Key-Insulated and Privacy-Preserving Signature Scheme with Publicly Derived Public Key, Revisited: Consistency, Outsider Strong Unforgeability, and Generic Construction

Keita Emura*[1,2]

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

December 13, 2024

Abstract

Liu et al. (EuroS&P 2019) introduced Key-Insulated and Privacy-Preserving Signature Scheme with Publicly Derived Public Key (PDPKS) to enhance the security of stealth address and deterministic wallet. In this paper, we point out that the current security notions are insufficient in practice, and introduce a new security notion which we call consistency. Moreover, we explore the unforgeability to provide strong unforgeability for outsider which captures the situation that nobody, except the payer and the payee, can produce a valid signature. From the viewpoint of cryptocurrency functionality, it allows us to implement a refund functionality. Finally, we propose a generic construction of PDPKS that provides consistency and outsider strong unforgeability. The design is conceptually much simpler than known PDPKS constructions. It is particularly note that the underlying strongly unforgeable signature scheme is required to provide the strong conservative exclusive ownership (S-CEO) security (Cremers et al., IEEE S&P 2021). Since we explicitly require the underlying signature scheme to be S-CEO secure, our security proof introduces a new insight of exclusive ownership security which may be of independent interest. As instantiations, we can obtain a pairing-based PDPKS scheme in the standard model, a discrete-logarithm based pairing-free PDPKS scheme in the random oracle model, and a lattice-based PDPKS scheme in the random oracle model, and so on.

1 Introduction

1.1 Background

Liu et al. [33][1] introduced Key-Insulated and Privacy-Preserving Signature Scheme with Publicly Derived Public Key (PDPKS) to capture and improve the functionality, security, and privacy requirements of stealth address [18, 42] and deterministic wallet [4, 23]. Briefly, the flow of PDPKS is described as follows (See Fig. 1). Assume that a payer Alice wants to transfer funds to a payee Bob. Bob publishes the master public key mpk_B where $(\mathsf{mpk}_B, \mathsf{msk}_B) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$ and $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^\lambda)$ is a common public parameter (here λ is a security parameter). Alice derives a fresh public key $\mathsf{dpk}_A \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_B)$, assigns a coin to dpk_A, and sends

*k-emura@se.kanazawa-u.ac.jp

[1]The full version of [33] is available in [34].

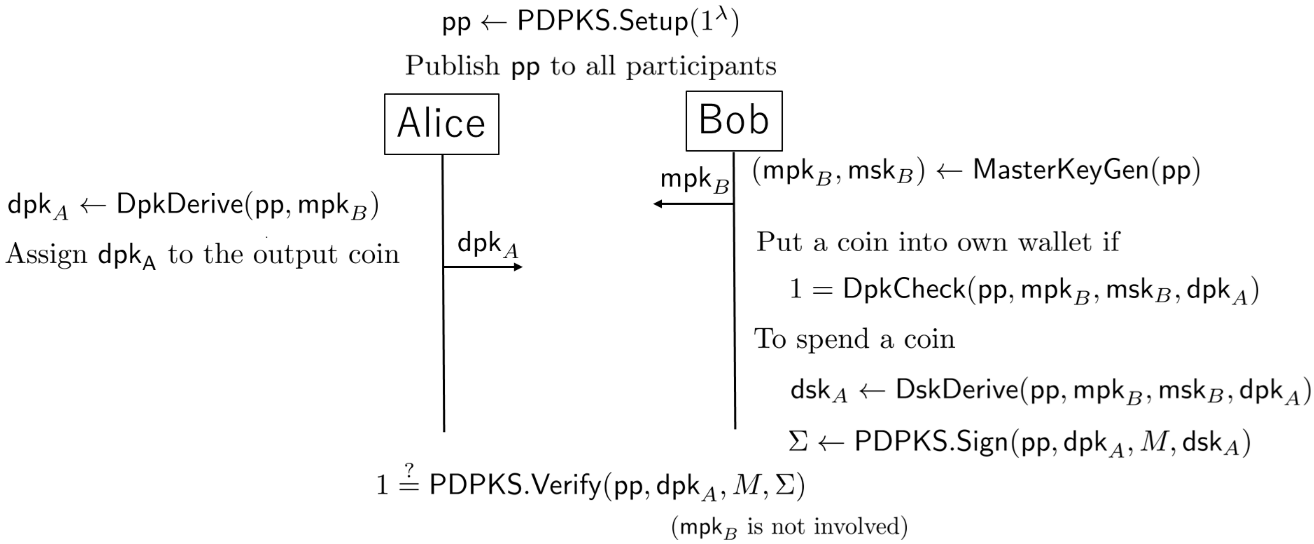

Figure 1: PDPKS Flow

dpk_A to Bob. We remark that this procedure is non-interactive, i.e., Alice can derive dpk_A from mpk_B without communicating Bob. To spend a coin, Bob checks whether dpk_A was derived from mpk_B or not by running $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_B, \mathsf{msk}_B, \mathsf{dpk}_A)$. If the algorithm returns 1 (meaning that dpk_A is linked to mpk_B), then Bob generates the corresponding derived secret key $\mathsf{dsk}_A \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}_B, \mathsf{msk}_B, \mathsf{dpk}_A)$. We remark that this procedure is also non-interactive, i.e., Bob can derive dsk_A from dpk_A without communicating Alice (after Bob obtains dpk_A). Bob generates a signature Σ on a message (transaction) M by running $\Sigma \leftarrow \mathsf{PDPKS.Sign}(\mathsf{pp}, \mathsf{dpk}_A, M, \mathsf{dsk}_A)$. Here, anyone can check the validity of (M, Σ) under dpk_A by running $\mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}_A, M, \Sigma)$ without using the corresponding master public key mpk_B.

Liu et al. [33] formalized security notions, correctness, unforgeability, and unlinkability. Correctness requires that honestly generated dpk and Σ are always accepted by $\mathsf{DpkCheck}$ and $\mathsf{PDPKS.Verify}$ algorithms, respectively. Unforgeability guarantees that no adversary $\mathcal{A}$ can produce $(\mathsf{dpk}^*, M^*, \Sigma^*)$ where $\mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}^*, M^*, \Sigma^*) = 1$. Since dpk^* is preduced by $\mathcal{A}$, it guarantees that even if anyone can derive a public key from a master public key mpk, nobody, except the corresponding master secret key holder, can produce a valid signature under the derived public key. Unlinkability guarantees that: (1) a derived public key does not leak information of the corresponding master public key, and (2) derived public keys do not leak information of whether two public keys are derived from the same master public key or not.

Liu et al. [33] pointed out that identity-based signature (IBS) is a promising tool to construct PDPKS but required a special property, that they called MPK-pack-able property. Briefly, it requires that there exists a function F and a verification algorithm Verify_F, where, for an identity ID and a message-signature pair (M, σ), $\mathsf{Verify}_F(F(\mathsf{IBS.mpk}, ID), M, \sigma) = \mathsf{IBS.Verify}(\mathsf{IBS.mpk}, ID, M, \sigma)$ holds and no information of $\mathsf{IBS.mpk}$ is leaked from $F(\mathsf{IBS.mpk}, ID)$.[2] They found that the Barreto et al. IBS scheme [7] has the property, and proposed a pairing-based PDPKS scheme which is secure under a q-type assumption in the random oracle model. We introduce their construction methodology for more detail in Appendix. As a subsequent work, Liu et al. [32] proposed a lattice-based PDPKS scheme by combining a lattice basis delegation technique [3] and public key encryption (PKE) with key privacy [10]. As a concrete lattice-based instantiation of the underlying

[2] We omit a common parameter from the input of the Verify_F algorithm.

key private PKE scheme, they proposed a lattice-based key private PKE scheme based on the Regev PKE scheme [39] with the Fujisaki-Okamoto transformation [26] to prevent chosen-ciphertext attacks (CCA).

1.2 Our Motivation

Generic Construction. Though the first construction [33] is based on IBS, it is not a purely generic construction (Liu et al. mentioned that "*We propose a (partially) generic approach on how to obtain a PDPKS construction*"). Moreover, no generic transformation for adding the MPK-packable property to an IBS scheme has not been proposed so far. The second construction [32] relies on the specific lattice basis delegation technique (though any key private PKE can be employed). Giving a generic construction highlights what a sufficient condition is to construct a cryptographic primitive, and provides several instantiations. Thus, proposing a generic construction of PDPKS is still open and is desirable.

Security Definitions. We also revisited the security definitions.

- First, we point out that the security notions defined in [32, 33] are not sufficient in practice because no security of the $\mathsf{DpkCheck}$ algorithm is defined. Due to the usage of PDPKS given in Fig. 1, a payer Alice uses dpk_A to specify the receiver of the transaction, and a payee Bob puts a coin into own wallet if $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_B, \mathsf{msk}_B, \mathsf{dpk}_A) = 1$. That is, Bob verifies whether dpk_A is linked to mpk_B or not before spending a coin. We remark that correctness just guarantees that $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_B, \mathsf{msk}_B, \mathsf{dpk}_A) = 1$ holds for a derived public key $\mathsf{dpk}_A \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_B)$, and does not guarantee that $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_B, \mathsf{msk}_B, \mathsf{dpk}_A) = 0$ if dpk_A is not derived from mpk_B. We need to strengthen the security of PDPKS in this perspective.

- Second, we explore unforgeability. In the current definition, $\mathcal{A}$ declares the challenge derived public key dpk^* which captures the situation that nobody, except the master secret key holder (the payee), can produce a valid signature, namely, the payer also is not allowed to produce a valid signature.[3] In the actual usage, however, it seems acceptable that the payer also can produce a valid signature unless a third person who just observes dpk cannot produce a valid signature. From this perspective, we can define a weaker variant of unforgeability, which we call outsider unforgeability, where dpk^* is sent to $\mathcal{A}$ from the challenger which captures the situation that nobody, except the payer and the payee, can produce a valid signature. This relaxation allows us to avoid the above IBS-like construction. From the viewpoint of cryptocurrency functionality, it allows us to implement a refund functionality, and it can be used for instantiating a refundable stealth address. Currently, basically there is no way to refund a coin when one mistakenly spends a coin to an address.

- Third, we further explore unforgeability. The original definition captures a conventional existential unforgeability: it is required that an adversary $\mathcal{A}$ never send a signing query (dpk^*, M^*) when $\mathcal{A}$ outputs a forgery $(\mathsf{dpk}^*, M^*, \Sigma^*)$. This means that a signature might be re-randomizable by definition. That is, without contradicting unforgeability, anyone may be able to produce a valid message-signature pair (M^*, Σ) by re-randomizing Σ^*. In the

[3] Liu et al. mentioned that "*For a target derived verification key, even if it was generated by the attacker from a target master public key, the attacker is not able to forge a valid signature to steal the coins on the target derived verification key*".

cryptocurrency context, a third person (who is neither the payer nor the payee) may be able to produce a valid (M^*, Σ) on an existing transaction M^*. Thus, to capture the concept of PDPKS, the strong unforgeability is mandatory: $\mathcal{A}$ is allowed to issue (dpk^*, M) as a signing query where $M = M^*$ is allowed, and obtains Σ. For the forgery $(\mathsf{dpk}^*, M^*, \Sigma^*)$, it is required that $(M^*, \Sigma^*) \notin \{(M, \Sigma)\}$ holds.

1.3 Our Contribution

In this paper, we revisited the security notions of PDPKS. First, we introduce a new security notion which we call consistency: for (distinct) two master public keys mpk_0 and mpk_1, it guarantees that $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}) = 0$ when $\mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_1)$ with overwhelming probability (we define consistency in a computational security manner later). Second, we weaken unforgeability which we call outsider unforgeability. Third, we strengthen unforgeability to capture strong unforgeability. By combining outsider unforgeability, which we call outsider strong unforgeability, our definition guarantees the situation that nobody, except the payer and the payee, can produce a valid signature.

As mentioned in the motivation part, from the viewpoint of functionality, outsider unforgeability allows us to implement a refund functionality. Moreover, it also allows us to remove the IBS-like construction procedure unlike the Liu et al.'s construction, and can provide a generic construction of PDPKS. The design is conceptually much simpler than known PDPKS constructions because the ingredients are signatures and PKE only, and thus we can easily provide strong unforgeability. The proposed generic construction provides correctness, outsider strong unforgeability, unlinkability, and consistency.

