Cybersecurity

RESEARCH Open Access

Key derivable signature and its application in blockchain stealth address

Ruida Wang[1,2], Ziyi Li[1,2], Xianhui Lu[1,2*], Zhenfei Zhang[3] and Kunpeng Wang[1,2]

Abstract
Stealth address protocol (SAP) is widely used in blockchain to achieve anonymity. In this paper, we formalize a key derivable signature scheme (KDS) to capture the functionality and security requirements of SAP. We then propose a framework to construct key separation KDS, which follows the *key separation* principle as all existing SAP solutions to avoid the reuse of the master keys in the derivation and signature component. We also study the joint security in KDS and construct a key reusing KDS framework, which implies the first compact stealth address protocol using a single key pair. Finally, we provide instantiations based on the elliptic curve (widely used in cryptocurrencies) and on the lattice (with quantum resistance), respectively.

Keywords Key derivable signature, Stealth address protocol, Blockchain, Compact stealth address protocol, Joint security

Introduction

Modern blockchain is a distributed ledger that allows transactions among different users. Typically, the ledger is in the form of a key-value storage, where the key is the user's *address*; and the value is the balance of this address. For engineering simplicity, the address is usually a public, efficient encoding of a group element over a certain elliptic curve. This group element is also used as a public key. To spend from this address, one simply needs to provide a digital signature and post the transaction to the blockchain. This method has been used by main cryptocurrency platforms (Nakamoto 2008; Wood 2014; Sasson et al. 2014; Wood 2016; Gilad et al. 2017).

A main drawback of the above design is the lack of privacy, i.e., everyone can figure out the transactions. Stealth address protocol (SAP), adopted by Monero, Bytecoin and Samourai Wallet (Saberhagen 2012; Noether and Mackenzie 2016), is a common cryptographic approach to hide the identity of the receiver. To date, those platforms have a combined market cap of 3.1 billion US dollars. In a nutshell, a stealth address can be seen as a derived public key from the user's master public key. In a Monero-like protocol, for an address $b\mathcal{G}$, where b is a field element and $\mathcal{G}$ is the group generator, its stealth address is in the form of $(r+b)\mathcal{G}$, where r is randomly chosen by the sender and shared with the receiver via key exchange scheme. Early designs adopted this idea and modified the protocol when suffering from security risks (Saberhagen 2012, 2013; Noether and Mackenzie 2016; Courtois and Mercer 2017), which lacks a formalized algorithm definition, and a security model including security/privacy requirements and capturing adversary capabilities.

In 2019, Liu et al. made the first attempt to define a crypiographic tool to formalize SAP named publicly derived public key scheme (PDPKS), and proposed an instantiation based on pairing friendly curve (Liu et al. 2019b). However, the definition in PDPKS does not allow for signing with user's master keys. This may create a discrepancy with the widespread application of SAP. When deploying SAP in a widely-used blockchain, there are

*Correspondence:
Xianhui Lu
luxianhui@iie.ac.cn
[1] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] Ethereum Foundation, Boston, USA

already numerous transactions in the system that have been signed by users' master keys. These transactions are publicly accessible and permanently recorded on the blockchain. Therefore, it is important for the SAP to have a well-defined approach to handling master key signatures and related security measures to ensure the continued safety and privacy of the system. In addition, the security model of PDPKS allows the adversary to corrupt the derived secret keys, which imposes a strong security requirement that is rarely targeted in subsequent stealth address protocols (Feng et al. 2020; Yu 2020). There is still a lack of cryptographic formalization that should match the SAP usage in blockchain, and a security model capturing the basic security requirements of Monero-like SAPs and the stronger security proposed in PDPKS, respectively.

On the other side, current SAPs and PDPKS are all dual-key schemes, that is, a scan key pair to share and check the stealth address, and a spend key pair to sign and verify transactions. This design is for security concern, especially using the principle of *key separation* (i.e., independence of keys, which is essential in cryptography) to avoid the reuse of the master key pair. However, key reusing is also possible with well-defined security. It implies a compact SAP using single key pair for simpler systems and broader application scenarios. We can deploy such a compact SAP into popular single-key cryptocurrency systems, especially Bitcoin. There is still a lack of such design after 10 years of SAP deployment.

In addition, current constructions of SAP and PDPKS are built on concrete assumptions. For example, Monero-like protocols use the discrete logarithm problem on the elliptic curve and the latest PDPKS construction (Liu et al. 2020) exploits the trapdoor function on the lattice. It is desirable to design a framework based on crypto primitives that captures multiple instantiations according to the application. Thus, the motivation of this paper is to revise a better formal definition to abstract SAP, propose framework constructions, and support key reusing for compact schemes.

Our results

This paper analyzes the main existing stealth address protocols and the concept of publicly derived public key scheme. We improve this cryptographic tool into the key derivable signature scheme (KDS), and propose general frameworks and instantiations. Specifically, our contributions can be summarized as follow:

KDS Scheme and Security Model.
We first introduce and formalize a KDS scheme and its security model to capture the functionality and security requirements of SAP. Compared to the similar concept PDPKS, KDS allows signing with master keys, which can better match SAP application scenarios on the blockchain; secondly, we revise the existential unforgeable (EUF) and public key unlinkable (UNL) security proposed in PDPKS accordingly, to allow the adversary to access the signing oracle via the master key and win the game by forging a master key signature. We enforce the basic security EUF/UNL of KDS to its strong version strong-EUF/strong-UNL (SUNF/SUNL), by considering the derived secret key leakage.

KDS Frameworks and Instantiations.
We give two KDS frameworks. To achieves this, two barriers should be addressed: the first one is the lack of a cryptographic primitive to capture the key derivation; the second one is the *key reuse* problem in KDS. To address the first issue, we define a key derivation scheme (KDV), as well as its *re-linkability* and *leakage-resistance*. Re-linkability of KDV requires that the derived public key does not leak information about master public key, whereas leakage-resistance is a stronger security and requires that the derived secret key does not leak information about the master secret key. We then overcome the second problem with two design approaches:

- *Key Separation KDS (KS-KDS) Framework* The first approach is a KS-KDS following *key separation* principle. Our framework is based on a KDV, a public key encryption (PKE), and a canonical identification protocol (CID). Specifically, we use the KDV to extract the derived key pair from user's master key pair, use Fujisaki-Okamoto transformation on PKE to get a KEM to share and check the stealth addresses, and use Fiat-Shamir transformation on CID to construct a signature scheme to sign and verify transactions. An EUF and UNL secure KS-KDS requires that the KDV is re-linkable, the PKE is indistinguishable under chosen-plaintext attack (IND-CPA) and indistinguishable of keys under chosen-plaintext attack (IK-CPA); and the CID should be secure against impersonation under key-only attack (IMP-KOA) and honest-verifier zero-knowledge (HVZK). Further, if the KDV also satisfies leakage-resistance, then we can obtain a strong KS-KDS satisfying SEUF and SUNL security. We give a KS-KDS instantiation on the elliptic curve, and a strong KS-KDS instantiation on the lattice.
- *Key Reusing KDS (KR-KDS) Framework* To enable a KR-KDS, where a single master key pair is reused for both key derivation and signature, we resort to the techniques of joint security (Haber and Pinkas 2001). We construct a jointly secure and key private combined signature and key encapsulation mechanism

Table 1 The comparison between our work and other SAP solutions

SAP	Sign_{msk}	EUF/UNL	SEUF/SUNL	Framework	Compact
Monero (Saberhagen 2013)	✓	✓	×	×	×
PDPKS 19 (Liu et al. 2019b)	×	✓	✓	×	×
PDPKS 20 (Liu et al. 2020)	×	✓	✓	×	×
Our KS-KDS	✓	✓	✓	✓	×
Our KR-KDS	✓	✓	✓	✓	✓

Here, Sign_{msk} means the construction contains a master key signature algorithm. Framework means the construction is based on crypto primitives rather than concrete assumptions. Compact means the construction is key reusing rather than key separation

(CSK) using CID, PKE and random oracles (RO). This construction follows an observation that the RO plays a similar role to key separation in our KS-KDS framework. We then construct a (strong) KR-KDS by combining such a CSK scheme with a re-linkable (and leakage-resistant) KDV scheme. To illustrate the effectiveness of our result, we give the first compact stealth address protocol instantiation which can be deployed in widely used cryptocurrencies such as Bitcoin to provide privacy protection.

The comparison between our work and other SAP solutions is shown in Table 1.

Related work

There has been a lot of recent work based on Monero-like stealth address. For example, Yu (2020) proposed a solution allowing for multiple addresses within one wallet, Liu et al. (2019a) proposed a lattice-based linkable ring signature scheme with stealth address, and Wang et al. (2024) proposed a universally composable (UC) linkable ring signature supporting stealth address. Motivated by these works, we present a formalized KDS definition and a security model to capture the essence of stealth address.

Another line of research on stealth address is based on publicly derived public key schemes (PDPKS), which is designed to maintain security even when derived keys have been corrupted. Liu's team first proposed this definition (Liu et al. 2019b), and later extended it to lattice-based (Liu et al. 2020) and UC (Zhu et al. 2023) settings. However, they do not consider the key-reusing property, which is essential to achieve a compact stealth address protocol that can be deployed into most blockchains, not just specially designed systems such as Monero. We revised their model and discuss the joint security in blockchain stealth address.

Joint security ensures the security of the cryptographic tools in the case of key reusing, and can significantly simplify the system and reduce storage in engineering implementations. Haber and Pinkas (2001) introduced the concept of the jointly secure combined public key scheme in 2001, which combines a signature scheme and an encryption scheme using the same key pair. Prior works mainly focus on combined signature and encryption schemes, such as ElGamal-based signature and encryption (Vasco et al. 2008), Schnorr and ECIES (Degabriele et al. 2012), and blind Schnorr signature and Schnorr-signed ElGamal encryption (Fuchsbauer et al. 2020). Adding a derivation algorithm based on such a combined scheme may lead to novel application scenarios, such as our key derivable signature. Chen et al. (2021) proposed a hierarchical integrated signature and encryption (HISE) scheme, in which the encryption component used a derived key pair from the signature key pair.

