyen. Public-key authenticated encryption with keyword search made easy. *IACR Communications in Cryptology*, 1(2):16, 2024.

[26] Wenling Liu, Zhen Liu, Khoa Nguyen, Guomin Yang, and Yu Yu. A lattice-based key-insulated and privacy-preserving signature scheme with publicly derived public key. In *ESORICS, Part II*, pages 357–377, 2020.

[27] Zhen Liu, Guomin Yang, Duncan S. Wong, Khoa Nguyen, and Huaxiong Wang. Key-insulated and privacy-preserving signature scheme with publicly derived public key. In *IEEE EuroS&P*, pages 215–230, 2019.

[28] Zhen Liu, Guomin Yang, Duncan S. Wong, Khoa Nguyen, Huaxiong Wang, Xiaorong Ke, and Yining Liu. Secure deterministic wallet and stealth address: Key-insulated and privacy-preserving signature scheme with publicly derived public key. *IEEE Transactions on Dependable and Secure Computing*, 19(5):2934–2951, 2022.

[29] Ryo Mizuno and Keita Emura. Generic construction of cca-secure pke from key-insulated and privacy-preserving signatures with publicly derived public key. In *ICICS, Part I*, pages 388–399, 2026.

[30] Alberto Maria Mongardini, Daniele Friolo, and Giuseppe Ateniese. Stealth and beyond: Attribute-driven accountability in bitcoin transactions. In *Applied Cryptography and Network Security, Part II*, pages 371–397, 2026.

[31] Steven A. Myers and Abhi Shelat. Bit encryption is complete. In *IEEE FOCS*, pages 607–616, 2009.

[32] Sihang Pu, Sri Aravinda Krishnan Thyagarajan, Nico Döttling, and Lucjan Hanzlik. Post quantum fuzzy stealth signatures and applications. In *ACM CCS*, pages 371–385, 2023.

[33] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT*, pages 552–565, 2001.

[34] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *ACM STOC*, pages 387–394, 1990.

[35] Yusuke Sakai, Keita Emura, Goichiro Hanaoka, Yutaka Kawai, Takahiro Matsuda, and Kazumasa Omote. Group signatures with message-dependent opening. In *Pairing-Based Cryptography*, pages 270–294, 2012.

[36] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In *Public Key Cryptography*, pages 715–732, 2012.

[37] Kyosuke Yamashita and Keisuke Hara. On the black-box impossibility of multi-designated verifiers signature schemes from ring signature schemes. *Journal of Mathematical Cryptology*, 18(1), 2024.

[38] Takeshi Yoshida and Keita Emura. On the relations between matchmaking public key encryption and public key authenticated encryption with keyword search. In *Cryptology and Network Security*, pages 272–291, 2025.