Feasibility of the Generic Construction. The underlying signature scheme is assumed to be strongly unforgeable. As a candidate, we can employ the Boneh-Shen-Waters signature scheme [13] which is strongly unforgeable under the computational Diffie-Hellman assumption in bilinear groups, and is secure in the standard model. We can also employ generic conversions [40, 41] to obtain a strongly unforgeable signature scheme from any signature scheme. We also require that the signature scheme provides strong conservative exclusive ownership (S-CEO) security [20]. S-CEO is recognized as one of BUFF (Beyond UnForgeability Features) security, and it informally guarantees that for a verification key vk^*, no adversary can produce vk such that there is (M, Σ) satisfying $\mathsf{Sig.Verify}(\mathsf{vk}^*, M, \sigma) = 1$, $\mathsf{Sig.Verify}(\mathsf{vk}, M, \sigma) = 1$, and $\mathsf{vk} \neq \mathsf{vk}^*$. Cremers et al. [20] gave a generic conversion to provide the S-CEO security by adding $\mathsf{Hash}(\mathsf{vk}, M)$ to a signature. Here, Hash is a collision resistant hash function and is *not* modeled as a random oracle. We can add the S-CEO security to the Boneh-Shen-Waters signature scheme via the generic transformation. As a concrete scheme, Brendel et al. [14] showed that the Ed25519-LibS signature scheme is strong unforgeable and provides the malicious-strong universal exclusive ownership (M-S-UEO) security that implies the S-CEO security. Moreover, Aulbach et al. [5] showed that NIST PQC candidates, CROSS[4], HAETAE[5], HAWK[6], RACCOON[7], and PROV[8], provide the S-CEO security. Among them, we can employ HAETAE, HAWK, and RACCOON because they are strongly unforgeable. See the algorithm specification documents[9] for details.

[4] `https://www.cross-crypto.com/`
[5] `https://kpqc.cryptolab.co.kr/haetae`
[6] `https://hawk-sign.info/`
[7] `https://raccoonfamily.org/`
[8] `https://prov-sign.github.io/`
[9] `https://csrc.nist.gov/projects/pqc-dig-sig`

Table 1: Comparisons of PDPKS Schemes

Scheme	Assumptions	ROM/SDM	Unforgeability
Liu et al. [33]	q-SDH, CDH (in bilinear groups)	ROM	Nomal
Liu et al. [32]	LWE, SIS	ROM	Nomal
Ours 1	CDH (in bilinear groups), DDH, CR	SDM	Strong and Outsider
Ours 2	ECDLP, ODH	ROM	Strong and Outsider
Ours 3	ECDLP, DDH, CR	ROM	Strong and Outsider
Ours 4-1	MLWE, SIS, CR, LWE, GapSVP	ROM	Strong and Outsider
Ours 4-2	One-more SVP, CR, LWE, GapSVP	ROM	Strong and Outsider
Ours 4-3	Self-target MSIS, CR, LWE, GapSVP	ROM	Strong and Outsider

Table 2: Efficiency Comparisons of DL/Pairing-based PDPKS Schemes

Scheme	$\lvert\mathsf{mpk}\rvert$	$\lvert\mathsf{msk}\rvert$	$\lvert\mathsf{dpk}\rvert$	$\lvert\mathsf{dsk}\rvert$	$\lvert\Sigma\rvert$
Liu et al. [33]	$2\lvert\mathbb{G}_2\rvert$	$2\lvert\mathbb{Z}_p\rvert$	$2\lvert\mathbb{G}_2\rvert$	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_1\rvert + \lvert\mathbb{Z}_p\rvert$
Generic Construction	$\lvert\mathsf{PKE.pk}\rvert$	$\lvert\mathsf{PKE.dk}\rvert$	$\lvert\mathsf{vk}\rvert + \lvert C_{\mathsf{PKE}}\rvert$	$\lvert\mathsf{sigk}\rvert$	$\lvert\Sigma\rvert$
Ours 1	$3\lvert\mathbb{G}\rvert$	$6\lvert\mathbb{Z}_p\rvert$	$O(\log\lambda)\lvert\mathbb{G}_1\rvert + 4\lvert\mathbb{G}\rvert$	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_2\rvert + \lvert\mathbb{Z}_p\rvert + \lvert\mathsf{CR}\rvert$
Ours 2	$\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_p\rvert$	$2\lvert\mathbb{G}\rvert + \lvert C_{\mathsf{SKE}}\rvert + \lvert\mathsf{MAC}\rvert$	$\lvert\mathbb{Z}_p\rvert$	$\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_p\rvert$
Ours 3	$3\lvert\mathbb{G}\rvert$	$6\lvert\mathbb{Z}_p\rvert$	$5\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_p\rvert$	$\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_p\rvert$

The underlying CCA-secure PKE scheme is assumed to be key private [10] (it ensures that a ciphertext does not leak information of the public key) and strongly robust [1] (it ensures that the decryption result of a ciphertext is $\perp$ if the ciphertext is not produced by the corresponding public key). Abdalla et al. [1] gave a strongly robust variant of the Cramer-Shoup PKE scheme [19] and the DHIES (Diffie-Hellman integrated encryption scheme) PKE scheme [2], respectively, that are also CCA secure and key private. Abdalla et al. [1] also proposed a generic transformation to add robustness to any key private and CCA-secure PKE scheme. The transformation additionally requires a commitment scheme that provides a standard hiding and binding properties. Thus, for a lattice-based instantiation of the PKE scheme, we can employ a key private variant of the Regev PKE given by Liu et al. [32], with the Abdalla et al. transformation.

Comparison. In summary, we compare interesting instantiations with previous schemes in Table 1. Here, ROM stands for random oracle model and SDM stands for standard model. SDH stands for strong Diffie-Hellman, CDH/DDH stands for computational/decisional Diffie-Hellman, (M)LWE stands for (module) learning with errors, SIS stands for short integer solution, CR stands for collision resistant hash, ODH stands for oracle Diffie-Hellman, and SVP stands for shortest vector problem.

Ours 1: A pairing-based PDPKS scheme without random oracles instantiated from the Boneh-Shen-Waters signature scheme [13] with Cremers et al.'s conversion [20] and a strongly robust variant of the Cramer-Shoup PKE scheme [1]. Since previous PDPKS schemes are secure in the random oracle model. Due to the result by Canetti et al. [15], random oracles should not be employed as much as possible.

Ours 2: A discrete-logarithm (DL) based pairing-free PDPKS scheme in the random oracle model instantiated from the Ed25519-LibS signature scheme [14] and a strongly robust variant of the DHIES PKE scheme [1].

Ours 3: A DL based pairing-free PDPKS scheme in the random oracle model instantiated from the Ed25519-LibS signature scheme [14] and a strongly robust variant of the Cramer-Shoup

PKE scheme [1]. Compared to the scheme described as "Ours 2", we can avoid to employ an oracle assumption, and the scheme is secure under standard assumptions in the random oracle model, whereas the Liu et al. scheme [33] relies on the q-SDH assumption. Due to the Cheon attack [17], q-type assumptions should not be employed as much as possible.

Ours 4: Lattice-based PDPKS schemes in the random oracle model instantiated from (4-1) HAETAE, (4-2) HAWK or (4-3) RACCOON, and a strongly robust variant of the Liu et al.'s key private PKE scheme (i.e., the key private PKE scheme given in [32] is converted by the Abdalla et al.'s transformation by using a lattice-based commitment scheme such as the KXT commitment [31]).

We also compare the size of keys and signatures among DL/pairing-based PDPKS schemes in Table 2. To clarify these sizes in the proposed generic constriction, we add the generic construction row in the table. In the generic construction, a PDPKS signature is a signature of the underlying signature scheme. Thus, we use the same notation Σ. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be bilinear groups with prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a pairing (i.e., BN curves [9] or BLS curves [8]). Let $\mathbb{G}$ be a group with prime order p (i.e, Curve25519 [11]). $|\mathsf{CR}|$ is due to Cremers et al.'s conversion, and is estimated as 2λ due to the birthday bound. For Ed25519, a signature contains an element of the elliptic curve and a value modulo L. Here, L is specified by the prime generator order where for a generator $g \in \mathbb{G}$, $g^L = 1$ holds.[10] Thus, we denote $|\mathbb{G}| + |\mathbb{Z}_p|$ in the table as the signature size of Ed25519.

Technical Overview. We further explore the reason why an IBS-like construction is required in the Liu et al. PDPKS scheme [33]. Due to the syntax of PDPKS, anyone who observes a master public key mpk_B can derive a public key dpk_A, and unforgeability guarantees that nobody, except the corresponding master secret key holder (the payee Bob), can produce a valid signature under the derived public key dpk_A. That is, dsk_A is derived from dpk_A by using msk_B *after* dpk_A is generated. This key generation process requires an IBS-like construction procedure. Here, we weaken the unforgeability that captures the situation that nobody, except the payer and the payee, can produce a valid signature. This relaxation allows a payer Alice to produce dsk_A together with dpk_A and can remove the IBS-like construction procedure. In our construction, a payer Alice chooses a verification and signing key pair of a signature scheme $(\mathsf{vk}_A, \mathsf{sigk}_A)$ on the fly, and encrypts sigk_A by using the payee's public key $\mathsf{mpk}_B = \mathsf{PKE.pk}_B$. That is, nobody, except Alice and the decryption key holder Bob, can produce the corresponding signing key sigk_A which means that vk_A is linked to mpk_B. By using sigk_A, Bob can produce a signature which is valid under vk_A. The design is conceptually much simpler than known PDPKS constructions.

We need to further assume that the underlying signature scheme is S-CEO secure, which is the most technical part of this paper. We give an intuition of the security proof of outsider strong unforgeability (See Section 5 for detail). Let $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$ where C^*_{PKE} is a ciphertext of sigk^*. Before reducing strong unforgeability of the underlying signature scheme, we need to guarantee that $\mathcal{A}$ does not produce dpk as a derived secret key corruption query such that $\mathsf{dpk} = (\mathsf{vk}, C^*_{\mathsf{PKE}})$, $\mathsf{vk} \neq \mathsf{vk}^*$, but vk is a valid verification key relative to sigk^*. Since the $\mathsf{DpkCheck}$ algorithm returns 1 for dpk, the reduction algorithm needs to respond sigk^* and fails to reduce strong unforgeability. To exclude the case, we employ the S-CEO security. The reduction algorithm takes vk^* from the challenger of the S-CEO security. C^*_{PKE} has been replaced to a ciphertext of $0^{|\mathsf{sigk}|}$ by employing the IND-CCA security of the PKE scheme. Thus, the reduction algorithm can produce $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$. If an adversary sends $\mathsf{dpk} = (\mathsf{vk}, C^*_{\mathsf{PKE}})$ where $\mathsf{vk} \neq \mathsf{vk}^*$, the reduction algorithm sends

[10] The authors of [14] employ additive operations. We introduce their notations: E is an elliptic curve, and B is a generator of the prime order subgroup of E, and c is the $\log_2$ of curve cofactor, Then, $LB = 0$ and $2^c L = |E|$ hold.

a signing query m to the challenger, and obtains a signature Σ. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \Sigma) = 1$, then the reduction algorithm breaks the S-CEO security, and does not have to respond the derived secret key corruption query for dpk to reduce strong unforgeability.

Related Work. As a similar primitive of PDPKS, Wang et al. [43] proposed key derivable signature (KDS) that allows a master secret key to sign a message unlike PDPKS, and proposed a generic construction of KDS from a key derivation scheme (KDV), PKE, and signatures *in the random oracle model*. Though their construction methodology could be employed to construct PDPKS, it deeply depends on random oracles, whereas our generic construction of PDPKS itself does not rely on random oracles. Pu et al. [38] proposed a post-quantum stealth signature scheme which they call Spirit. They also proposed a generic transformation from a stealth signature scheme without key-exposure into a scheme with unbounded key-exposure. Their construction is not purely generic in the sense that they first constructed a stealth signature scheme based on Dilithium and transformed it to a scheme with key exposure. As an independent and concurrent work, Mongardini et al. [36] proposed identity-based matchmaking signatures (IB-MSS) based on the stealth signatures model in [38]. As in our outsider unforgeabiity, the challenge one-time public key is generated by the challenger, and is not generated by the adversary. One crucial difference is IB-MSS involves a certification authority (CA) that generates sender/receiver keys using a master secret key to ensure that participants comply with specific regulations, such as anti-money laundering (AML) and know your customer (KYC) requirements.

Related Work in terms of Exclusive Ownership security. We explicitly assume that the underlying signature scheme is S-CEO secure. Unlike strong unforgeability, which is widely used for enhancing a security level, e.g., the CHK transformation [16], exclusive ownership security notions have not been widely employed as a security property of the underlying signature scheme, with the following one exception (to the best of our knowledge). Gunther et al. [29] showed that a lightweight authenticated key exchange protocol for IoT communication, EDHOC (Ephemeral Diffie-Hellman Over COSE, COSE stands for CBOR Object Signing and Encryption, and CBOR stands for Concise Binary Object Representation), the underlying signature scheme is explicitly assumed to be universal exclusive ownership (S-UEO) secure to provide multi-stage key exchange security.

As another related works, Boneh et al. [12] introduced strong binding for multi-signatures as a related definition of message-bound signatures. Ferreira and Pascal [24] employed Dilithium and EdDSA to instantiate their post-quantum secure ZRTP (which is an authenticated key exchange protocol for establishing secure communications for Voice over IP applications) and mentioned that both signature schemes are S-UEO secure. But they did not explicitly require the S-UEO security to prove the security of their protocol. Jiang and Wang [30] also mentioned exclusive ownership security notions, but they did not employ the notions and claimed that their protocol is secure under the standard unforgeability because both the user and servers employ the user's fixed public key for verification.

Since we explicitly require the underlying signature scheme to be S-CEO secure, our security proof introduces a new insight of exclusive ownership security which may be of independent interest.