Paper organization

This paper is organized as follows: "Preliminaries" section reviews the crypto primitives; "Key derivable signature scheme" section presents our key derivable signature scheme (KDS) definition and security model, and compares it with prior works; "The construction of (strong) key separation key derivable signature scheme" section describes our key derivation scheme (KDV) definition and the framework to construct (strong) KS-KDS; "The construction of (strong) key reusing key derivation scheme" section improves this framework to construct KR-KDS; "Conclusion" section concludes the paper.

Preliminaries

Canonical identification protocol (CID)

Definition 1 (*Canonical Identification Protocol*) A canonical identification (CID) protocol with commitment space *COM*, challenge space *CH*, and response space *RSP* consists of a triple of PPT algorithms $(\mathsf{CID.Gen}, \mathsf{CID.P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{CID.V} = (\mathsf{V}_1, \mathsf{V}_2))$ as follows:

```
$\mathsf{G}^{\mathsf{IMP\text{-}KOA}}_{\mathsf{CID},\mathcal{A}}(pp)$:
1: $(pk, sk) \leftarrow \mathsf{CID.Gen}(\mathsf{pp})$
2: $ch \leftarrow \mathsf{V}_1(pk)$
3: $(com, \mathsf{st}) \leftarrow \mathcal{A}(pk)$
1: $rsp \leftarrow \mathcal{A}(pk, com, ch, \mathsf{st})$
2: $\mathbf{return}\ \|\mathsf{V}_2(pk, com, ch, rsp)\|$
```

Fig. 1 IMP-KOA game for $\mathcal{CID}$

- $\mathsf{CID.Gen}(pp) \rightarrow (pk, sk)$: The key generation takes the public parameters *pp* as input and outputs a public key *pk* and a secret key *sk*.
- $\mathsf{CID.P} = (\mathsf{P}_1, \mathsf{P}_2)$: The prover is a two-stage algorithm that takes a secret key *sk*. P_1 takes the secret key *sk* as input and outputs a commitment $com \in COM$ and a state st; P_2 takes the secret key *sk*, a commitment $com \in COM$, a challenge $ch \in CH$ and a state st as input and outputs a response $rsp \in RSP$.
- $\mathsf{CID.V} = (\mathsf{V}_1, \mathsf{V}_2)$: The verifier is a two-stage algorithm that takes a public key *pk* as input. V_1 takes the public key *pk* as input, chooses a random challenge $ch \overset{\$}{\leftarrow} CH$ and sends it to the prover. V_2 takes the public key *pk* and the conversation transcript (*com*, *ch*, *rsp*) as input and outputs a deterministic decision, 1 (acceptance) or 0 (rejection).

$$\mathsf{Adv}^{\mathsf{IMP-KOA}}_{\mathsf{CID},\mathcal{A}}(pp) = \Pr\left[\mathsf{G}^{\mathsf{IMP-KOA}}_{\mathsf{CID},\mathcal{A}}(pp) \rightarrow 1\right],$$

where $\mathsf{G}^{\mathsf{IMP-KOA}}_{\mathsf{CID},\mathcal{A}}(pp)$ is describe as in Fig. 1.

Honest-verifier zero-knowledge (HVZK) is a formalization of the property where adversaries do not gain any additional knowledge from honest interactions.

Definition 3 (*HVZK*) A CID protocol is HVZK if there exists a PPT algorithm Sim and a negligible function $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp) \le negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage $\mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp)$ is defined as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp) =& |\Pr[\mathcal{A}(pk, com, ch, rsp) \rightarrow 1 | (com, ch, rsp) \leftarrow \mathsf{Sim}(pk), (pk, sk)\\ &\leftarrow \mathsf{CID.Gen}(pp)] - \Pr[\mathcal{A}(pk, (com, ch, rsp)) \rightarrow 1 | (com, ch, rsp)\\ &\leftarrow (\mathsf{P}(sk), \mathsf{V}(pk)), (pk, sk) \leftarrow \mathsf{CID.Gen}(pp)]|.\end{aligned}$$

We denote the conversation transcript (*com*, *ch*, *rsp*) as $\mathsf{P}(sk) \leftrightarrow \mathsf{V}(pk)$. Here, (*pk*, *sk*) represents the (public key, secret key) pair generated by $\mathsf{CID.Gen}(pp)$, and $(com, ch, rsp) := \mathsf{P}(sk) \leftrightarrow \mathsf{V}(pk)$ denotes the information exchanged between $\mathsf{CID.P}$ and $\mathsf{CID.V}$ throughout the CID protocol interaction under (*pk*, *sk*).
Security against impersonation under key-only attacks (IMP-KOA) serves as a basic security for CID. It guarantees that adversaries cannot impersonate the prover to deceive an honest verifier without having access to the secret key.

Definition 2 (*IMP-KOA Security*) A CID protocol $\mathsf{CID} = (\mathsf{CID.Gen}, \mathsf{CID.P}, \mathsf{CID.V})$ is IMP-KOA secure, if there exists a negligible function $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{IMP-KOA}}_{\mathsf{CID},\mathcal{A}}(pp) \le negl(\lambda)$[1] for any PPT adversary $\mathcal{A}$. The advantage is defined as follows:

[1] The λ, which appears here and later in this paper, is a security parameter. It is determined by *pp*.

Key privacy of public key encryption

The key privacy of a public key encryption scheme is effectively captured by the property known as "indistinguishability of keys under chosen-plaintext attack" (IK-CPA). It ensures that adversaries, under a chosen-plaintext attack, are unable to distinguish ciphertexts generated by different encryption keys.

Definition 4 (*IK-CPA Security*) A PKE scheme $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ is IK-CPA secure if there exists a negligible function $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{IK-CPA}}_{\mathsf{PKE},\mathcal{A}}(pp) \le negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage $\mathsf{Adv}^{\mathsf{IK-CPA}}_{\mathsf{PKE},\mathcal{A}}(pp)$ is defined as follows:

$$\mathsf{Adv}^{\mathsf{IK-CPA}}_{\mathsf{PKE},\mathcal{A}}(pp) = |\Pr[\mathsf{G}^{\mathsf{IK-CPA}}_{\mathsf{PKE},\mathcal{A}}(pp) \rightarrow 1] - 1/2|,$$

where the IK-CPA game $\mathsf{G}^{\mathsf{IK-CPA}}_{\mathsf{PKE},\mathcal{A}}(pp)$ is described as Fig. 2.

```
$\underline{\mathsf{G}^{\mathsf{IK\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(pp)}$:
1: $(pk_0, sk_0) \leftarrow \mathsf{PKE.Gen}(\mathsf{pp})$
2: $(pk_1, sk_1) \leftarrow \mathsf{PKE.Gen}(\mathsf{pp})$
3: $m^* \leftarrow \mathcal{A}(pk_0, pk_1)$
4: $b \xleftarrow{\$} \{0,1\}$
5: $c^* \leftarrow \mathsf{PKE.Enc}(pk_b, m^*)$
6: $b' \leftarrow \mathcal{A}(pk_0, pk_1, c^*)$
7: return $\|b' = b\|$
```

Fig. 2 IK-CPA game for PKE

Key derivable signature scheme

Algorithm definition

In this section, we introduce the key derivable signature (KDS). This scheme enables the generation of derived key pairs from the master key pair. Our formal definition of KDS is as follows:

Definition 5 (*Key Derivable Signature*) A KDS scheme with message space M consists of following algorithms:

- $\mathsf{KeyGen}(\mathsf{pp}) \to (mpk, msk)$. The key generation algorithm takes the public parameters pp as input and outputs a master public key/secret key pair (mpk, msk).
- $\mathsf{DpkDerive}(mpk) \to dpk$. The public key derivation algorithm takes a master public key mpk and outputs a derived public key dpk.
- $\mathsf{DpkCheck}(mpk, msk, dpk) \to 0/1$. The derived public key checking algorithm takes a master key pair (mpk, msk) and a derived public key dpk as input, and outputs a bit b, where $b = 1$ means that dpk is a derived key generated from mpk and $b = 0$ means not.
- $\mathsf{DskDerive}(mpk, msk, dpk) \to dsk$. The secret key derivation algorithm takes a master key pair (mpk, msk) and a derived public key dpk as input, and outputs a derived secret key dsk.
- $\mathsf{Sign}(sk, m) \to \sigma$. The signing algorithm takes a secret key sk (a master secret key msk or a derived secret key dsk) and a message $m \in M$ as input, and outputs a signature σ.
- $\mathsf{Verify}(pk, m, \sigma) \to 0/1$. The verification algorithm takes as a public key pk (a master public key mpk or a derived public key dpk), a message m and a signature σ as input, and outputs a bit b, where $b = 1$ means that the signature is valid and $b = 0$ means not.

Additionally, the above algorithms must satisfy the following correctness properties:

- For any $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$, there exists a negligible function $negl(\cdot)$ such that
$$\Pr_{dpk \leftarrow \mathsf{DpkDerive}(mpk)}[\mathsf{DpkCheck}(mpk, msk, dpk) \neq 1] \leq neg(\lambda).$$
- For any $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$ and $m \in M$, there exists a negligible function $negl(\cdot)$ such that
$$\Pr_{\sigma \leftarrow \mathsf{Sign}(msk, m)}[\mathsf{Verify}(mpk, m, \sigma) \neq 1] \leq neg(\lambda).$$
- For $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$ and $m \in M$,
$$\Pr_{dpk \leftarrow \mathsf{DpkDerive}(mpk)}[\mathsf{Verify}(dpk, m, \sigma) \neq 1 | dsk \leftarrow \mathsf{DskDerive}(mpk, msk, mpk), \sigma \leftarrow \mathsf{Sign}(dsk, m)] \leq neg(\lambda).$$