Adaptor Signatures. We regard refundability is a suitable functionality, and leave further considerations on the functionality in detail as a future work. Nevertheless, we would like to mention that refundability has been implicitly realized via adaptor signatures, e.g., [6,21,27,37]. When Alice has a token c_A for some cryptocurrency and Bob has a witness y of some instance Y, we consider the case that Alice and Bob would like to trade c_A and y. Alice generates a pair of verification key and signing key $(\mathsf{vk}, \mathsf{sigk})$ and posts a transaction to the blockchain that transfers c_A to Bob if a valid signature σ under vk is sent. Then, Alice generates a pre-signature on the transaction using

sigk and Y, and sends it to Bob. Bob adopts the pre-signature and generates a full signature using y. Since the full signature is valid under vk, Bob can obtain c_A by sending σ to the blockchain. After σ is published, Alice can extract y from the pre-signature and the full signature. Thus, a fair exchange completes. Obviously, Alice can withdraw c_A because Alice has sigk. One crucial difference from ours is that the adaptor signature based refundable address is not stealth. Since, in this example, Alice does not spend a coin to an address and just uses own wallet, we should not have called the functionality refundable. The common point is both Alice (the payer) and Bob (the payee) can issue the transaction to spend c_A. From this perspective, PDPKS and adaptive signatures may have a relationship.

2 Preliminaries

2.1 Signatures and PKE

In this section, we define signatures and PKE. We introduce setup algorithms to employ the same parameter among all users (payers and payees).

Signatures. Let $\mathsf{Sig} = (\mathsf{Sig.Setup}, \mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ be a signature scheme. The setup algorithm $\mathsf{Sig.Setup}$ takes a security parameter λ, and outputs a common parameter $\mathsf{pp}_{\mathsf{Sig}}$ that implicitly contains a message space $\mathsf{MS}_{\mathsf{Sig}}$. The key generation algorithm takes $\mathsf{pp}_{\mathsf{Sig}}$ as input and outputs a verification and signing key pair $(\mathsf{vk}, \mathsf{sigk})$. The signing algorithm $\mathsf{Sig.Sign}$ takes sigk and a message to be signed $M \in \mathsf{MS}_{\mathsf{Sig}}$ as input and outputs a signature Σ. The verification algorithm takes vk, M, and Σ as input and outputs 0 or 1. The correctness requires that for any $\lambda \in \mathbb{N}$, any $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(\mathsf{pp}_{\mathsf{Sig}})$, any $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$, and any $M \in \mathsf{MS}_{\mathsf{Sig}}$, $\mathsf{Sig.Verify}(\mathsf{vk}, M, \mathsf{Sig.Sign}(\mathsf{sigk}, M)) = 1$ holds with overwhelming probability in the security parameter λ.

Strong unforgeability is defined as follows. Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(\mathsf{pp}_{\mathsf{Sig}})$ and $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$, and gives $(\mathsf{pp}_{\mathsf{Sig}}, \mathsf{vk})$ to $\mathcal{A}$. $\mathcal{C}$ initiates $\mathsf{SigSet} = \emptyset$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ generates $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$ and returns Σ to $\mathcal{A}$. $\mathcal{C}$ stores (M, Σ) to SigSet. Finally, $\mathcal{A}$ outputs (M^*, Σ^*). We say that $\mathcal{A}$ wins if $\mathsf{Sig.Verify}(\mathsf{vk}, M^*, \Sigma^*) = 1$ and $(M^*, \Sigma^*) \notin \mathsf{SigSet}$. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathrm{strong}}_{\mathcal{A},\mathsf{Sig}}(\lambda) = \Pr[\mathcal{A} \text{ wins}]$. We say that Sig is strongly unforgeable if $\mathsf{Adv}^{\mathrm{strong}}_{\mathcal{A},\mathsf{Sig}}(\lambda)$ is negligible for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$.

The S-CEO security is defined as follows. Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(\mathsf{pp}_{\mathsf{Sig}})$ and $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$, and gives $(\mathsf{pp}_{\mathsf{Sig}}, \mathsf{vk})$ to $\mathcal{A}$. $\mathcal{C}$ initiates $\mathsf{SigSet} = \emptyset$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ generates $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$ and returns Σ to $\mathcal{A}$. $\mathcal{C}$ stores (M, Σ) to SigSet. Finally, $\mathcal{A}$ outputs $(\mathsf{vk}^*, M^*, \Sigma^*)$. We say that $\mathcal{A}$ wins if $\mathsf{Sig.Verify}(\mathsf{vk}^*, M^*, \Sigma^*) = 1$, $(M^*, \Sigma^*) \in \mathsf{SigSet}$ (that implies $\mathsf{Sig.Verify}(\mathsf{vk}, M^*, \Sigma^*) = 1$), and $\mathsf{vk}^* \neq \mathsf{vk}$. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathrm{S\text{-}CEO}}_{\mathcal{A},\mathsf{Sig}}(\lambda) = \Pr[\mathcal{A} \text{ wins}]$. We say that Sig is S-CEO secure if $\mathsf{Adv}^{\mathrm{S\text{-}CEO}}_{\mathcal{A},\mathsf{Sig}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

PKE. Let $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE scheme. The setup algorithm $\mathsf{PKE.Setup}$ takes a security parameter λ as input, and outputs a common parameter $\mathsf{pp}_{\mathsf{PKE}}$ that implicitly contains a message space $\mathsf{MS}_{\mathsf{PKE}}$. The key generation algorithm $\mathsf{PKE.KeyGen}$ takes $\mathsf{pp}_{\mathsf{PKE}}$, and outputs a key pair $(\mathsf{PKE.pk}, \mathsf{PKE.dk})$. The encryption algorithm $\mathsf{PKE.Enc}$ takes $\mathsf{PKE.pk}$ and a plaintext M, and outputs a ciphertext C_{PKE}. The decryption algorithm $\mathsf{PKE.Dec}$ takes $\mathsf{PKE.dk}$ and C_{PKE} as input, and outputs M or $\perp$. Correctness requires that for any $\lambda \in \mathbb{N}$, any $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, any $(\mathsf{PKE.pk}, \mathsf{PKE.dk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, any $M \in \mathsf{MS}_{\mathsf{PKE}}$,

$\mathsf{PKE.Dec}(\mathsf{PKE.dk}, \mathsf{PKE.Enc}(\mathsf{PKE.pk}, M)) = M$ holds with overwhelming probability in the security parameter λ.

The CCA security is defined as follows. Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and $(\mathsf{PKE.pk}, \mathsf{PKE.dk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and gives $(\mathsf{pp}_{\mathsf{PKE}}, \mathsf{PKE.pk})$ to $\mathcal{A}$. $\mathcal{A}$ is allowed to issue decryption queries C_{PKE}. $\mathcal{C}$ returns the result of $\mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$. $\mathcal{A}$ declares two equal-length plaintexts M_0^* and M_1^*. $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$, computes the challenge ciphertext $C^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}, M_b^*)$, and gives C^*_{PKE} to $\mathcal{A}$. $\mathcal{A}$ is further allowed to issue decryption queries $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$. $\mathcal{C}$ returns the result of $\mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$. $\mathcal{A}$ wins if $b = b'$. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{IND-CCA}}_{\mathcal{A},\mathsf{PKE}}(\lambda) = |\Pr[b = b'] - 1/2|$. We say that PKE is IND-CCA secure (or simply CCA secure) if $\mathsf{Adv}^{\text{IND-CCA}}_{\mathcal{A},\mathsf{PKE}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

Key privacy is defined as follows. Note that the following definition contains CCA security. Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, $(\mathsf{PKE.pk}_0, \mathsf{PKE.dk}_0) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and $(\mathsf{PKE.pk}_1, \mathsf{PKE.dk}_1) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and gives $(\mathsf{pp}_{\mathsf{PKE}}, \mathsf{PKE.pk}_0, \mathsf{PKE.pk}_1)$ to $\mathcal{A}$. $\mathcal{A}$ is allowed to issue decryption queries (C_{PKE}, i) where $i \in \{0,1\}$. $\mathcal{C}$ returns the result of $\mathsf{PKE.Dec}(\mathsf{PKE.dk}_i, C_{\mathsf{PKE}})$. $\mathcal{A}$ declares the challenge plaintext M^*. $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$, computes the challenge ciphertext $C^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}_b, M^*)$, and gives C^*_{PKE} to $\mathcal{A}$. $\mathcal{A}$ is further allowed to issue decryption queries (C_{PKE}, i) where $i \in \{0,1\}$ and $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$. $\mathcal{C}$ returns the result of $\mathsf{PKE.Dec}(\mathsf{PKE.dk}_i, C_{\mathsf{PKE}})$. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$. $\mathcal{A}$ wins if $b = b'$. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{Key-Privacy}}_{\mathcal{A},\mathsf{PKE}}(\lambda) = |\Pr[b = b'] - 1/2|$. We say that PKE is key private if $\mathsf{Adv}^{\text{Key-Privacy}}_{\mathcal{A},\mathsf{PKE}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

Strong robustness is defined as follows (here strong means that robustness holds for ciphertexts declared by an adversary). Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, $(\mathsf{PKE.pk}_0, \mathsf{PKE.dk}_0) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and $(\mathsf{PKE.pk}_1, \mathsf{PKE.dk}_1) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and gives $(\mathsf{pp}_{\mathsf{PKE}}, \mathsf{PKE.pk}_0, \mathsf{PKE.pk}_1)$ to $\mathcal{A}$. $\mathcal{A}$ declares the challenge ciphertext C^*_{PKE}. $\mathcal{C}$ runs $M_0 \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}_0, C^*_{\mathsf{PKE}})$ and $M_1 \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}_1, C^*_{\mathsf{PKE}})$. $\mathcal{C}$ outputs 1 if $M_0 \neq \perp$ and $M_1 \neq \perp$ hold, and 0 otherwise. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{strong-robust}}_{\mathcal{A},\mathsf{PKE}}(\lambda) = \Pr[\mathcal{C} \rightarrow 1]$. We say that PKE is strongly robust if $\mathsf{Adv}^{\text{strong-robust}}_{\mathcal{A},\mathsf{PKE}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

2.2 PDPKS

Next, we introduce the syntax and security notions of PDPKS defined by Liu et al. [33]. We note that the definition of consistency and outsider unforgeability, that are newly introduced in this paper, are given in Section 3. Let PDPKS be a PDPKS scheme that consists of seven algorithms ($\mathsf{PDPKS.Setup}$, $\mathsf{MasterKeyGen}$, $\mathsf{DpkDerive}$, $\mathsf{DpkCheck}$, $\mathsf{DskDerive}$, $\mathsf{PDPKS.Sign}$, $\mathsf{PDPKS.Verify}$) defined as follows.

$\mathsf{PDPKS.Setup}$: The setup algorithm takes a security parameter λ as input, and outputs a common parameter pp that implicitly contains a message space $\mathsf{MS}_{\mathsf{PDPKS}}$.

$\mathsf{MasterKeyGen}$: The master key generation algorithm takes pp as input, and outputs a master public key and a master secret key pair $(\mathsf{mpk}, \mathsf{msk})$.

$\mathsf{DpkDerive}$: The public key derivation algorithm takes pp and mpk as input, and outputs a derived public key dpk.

$\mathsf{DpkCheck}$: The derived public key checking algorithm takes pp, mpk, msk, and dpk as input, and outputs 1 (meaning that dpk is linked to mpk) or 0 (meaning that dpk is not linked to mpk).

$\mathsf{DskDerive}$: The secret key derivation algorithm takes pp, mpk, msk, and dpk as input. The algorithm outputs $\perp$ if $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 0$. Otherwise, the algorithm outputs a derived secret key dsk corresponding to dpk.

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

$\mathsf{PDPKS.Verify}$: The verification algorithm tales pp, dpk, M, and Σ as input, and outputs 1 (valid) or 0 (invalid).

Correctness. It requires that for any $\lambda \in \mathbb{N}$, any $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^\lambda)$, any $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, any $\mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk})$, and any $M \in \mathsf{MS}_{\mathsf{PDPKS}}$,

$$\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 1 \text{ and } \mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}, M, \Sigma) = 1$$

hold with overwhelming probability in the security parameter λ, where $\Sigma \leftarrow \mathsf{PDPKS.Sign}(\mathsf{pp}, \mathsf{dpk}, \mathsf{dsk}, M)$ and $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk})$.

Next, we define unforgeability as follows. We explicitly return $\perp$ if the $\mathsf{DpkCheck}$ algorithm returns 0 for derived secret key corruption queries. Note that $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 1$ holds for $\mathsf{dpk} \in L_{\mathsf{dpk}}$. Thus, the challenger does not return $\perp$ for signing queries. Since it is required that $\mathcal{A}$ did not send (dpk^*, M^*) as a signing query, the following does not capture strongly unforgeability.

Definition 1 (Unforgeability)**.** *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^\lambda)$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, and gives $(\mathsf{pp}, \mathsf{mpk})$ to $\mathcal{A}$. $\mathcal{C}$ initializes $L_{\mathsf{dpk}} := \emptyset$. $\mathcal{A}$ is allowed to issue the following queries.*

Derived Public Key Check Query: *$\mathcal{A}$ sends dpk to $\mathcal{C}$. $\mathcal{C}$ returns the result of $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk})$. If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 1$, then $\mathcal{C}$ updates $L_{\mathsf{dpk}} \leftarrow L_{\mathsf{dpk}} \cup \{\mathsf{dpk}\}$.*

Derived Secret Key Corruption Query: *$\mathcal{A}$ sends $\mathsf{dpk} \in L_{\mathsf{dpk}}$ to $\mathcal{C}$. $\mathcal{C}$ returns $\perp$ if $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 0$. Otherwise, $\mathcal{C}$ returns $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk})$.*

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

Finally, $\mathcal{A}$ outputs $(\mathsf{dpk}^*, M^*, \Sigma^*)$ where $M^* \in \mathsf{MS}_{\mathsf{PDPKS}}$. $\mathcal{A}$ wins if $\mathsf{dpk}^* \in L_{\mathsf{dpk}}$, $\mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}^*, M^*, \Sigma^*) = 1$, $\mathcal{A}$ did not send dpk^* as a derived secret key corruption query, and $\mathcal{A}$ did not send (dpk^*, M^*) as a signing query. The advantage is defined as

$$\mathsf{Adv}^{unforge}_{\mathcal{A},\mathsf{PDPKS}}(\lambda) = \Pr[\mathcal{A} \textit{ wins}]$$

We say that a PDPKS scheme PDPKS is unforgeable if $\mathsf{Adv}^{unforge}_{\mathcal{A},\mathsf{PDPKS}}(\lambda)$ is negligible in the security parameter λ for all PPT adversaries $\mathcal{A}$.

Next, we define unlinkability as follows. As in unforgeability, we explicitly return $\perp$ if the $\mathsf{DpkCheck}$ algorithm returns 0 for derived secret key corruption queries.

Definition 2 (Unlinkability)**.** *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^\lambda)$, $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, and $(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, $\mathcal{C}$ chooses $b \leftarrow \{0, 1\}$ and computes $\mathsf{dpk}^* \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_b)$. $\mathcal{C}$ initializes $L_{\mathsf{dpk},0} := \emptyset$ and $L_{\mathsf{dpk},1} := \emptyset$. We remark that $\mathsf{dpk}^* \notin L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1}$. $\mathcal{C}$ gives $(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1, \mathsf{dpk}^*)$ to $\mathcal{A}$. $\mathcal{A}$ is allowed to issue the following queries.*

Derived Public Key Check Query: *$\mathcal{A}$ sends $\mathsf{dpk} \neq \mathsf{dpk}^*$ and 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 Corruption Query: *$\mathcal{A}$ sends $\mathsf{dpk} \in L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1}$ to $\mathcal{C}$. $\mathcal{C}$ returns $\perp$ if $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{dpk}) = 0$ where $\mathsf{dpk} \in L_{\mathsf{dpk},i}$. Otherwise, $\mathcal{C}$ returns $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{dpk})$.*

Signing Query: *$\mathcal{A}$ sends $M \in \mathsf{MS}_{\mathsf{PDPKS}}$ and $\mathsf{dpk} \in L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1} \cup \{\mathsf{dpk}^*\}$ to $\mathcal{C}$. $\mathcal{C}$ returns $\Sigma \leftarrow \mathsf{PDPKS.Sign}(\mathsf{pp}, \mathsf{dpk}, \mathsf{dsk}, M)$. Here $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{dpk})$ where $\mathsf{dpk} \in L_{\mathsf{dpk},i}$ and $i = b$ if $\mathsf{dpk} = \mathsf{dpk}^*$.*

Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$. $\mathcal{A}$ wins if $b = b'$. The advantage is defined as

$$\mathsf{Adv}^{unlink}_{\mathcal{A},\mathsf{PDPKS}}(\lambda) = |\Pr[b = b'] - 1/2|$$

We say that a PDPKS scheme PDPKS is unlinkable if $\mathsf{Adv}^{unlink}_{\mathcal{A},\mathsf{PDPKS}}(\lambda)$ is negligible in the security parameter λ for all PPT adversaries $\mathcal{A}$.

3 New Definitions: Consistency and Outsider Strong Unforgeability

3.1 Definition of Consistency

To capture the condition that $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 0$ if dpk is not derived from mpk, we define consistency as follows. Here, $\mathcal{A}$ is not allowed to issue a derived public key check query unlike the other definition. This is reasonable because the $\mathsf{DpkCheck}$ algorithm is internally run to respond the query, and consistency considers a security of the $\mathsf{DpkCheck}$ algorithm. On the other hand, we need to guarantee that no adversary can break consistency even if the adversary has observed a valid signature. Thus, the adversary is allowed to issue signing queries but dpk is derived by the challenger.

Definition 3 (Consistency)**.** *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^\lambda)$, $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, 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 $M \in \mathsf{MS}_{\mathsf{PDPKS}}$ and index $i \in \{0,1\}$ to $\mathcal{C}$. $\mathcal{C}$ runs $\mathsf{dpk} \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk}_i)$, $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{dpk})$, and $\Sigma \leftarrow \mathsf{PDPKS.Sign}(\mathsf{pp}, \mathsf{dpk}, \mathsf{dsk}, M)$, and returns (dpk, Σ) to $\mathcal{A}$.*

Finally, $\mathcal{A}$ outputs dpk^*. $\mathcal{A}$ wins if $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}^*) = 1$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}^*) = 1$ hold. The advantage is defined as

$$\mathsf{Adv}^{consist}_{\mathcal{A},\mathsf{PDPKS}}(\lambda) = \Pr[\mathcal{A} \text{ wins}]$$

We say that a PDPKS scheme PDPKS is consistent if $\mathsf{Adv}^{consist}_{\mathcal{A},\mathsf{PDPKS}}(\lambda)$ is negligible in the security parameter λ for all PPT adversaries $\mathcal{A}$.

Analysis of Previous PDPKS schemes: Here, we demonstrate whether previous PDPKS schemes provide consistency or not. We give a brief analysis and decline to give a formal proof here. The pairing-based Liu et al. PDPKS scheme [33] is briefly described as follows (Appendix may help the reader to understand the scheme). Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a pairing where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups with prime order p, and $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generators. $\mathsf{mpk} = (\mathsf{mpk}_1, \mathsf{mpk}_2) = (g_2^{\alpha}, g_2^{\beta}) \in \mathbb{G}_2^2$ and $\mathsf{msk} = (\mathsf{msk}_1, \mathsf{msk}_2) = (\alpha, \beta) \in \mathbb{Z}_p^2$. To derive dpk, choose $r \xleftarrow{\$} \mathbb{Z}_p$ and compute $\mathsf{dpk} = (\mathsf{dpk}_1, \mathsf{dpk}_2) = (g_2^r, \mathsf{mpk}_2 \cdot g_2^{\mathsf{Hash}(g_2^r, \mathsf{mpk}_1^r)})$. Here, Hash is modeled as a random oracle. $\mathsf{mpk}_1^r = g_2^{\alpha r} = \mathsf{dpk}_1^{\mathsf{msk}_1}$ can be seen as a non-interactive key exchange (NIKE) [25] key and $\mathsf{mpk}_2 \cdot g_2^{\mathsf{Hash}(g_2^r, \mathsf{mpk}_1^r)}$ can be seen as a Pedersen commitment for mpk_2 with the randomness (decommit) $\mathsf{Hash}(g_2^r, \mathsf{mpk}_1^r)$. To check whether dpk is derived from mpk, check whether $\mathsf{dpk}_2 = \mathsf{mpk}_2 \cdot g_2^{\mathsf{Hash}(\mathsf{dpk}_1, \mathsf{dpk}_1^{\mathsf{msk}_1})}$ holds or not. Here, $\mathsf{dpk}_1^{\mathsf{msk}_1} = g_2^{r\alpha}$ is the NIKE key. If dpk is linked to two different master public keys $\mathsf{mpk}_0 = (g_2^{\alpha_0}, g_2^{\beta_0})$ and $\mathsf{mpk}_1 = (g_2^{\alpha_1}, g_2^{\beta_1})$, then $\mathsf{dpk}_2 = g_2^{\beta_0} \cdot g_2^{\mathsf{Hash}(\mathsf{dpk}_1, \mathsf{dpk}_1^{\alpha_0})} = g_2^{\beta_1} \cdot g_2^{\mathsf{Hash}(\mathsf{dpk}_1, \mathsf{dpk}_1^{\alpha_1})}$ holds. Since Hash is modeled as a random oracle, $h_0 = \mathsf{Hash}(\mathsf{dpk}_1, \mathsf{dpk}_1^{\alpha_0})$ and $h_1 = \mathsf{Hash}(\mathsf{dpk}_1, \mathsf{dpk}_1^{\alpha_1})$ are uniformly distributed over $\mathbb{Z}_p$ and $h_0 \neq h_1$. Thus, the probability that $\mathsf{dpk}_2 = g_2^{\beta_0 + h_0} = g_2^{\beta_1 + h_1}$ holds is negligible. This implies the Liu et al. scheme is consistent. The Liu et al. lattice-based PDPKS scheme [32] also provides the consistency if the underlying hash functions are modeled as random oracles. Briefly, dpk is computed by a hash value of a plaintext t and the ciphertext τ of the underlying PKE scheme, and τ is contained in dpk. That is, $\mathsf{Hash}(t, \tau)$ can be (informally) seen as a shared key because a master secret key (decryption key of the PKE scheme) holder can obtain t from τ. As in the pairing-based scheme, the probability that a random value (generated via the random oracle) coincidentally satisfies a checking equation is negligible. To sum up, previous PDPKS schemes are consistent if the underlying hash function is modeled as a random oracle.

3.2 Definition of Outsider Strong Unforgeability

Next, we define outsider strong unforgeability as follows. As mentioned before, we weaken unforgeability in the sense that $\mathcal{C}$ sends dpk^* to $\mathcal{A}$, and strengthen unforgeability in the sense that $\mathcal{A}$ is allowed to issue a signing query (dpk^*, M^*).

Definition 4 (Outsider Strong Unforgeability)**.** *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PDPKS.Setup}(1^{\lambda})$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp})$, computes $\mathsf{dpk}^* \leftarrow \mathsf{DpkDerive}(\mathsf{pp}, mpk)$, and gives $(\mathsf{pp}, \mathsf{mpk}, \mathsf{dpk}^*)$ to $\mathcal{A}$. $\mathcal{C}$ initializes $L_{\mathsf{dpk}} := \{\mathsf{dpk}^*\}$ and $L_{\mathsf{Sig}} := \emptyset$. $\mathcal{A}$ is allowed to issue the following queries.*

Derived Public Key Check Query: *$\mathcal{A}$ sends dpk to $\mathcal{C}$ ($\mathsf{dpk} = \mathsf{dpk}^*$ is allowed). $\mathcal{C}$ returns the result of $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk})$. If $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 1$, then $\mathcal{C}$ updates $L_{\mathsf{dpk}} \leftarrow L_{\mathsf{dpk}} \cup \{\mathsf{dpk}\}$.*

Derived Secret Key Corruption Query: *$\mathcal{A}$ sends $\mathsf{dpk} \in L_{\mathsf{dpk}} \setminus \{\mathsf{dpk}^*\}$ to $\mathcal{C}$. $\mathcal{C}$ returns $\perp$ if $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 0$. Otherwise, $\mathcal{C}$ returns $\mathsf{dsk} \leftarrow \mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk})$.*

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

Finally, $\mathcal{A}$ outputs (M^*, Σ^*) where $M^* \in \mathsf{MS}_{\mathsf{PDPKS}}$. $\mathcal{A}$ wins if $\mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}^*, M^*, \Sigma^*) = 1$ and $(M^*, \Sigma^*) \notin L_{\mathsf{Sig}}$. The advantage is defined as

$$\mathsf{Adv}_{\mathcal{A},\mathsf{PDPKS}}^{\textit{outsider-strong-unforge}}(\lambda) = \Pr[\mathcal{A} \textit{ wins}]$$

We say that a PDPKS scheme PDPKS is strongly unforgeable for outsider if $\mathsf{Adv}_{\mathcal{A},\mathsf{PDPKS}}^{\textit{outsider-strong-unforge}}(\lambda)$ is negligible in the security parameter λ for all PPT adversaries $\mathcal{A}$.

4 Proposed Generic Construction

In this section, we give the proposed generic construction. Let $\mathsf{Sig} = (\mathsf{Sig.Setup}, \mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ and $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a signature scheme and a PKE scheme, respectively. A payer (who generates dpk) chooses a verification key and signing key pair $(\mathsf{vk}, \mathsf{sigk})$ and encrypts sigk by using mpk where $\mathsf{mpk} = \mathsf{PKE.pk}$. One may think that the construction is trivial since a signing key is sent via a secure channel encrypted by $\mathsf{PKE.pk}$. This intuition is true and it well explains the fact that the design is conceptually much simpler than known PDPKS constructions. However, the security proof is not trivial, e.g., the underlying signature scheme is required to provide the S-CEO security. First, we give an intuition of the proposed generic construction as follows.