KDS is a revised definition of PDPKS, the security model of PDPKS is detailed in A. The major difference is that we consider a master key/derived key signature algorithm, while PDPKS only allows signing with derived keys. This change is motivated by practical considerations. In most existing blockchain systems, except for those specifically designed for stealth address schemes (such as Monero), there is no definition or transaction procedure related to derived keys. Transactions are signed using master keys, recorded on-chain, and publicly accessible. When deploying stealth address on such blockchains, it is necessary to have a well-defined approach to handling master key signatures and related security measures to ensure the safety and privacy of the system. According to our KDS definition, the stealth address protocol can be deployed as a plug-in. It effectively protects the identity privacy while allowing users to still utilize master key signatures without affecting

```
$\underline{\mathsf{G}^{\mathrm{S}\,\mathsf{EUF}}_{\mathsf{KDS},\mathcal{A}}(pp)}$:
1: $L_{Sig} \leftarrow \emptyset$, $L_{Dpk} \leftarrow \emptyset$, $L_{Dsk} = \emptyset$
2: $(mpk, msk) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
3: $(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{O_{Sig}(\cdot), O_{Dpk}(\cdot)}(mpk)$
4: $(pk^*, m^*, \sigma^*) \leftarrow$
     $\mathcal{A}^{O_{Sig}(\cdot), O_{Dpk}(\cdot), O_{Dsk}(\cdot)}(mpk)$
5: $b_0 := \|pk^* \notin L_{Dsk}\|$
6: $b_1 := \mathsf{Verify}(pk^*, m^*, \sigma^*)$
7: $b_2 := \|(pk^*, m^*) \notin L_{Sig}\|$
8: $b_3 := \|pk^* \in L_{Dpk} \cup \{mpk\}\|$
9: return $\|b_1 \wedge b_2 \wedge b_3\|$
10: return $\|b_0 \wedge b_1 \wedge b_2 \wedge b_3\|$

$\underline{O_{Dpk}(dpk)}$:
1: $b \leftarrow \mathsf{DpkCheck}(mpk, msk, dpk)$
2: if $b = 1$
3:     $L_{Dpk} \leftarrow L_{Dpk} \cup \{dpk\}$
4: return $b$

$\underline{O_{Sig}(pk, m)}$:
1: if $pk = mpk$
2:     $\sigma \leftarrow \mathsf{Sign}(msk, m)$
3: if $pk \in L_{Dpk}$
4:     $sk \leftarrow \mathsf{DskDerive}(mpk, msk, pk)$
5:     $\sigma \leftarrow \mathsf{Sign}(sk, m)$
6: if $pk \notin L_{Dpk}$
7:     return $\perp$
8: $L_{Sig} \leftarrow L_{Sig} \cup \{(pk, m)\}$
9: return $\sigma$

$\underline{O_{Dsk}(dpk)}$:
1: if $dpk \in L_{Dpk}$
2:     $dsk \leftarrow$
       $\mathsf{DskDerive}(mpk, msk, dpk)$
3: if $dpk \notin L_{Dpk}$
4:     return $\perp$
5: $L_{Dsk} = L_{Dsk} \cup \{dsk\}$
6: return $dsk$
```

Fig. 3 EUF security and **SEUF security** of KDS with differences highlight in bold

their token balance in permanent addresses. The stealth address works as follows:

For a sender Alice (A), who wants to send a transaction to a receiver Bob (B), A first runs $\mathsf{DpkDerive}(mpk_B) \rightarrow dpk_B$ to derive a stealth address for B, where mpk_B is B's master public key. Then A runs $\mathsf{Sign}(sk_A, m_A) \rightarrow \sigma_A$ to send a transaction, where sk_A can be either A's master secret key or derived secret key, and m_A records the information such as the amount and receiver address dpk_B. To receive the transaction, B actively monitors the blocks and runs $\mathsf{DpkCheck}(mpk_B, msk_B, dpk') \rightarrow 0/1$ to check all potential dpk', until finding his derived address dpk_B and the corresponding transaction. Then he validates the transaction by using $\mathsf{Verify}(pk_A, m'_A, \sigma'_A) \rightarrow 0/1$. To spend the coin in dpk_B, B runs $\mathsf{DskDerive}(mpk_B, msk_B, dpk_B) \rightarrow dsk_B$ to derive the secret key dsk_B. Then he can use $\mathsf{Sign}(dsk_B, m_B) \rightarrow \sigma_B$ to sign a transaction spending the coin. Anyone else can validate the transaction by using $\mathsf{Verify}(dpk_B, m'_B, \sigma'_B) \rightarrow 0/1$.

Security model

In this section, we provide a formal security model to capture the security and privacy requirements of KDS. We need to consider two cases motivated by the security risks of cryptocurrencies:

1. The adversary should not be able to forge the receiver's master/derived key signatures to steal their balance.
2. The adversary should not be able to trace the receiver's permanent address from the derived address.

Security We provide basic security properties: existential unforgeability (EUF) and public key unlinkability (UNL), as well as the stronger security properties: strong existential unforgeability (SEUF) and strong public key unlinkability (SUNL) of KDS, by distinguishing the adversary's capabilities.

The EUF security property states that an adversary, with access to the signature algorithm and derived public key checking algorithm, cannot produce a valid forgery. A more detailed definition is as follows:

Definition 6 (*EUF Security*) A KDS scheme $\mathsf{KDS} = (\mathsf{KeyGen}, \mathsf{DpkDerive}, \mathsf{DpkCheck}, \mathsf{DskDerive}, \mathsf{Sign}, \mathsf{Verify})$ is EUF secure, if there exists a negligible function $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{KDS},\mathcal{A}}(pp) \le negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage is defined as follows:

$$\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{KDS},\mathcal{A}}(pp) = \Pr\left[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KDS},\mathcal{A}}(pp) \to 1\right],$$

where the EUF game $\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KDS},\mathcal{A}}(pp)$ is described as in Fig. 3.

The UNL security states that an adversary, even with access to the signature algorithm and derived public key checking algorithm, cannot distinguish which of two master public keys was used to derive a specific derived public key. A more detailed definition is as follows:

Definition 7 (*UNL Security*) A KDS scheme $\mathsf{KDS} = (\mathsf{KeyGen}, \mathsf{DpkDerive}, \mathsf{DpkCheck}, \mathsf{DskDerive}, \mathsf{Sign}, \mathsf{Verify})$ is UNL secure, if there exists a negligible function $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{UNL}}_{\mathsf{KDS},\mathcal{A}}(pp) \le negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage $\mathsf{Adv}^{\mathsf{UNL}}_{\mathsf{KDS},\mathcal{A}}(pp)$ is defined as follows:

$$\mathsf{Adv}^{\mathsf{UNL}}_{\mathsf{KDS},\mathcal{A}}(pp) = |\Pr\left[\mathsf{G}^{\mathsf{UNL}}_{\mathsf{KDS},\mathcal{A}}(pp) \to 1\right] - 1/2|,$$

```
$\mathsf{G}^{\mathsf{S}\ \mathsf{UNL}}_{\mathsf{KDS},\mathcal{A}}(pp)$:
1: $L_{Dpk,0} \leftarrow \emptyset$, $L_{Dpk,1} \leftarrow \emptyset$, $L_{Dsk} = \emptyset$
2: $(mpk_0, msk_0) \leftarrow \mathsf{KeyGen}(pp)$
3: $(mpk_1, msk_1) \leftarrow \mathsf{KeyGen}(pp)$
4: $b \xleftarrow{\$} \{0,1\}$
5: $dpk^* \leftarrow \mathsf{DpkDerive}(mpk_b)$
6: $dsk^* \leftarrow \mathsf{DskDerive}(mpk_b, msk_b, dpk^*)$
7: $b' \leftarrow \mathcal{A}^{O_{Dpk}(\cdot),O_{Sig}(\cdot)}(mpk_0, mpk_1, dpk^*)$
8: $b' \leftarrow \mathcal{A}^{O_{Dpk}(\cdot),O_{Sig}(\cdot),O_{Dsk}(\cdot)}(mpk_0,$
   $mpk_1, dpk^*)$
9: return $\|b' = b\|$

$O_{Dsk}(dpk, i \in \{0,1\})$:
1: if $dpk \in L_{Dpk,i}$
2:     $dsk = \mathsf{DskDerive}(mpk_i, msk_i, dpk)$
3: if $dpk \notin L_{Dpk,i}$
4:     return $\perp$
5: end
6: $L_{Dsk} = L_{Dsk} \cup \{dsk\}$
7: return $dsk$

$O_{Dpk}(dpk \neq dpk^*, i \in \{0,1\})$:
1: $t \leftarrow \mathsf{DpkCheck}(mpk_i, msk_i, dpk)$
2: if $t = 1$
3:     $L_{Dpk,i} \leftarrow L_{Dpk,i} \cup \{dpk\}$
4: return $t$

$O_{Sig}(pk, m, i \in \{0,1\})$:
1: if $pk = mpk_i$
2:     $\sigma \leftarrow \mathsf{Sign}(msk_i, m)$
3: if $pk = dpk^*$
4:     $\sigma \leftarrow \mathsf{Sign}(dpk^*, dsk^*, m)$
5: if $pk \in L_{Dpk,i}$
6:     $sk \leftarrow$
       $\mathsf{DskDerive}(mpk_i, msk_i, pk)$
7:     $\sigma \leftarrow \mathsf{Sign}(sk, m)$
8: if $pk \neq mpk_i$ and $pk \notin L_{Dpk,i}$
9:     return $\perp$
10: return $\sigma$
```

Fig. 4 UNL security and **SUNL security** of KDS with differences highlight in bold

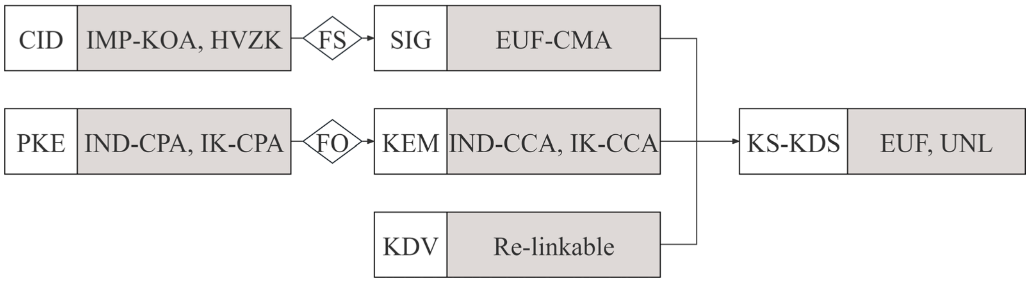

Fig. 5 A Framework to Construct KDS. Rectangles hold the crypto primitives, with security/privacy properties highlight in gray. Diamonds hold transformation algorithms

where the UNL game $\mathsf{G}^{\mathsf{UNL}}_{\mathsf{KDS},\mathcal{A}}(pp)$ is described as in Fig. 4.