- The $\mathsf{DpkDerive}$ algorithm internally generates a verification key and signing key pair $(\mathsf{vk}, \mathsf{sigk})$ on the fly. Let $\mathsf{mpk} = \mathsf{PKE.pk}$ and $\mathsf{msk} = \mathsf{PKE.dk}$. The algorithm encrypts sigk by using $\mathsf{mpk} = \mathsf{PKE.pk}$ such that $C_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathsf{sigk})$. Set $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. Now, vk is independent to mpk but C_{PKE} depends on mpk. To hide information of mpk, we assume that the underlying PKE scheme is key private. Moreover, the underlying signature scheme is required to provide the S-CEO security (See Section 5 for details).

- The $\mathsf{DskDerive}$ algorithm decrypts C_{PKE} by using $\mathsf{msk} = \mathsf{PKE.dk}$ such that $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$. The algorithm also checks whether sigk is a valid signing key for the verification key vk by generating a signature on a random message. To provide the consistency, we assume that the underlying PKE scheme is robust. Then, $\perp \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$ holds with overwhelming probability if $\mathsf{PKE.dk} \neq \mathsf{msk}$.

- To sign a message (transaction) M, the $\mathsf{PDPKS.Sign}$ algorithm simply signs M using sigk. Then, mpk is not required for verifying a PDPKS signature. To provide strong unforgeability, we assume that the underlying signature scheme is strongly unforgeable. Moreover, the underlying signature scheme is required to provide the S-CEO security (See Section 5 for details).

Here, we give the proposed generic construction.

$\mathsf{PDPKS.Setup}(1^\lambda)$**:** Run $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, and output $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$.

$\mathsf{MasterKeyGen}(\mathsf{pp})$**:** Parse $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$. Run $(\mathsf{PKE.pk}, \mathsf{PKE.dk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$ and output $(\mathsf{mpk}, \mathsf{msk}) = (\mathsf{PKE.pk}, \mathsf{PKE.dk})$.

$\mathsf{DpkDerive}(\mathsf{pp}, \mathsf{mpk})$**:** Parse $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$ and $\mathsf{mpk} = \mathsf{PKE.pk}$. Run $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$ and $C_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathsf{sigk})$. Output $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$.

$\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk})$**:** Parse $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$, $\mathsf{mpk} = \mathsf{PKE.pk}$, $\mathsf{msk} = \mathsf{PKE.dk}$, and $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. Output 0 if $\perp \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$. Otherwise, let $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$. Choose $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$,[11] and output the result of $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}, m))$.

[11] In our construction, $\mathsf{DpkCheck}$ is a probabilistic algorithm though it is a deterministic algorithm in the original definition. This difference does not affect the security.

$\mathsf{DskDerive}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk})$**:** Parse $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$, $\mathsf{mpk} = \mathsf{PKE.pk}$, $\mathsf{msk} = \mathsf{PKE.dk}$, and $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. Output $\mathsf{dsk} = \perp$ if $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 0$. Otherwise, if $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}, \mathsf{msk}, \mathsf{dpk}) = 1$, then sigk are obtained. Output $\mathsf{dsk} = \mathsf{sigk}$.

$\mathsf{PDPKS.Sign}(\mathsf{pp}, \mathsf{dpk}, M, \mathsf{dsk})$**:** Parse $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$, $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$, and $\mathsf{dsk} = \mathsf{sigk}$. Output $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$.

$\mathsf{PDPKS.Verify}(\mathsf{pp}, \mathsf{dpk}, M, \Sigma)$**:** Parse $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$ and $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. Output 1 if $\mathsf{Sig.Verify}(\mathsf{vk}, M, \Sigma) = 1$ holds, and 0 otherwise.

5 Security Analysis

Obviously, correctness directly holds if Sig and PKE are correct. Next, we prove that the proposed construction is consistent.

Theorem 1. *The proposed construction is consistent if the underlying PKE scheme is strongly robust.*

Proof. Let $\mathcal{A}$ be an adversary of consistency and $\mathcal{C}$ be the challenger of strong robustness. We construct an algorithm $\mathcal{B}$ that breaks strong robustness by using $\mathcal{A}$ as follows. First, $\mathcal{C}$ generates $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, $(\mathsf{PKE.pk}_0, \mathsf{PKE.dk}_0) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and $(\mathsf{PKE.pk}_1, \mathsf{PKE.dk}_1) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and gives $(\mathsf{pp}_{\mathsf{PKE}}, \mathsf{PKE.pk}_0, \mathsf{PKE.pk}_1)$ to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{mpk}_0 = \mathsf{PKE.pk}_0$ and $\mathsf{mpk}_1 = \mathsf{PKE.pk}_1$. $\mathcal{B}$ runs $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and sets $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$. $\mathcal{B}$ gives $(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1)$ to $\mathcal{A}$. $\mathcal{B}$ responds queries issued by $\mathcal{A}$ as follows.

Signing Query: $\mathcal{A}$ sends $M \in \mathsf{MS}_{\mathsf{PDPKS}}$ and index $i \in \{0, 1\}$ to $\mathcal{B}$. $\mathcal{B}$ runs $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$, $C_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}_i, \mathsf{sigk})$, and $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$. $\mathcal{B}$ returns $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$ and Σ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$. Since $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{dpk}^*) = 1$ and $\mathsf{DpkCheck}(\mathsf{pp}, \mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{dpk}^*) = 1$ hold, the decryption results of C^*_{PKE} by using $\mathsf{PKE.dk}_0$ and $\mathsf{PKE.dk}_1$ are both non-$\perp$. $\mathcal{B}$ outputs C^*_{PKE} and breaks strong robustness. $\square$

Theorem 2. *The proposed construction is strongly unforgeable for outsider if the underlying PKE scheme is CCA secure and the underlying signature scheme is S-CEO secure and strongly unforgeable.*

Before giving our security proof, we give an intuition of the proof. Let $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$ where C^*_{PKE} is a ciphertext of sigk^*. Our final goal is to reduce strong unforgeability. Then, the challenger of the signature scheme sends the challenge verification key vk^* to the reduction algorithm. Since C^*_{PKE} is a ciphertext of sigk^*, the reduction algorithm cannot produce dpk^*. Thus, before reducing to strong unforgeability, we replace C^*_{PKE} to a ciphertext of $0^{|\mathsf{sigk}|}$ due to the IND-CCA security of the PKE scheme. Here, we implicitly assume that for any $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$, $|\mathsf{sigk}|$ is the same.[12] Now, the reduction algorithm can produce dpk^* by obtaining vk^* from the challenger of the signature scheme and by computing $C^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}, 0^{|\mathsf{sigk}|})$. The remaining issue is how to respond a derived public key check query and a derived secret key corruption query for $\mathsf{dpk} = (\mathsf{vk}, C^*_{\mathsf{PKE}})$ where $\mathsf{vk} \neq \mathsf{vk}^*$ but vk is a valid verification key relative to sigk^*. Since the reduction algorithm needs to return 1 for the derived public key check query dpk, the reduction

[12]This is not a strong requirement. Even if each signing key has a different size, we can artificially add some paddings.

algorithm needs to return sigk^* for the derived secret key corruption query dpk. Thus, we need to guarantee that $\mathcal{A}$ does not produce such a dpk. Now, it is the turn of the S-CEO security. If an adversary produces such a dpk, the reduction algorithm aborts and outputs vk that breaks the S-CEO security. Finally, we show that an algorithm exists that breaks strong unforgeability of the signature scheme.

Proof. We use a game sequence Game_0, Game_1, and Game_2. Let E_i be an event that $\mathcal{A}$ wins in Game_i.

Game_0**.** This is the security game of outsider strong unforgeability. By definition,

$$\mathsf{Adv}_{\mathcal{A},\mathsf{PDPKS}}^{\text{outsider-strong-unforge}}(\lambda) = \Pr[E_0]$$

Game_1**.** This is the same as Game_0 except that C^*_{PKE} is replaced to a ciphertext of $0^{|\mathsf{sigk}^*|}$. We show that there exists an algorithm $\mathcal{B}_1$ such that $|\Pr[E_0] - \Pr[E_1]| \leq \mathsf{Adv}_{\mathcal{B}_1,\mathsf{PKE}}^{\text{IND-CCA}}(\lambda)$ as follows. Let $\mathcal{A}$ be an adversary of the outsider strong unforgeability and $\mathcal{C}$ be the challenger of the PKE scheme. $\mathcal{C}$ generates $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and $(\mathsf{PKE.pk}, \mathsf{PKE.dk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and gives $(\mathsf{pp}_{\mathsf{PKE}}, \mathsf{PKE.pk})$ to $\mathcal{B}_1$. $\mathcal{B}_1$ runs $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and sets $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$ and $\mathsf{mpk} = \mathsf{PKE.pk}$. $\mathcal{B}_1$ runs $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$ and sends $(M_0^*, M_1^*) = (\mathsf{sigk}^*, 0^{|\mathsf{sigk}^*|})$ to $\mathcal{C}$ as the challenge query. $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$, computes the challenge ciphertext $C^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}, M_b^*)$, and returns C^*_{PKE} to $\mathcal{B}_1$. $\mathcal{B}_1$ sets $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$ and gives $(\mathsf{pp}, \mathsf{mpk}, \mathsf{dpk}^*)$ to $\mathcal{A}$. $\mathcal{B}_1$ initializes $L_{\mathsf{dpk}} = \{\mathsf{dpk}^*\}$. $\mathcal{B}_1$ responds queries issued by $\mathcal{A}$ as follows.

Derived Public Key Check Query: $\mathcal{A}$ sends $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$ to $\mathcal{B}_1$. If $\mathsf{dpk} = \mathsf{dpk}^*$, then $\mathcal{B}_1$ returns 1. Otherwise, if $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, then $\mathcal{B}_1$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 1$, then $\mathcal{B}_1$ updates $L_{\mathsf{dpk}} \leftarrow L_{\mathsf{dpk}} \cup \{\mathsf{dpk}\}$ and returns 1. Otherwise, $\mathcal{B}_1$ returns 0. If $\mathsf{vk} \neq \mathsf{vk}^*$ and $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$, $\mathcal{B}_1$ sends C_{PKE} to $\mathcal{C}$ as a decryption query. $\mathcal{C}$ returns $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$ to $\mathcal{B}_1$. $\mathcal{B}_1$ returns 0 if $\mathsf{sigk} = \perp$. If $\mathsf{sigk} \neq \perp$, $\mathcal{B}_1$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 1$, then $\mathcal{B}_1$ updates $L_{\mathsf{dpk}} \leftarrow L_{\mathsf{dpk}} \cup \{\mathsf{dpk}\}$ and returns 1. Otherwise, $\mathcal{B}_1$ returns 0.

Derived Secret Key Corruption Query: $\mathcal{A}$ sends $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$ where $\mathsf{dpk} \in L_{\mathsf{dpk}} \setminus \{\mathsf{dpk}^*\}$. If $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, then $\mathcal{B}_1$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 1$, then $\mathcal{B}_1$ returns sigk^*, and $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 0$, then $\mathcal{B}_1$ returns $\perp$. If $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$, then $\mathcal{B}_1$ sends C_{PKE} to $\mathcal{C}$ as a decryption query. $\mathcal{C}$ returns $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$ to $\mathcal{B}_1$. $\mathcal{B}_1$ returns $\perp$ if $\mathsf{sigk} = \perp$. Otherwise, $\mathcal{B}_1$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$ and returns sigk to $\mathcal{A}$ if $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}, m)) = 1$, and $\perp$ if $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}, m)) = 0$.

Signing Query: $\mathcal{A}$ sends $M \in \mathsf{MS}_{\mathsf{PDPKS}}$ and $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}}) \in L_{\mathsf{dpk}}$ to $\mathcal{B}_1$. Since $\mathsf{dpk} \in L_{\mathsf{dpk}}$, $\mathcal{A}$ has sent dpk as a derived public key check query, and $\mathcal{B}_1$ retrieves $\mathsf{dsk} = \mathsf{sigk}$ by internally issuing a derived secret key corruption query or uses sigk if $\mathcal{A}$ has issued dpk as a derived secret key corruption query. $\mathcal{B}_1$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$.

If $b = 0$, then $\mathcal{B}_1$ simulates Game_0 and if $b = 1$, $\mathcal{B}_1$ simulates Game_1. Thus, $|\Pr[E_0] - \Pr[E_1]| \leq \mathsf{Adv}_{\mathcal{B}_1,\mathsf{PKE}}^{\text{IND-CCA}}(\lambda)$ holds.

Game_2**.** This is the same as Game_1 except that the response of derived public key check queries is changed. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$ be a derived public key check query. If $\mathsf{vk} \neq \mathsf{vk}^*$, $C_{\mathsf{PKE}} =$

C^*_{PKE}, and $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 1$ holds for a random choice of $m \in \mathsf{MS}_{\mathsf{Sig}}$, then the challenger output 0. If this event does not happen, Game_1 and Game_2 are identical. Thus, $|\Pr[E_1]-\Pr[E_2]| \leq \Pr[E]$ holds where $\Pr[E]$ is the probability that the event happens. We show that $\Pr[E]$ is negligible if Sig provides the S-CEO security as follows. Let $\mathcal{A}$ be an adversary of outsider strong unforgeability and $\mathcal{C}$ be the challenger of the signature scheme. We construct an algorithm $\mathcal{B}_2$ that breaks the S-CEO security if the event happens as follows. $\mathcal{C}$ runs $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$ and sends $(\mathsf{pp}_{\mathsf{Sig}}, \mathsf{vk}^*)$ to $\mathcal{B}_2$. $\mathcal{B}_2$ runs $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and $(\mathsf{PKE.pk}, \mathsf{PKE.dk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, computes $C^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}, 0^{|\mathsf{sigk}|})$, sets $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$, $\mathsf{mpk} = \mathsf{PKE.pk}$, and $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$, and gives $(\mathsf{pp}, \mathsf{mpk}, \mathsf{dpk}^*)$ to $\mathcal{A}$. $\mathcal{B}_2$ initializes $L_{\mathsf{dpk}} = \{\mathsf{dpk}^*\}$. $\mathcal{B}_2$ responds queries issued by $\mathcal{A}$ as follows.

Derived Public Key Check Query: $\mathcal{A}$ sends $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$ to $\mathcal{B}_2$. If $\mathsf{dpk} = \mathsf{dpk}^*$, then $\mathcal{B}_1$ returns 1. Otherwise, if $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, then $\mathcal{B}_2$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$ and sends m to $\mathcal{C}$ as a signing query. $\mathcal{C}$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \Sigma) = 1$ (i.e., the event happens), then $\mathcal{B}_2$ updates $L_{\mathsf{dpk}} \leftarrow L_{\mathsf{dpk}} \cup \{\mathsf{dpk}\}$, outputs (vk, m, Σ) that breaks the S-CEO security. Otherwise, $\mathcal{B}_2$ computes $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$. $\mathcal{B}_2$ returns 0 if $\mathsf{sigk} = \perp$. Otherwise, $\mathcal{B}_2$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 1$, then $\mathcal{B}_2$ updates $L_{\mathsf{dpk}} \leftarrow L_{\mathsf{dpk}} \cup \{\mathsf{dpk}\}$ and returns 1. Otherwise, $\mathcal{B}_2$ returns 0.

Derived Secret Key Corruption Query: $\mathcal{A}$ sends $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$ where where $\mathsf{dpk} \in L_{\mathsf{dpk}} \setminus \{\mathsf{dpk}^*\}$. If $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, then $\mathcal{B}_2$ has broken the S-CEO security. Otherwise, $\mathcal{B}_2$ computes $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$ and returns sigk to $\mathcal{A}$.

Signing Query: $\mathcal{A}$ sends $M \in \mathsf{MS}_{\mathsf{PDPKS}}$ and $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}}) \in L_{\mathsf{dpk}}$ to $\mathcal{B}_2$. If $\mathsf{dpk} = \mathsf{dpk}^*$, then $\mathcal{B}_2$ sends M to $\mathcal{C}$ as a signing query, obtains Σ, and returns Σ. Otherwise, $\mathcal{B}_2$ retrieves $\mathsf{dsk} = \mathsf{sigk}$ by internally issuing a derived secret key corruption query or uses sigk if $\mathcal{A}$ has issued dpk as a derived secret key corruption query. $\mathcal{B}_2$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$.

If the event happens, then $\mathcal{B}_2$ breaks the S-CEO security. Thus, $\Pr[E] \leq \mathsf{Adv}^{\text{S-CEO}}_{\mathcal{B}_2,\mathsf{Sig}}(\lambda)$ holds.

Finally, in Game_2, we construct an algorithm $\mathcal{B}_3$ that breaks strong unforgeability, i.e., $\Pr[E_2] \leq \mathsf{Adv}^{\text{strong}}_{\mathcal{B}_3,\mathsf{Sig}}(\lambda)$, as follows. In this game, $\mathcal{B}_3$ returns 0 for a derived public key check query dpk where $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$ due to the modification of the previous game. Let $\mathcal{A}$ be an adversary of outsider strong unforgeability and $\mathcal{C}$ be the challenger of the signature scheme. $\mathcal{C}$ runs $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$ and sends $(\mathsf{pp}_{\mathsf{Sig}}, \mathsf{vk}^*)$ to $\mathcal{B}_3$. $\mathcal{B}_3$ runs $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and $(\mathsf{PKE.pk}, \mathsf{PKE.dk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, computes $C^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}, 0^{|\mathsf{sigk}|})$, sets $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$, $\mathsf{mpk} = \mathsf{PKE.pk}$, and $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$, and gives $(\mathsf{pp}, \mathsf{mpk}, \mathsf{dpk}^*)$ to $\mathcal{A}$. $\mathcal{B}_3$ initializes $L_{\mathsf{dpk}} = \{\mathsf{dpk}^*\}$. $\mathcal{B}_3$ responds queries issued by $\mathcal{A}$ as follows.

Derived Public Key Check Query: $\mathcal{A}$ sends $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$ to $\mathcal{B}_3$. If $\mathsf{dpk} = \mathsf{dpk}^*$, then $\mathcal{B}_3$ returns 1. Otherwise, if $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, then $\mathcal{B}_3$ returns 0. Otherwise, $\mathcal{B}_3$ computes $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$. $\mathcal{B}_3$ returns 0 if $\mathsf{sigk} = \perp$. Otherwise, $\mathcal{B}_3$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 1$, then $\mathcal{B}_3$ updates $L_{\mathsf{dpk}} \leftarrow L_{\mathsf{dpk}} \cup \{\mathsf{dpk}\}$ and returns 1. Otherwise, $\mathcal{B}_3$ returns 0.

Derived Secret Key Corruption Query: $\mathcal{A}$ sends $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$ where where $\mathsf{dpk} \in L_{\mathsf{dpk}} \setminus \{\mathsf{dpk}^*\}$. If $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, then $\mathcal{B}_3$ returns $\perp$. Otherwise, $\mathcal{B}_3$ computes $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$ and returns sigk.

Signing Query: $\mathcal{A}$ sends $M \in \mathsf{MS}_{\mathsf{PDPKS}}$ and $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}}) \in L_{\mathsf{dpk}}$ to $\mathcal{B}_3$. If $\mathsf{dpk} = \mathsf{dpk}^*$, then $\mathcal{B}_3$ sends M to $\mathcal{C}$ as a signing query. $\mathcal{C}$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}^*, M)$ to $\mathcal{B}_3$. $\mathcal{B}_3$ returns Σ to $\mathcal{A}$. Otherwise, $\mathcal{B}_3$ retrieves $\mathsf{dsk} = \mathsf{sigk}$ by internally issuing a derived secret key corruption query or uses sigk if $\mathcal{A}$ has issued dpk as a derived secret key corruption query. $\mathcal{B}_3$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$.

Finally, $\mathcal{A}$ outputs (M^*, Σ^*). $\mathcal{B}_3$ outputs (M^*, Σ^*) as a forgery and breaks strong unforgeability.

Now, we have

$$\begin{aligned}
\mathsf{Adv}^{\text{outsider-strong-unforge}}_{\mathcal{A},\mathsf{PDPKS}}(\lambda) &= \Pr[E_0] \\
&= \Pr[E_0] - \Pr[E_1] + \Pr[E_1] - \Pr[E_2] + \Pr[E_2] \\
&\leq |\Pr[E_0] - \Pr[E_1]| + |\Pr[E_1] - \Pr[E_2]| + \Pr[E_2] \\
&\leq |\Pr[E_0] - \Pr[E_1]| + \Pr[E] + \Pr[E_2] \\
&\leq \mathsf{Adv}^{\text{IND-CCA}}_{\mathcal{B}_1,\mathsf{PKE}}(\lambda) + \mathsf{Adv}^{\text{S-CEO}}_{\mathcal{B}_2,\mathsf{Sig}}(\lambda) + \mathsf{Adv}^{\text{strong}}_{\mathcal{B}_3,\mathsf{Sig}}(\lambda)
\end{aligned}$$

This concludes the proof. □

Theorem 3. *The proposed construction is unlinkable if the underlying signature scheme is S-CEO secure and the underlying PKE scheme is CCA secure and key private.*

Before giving our security proof, we give an intuition of the proof. Basically, for the challenge derived public key $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$, no information of mpk is revealed if the PKE scheme is key private. However, if an adversary issues a derived public key check query $\mathsf{dpk} = (\mathsf{vk}, C^*_{\mathsf{PKE}})$ and $i \in \{0,1\}$ where $\mathsf{vk} \neq \mathsf{vk}^*$, the reduction algorithm needs to know not only whether vk is a valid verification key relative to sigk^* but also C^*_{PKE} is generated by mpk_i or not. This implies the reduction algorithm breaks key privacy without using the adversary, and the reduction algorithm fails the simulation. Thus, we need to change the game description where the reduction algorithm returns 0 for a derived public key check query $\mathsf{dpk} = (\mathsf{vk}, C^*_{\mathsf{PKE}})$ if $\mathsf{vk} \neq \mathsf{vk}^*$ regardless of i. Here, the reduction algorithm can respond 0 to the query regardless of i if vk is not a valid verification key relative to sigk^*. Here, if vk is a valid verification key relative to sigk^*, then we can construct an algorithm that breaks the S-CEO security, as in the security proof of outsider strong unforgeability. Thanks to the game modifications, the reduction algorithm (for key privacy) responds 0 for a derived public key check query $\mathsf{dpk} = (\mathsf{vk}, C^*_{\mathsf{PKE}})$ regardless of i.

Proof. We use a game sequence Game_0, Game_1, and Game_2. Let E_i be an event that $\mathcal{A}$ wins in Game_i.

Game_0**.** This is the security game of unlinkability. By definition,

$$\mathsf{Adv}^{\text{unlink}}_{\mathcal{A},\mathsf{PDPKS}}(\lambda) = |\Pr[E_0] - 1/2|$$

Game_1**.** This is the same as Game_0 except that C^*_{PKE} is replaced to a ciphertext of $0^{|\mathsf{sigk}^*|}$. We show that there exists an algorithm $\mathcal{B}_1$ such that $|\Pr[E_0] - \Pr[E_1]| \leq \mathsf{Adv}^{\text{IND-CCA}}_{\mathcal{B}_1,\mathsf{PKE}}(\lambda)$ as follows. Let $\mathcal{A}$ be an adversary of unlinkability and $\mathcal{C}$ be the challenger of the PKE scheme. Note that the definition of key privacy contains CCA security but here $\mathcal{C}$ is the IND-CCA challenger. $\mathcal{C}$ generates $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and $(\mathsf{PKE.pk}, \mathsf{PKE.dk}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and gives $(\mathsf{pp}_{\mathsf{PKE}}, \mathsf{PKE.pk})$ to $\mathcal{B}_1$. $\mathcal{B}_1$ runs $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and sets $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$. $\mathcal{B}_1$ chooses $b'' \xleftarrow{\$} \{0,1\}$ and sets $\mathsf{mpk}_{b''} = \mathsf{PKE.pk}$. $\mathcal{B}_1$ runs $(\mathsf{mpk}_{1-b''}, \mathsf{msk}_{1-b''}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$. $\mathcal{B}_1$ runs $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$ and sends $(M_0^*, M_1^*) = (\mathsf{sigk}^*, 0^{|\mathsf{sigk}^*|})$ to $\mathcal{C}$ as the challenge query. $\mathcal{C}$ chooses

$b \xleftarrow{\$} \{0,1\}$, computes the challenge ciphertext $C^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}, M^*_b)$, and returns C^*_{PKE} to $\mathcal{B}_1$. $\mathcal{B}_1$ sets $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$ and gives $(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1, \mathsf{dpk}^*)$ to $\mathcal{A}$. $\mathcal{B}_1$ initializes $L_{\mathsf{dpk},0} := \emptyset$ and $L_{\mathsf{dpk},1} := \emptyset$. $\mathcal{B}_1$ responds queries issued by $\mathcal{A}$ as follows.

Derived Public Key Check Query: $\mathcal{A}$ sends $\mathsf{dpk} \neq \mathsf{dpk}^*$ and index $i \in \{0,1\}$ to $\mathcal{B}$. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. If $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, $\mathcal{B}_1$ returns 0 if $i \neq b''$. Otherwise, if $i = b''$, then $\mathcal{B}_1$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 1$, then $\mathcal{B}$ updates $L_{\mathsf{dpk},i} \leftarrow L_{\mathsf{dpk},i} \cup \{\mathsf{dpk}\}$ and returns 1. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 0$, then $\mathcal{B}_1$ returns 0. If $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$ and $i = b''$, $\mathcal{B}_1$ sends C_{PKE} to $\mathcal{C}$ as a decryption query. $\mathcal{C}$ returns $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}, C_{\mathsf{PKE}})$ to $\mathcal{B}_1$. $\mathcal{B}_1$ returns 0 if $\mathsf{sigk} = \perp$. Otherwise, $\mathcal{B}_1$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}, m)) = 1$, then $\mathcal{B}_1$ updates $L_{\mathsf{dpk},i} \leftarrow L_{\mathsf{dpk},i} \cup \{\mathsf{dpk}\}$ and returns 1. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 0$, then $\mathcal{B}_1$ returns 0. If $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$ and $i \neq b''$, $\mathcal{B}_1$ computes $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{msk}_{1-b''}, C_{\mathsf{PKE}})$. $\mathcal{B}_1$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}, m)) = 1$, then $\mathcal{B}_1$ updates $L_{\mathsf{dpk},i} \leftarrow L_{\mathsf{dpk},i} \cup \{\mathsf{dpk}\}$ and returns 1. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 0$, then $\mathcal{B}_1$ returns 0.