The strong-EUF (SEUF) and strong-UNL (SUNL) security, which correspond to EUF and UNL security, respectively, provide the adversary with the additional capability of querying the secret key derivation algorithm. Their more detailed definitions are as follows:

Definition 8 (*SEUF Security*) A KDS scheme $\mathsf{KDS} = (\mathsf{KeyGen}, \mathsf{DpkDerive}, \mathsf{DpkCheck}, \mathsf{DskDerive}, \mathsf{Sign}, \mathsf{Verify})$ is SEUF secure, if there exists a negligible $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{SEUF}}_{\mathsf{KDS},\mathcal{A}}(pp) \leq negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage is defined as follows:

$$\mathsf{Adv}^{\mathsf{SEUF}}_{\mathsf{KDS},\mathcal{A}}(pp) = \Pr\left[\mathsf{G}^{\mathsf{SEUF}}_{\mathsf{KDS},\mathcal{A}}(pp) \to 1\right],$$

where the SEUF game $\mathsf{G}^{\mathsf{SEUF}}_{\mathsf{KDS},\mathcal{A}}(pp)$ is described as in Fig. 3.

Definition 9 (*SUNL Security*) A KDS scheme $\mathsf{KDS} = (\mathsf{KeyGen}, \mathsf{DpkDerive}, \mathsf{DpkCheck}, \mathsf{DskDerive}, \mathsf{Sign}, \mathsf{Verify})$ is UNL secure, if there exists a negligible function $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{SUNL}}_{\mathsf{KDS},\mathcal{A}}(pp) \leq negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage $\mathsf{Adv}^{\mathsf{SUNL}}_{\mathsf{KDS},\mathcal{A}}(pp)$ is defined as follows:

$$\mathsf{Adv}^{\mathsf{SUNL}}_{\mathsf{KDS},\mathcal{A}}(pp) = |\Pr\left[\mathsf{G}^{\mathsf{SUNL}}_{\mathsf{KDS},\mathcal{A}}(pp) \to 1\right] - 1/2|,$$

where the SUNL game $\mathsf{G}^{\mathsf{UNL}}_{\mathsf{KDS},\mathcal{A}}(pp)$ is described as in Fig. 4.

There are several differences between our security model and PDPKS's. First, we allow the adversary to access O_{Sig} with the master key and win the game by forging the master key signatures. This modification captures the fact that an adversary can request transactions or look up transactions signed by a user with his master key from the blockchain ledger, and steal the balance in the user's permanent address by forging his master key signature. Second, we define security as the basic security (EUF/UNL) and the strong security (SEUF/SUNL) by dividing adversary capabilities. In the basic security, we limit the adversary can only access O_{Dpk} and O_{Sig}, as considered in most SAP works. In the strong security, the adversary has an additional access to O_{Dsk}, which captures the assumption that user leaks his derived secret key, as PDPKS considered.

The construction of (strong) key separation key derivable signature scheme

In this section, we define a key derivation scheme (KDV) to extract derived keys from a master key pair, and propose a framework to construct (strong) key separation KDS (KS-KDS) as illustrated in Fig. 5.

Key derivation scheme

Definition 10 (*Key Derivation Scheme*) A KDV scheme consists of a triple PPT algorithms $(\mathsf{KDV.Gen}, \mathsf{KDV.Dpk}, \mathsf{KDV.Dsk}, \mathsf{KDV.Chk})$ as follows:

- $\mathsf{KDV.Gen}(pp) \to (mpk, msk, \mathcal{R})$:The key generation algorithm takes the public parameters as input and outputs a master public key/secret key pair (mpk, msk) and a deterministic polynomial-time verifiable relationship $\mathcal{R}$.
- $\mathsf{KDV.Dpk}(mpk, \mathcal{R}) \to dpk$: The public key derivation generation algorithm takes a master public key mpk and a deterministic polynomial-time verifiable relationship $\mathcal{R}$ as input, selects a random number $r \xleftarrow{\$} R$,[2] and outputs a derived public key $dpk := \mathsf{DKV.Dpk}(mpk, \mathcal{R}; r)$.

[2] R is determined by $\mathcal{R}$ and mpk.

```
$\mathsf{G}^{\mathsf{RLink}}_{\mathsf{KDV},\mathcal{A}}(pp)$:
1: $(mpk_0, msk_0, \mathcal{R}) \leftarrow \mathsf{KDV.Gen}(pp)$
2: $b \overset{\$}{\leftarrow} \{0,1\}$
3: $r_0 \overset{\$}{\leftarrow} R$
4: $dpk \leftarrow \mathsf{KDV.Dpk}(mpk_0, \mathcal{R}; r_0)$
5: $(mpk_1, r_1) \leftarrow \mathsf{Rel}(dpk, \mathcal{R})$
6: $b' \leftarrow \mathcal{A}(\mathcal{R}, mpk_b, dpk, r_b)$
7: $\mathbf{return}\ \|b' = b\|$
```

```
$\mathsf{G}^{\mathsf{LR}}_{\mathsf{KDV},\mathcal{A}}(pp)$:
1: $(mpk, msk, \mathcal{R}) \leftarrow \mathsf{KDV.Gen}(pp)$
2: $b \overset{\$}{\leftarrow} \{0,1\}, r_0 \overset{\$}{\leftarrow} R$
3: $dpk_0 \leftarrow \mathsf{KDV.Dpk}(mpk, \mathcal{R}; r_0)$
4: $dsk_0 \leftarrow \mathsf{KDV.Dsk}(mpk, msk, \mathcal{R}; r_0)$
5: $(r_1, aux) \leftarrow \mathsf{Alt}_1(mpk, \mathcal{R})$
6: $(dpk_1, dsk_1) \leftarrow \mathsf{Alt}_2(mpk, \mathcal{R}, r_1, aux)$
7: $b' \leftarrow \mathcal{A}(mpk, r_b, dpk_b, dsk_b)$
8: $\mathbf{return}\ \|b' = b\|$
```

Fig. 6 The re-linkability and leakage-resistance game for the KDV scheme KDV

- $\mathsf{KDV.Dsk}(mpk, msk, \mathcal{R}) \to dsk$: The secret key derived generation algorithm takes a master public/secret key pair (mpk, msk) and a deterministic polynomial-time verifiable relationship $\mathcal{R}$, selects a random number $r \overset{\$}{\leftarrow} R$ and outputs a derived secret key $dsk := \mathsf{KDV.Dsk}(mpk, msk, \mathcal{R}; r)$.
- $\mathsf{KDV.Chk}(pk, sk, \mathcal{R}) \to 0/1$:The validity checking algorithm takes a public/secret key pair (pk, sk) and a deterministic polynomial-time verifiable relationship $\mathcal{R}$ and outputs a bit b, where $b = 1$ means that $(pk, sk) \in \mathcal{R}$ and $b = 0$ means that $(pk, sk) \notin \mathcal{R}$

Additionally, the above algorithms must satisfy the following correctness:

- For any $(mpk, msk, \mathcal{R}) \leftarrow \mathsf{KDV.Gen}(pp)$,
$$\mathsf{KDV.Chk}(mpk, msk, \mathcal{R}) = 1.$$
- For any $(mpk, msk, \mathcal{R}) \leftarrow \mathsf{KDV.Gen}(pp)$ and $r \in R$,
$$\mathsf{KDV.Chk}(dpk, dsk, \mathcal{R}) = 1,$$
where $dpk := \mathsf{KDV.Dpk}(mpk, \mathcal{R}; r)$ and $dsk := \mathsf{KDV.Dsk}(mpk, msk, \mathcal{R}; r)$
- For any $(mpk, msk, \mathcal{R}) \leftarrow \mathsf{KDV.Gen}(pp)$, if $r \neq r'$, then
$$\mathsf{KDV.Chk}(dpk, dsk, \mathcal{R}) = 0,$$
where $dpk := \mathsf{KDV.Dpk}(mpk, \mathcal{R}; r)$ and $dsk := \mathsf{KDV.Dsk}(mpk, msk, \mathcal{R}; r')$.

For a key generation algorithm Gen, if the relationship $\mathcal{R} := \{(pk, sk) | (pk, sk) \leftarrow \mathsf{Gen}(pp)\}$ is polynomial-time verifiable, we can define a KDV for Gen as follows:

- $\mathsf{KDV.Gen}(pp) \to (mpk, msk, \mathcal{R})$: It runs $(mpk, msk) \leftarrow \mathsf{Gen}(pp)$ and extracts a deterministic polynomial-time verifiable relation $\mathcal{R} := \{(pk, sk) | (pk, sk) \leftarrow \mathsf{Gen}(pp)\}$. Then outputs $(mpk, msk, \mathcal{R})$.

Security. We define the re-linkability and leak-resistance for KDV. Re-linkability states that a derived public key $dpk := \mathsf{KDV.Dpk}(mpk, \mathcal{R}; r)$ can be relinked to $mpk' \neq mpk$. It ensures that dpk does reveal any information about the master public key mpk. Leak-resistance states that derived public/secret key pairs (dpk, dsk) do not reveal any information about the master secret key msk.

Definition 11 (*Re-linkability*) A KDV scheme $\mathsf{KDV} = (\mathsf{KDV.Gen}, \mathsf{KDV.Dpk}, \mathsf{KDV.Dsk}, \mathsf{KDV.Chk})$ is re-linkable, if there exists a PPT "re-link" algorithm Rel and a negligible function $negl(\cdot)$ such that $\mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{KDV},\mathcal{A}}(pp) \leq negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage is defined as follows:

$$\mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{KDV},\mathcal{A}}(pp) = |\Pr\left[\mathsf{G}^{\mathsf{RLink}}_{\mathsf{KDV},\mathcal{A}}(pp) \to 1\right] - 1/2|,$$

where the game $\mathsf{G}^{\mathsf{RLink}}_{\mathsf{KDV},\mathcal{A}}(pp)$ is described as Fig. 6.