Derived Secret Key Corruption Query: $\mathcal{A}$ sends $\mathsf{dpk} \in L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1}$ to $\mathcal{B}$. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. If $\mathsf{dpk} \neq \mathsf{dpk}^*$, $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, $\mathcal{B}_1$ returns sigk^* to $\mathcal{A}$. If $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$ and $i = b''$, $\mathcal{B}_1$ sends C_{PKE} to $\mathcal{C}$ as a decryption query. $\mathcal{C}$ returns $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}_i, C_{\mathsf{PKE}})$ to $\mathcal{B}_1$. $\mathcal{B}$ returns sigk to $\mathcal{A}$. If $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$ and $i \neq b''$, $\mathcal{B}_1$ computes $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{msk}_{1-b''}, C_{\mathsf{PKE}})$ and returns sigk.

Signing Query: $\mathcal{A}$ sends $M \in \mathsf{MS}_{\mathsf{PDPKS}}$ and $\mathsf{dpk} \in L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1} \cup \{\mathsf{dpk}^*\}$ to $\mathcal{C}$. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. If $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$ and $i = b''$, then $\mathcal{B}_1$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}^*, M)$. Otherwise, $\mathcal{B}_1$ retrieves $\mathsf{dsk} = \mathsf{sigk}$ by internally issuing a derived secret key corruption query or uses sigk if $\mathcal{A}$ has issued dpk as a derived secret key corruption query. $\mathcal{B}_1$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$.

If $b = 0$, then $\mathcal{B}_1$ simulates Game_0 and if $b = 1$, $\mathcal{B}_1$ simulates Game_1. Thus, $|\Pr[E_0] - \Pr[E_1]| \leq \mathsf{Adv}^{\text{IND-CCA}}_{\mathcal{B}_1,\mathsf{PKE}}(\lambda)$ holds.

Game_2**.** This is the same as Game_1 except that the response of derived public key check queries is changed. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$ be a derived public key check query. If $\mathsf{vk} \neq \mathsf{vk}^*$, $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, and $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 1$ holds for a random choice of $m \in \mathsf{MS}_{\mathsf{Sig}}$, then the challenger outputs 0 for a derived public key check query. If this event does not happen, Game_1 and Game_2 are identical. Thus, $|\Pr[E_1] - \Pr[E_2]| \leq \Pr[E]$ holds where $\Pr[E]$ is the probability that the event happens. We show that $\Pr[E]$ is negligible if Sig provides the S-CEO security as follows. Let $\mathcal{A}$ be an adversary of unlinkability and $\mathcal{C}$ be the challenger of the signature scheme. We construct an algorithm $\mathcal{B}_2$ that breaks the S-CEO security if the event happens as follows. $\mathcal{C}$ runs $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$ and sends $(\mathsf{pp}_{\mathsf{Sig}}, \mathsf{vk}^*)$ to $\mathcal{B}_2$. $\mathcal{B}_2$ chooses $b'' \xleftarrow{\$} \{0,1\}$, runs $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$, $(\mathsf{mpk}_{b''}, \mathsf{msk}_{b''}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and $(\mathsf{mpk}_{1-b''}, \mathsf{msk}_{1-b''}) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, computes $C^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{mpk}_{b''}, 0^{|\mathsf{sigk}|})$, sets $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$ and $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$, and gives $(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1, \mathsf{dpk}^*)$ to $\mathcal{A}$. $\mathcal{B}_2$ initializes $L_{\mathsf{dpk},0} := \emptyset$ and $L_{\mathsf{dpk},1} := \emptyset$. $\mathcal{B}_2$ responds queries issued by $\mathcal{A}$ as follows.

Derived Public Key Check Query: $\mathcal{A}$ sends $\mathsf{dpk} \neq \mathsf{dpk}^*$ and index $i \in \{0,1\}$ to $\mathcal{B}_2$. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. If $\mathsf{dpk} \neq \mathsf{dpk}^*$, $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, $\mathcal{B}_2$ returns 0 if $i \neq b''$. Otherwise, if $i = b''$, then $\mathcal{B}_2$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$ and sends m to $\mathcal{C}$ as a signing query, and obtains Σ. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \Sigma) = 1$ (i.e., the event happens), then $\mathcal{B}_2$ updates $L_{\mathsf{dpk},i} \leftarrow L_{\mathsf{dpk},i} \cup \{\mathsf{dpk}\}$, outputs (vk, m, Σ) that breaks the S-CEO security. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \Sigma) = 0$, then $\mathcal{B}_2$ returns 0. If $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$, $\mathcal{B}_2$ computes $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{msk}_i, C_{\mathsf{PKE}})$. $\mathcal{B}_2$ outputs 0 if $\mathsf{sigk} = \bot$. Otherwise, $\mathcal{B}_2$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}, m)) = 1$, then $\mathcal{B}_2$ updates $L_{\mathsf{dpk},i} \leftarrow L_{\mathsf{dpk},i} \cup \{\mathsf{dpk}\}$ and returns 1. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 0$, then $\mathcal{B}_2$ returns 0.

Derived Secret Key Corruption Query: $\mathcal{A}$ sends $\mathsf{dpk} \in L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1}$ to $\mathcal{B}_2$. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. If $\mathsf{dpk} \neq \mathsf{dpk}^*$, $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, and $i = b''$, $\mathcal{B}_2$ has broken the S-CEO security. Now, $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$. $\mathcal{B}_2$ computes $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{msk}_i, C_{\mathsf{PKE}})$ and returns sigk.

Signing Query: $\mathcal{A}$ sends $M \in \mathsf{MS}_{\mathsf{PDPKS}}$ and $\mathsf{dpk} \in L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1} \cup \{\mathsf{dpk}^*\}$ to $\mathcal{B}_2$. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. If $\mathsf{dpk} = \mathsf{dpk}^*$, then $\mathcal{B}_2$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}^*, M)$. Otherwise, $\mathcal{B}_2$ retrieves $\mathsf{dsk} = \mathsf{sigk}$ by internally issuing a derived secret key corruption query or uses sigk if $\mathcal{A}$ has issued dpk as a derived secret key corruption query. $\mathcal{B}_2$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$.

If the event happens, then $\mathcal{B}_2$ breaks the S-CEO security. Thus, $\Pr[E] \leq \mathsf{Adv}^{\text{S-CEO}}_{\mathcal{B}_2,\mathsf{Sig}}(\lambda)$ holds.

Finally, in Game_2, we construct an algorithm $\mathcal{B}_3$ that breaks key privacy, i.e., $|\Pr[E_2] - 1/2| \leq \mathsf{Adv}^{\text{Key-Privacy}}_{\mathcal{B}_3,\mathsf{PKE}}(\lambda)$, as follows. Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger of the PKE scheme. $\mathcal{C}$ generates $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, $(\mathsf{PKE.pk}_0, \mathsf{PKE.dk}_0) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and $(\mathsf{PKE.pk}_1, \mathsf{PKE.dk}_1) \leftarrow \mathsf{PKE.KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$, and gives $(\mathsf{pp}_{\mathsf{PKE}}, \mathsf{PKE.pk}_0, \mathsf{PKE.pk}_1)$ to $\mathcal{B}_3$. $\mathcal{B}_3$ runs $\mathsf{pp}_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp}_{\mathsf{Sig}})$. $\mathcal{B}_3$ sends $0^{|\mathsf{sigk}^*|}$ to $\mathcal{C}$ as the challenge plaintext. $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$, computes the challenge ciphertext $C^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{PKE.pk}_b, 0^{|\mathsf{sigk}^*|})$, and gives C^*_{PKE} to $\mathcal{B}_3$. $\mathcal{B}_3$ sets $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sig}}, \mathsf{pp}_{\mathsf{PKE}})$, $(\mathsf{mpk}_0, \mathsf{mpk}_1) = (\mathsf{PKE.pk}_0, \mathsf{PKE.pk}_1)$, and $\mathsf{dpk}^* = (\mathsf{vk}^*, C^*_{\mathsf{PKE}})$, and sends $(\mathsf{pp}, \mathsf{mpk}_0, \mathsf{mpk}_1, \mathsf{dpk}^*)$ to $\mathcal{A}$. $\mathcal{B}_3$ initializes $L_{\mathsf{dpk},0} := \emptyset$ and $L_{\mathsf{dpk},1} := \emptyset$. $\mathcal{B}_3$ responds queries issued by $\mathcal{A}$ as follows.

Derived Public Key Check Query: $\mathcal{A}$ sends $\mathsf{dpk} \neq \mathsf{dpk}^*$ and index $i \in \{0,1\}$ to $\mathcal{B}_3$. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. If $\mathsf{dpk} \neq \mathsf{dpk}^*$ and $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$, then $\mathcal{B}_3$ returns 0 regardless of i due to the modification of the previous game. If $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$, $\mathcal{B}_3$ sends (C_{PKE}, i) to $\mathcal{C}$ as a decryption query where $\mathsf{dpk} \in L_{\mathsf{dpk},i}$. $\mathcal{C}$ returns $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}_i, C_{\mathsf{PKE}})$ to $\mathcal{B}_3$. $\mathcal{B}_3$ returns 0 if $\mathsf{sigk} = \bot$. Otherwise, $\mathcal{B}_3$ chooses $m \xleftarrow{\$} \mathsf{MS}_{\mathsf{Sig}}$. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}, m)) = 1$, then $\mathcal{B}_3$ updates $L_{\mathsf{dpk},i} \leftarrow L_{\mathsf{dpk},i} \cup \{\mathsf{dpk}\}$ and returns 1. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \mathsf{Sig.Sign}(\mathsf{sigk}^*, m)) = 0$, then $\mathcal{B}_3$ returns 0.

Derived Secret Key Corruption Query: $\mathcal{A}$ sends $\mathsf{dpk} \in L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1}$ to $\mathcal{B}_3$. Let $\mathsf{dpk} = (\mathsf{vk}, C_{\mathsf{PKE}})$. Note that if $C_{\mathsf{PKE}} = C^*_{\mathsf{PKE}}$ (i.e., $\mathsf{vk} \neq vk^*$), then $\mathsf{dpk} \notin L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1}$. Thus, we consider the case that $C_{\mathsf{PKE}} \neq C^*_{\mathsf{PKE}}$. $\mathcal{B}_3$ sends (C_{PKE}, i) to $\mathcal{C}$ as a decryption query where $\mathsf{dpk} \in L_{\mathsf{dpk},i}$. $\mathcal{C}$ returns $\mathsf{sigk} \leftarrow \mathsf{PKE.Dec}(\mathsf{PKE.dk}_i, C_{\mathsf{PKE}})$ to $\mathcal{B}_3$. $\mathcal{B}_3$ returns sigk to $\mathcal{A}$.

Signing Query: $\mathcal{A}$ sends $M \in \mathsf{MS}_{\mathsf{PDPKS}}$ and $\mathsf{dpk} \in L_{\mathsf{dpk},0} \cup L_{\mathsf{dpk},1} \cup \{\mathsf{dpk}^*\}$ to $\mathcal{C}$. If $\mathsf{dpk} = \mathsf{dpk}^*$, then $\mathcal{B}_3$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}^*, M)$. Otherwise, $\mathcal{B}_3$ retrieves $\mathsf{dsk} = \mathsf{sigk}$ by internally issuing a derived secret key corruption query or uses sigk if $\mathcal{A}$ has issued dpk as a derived secret key corruption query. $\mathcal{B}_3$ returns $\Sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sigk}, M)$.

Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$. $\mathcal{B}_3$ outputs b' and breaks key privacy with the advantage at least $\mathsf{Adv}^{\text{unlink}}_{\mathcal{A},\mathsf{PDPKS}}(\lambda)$. Thus, $|\Pr[E_2] - 1/2| \leq \mathsf{Adv}^{\text{Key-Privacy}}_{\mathcal{B}_3,\mathsf{PKE}}(\lambda)$ holds.

Now, we have

$$\begin{aligned}
\mathsf{Adv}^{\text{unlink}}_{\mathcal{A},\mathsf{PDPKS}}(\lambda) &= |\Pr[E_0] - 1/2| \\
&= |\Pr[E_0] - \Pr[E_1] + \Pr[E_1] - \Pr[E_2] + \Pr[E_2] - 1/2| \\
&\leq |\Pr[E_0] - \Pr[E_1]| + |\Pr[E_1] - \Pr[E_2]| + |\Pr[E_2] - 1/2| \\
&\leq |\Pr[E_0] - \Pr[E_1]| + \Pr[E] + |\Pr[E_2] - 1/2| \\
&\leq \mathsf{Adv}^{\text{IND-CCA}}_{\mathcal{B}_1,\mathsf{PKE}}(\lambda) + \mathsf{Adv}^{\text{S-CEO}}_{\mathcal{B}_2,\mathsf{Sig}}(\lambda) + \mathsf{Adv}^{\text{Key-Privacy}}_{\mathcal{B}_3,\mathsf{PKE}}(\lambda)
\end{aligned}$$

This concludes the proof. □

6 One-time PDPKS

Liu et al. [33] mentioned that "*Note that the concept of PDPKS is motivated by the security and privacy problems in cryptocurrency, where it is suggested that each public/verification key, as the coin address, is used only once. But in this paper we do not restrict the concept to one-time signature scheme, which requires that for each public key the signing oracle can be queried at most once. Our proposed PDPKS requires stronger security, namely, even if the users use the freshly derived key pairs multiple times, the system is still safe.*". Thus, in their security model (and our outsider strong unforgeability), an adversary $\mathcal{A}$ is allowed to issue signing queries (dpk^*, M) in multiple times.

Again, each dpk is used only once as a coin-receiving address, and each fresh dpk is a different value (if a different randomness is used for key derivation). From this perspective, we can define a one-time variant of outsider strong unforgeability where $\mathcal{A}$ is allowed to issue a signing query (dpk^*, M) only once. Intuitively, we can employ a one-time signature scheme instead of a signature scheme that seems effective to improve the efficiency of PDPKS schemes instantiated via the proposed generic construction. For example, with the Cremers et al.'s conversion [20] to add the S-CEO security, we may be able to employ the DL based Groth strongly unforgeable one-time signature scheme [28, 44] or the lattice-based Lyubashevsky-Micciancio strongly unforgeable one-time signature scheme [35]. However, in the security proof of outsider strong unforgeability, the reduction algorithm $\mathcal{B}_2$ may issue signing queries to the challenger more than once. Concretely, let $\mathcal{A}$ issue $\mathsf{dpk} = (\mathsf{vk}, C^*_{\mathsf{PKE}})$ where $\mathsf{vk} \neq \mathsf{vk}^*$ as a derived public key check query (resp. a derived secret key key query). Then, $\mathcal{B}_2$ sends m to $\mathcal{C}$ as a signing query and obtains Σ. If $\mathsf{Sig.Verify}(\mathsf{vk}, m, \Sigma) = 1$ holds, then $\mathcal{B}_2$ breaks the S-CEO security. However, if $\mathsf{Sig.Verify}(\mathsf{vk}, m, \Sigma) = 0$, then $\mathcal{B}_2$ returns 0 (resp. $\perp$) to $\mathcal{A}$ and the game goes on. Later, if $\mathcal{A}$ issues a signing query (dpk^*, M), then $\mathcal{B}_2$ needs to send a signing query M to $\mathcal{C}$ but it is prohibited if the signature scheme is restricted as one-time use. The same thing happens in the proof of unlinkability. Thus, though one-time PDPKS seems sufficient for cryptocurrency applications such as stealth address and deterministic wallet, our generic construction is currently not applicable for instantiating a one-time PDPKS scheme. We leave how to construct an efficient one-time PDPKS scheme is left as a future work of this paper.

7 Conclusion

In this paper, we introduced consistency and outsider strong unforgeability, and proposed a generic construction of PDPKS from signatures and PKE. To the best of our knowledge, no isogeny-based CCA secure key private PKE scheme has been proposed so far. Thus, currently our generic construction does not give an isogeny-based PDPKS scheme. Das et al. [22] proposed an isogeny-based deterministic threshold wallet. Though the functionality is incompatible to PDPKS, their construction technique may be applicable to PDPKS. We leave it as a future work of this paper. Zhu et al. [45] investigated universal composability (UC) of PDPKS and showed that the game-based definitions (unforgeability and unlinkability) given by Liu t al. [33] and the UC-security of PDPKS are equivalent. Since we gave a new security definitions of PDPKS, it would be desirable to investigate whether our definition is still equivalent to the UC-security or not. We leave it as a future work of this paper.

Acknowledgment: This work was supported by JSPS KAKENHI Grant Number JP21K11897 and JP23K24844.

References

[1] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In *TCC*, pages 480–497, 2010.

[2] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *CT-RSA*, pages 143–158, 2001.

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

[4] Nabil Alkeilani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. Deterministic wallets in a quantum world. In *ACM CCS*, pages 1017–1031, 2020.

[5] Thomas Aulbach, Samed Düzlü, Michael Meyer, Patrick Struck, and Maximiliane Weishäupl. Hash your keys before signing: BUFF security of the additional NIST PQC signatures. In *Post-Quantum Cryptography*, pages 301–335, 2024.

[6] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In *ASIACRYPT*, pages 635–664, 2021.

[7] 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.

[8] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *SCN*, pages 257–267, 2002.

[9] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, pages 319–331, 2005.

[10] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, pages 566–582, 2001.

[11] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In *Public Key Cryptography*, pages 207–228, 2006.

[12] Dan Boneh, Aditi Partap, and Brent Waters. Accountable multi-signatures with constant size public keys. *IACR Cryptology ePrint Archive*, page 1793, 2023.

[13] Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In *Public Key Cryptography*, pages 229–240, 2006.

[14] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. The provable security of Ed25519: Theory and practice. In *IEEE S&P*, pages 1659–1676, 2021.

[15] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[16] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[17] Jung Hee Cheon. Security analysis of the Strong Diffie-Hellman problem. In *EUROCRYPT*, pages 1–11, 2006.

[18] Nicolas T. Courtois and Rebekah Mercer. Stealth address and key management techniques in blockchain systems. In *ICISSP*, pages 559–566, 2017.

[19] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25, 1998.

[20] 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 S&P*, pages 1696–1714, 2021.

[21] Wei Dai, Tatsuaki Okamoto, and Go Yamamoto. Stronger security and generic constructions for adaptor signatures. In *INDOCRYPT*, pages 52–77, 2022.

[22] Poulami Das, Andreas Erwig, Michael Meyer, and Patrick Struck. Efficient post-quantum secure deterministic threshold wallets from isogenies. In *ACM AsiaCCS*, 2024.

[23] Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. In *ACM CCS*, pages 651–668, 2019.

[24] Loïc Ferreira and Johan Pascal. Post-quantum secure ZRTP. In *Post-Quantum Cryptography*, pages 3–36, 2024.

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

[26] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, pages 537–554, 1999.

[27] Paul Gerhart, Dominique Schröder, Pratik Soni, and Sri Aravinda Krishnan Thyagarajan. Foundations of adaptor signatures. In *EUROCRYPT*, pages 161–189, 2024.

[28] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT*, pages 444–459, 2006.

[29] Felix Günther and Marc Ilunga Tshibumbu Mukendi. Careful with MAc-then-SIGn: A computational analysis of the EDHOC lightweight authenticated key exchange protocol. In *IEEE EuroS&P*, pages 773–796, 2023.

[30] Jingwei Jiang and Ding Wang. QPASE: quantum-resistant password-authenticated searchable encryption for cloud storage. *IEEE Transactions on Information Forensics and Security*, 19:4231–4246, 2024.

[31] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, pages 372–389, 2008.

[32] 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*, pages 357–377, 2020.

[33] 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.

[34] 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.

[35] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In *TCC*, pages 37–54, 2008.

[36] Alberto Maria Mongardini, Daniele Friolo, and Giuseppe Ateniese. Stealth and beyond: Attribute-driven accountability in bitcoin transactions. *IACR Cryptology ePrint Archive*, page 1789, 2024.

[37] Andrew Poelstra. Scriptless scripts. `https://tinyurl.com/ludcxyz`, 2017.

[38] 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.

[39] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *ACM STOC*, pages 84–93, 2005.

[40] Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In *CT-RSA*, pages 357–371, 2007.

[41] Isamu Teranishi, Takuro Oyama, and Wakaha Ogata. General conversion for obtaining strongly existentially unforgeable signatures. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 91-A(1):94–106, 2008.

[42] Anton Wahrstätter, Matthew Solomon, Ben DiFrancesco, Vitalik Buterin, and Davor Svetinovic. BaseSAP: Modular stealth address protocol for programmable blockchains. *IEEE Transactions on Information Forensics and Security*, 19:3539–3553, 2024.

[43] Ruida Wang, Ziyi Li, Xianhui Lu, Zhenfei Zhang, and Kunpeng Wang. Key derivable signature and its application in blockchain stealth address. *Cybersecurity*, 7(1):43, 2024.

[44] Hoeteck Wee. Public key encryption against related key attacks. In *Public Key Cryptography*, pages 262–279, 2012.

[45] Chunping Zhu, Xingkai Wang, and Zhen Liu. Universally composable key-insulated and privacy-preserving signature scheme with publicly derived public key. In *Inscrypt*, pages 44–64, 2023.

Appendix

In this Appendix, we introduce the construction methodology of Liu et al. [33] which may help the reader to understand their scheme. They pointed out that IBS is a promising tool to construct PDPKS but required a special property, that they called MPK-pack-able property. Briefly, it requires that there exists a function F and a verification algorithm Verify_F, where, for an identity ID and a message-signature pair (M, σ),

$$\mathsf{Verify}_F(F(\mathsf{IBS.mpk}, ID), M, \sigma) = \mathsf{IBS.Verify}(\mathsf{IBS.mpk}, ID, M, \sigma)$$

holds and no information of $\mathsf{IBS.mpk}$ is leaked from $F(\mathsf{IBS.mpk}, ID)$. The main reason why Liu et al. introduced the MPK-pack-able property is the verification algorithm of IBS needs to take the corresponding master public key that violates the unlinkability. Intuitively, for $(\mathsf{IBS.mpk}_B, \mathsf{IBS.msk}_B) \leftarrow \mathsf{IBS.MasterKeyGen}(\mathsf{pp}_{\mathsf{IBS}})$, where $\mathsf{pp}_{\mathsf{IBS}}$ is a common parameter, a payee Bob sets $\mathsf{mpk}_B = \mathsf{IBS.mpk}_B$ and $\mathsf{msk}_B = \mathsf{IBS.msk}_B$. A payer Alice computes $F(\mathsf{mpk}, ID)$ for some ID, and sets $\mathsf{dpk}_A = F(\mathsf{mpk}_B, ID)$. Bob generates $\mathsf{IBS.sk} \leftarrow \mathsf{IBS.KeyDer}(\mathsf{mpk}_B, \mathsf{msk}_B, ID)$ and sets $\mathsf{dsk}_B = \mathsf{IBS.sk}$. Then, due to the MPK-pack-able property, an IBS signature σ generated by dsk_B can be verified by $\mathsf{dpk}_A = F(\mathsf{mpk}_B, ID)$ and information of mpk_B is not leaked from dpk_A.

The last piece is how Bob to know ID and how Bob to check the validity of dpk_A. Liu et al. implicitly employed non-interactive key exchange (NIKE) [25].[13] By using a NIKE protocol, the construction idea of Liu et al. is explained as follows. A payee Bob sets $\mathsf{mpk}_B = (\mathsf{IBS.mpk}_B, \mathsf{NIKE.pk}_B)$ and $\mathsf{msk}_B = (\mathsf{IBS.msk}_B, \mathsf{NIKE.sk}_B)$. A payer Alice generates a fresh key pair $(\mathsf{NIKE.pk}_A, \mathsf{NIKE.sk}_A)$ and computes a shared key shk_{AB} by $(\mathsf{NIKE.sk}_A, \mathsf{NIKE.pk}_B)$. Let $F = (\mathsf{Commit}, \mathsf{ComOpen})$ be a commitment scheme. Alice sets

$$\mathsf{dpk}_A = (c, \mathsf{NIKE.pk}_A)$$

where $c = \mathsf{Commit}(\mathsf{mpk}_B, \mathsf{shk}_{AB})$, i.e., the shared key shk_{AB} is set as the decommit (and ID in the above explanation). Due to the hiding property of the commitment scheme, no information of mpk_B is leaked from c. Moreover, $\mathsf{NIKE.pk}_A$ is independent to mpk_B. Due to the binding property of the commitment scheme, mpk_B is linked to c. Bob also generates shk_{AB} by $(\mathsf{NIKE.pk}_A, \mathsf{NIKE.sk}_B)$ and checks whether c is a commitment of mpk_B by $\mathsf{ComOpen}(\mathsf{mpk}_B, \mathsf{shk}_{AB}, c)$. To support the condition $\mathsf{Verify}_F(\mathsf{Commit}(\mathsf{mpk}_B, \mathsf{shk}_{AB}), M, \sigma) = \mathsf{IBS.Verify}(\mathsf{IBS.mpk}_B, \mathsf{shk}_{AB}, M, \sigma)$, the MPK-packable property has the central roles of this construction methodology.

[13] Let two users, Alice and Bob, would like to share a key. Then, Alice and Bob generate $(\mathsf{NIKE.pk}_A, \mathsf{NIKE.sk}_A)$ and $(\mathsf{NIKE.pk}_B, \mathsf{NIKE.sk}_B)$, respectively. Then, a shared key shk_{AB} can be generated by either $(\mathsf{NIKE.pk}_A, \mathsf{NIKE.sk}_B)$ or $(\mathsf{NIKE.sk}_A, \mathsf{NIKE.pk}_B)$ and is indistinguishable from random. In the actual syntax of NIKE, identities (Alice and Bob here) are also included to generate shk_{AB}. We omit them here.