Definition 12 (*Leakage Resistance*) A KDV scheme $\mathsf{KDV} = (\mathsf{KDV.Gen}, \mathsf{KDV.Dpk}, \mathsf{KDV.Dsk}, \mathsf{KDV.Chk})$ is leakage resistant, if there exist a PPT algorithm $\mathsf{Alt} = (\mathsf{Alt}_1, \mathsf{Alt}_2)$ and a negligible function $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{LR}}_{\mathsf{KDV},\mathcal{A}}(pp) \leq negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage is defined as follows:

$$\mathsf{Adv}^{\mathsf{LR}}_{\mathsf{KDV},\mathcal{A}}(pp) = |\Pr[\mathsf{G}^{\mathsf{LR}}_{\mathsf{KDV},\mathcal{A}}(pp) \to 1] - 1/2|,$$

where the game $\mathsf{G}^{\mathsf{LR}}_{\mathsf{KDV},\mathcal{A}}(pp)$ is described as Fig. 6.

A framework to construct (strong) KS-KDS

We propose a KS-KDS framework that is based on a canonical identification protocol CID, a public key encryption PKE, and a key derivation scheme KDV.

Let CID $=$ (CID.Gen, CID.P, CID.V) be a CID protocol and KDV $=$ (KDV.Gen, KDV.Dpk, KDV.Dsk, KDV.Chk) be a KDV scheme for CID. Let PKE $=$ (PKE.Gen, PKE.Enc, PKE.Dec) be a PKE scheme. We compose them to construct a KS-KDS as shown in Algorithm 1. We use Fiat-Shamir transformation (FS) on CID to get a signature scheme, use Fujisaki-Okamoto transformation (FO) on PKE to get a KEM. This KEM scheme is used to generated the k which is the randomness used to generated the derivable key pair. G, H and H_1 are random oracles.

Remark. Note that constructing KS-KDS using SIG, KEM, and KDV is also feasible. However, our approach prefers to build upon more fundamental cryptographic primitives, and this method is particularly useful for designing the KR-KDS algorithm in "The construction of (strong) key reusing key derivation scheme" section.

Security. We then discuss the security of the above KS-KDS construction.

Theorem 1 *Let* CID *be an IMP-KOA secure CID protocol with HVZK. Let* KDV *be a re-linkable KDV scheme for* CID*. Let* PKE *be an IND-CPA secure PKE scheme. Let G, H and H_1 are random oracles. Then, the KS-KDS scheme is EUF secure in the random oracle model.*

Algorithm 1 Key Separation Key Derivation Signature Scheme

```
KeyGen(GP):
1:  $(spk, ssk, \mathcal{R}) \leftarrow \mathsf{KDV.Gen}(pp)$
2:  $(epk, esk) \leftarrow \mathsf{PKE.Gen}(pp)$
3:  $mpk := spk||epk||\mathcal{R}$
4:  $msk := ssk||esk$
5:  return $(mpk, msk)$
DpkDerive(mpk):
6:  $m \overset{\$}{\leftarrow} M$
7:  $c := \mathsf{PKE.Enc}(epk, m; G(m))$
8:  $k = H(m, c)$
9:  $sdpk := \mathsf{KDV.Dpk}(spk, \mathcal{R}; k)$
10: $dpk := (c, sdpk)$
11: return $dpk$
DpkCheck(mpk, msk, dpk):
12: Parse $mpk$ as $spk$, $epk$ and $\mathcal{R}$
13: Parse $msk$ as $ssk$ and $esk$
14: Parse $dpk$ as $c$ and $sdpk$
15: $m := \mathsf{PKE.Dec}(esk, c)$
16: $k := H(m, c)$
17: $sdpk' := \mathsf{KDV.Dpk}(spk, \mathcal{R}; k)$
18: if $sdpk' \neq sdpk$
19:     return 0
20: return 1
DskDerive(mpk, msk, dpk):
21: Parse $mpk$ as $spk$, $epk$ and $\mathcal{R}$
22: Parse $msk$ as $ssk$ and $esk$
23: Parse $dpk$ as $c$ and $sdpk$
24: if $\mathsf{DpkCheck}(mpk, msk, dpk) = 0$
25:     return $\perp$
26: $m := \mathsf{PKE.Dec}(esk, c)$
27: $k := H(m, c)$
28: $dsk := \mathsf{KDV.Dsk}(spk, ssk, \mathcal{R}; k)$
29: return $dsk$
Sign(sk, m):
30: $(com, \mathsf{st}) \leftarrow \mathsf{CID.P}_1(sk)$
31: $ch := H_1(com, m)$
32: $rsp \leftarrow \mathsf{CID.P}_2(\mathsf{st}, com, ch)$
33: $\sigma := (com, ch, rsp)$
34: return $\sigma$
Verify(pk, m, σ):
35: Parse $pk$ as $c$ and $sdpk$ or parse
    $pk$ as $spk||epk||\mathcal{R}$
36: $b \leftarrow \mathsf{CID.V}_2(sdpk, m, \sigma)$ or
    $b \leftarrow \mathsf{CID.V}_2(spk, m, \sigma)$
37: return $b$
```

Proof Assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ that breaks the EUF security of the KS-KDS scheme with noticeable probability. That is, the following advantage is noticeable:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) &= \Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1]\\ &= \Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1, pk^* = mpk] + \Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1, pk^* \in L_{Dpk}].\end{aligned}$$

We consider two different cases and construct PPT adversary $\mathcal{B}$ or $\mathcal{C}$ against the IMP-POA security of CID, respectively:

Case 1 ($pk^* = mpk$). $\mathcal{A}$ generate the valid forge signature for mpk. We then construct $\mathcal{B}$ to attack CID as follows:

- *Setup Phase* The IMP-KOA challenger generates a pair of public/secret keys $(spk, ssk) \leftarrow \mathsf{CID.Gen}(pp)$. The IMP-KOA adversary $\mathcal{B}$ receives spk and public parameters pp. It runs $(epk, esk) \leftarrow \mathsf{PKE.Gen}(pp)$ and extracts a deterministic polynomial time verifiable relationship $\mathcal{R} := \{(pk, sk)|(pk, sk) \leftarrow \mathsf{CID.Gen}(pp)\}$. Let $mpk := spk||epk||\mathcal{R}$ and $msk := ssk||esk$. It sends mpk to $\mathcal{A}$ and initializes three query lists L_{H_1}, L_{Dpk} and L_{Sig} as empty sets. Let Sim be a PPT algorithm defined as in the Definition 3.
- *Challenge Phase* The challenger picks a challenge $ch^* \xleftarrow{\$} CH$ and sends it to $\mathcal{B}$.
- *Query Phase* For each $\mathcal{A}$'s query m and (m, c), O_G and O_H return $H(m)$ and $H(m, c)$, respectively. Additionally, O_{H_1}, O_{Dpk} and O_{Sig} are simulated by the adversary $\mathcal{B}$ as follows:
  - $O_{H_1}(com, m)$: At beginning, the adversary $\mathcal{B}$ picks $i^* \xleftarrow{\$} [1, q_{H_1}]$. For $\mathcal{A}$'s i-th ($i \neq i^*$) query (com, m), if there exists $(com||m, ch) \in L_{H_1}$, it returns ch. Otherwise, it randomly selects ch and lets $L_{H_1} = L_{H_1} \cup \{(com||m, ch)\}$; then returns h_1. Specially, for $\mathcal{A}$'s i^*-th query (com, m),[3] it returns ch^* and lets $L_{H_1} = L_{H_1} \cup \{(com||m, ch^*)\}$.
  - $O_{Dpk}(dpk)$: For each $\mathcal{A}$'s query $dpk = (c, sdpk)$ to O_{Dpk}, the adversary $\mathcal{B}$ calculates $m := \mathsf{PKE.Dec}(esk, c)$. If $m = \perp$ or $\mathsf{PKE.Enc}(esk, m; G(m)) \neq c$, it returns 0. It calculates $k := H(m, c)$, if $sdpk = \mathsf{KDV.Dpk}(spk, \mathcal{R}; k)$, it returns 1 and lets $L_{Dpk} := L_{Dpk} \cup \{dpk\}$. Otherwise,it returns 0.
  - $O_{Sig}(pk, m)$: For each $\mathcal{A}$'s query (pk, m) to O_{Sig}, if $pk = (c, sdpk) \in L_{Dpk}$ or $pk = mpk$, $\mathcal{B}$ keeps running $(com, ch, rsp) \leftarrow \mathsf{Sim}(sdpk)$ or $(com, ch, rsp) \leftarrow \mathsf{Sim}(spk)$ respectively until there does not exist $com' \neq com$ and $ch' \neq ch$ such that $(com'||m, ch) \in L_{H_1}$ and $(com||m, ch') \in L_{H_1}$. It returns $\sigma := (com, ch, rsp)$ and sets $L_{H_1} = L_{H_1} \cup \{(com||m, ch)\}$ and $L_{Sig} = L_{Sig} \cup \{(pk, m)\}$. Otherwise, it returns $\perp$.
- *Output Phase* When $\mathcal{A}$ outputs (pk^*, m^*, σ^*), $\mathcal{B}$ parses σ^* as (com^*, ch'^*, rsp^*) and outputs (com^*, rsp^*).

Since the adversary $\mathcal{B}$ has the secret key esk of the PKE, the simulation of O_{Dpk} is perfect. Since H_1 is a random oracle, the simulation of O_{H_1} is perfect. Additionally, $\mathcal{B}$ simulates the oracle O_{Sig} without using of ssk. Because CID is HVZK, the probability that the simulation of O_{Sig} can be distinguished by the adversary $\mathcal{A}$ is at most $q_{Sig} \cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}$. Then for *Case 1*, we have (detailed derivation can be found in E):

$$\Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1, pk^* = mpk] \leq q_{H_1} \cdot \mathsf{Adv}^{\mathsf{IMP-POA}}_{\mathsf{CID},\mathcal{B}}(\mathrm{pp}) + q_{Sig} \cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp) + \frac{1}{|CH|},$$

where q_{H_1}, q_{Sig} are the numbers of $\mathcal{A}$'queries to O_{H_1} and O_{Sig}, respectively.

Case 2 ($pk^* \in L_{Dpk}$). $\mathcal{A}$ generate the valid forge signature for a $dpk \in L_{Dpk}$. We then construct $\mathcal{C}$ to attack CID. Naturally, the challenge public key for $\mathcal{C}$ is a derived key which belongs to L_{Dpk}. Therefore $\mathcal{C}$ needs to generate related master public key of the challenge public key by using the re-linkability of KDV. The details of $\mathcal{C}$ are as follows:

- *Setup Phase* The IMP-KOA challenger generates a key pair $(dspk, dssk) \leftarrow \mathsf{CID.Gen}(pp')$. The IMP-KOA adversary $\mathcal{C}$ receives $dspk$ and public parameters pp. It runs $(epk, esk) \leftarrow \mathsf{PKE.Gen}(pp)$ and extracts a deterministic polynomial-time verifiable relationship $\mathcal{R} := \{(pk, sk)|(pk, sk) \leftarrow \mathsf{CID.Gen}(pp)\}$. It runs $(spk, k^*) \leftarrow \mathsf{Rel}(dspk, \mathcal{R})$ and lets $mpk := spk||epk||\mathcal{R}$. Then, it sends mpk and pp to $\mathcal{A}$ and initializes query lists L_H, L_{H_1}, L_{Dpk} and L_{Sig} as empty sets. Let Sim be a PPT algorithm defined as in the Definition 3.

[3] Without loss of generality, we assume that (com, m) has not been queried.

- *Challenge Phase* The challenger generates a challenge ch^* and sends it to $\mathcal{C}$.
- *Query Phase* For each $\mathcal{A}$' query m to O_G, it returns $r = G(m)$. O_{H_1}, O_{Dpk} and O_{Sig} are simulated as in the construction of $\mathcal{B}$. Additionally, O_H is simulates by the adversary $\mathcal{C}$ as follows:
  - $O_H(m, c)$: The adversary $\mathcal{C}$ picks $j^* \xleftarrow{\$} [1, q_H]$. For $\mathcal{A}$'s j-th ($j \neq j^*$) query (m, c) to O_H, if there exists $(m||c, k) \in L_{H_1}$, it returns k. Otherwise, it randomly selects k and lets $L_H = L_H \cup \{(m||c, k)\}$; then returns h_1. Specially, for $\mathcal{A}$'s j^*-th query (m, c),[4] it returns k^* and lets $L_H = L_H \cup \{(m||c, k^*)\}$.
- *Output Phase* When $\mathcal{A}$ outputs (pk^*, m^*, σ^*), $\mathcal{C}$ parse σ^* as (com^*, ch'^*, rsp^*) and outputs (com^*, σ^*).

Applying the same reasoning as in **Case 1**, the simulation of O_{Dpk} and O_{H_1} is perfect and the simulation of O_{Sig} can be distinguished by the adversary $\mathcal{A}$ with probability at most $q_{Sig} \cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp)$. Since the re-linkability of KDV and O_H is also a random oracle, the $(mpk, dspk, k^*)$ and the simulation of O_H can be distinguished by $\mathcal{A}$ with probability at most $\mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{CID},\mathcal{A}}(pp)$. Then for *Case 2*, we have (detailed derivation can be found in E):

$$\Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1, pk^* \in L_{Dpk}] \leq q_{H_1} q_H \cdot \mathsf{Adv}^{\mathsf{IMP-POA}}_{\mathsf{CID},\mathcal{C}}(\mathsf{pp}) + q_{Sig} \cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp) + \mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{CID},\mathcal{A}}(pp) + \frac{1}{|CH|},$$

Put these results together,

$$\Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1] \leq q_{H_1} \cdot \mathsf{Adv}^{\mathsf{IMP-POA}}_{\mathsf{CID},\mathcal{B}}(\mathsf{pp}) + q_{H_1} q_H \cdot \mathsf{Adv}^{\mathsf{IMP-POA}}_{\mathsf{CID},\mathcal{C}}(pp) + 2q_{Sig} \cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp) + \mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{CID},\mathcal{A}}(pp) + \frac{2}{|CH|}.$$

Thus, the KS-KDS scheme is EUF secure in the random oracle model. □

Theorem 2 *Let* CID *be an IMP-KOA secure CID protocol with HVZK. Let* KDV *be a re-linkable and leakage-resistant KDV scheme for* CID. *Let* PKE *be an IND-CPA secure PKE scheme. Let G, H and H_1 are random oracles. Then, the KS-KDS scheme is SEUF secure in the random oracle model.*

Proof This theorem can be proved using the same proof strategy for Theorem 1. We put the full proof in E for self-completeness. □

Theorem 3 *Let* CID *be an IMP-KOA secure CID protocol with HVZK. Let* KDV *be a re-linkable KDV scheme for* CID. *Let* PKE *be an IND-CPA secure and IK-CPA secure PKE scheme. Let G, H and H_1 are random oracles. Then, the KS-KDS scheme is UNL secure in the random oracle model.*

Proof Assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ that breaks the UNL security of the KS-KDS scheme. We can construct a PPT adversary $\mathcal{B}$, breaking the IK-CPA security of the PKE scheme, we put the full proof in E for self-completeness. □

Theorem 4 *Let* CID *be an IMP-KOA secure protocol with HVZK. Let* KDV *be a re-linkable and leakage-resistant KDV scheme for* CID. *Let* PKE *be an IND-CPA secure and IK-CPA secure PKE scheme. Let G, H and H_1 are random oracles. Then, the KS-KDS scheme is SUNL secure in the random oracle model.*

Proof One can use the same proof strategy for Theorems 2 and 3 to prove this theorem. We give the full proof in E for self-completeness. □

Instantiation

Finally, we give two instantiations of our KS-KDS framework with EUF/UNL security and SEUF/SUNL security, respectively.

A KS-KDS instantiation on the elliptic curve

We instantiate the CID in Algorithm 1 to the Schnorr underlying identification protocol, which is HVZK and

[4] Without loss of generality, we assume that (m, c) has not been queried.

IMP-KOA secure, the full proof could be derived from Pointcheval and Stern (1996); we instantiate the PKE to EC-Elgamal, which is IND-CPA secure and IK-CPA private (this is noted in Naor and Reingold 1997; Cramer and Shoup 1998 and fully treated in Tsiounis and Yung 1998). Let $pp = (E, \mathcal{G}, q)$ be system parameters specifying a secure elliptic curve E with a generator $\mathcal{G}$ that generates a secure cyclic subgroup $\mathbb{G}$ with prime order q. Then we propose a KDV instantiation on the elliptic curve as follows:

- $\mathsf{KDV.Dpk}(pp) \to (B, b, \mathcal{R})$: It randomly chooses $b \in \mathbb{Z}_q$ and calculates $B = b \cdot \mathcal{G}$. Let $\mathcal{R} := \{(pk, sk) | pk = sk \cdot \mathcal{G}, sk \in \mathbb{Z}_q\}$. Then, it outputs $(B, b, \mathcal{R})$.
- $\mathsf{KDV.Dpk}(B, \mathcal{R}) \to \tilde{B}$: It randomly chooses $r \in \mathbb{Z}_q$ and outputs $\tilde{B} := r \cdot B$
- $\mathsf{KDV.Dsk}(B, b, \mathcal{R}) \to \tilde{b}$: It randomly chooses $r \in \mathbb{Z}_q$ and outputs $\tilde{b} := r \cdot b \mod q$.
- $\mathsf{KDV.Chk}(pk, sk, \mathcal{R}) \to \tilde{b}$: It checks whether $(pk, sk) \in \mathcal{R}$. If $(pk, sk) \in \mathcal{R}$, it outputs 1. Otherwise, outputs 0.

This KDV is re-linkable since we can construct Rel as follow: It takes as input $\tilde{B}$, and randomly chooses $a \in \mathbb{Z}_q$. It calculates $B' = a \cdot \tilde{B}$ and $r' = a^{-1}$, then outputs (B', r').

A strong KS-KDS instantiation on the lattice

We first propose a CID instantiation on the lattice using trapdoor generation algorithm TrapGen defined in Alwen and Peikert (2009), Boyen (2010) and pre-image sample algorithm SamplePre proposed in Gentry et al. (2008):

- $\mathsf{CID.Gen}(pp)$. The algorithm runs $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(pp)$, then outputs the public key $\mathbf{A}$ and the secret key $\mathbf{T_A}$.
- $\mathsf{CID.P_1}(\mathbf{A}) \to (com, \mathsf{st})$. $\mathsf{CID.P_1}$ randomly chooses $\mathbf{t} \leftarrow \mathbb{Z}_q^n$, computes $com = \mathbf{At}$ and generates the state st. It sends com to verifier V.
- $\mathsf{CID.V_1}(\mathbf{A}, com) \to ch$. On receiving $com = \mathbf{At}$ from P, $\mathsf{CID.V_1}$ randomly chooses $r \in \mathbb{Z}_q$ and $\mathbf{x} \in \mathbb{Z}_q^n$. Calculates $\mathbf{u} = \mathbf{Ax}$ and returns $ch := (r, \mathbf{u})$.
- $\mathsf{CID.P_2}(\mathsf{st}, \mathbf{A}, \mathbf{T_A}, com, ch) \to rsp$. On receiving $ch := (r, \mathbf{u})$, $\mathsf{CID.P_2}$ runs $\mathbf{x}' \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{u})$, and computes and returns $rsp = \mathbf{t} + r\mathbf{x}'$.
- $\mathsf{CID.V_2}(\mathbf{A}, com, ch, rsp) \to 1/0$. On receiving $rsp = \mathbf{t} + r\mathbf{x}'$, $\mathsf{CID.V_2}$ computes and checks $com + r \cdot \mathbf{u} = \mathbf{A} \cdot rsp$. It accepts the proof when the check passes.

This CID is HVZK and IMP-KOA secure, the full proof could be derived from Lyubashevsky (2012). We then instantiate the PKE in Algorithm 1 to LWEPKE, which is IND-

CPA secure and IK-CPA private (Regev 2009); and propose a re-linkable and leakage-resistant KDV inspired by the lattice-based PDPKS scheme (Liu et al. 2020):

- $\mathsf{KDV.Gen}(pp) \to (\mathbf{A}, \mathbf{T_A}, \mathcal{R})$: It runs $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(pp)$, then outputs the master public key $\mathbf{A}$, the master secret key $\mathbf{T_A}$, and a relation $\mathcal{R} := \{(pk, sk) | pk \cdot sk = 0\}$.
- $\mathsf{KDV.Dpk}(\mathbf{A}, \mathcal{R}) \to \mathbf{F}$: It randomly chooses $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{R} \in \{-1, 1\}^{m \times m}$ and calculates $\mathbf{F} = [\mathbf{B} | \mathbf{BR} + \mathbf{A}]$. Returns $\mathbf{F}$.
- $\mathsf{KDV.Dsk}(\mathbf{A}, \mathbf{T_A}, \mathcal{R}) \to \mathbf{T_F}$: It randomly chooses $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{R} \in \{-1, 1\}^{m \times m}$ and calculates $\mathbf{T_F} \leftarrow \mathsf{DeleRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}, \mathbf{R}, \mathsf{GP})$. Returns $\mathbf{T_F}$.
- $\mathsf{KDV.Chk}(\mathbf{F}', \mathbf{T_F}', \mathcal{R}) \to 0/1$: it returns 1 if $(\mathbf{F}', \mathbf{T_F}') \in \mathcal{R}$, and returns 0 otherwise.

DeleRight is a trapdoor delegation algorithm defined by Liu et al. (2020). This KDV is re-linkable and leakage-resistant, we give the proof sketch in D.

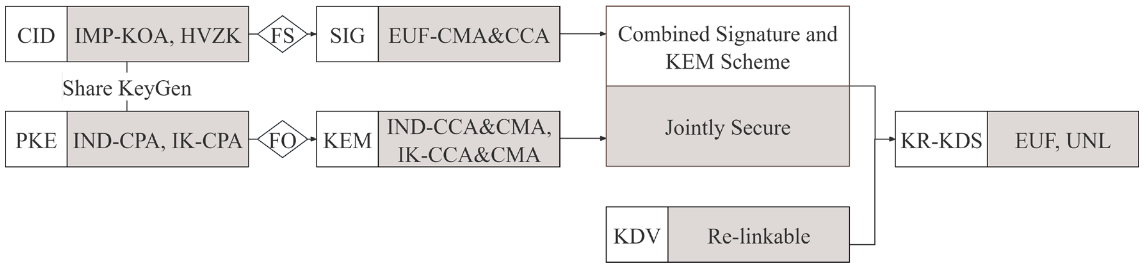

Fig. 7 A Framework to Construct KR-KDS. Rectangles hold the crypto primitives, with security/privacy properties highlight in gray. Diamonds hold transformation algorithms

The construction of (strong) key reusing key derivation scheme

In "The construction of (strong) key separation key derivable signature scheme" section we propose a framework to construct KS-KDS. It is a dual-key framework that follows the key separation principle. We observe that the random oracles in our framework play a similar separation role. Therefore, in this section we revise the key generation part to obtain a key reusing KDS (KR-KDS) framework as shown in Fig. 7.

A framework to construct (strong) KR-KDS

We construct a KR-KDS based on a canonical identification protocol CID and a public key encryption PKE sharing the same key generation algorithm Gen, and a key derivation scheme KDV.

Let the $\mathsf{CID} = (\mathsf{Gen}, \mathsf{CID.P}, \mathsf{CID.V})$ is IMP-KOA secure and HVZK, the $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ is IND-CPA and IK-CPA secure, and $\mathsf{KDV} = (\mathsf{Gen}, \mathsf{KDV.Dpk}, \mathsf{KDV.Dsk}, \mathsf{KDV.Chk})$ is a KDV scheme for CID. Then we can construct a KR-KDS by simply replacing $(epk, esk) = (spk, ssk) \to (pk, sk)$ in Algorithm 1. The KeyGen algorithm of the KR-KDS scheme is as follows:

- $\mathsf{KeyGen}(\mathsf{GP}) \to (pk, sk)$: It runs $\mathsf{Gen}(pp)$ to gets (pk, sk) and returns them.

We put the full algorithm in B for self-completeness.

Remark We remark that our KR-KDS framework contains a combined signature and KEM scheme (CSK): let $\mathsf{SIG} = \mathsf{FS}(\mathsf{CID}, H_1) = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, $\mathsf{KEM} = \mathsf{FO}(\mathsf{PKE}, H) = (\mathsf{Gen}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$, then we obtain a CSK scheme $\mathsf{CSK} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$. This is a novel framework to construct CSK that is jointly secure. It has independent interests outside of stealth address, which we present further discussion in C.

Security Our KR-KDS construction has EUF/UNL security when KDV is UNL secure, and SEUF/SUNL security when KDV is UNL and IND secure.

Theorem 5 *Let* CID *be and IMP-KOA secure CID protocol with HVZK. Let* PKE *be an IND-CPA secure and IK-CPA secure PKE scheme. Let* KDV *be a re-linkable KDV scheme for* CID *and* PKE*. Let G, H and H_1 be random oracles. Then the KR-KDS scheme is EUF and UNL secure in the random oracle model.*

Proof The proof is similar to Theorems 1 and 3. We put the proof sketch in E. □

Theorem 6 *Let* CID *be and IMP-KOA secure CID protocol with HVZK. Let* PKE *be an IND-CPA secure and IK-CPA secure PKE scheme. Let* KDV *be a re-linkable and leakage-resistant KDV scheme for* CID *and* PKE*. Let G, H and H_1 be random oracles. Then the KR-KDS scheme is SEUF and SUNL secure in the random oracle model.*

Proof The proof is similar to Theorems 2 and 4. We put the proof sketch in E. □

Instantiation

We then improve the two KS-KDS instantiations in "Instantiation" section to KR-KDS schemes.

The KR-KDS instantiation on the elliptic curve

We still use the Schnorr CID, EC-Elgamal encryption and our elliptic curve KDV in "Instantiation" section to instantiate KR-KDS, since their key pair have the same relation. It implies a Schnorr-based compact stealth address protocol that can be used in the single-key cryptocurrency systems that support Schnorr signature such as Bitcoin (support from 2021.11), BCH and Polkadot Wuille et al. (2021). The full algorithm in shown follows:

Let $pp = (E, \mathcal{G}, q)$ be system parameters specifying a secure elliptic curve E with a generator point $\mathcal{G}$ that generates a secure cyclic subgroup $\mathbb{G}$ with prime order q. Then the CKDS scheme can be described as follows:

- $\mathsf{KeyGen}(E, \mathcal{G}, q) \to ((E, \mathcal{G}, q, B), b)$: It picks $b \xleftarrow{\$} \mathbb{Z}_q$, calculates $B = b \cdot \mathcal{G} \mod p$ and lets $pk := (E, \mathcal{G}, q, B)$ and $sk := (b, pk)$. Then, it returns (pk, sk).
- $\mathsf{DpkDerive}(pk) \to (c, (E, \mathcal{G}, q, \tilde{B}))$: It picks $r \xleftarrow{\$} \mathbb{Z}_q$ and calculates $c := (c_1, c_2)$, where $c_1 = G(r) \cdot \mathcal{G}$ and $c_2 = rG(r) \cdot B$. Lets $k = H(r||c)$ and $\tilde{B} := k \cdot B$. Then, it returns $(c, \tilde{B})$.
- $\mathsf{DpkCheck}(pk, sk, (c, \tilde{B})) \to 1/0$: It calculates $r = c_2/(b \cdot c_1) \mod q$. If $c_1 = G(r) \cdot \mathcal{G}$, $c_2 = rG(r) \cdot B$ and $\tilde{B} = H(r||c) \cdot B$, returns 1. Otherwise 0.

- $\mathsf{DskDerive}(pk, sk, (c, \tilde{B})) \rightarrow dsk/\perp$: If $\mathsf{DpkCheck}(pk, sk, (c, \tilde{B})) = 1$, it calculates $r = c_2/(b \cdot c_1) \bmod q$ and lets $\tilde{b} = H(r, c) \cdot b$. Then, it returns $dsk := (\tilde{b}, pk)$. Otherwise, it returns $\perp$.
- $\mathsf{Sign}(sk/dsk, m) \rightarrow \sigma$: It parse sk (or dsk) as b' and pk. It picks $z \xleftarrow{\$} \mathbb{Z}_q$ and calculates $com := z \cdot \mathcal{G}$. Lets $ch := H_1(f(com_x)||f(com_y)||m)$ where $com_{x/y}$ denotes the x/y-coordinate of com and $f(\cdot)$ is a conversion function that converts a field element into a bit-string. It computes $rsp := z + hb' \bmod q$ and returns (ch, rsp) unless $rsp = 0$ or $ch = 0$. In this latter case the whole procedure is repeated.
- $\mathsf{Verify}((E, \mathcal{G}, q, B'), m, \sigma) \rightarrow 1/0$: If $ch \neq 0$ and $rsp \in \mathbb{Z}_q$, it computes $com' = rsp \cdot \mathcal{G} - ch \cdot B'$ and $ch' = H_1(f(com'_x)||f(com'_y)||m)$. If $ch' = ch$, it returns 1. Otherwise, it returns 0.

Remark. Our solution is a framework that can have multiple instantiations, such as using EC-IES instead of EC-Elgamal. EC-IES satisfies the requirements set forth in "The construction of (strong) key reusing key derivation scheme" section and is jointly secure with Schnorr (Degabriele et al. 2012):

Lemma 1 (Degabriele et al. 2012) *ECIES-KEM and EC-Schnorr are jointly secure in the random oracle model, if the gap-DLP problem and gap-DH problem are both hard.*

The strong KR-KDS instantiation on the lattice

Unlike the KR-KDS implementation on the elliptic curve, we cannot use the same components as in the lattice instantiation in "Instantiation" section to construct the KR-KDS scheme. This is because our CID and KDV use a lattice basis and its trapdoor as a public and secret key pair, while LWEPKE use a different relation. We then propose a trapdoor encryption algorithm instead of LWEPKE:

According to the parameters of learning with errors problem (LWE) and the chosen ciphertext-secure encryption scheme constructed by Micciancio and Peikert (2012), let $\mathcal{D} := D^{\bar{m} \times nk}_{\mathbb{Z}, \omega(\sqrt{logn})}$, $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ is a gadget matrix for large enough prime power $q = p^e$ and $k = O(logn) = O(logq)$. $\mathbf{A}^{\perp}$ denotes the transpose of $\mathbf{A}$. We give a PKE scheme $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ as follows:

- $\mathsf{PKE.Gen}(pp) \rightarrow (\mathbf{A}, \mathbf{T_A})$: it randomly chooses $\mathbf{A_1} \in \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$. Then lets $\mathbf{A_2} = -\mathbf{A_1R} \bmod q$, $\mathbf{A} = [\mathbf{A_1}|\mathbf{A_2}]$ and $\mathbf{T_A} = [\mathbf{R}^{\perp}|\mathbf{I}]^{\perp}$. Returns $(\mathbf{A}, \mathbf{T_A})$.
- $\mathsf{PKE.Enc}(\mathbf{A} = [\mathbf{A_1}|\mathbf{A_2}], \mathbf{s}) \rightarrow c$: it samples $\mathbf{e_1} \leftarrow D^{\bar{m}}_{\mathbb{Z},\alpha q}$ and $\mathbf{e_2} \leftarrow D^{nk}_{\mathbb{Z},\alpha q}$. And calculates $c_1 := \mathbf{sA_1} + \mathbf{e_1}$ and $c_1 := \mathbf{sG} + \mathbf{sA_2} + \mathbf{e_2}$.
- $\mathsf{PKE.Dec}(\mathbf{T_A} = [\mathbf{R}|\mathbf{I}], c = (c_1, c_2)) \rightarrow \mathbf{s}/\perp$: it calculates $\mathbf{sG} + \mathbf{e_1R} + \mathbf{e_2} = c_2 + c_1 \cdot \mathbf{R}$ and $\mathbf{s} \leftarrow \mathbf{sG} + \mathbf{e_1R} + \mathbf{e_2}$.

Conclusion

Stealth address protocol, a cryptographic technique widely used in blockchain systems, lacks formalized definition, theoretical analysis and frameworks. Additionally, there is no existing work that studies the key reuse in such a crypto primitive that contains both a derivation scheme and a signature scheme. In this paper, we fill all these gaps.

We proposed a key derivable signature scheme (KDS) to formalize the stealth address protocol, and propose frameworks and the first compact scheme by constructing a jointly secure and key private CSK. Our construction can not only effectively simplify the stealth address protocol and privacy-preserving blockchain systems, but it also has the potential to be deployed into existing widely used cryptocurrency systems to provide privacy protection.

Appendix A: Security models of PDPKS

Definition 13 (*PDPKS* Liu et al. 2019b) A PDPKS scheme with message space M consists of following algorithms:

- $\mathsf{Setup}(1^{\lambda}) \rightarrow \mathsf{pp}$. The setup algorithm takes the security parameters 1^{λ} as input and outputs public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}) \rightarrow (mpk, msk)$. The key generation algorithm takes the public parameters pp as input and outputs a master public key/secret key pair (mpk, msk).
- $\mathsf{DpkDerive}(mpk) \rightarrow dpk$. The public key derivation algorithm takes a master public key mpk and outputs a derived public key dpk.

- $\mathsf{DpkCheck}(mpk, msk, dpk) \rightarrow 0/1$. The derived public key checking algorithm takes a master key pair (mpk, msk) and a derived public key dpk as input, and outputs a bit b, where $b = 1$ means that dpk is a derived key generated from mpk and $b = 0$ means not.
- $\mathsf{DskDerive}(mpk, msk, dpk) \rightarrow dsk$. The secret key derivation algorithm takes a master key pair (mpk, msk) and a derived public key dpk as input, and outputs a derived secret key dsk.
- $\mathsf{Sign}(sk, m) \rightarrow \sigma$. The signing algorithm takes a secret key sk (a derived secret key dsk) and a message $m \in M$ as input, and outputs a signature σ.
- $\mathsf{Verify}(pk, m, \sigma) \rightarrow 0/1$. The verification algorithm takes as a public key pk (a derived public key dpk), a message m and a signature σ as input, and outputs a bit b, where $b = 1$ means that the signature is valid and $b = 0$ means not.

Definition 14 (*EUF Security of PDPKS* Liu et al. 2019b) A PDPKS scheme $\mathsf{PDPKS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DpkDerive}, \mathsf{DpkCheck}, \mathsf{DskDerive}, \mathsf{Sign}, \mathsf{Verify})$ is EUF secure, if there exists a negligible function $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda) \leq negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage is defined as follows:

$$\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda) = \Pr\left[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda) \rightarrow 1\right],$$

```
$\mathsf{G}^{\mathsf{EUF}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda)$:
1: $L_{Sig} \leftarrow \emptyset$, $L_{Dpk} \leftarrow \emptyset$,$L_{Dsk} = \emptyset$
2: $(mpk, msk) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
3: $(pk^*, m^*, \sigma^*) \leftarrow$
       $\mathcal{A}^{O_{Dpk}(\cdot), O_{Sig}(\cdot), O_{Dsk}(\cdot)}(mpk)$
4: $b_0 := \|pk^* \notin L_{Dsk}\|$
5: $b_1 := \mathsf{Verify}(pk^*, m^*, \sigma^*)$
6: $b_2 := \|(pk^*, m^*) \notin L_{Sig}\|$
7: $b_3 := \|pk^* \in L_{Dpk}\|$
8: return $\|b_0 \wedge b_1 \wedge b_2 \wedge b_3\|$

$O_{Dpk}(dpk)$:
1: $b \leftarrow \mathsf{DpkCheck}(mpk, msk, dpk)$
2: if $b = 1$
3:     $L_{Dpk} \leftarrow L_{Dpk} \cup \{dpk\}$
4: return $b$

$O_{Sig}(pk, m)$:
1: if $pk \in L_{Dpk}$
2:     $sk \leftarrow \mathsf{DskDerive}(mpk, msk, pk)$
3:     $\sigma \leftarrow \mathsf{Sign}(sk, m)$
4: if $pk \notin L_{Dpk}$
5:     return $\perp$
6: $L_{Sig} \leftarrow L_{Sig} \cup \{(pk, m)\}$
7: return $\sigma$

$O_{Dsk}(dpk)$:
1: if $dpk \in L_{Dpk}$
2:     $dsk \leftarrow \mathsf{DskDerive}(mpk, msk, dpk)$
3: if $dpk \notin L_{Dpk}$
4:     return $\perp$
5: $L_{Dsk} = L_{Dsk} \cup \{dsk\}$
6: return $dsk$
```

Fig. 8 EUF security of PDPKS

where the EUF game $\mathsf{G}^{\mathsf{EUF}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda)$ is described as in Fig. 8.

Definition 15 (*MPK-UNL Security of PDPKS* Liu et al. 2019b) A PDPKS scheme $\mathsf{PDPKS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DpkDerive}, \mathsf{DpkCheck}, \mathsf{DskDerive}, \mathsf{Sign}, \mathsf{Verify})$ is MPK-UNL secure, if there exists a negligible function $negl(\cdot)$ such that the advantage $\mathsf{Adv}^{\mathsf{MPK-UNL}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda) \le negl(\lambda)$ for any PPT adversary $\mathcal{A}$. The advantage $\mathsf{Adv}^{\mathsf{MPK-UNL}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda)$ is defined as follows:

$$\mathsf{Adv}^{\mathsf{MPK-UNL}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda) = |\Pr\left[\mathsf{G}^{\mathsf{MPK-UNL}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda) \to 1\right] - 1/2|,$$

where the UNL game $\mathsf{G}^{\mathsf{MPK-UNL}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda)$ is described as in Fig. 9.

```
$\underline{\mathsf{G}^{\mathsf{MPK-UNL}}_{\mathsf{PDPKS},\mathcal{A}}(1^\lambda)}$:
1: $L_{Dpk,0} \leftarrow \emptyset,\ L_{Dpk,1} \leftarrow \emptyset$
2: $(mpk_0, msk_0) \leftarrow \mathsf{KeyGen}(pp)$
3: $(mpk_1, msk_1) \leftarrow \mathsf{KeyGen}(pp)$
4: $b \xleftarrow{\$} \{0,1\}$
5: $dpk^* \leftarrow \mathsf{DpkDerive}(mpk_b)$
6: $dsk^* \leftarrow \mathsf{DskDerive}(mpk_b, msk_b, dpk^*)$
7: $b' \leftarrow \mathcal{A}^{O_{Dpk}(\cdot),O_{Sig}(\cdot),O_{Dsk}(\cdot)}(mpk_0, mpk_1, dpk^*)$
8: return $\|b' = b\|$

$\underline{O_{Dpk}(dpk \neq dpk^*, i \in \{0,1\})}$:
1: $t \leftarrow \mathsf{DpkCheck}(mpk_i, msk_i, dpk)$
2: if $t = 1$
3:     $L_{Dpk,i} \leftarrow L_{Dpk,i} \cup \{dpk\}$
4: return $t$

$\underline{O_{Dsk}(dpk, i \in \{0,1\})}$:
1: if $dpk \in L_{Dpk,i}$
2:     $dsk = \mathsf{DskDerive}(mpk_i, msk_i, dpk)$
3: if $dpk \notin L_{Dpk,i}$
4:     return $\perp$
5: return $dsk$

$\underline{O_{Sig}(pk, m, i \in \{0,1\})}$:
1: if $pk = dpk^*$
2:     $\sigma \leftarrow \mathsf{Sign}(dpk^*, dsk^*, m)$
3: if $pk \in L_{Dpk,i}$
4:     $sk \leftarrow \mathsf{DskDerive}(mpk_i, msk_i, pk)$
5:     $\sigma \leftarrow \mathsf{Sign}(sk, m)$
6: if $pk \neq dpk^*$ and $pk \notin L_{Dpk,i}$
7:     return $\perp$
8: return $\sigma$
```

Fig. 9 MPK-UNL Security of PDPKS

Appendix B: Key reusing key derivable signature

Algorithm 2 Key Reusing Key Derivation Signature Scheme

```
KeyGen(GP):
1: $(pk, sk, \mathcal{R}) \leftarrow \mathsf{Gen}(pp)$
2: $mpk := pk||\mathcal{R}$
3: $msk := sk$
4: return $(mpk, msk)$
DpkDerive($mpk$):
5: $m \xleftarrow{\$} M$
6: $c := \mathsf{PKE.Enc}(pk, m; G(m))$
7: $k = H(m, c)$
8: $sdpk := \mathsf{KDV.Dpk}(pk, \mathcal{R}; k)$
9: $dpk := (c, sdpk)$
10: return $dpk$
DpkCheck($mpk, msk, dpk$):
11: Parse $mpk$ as $pk$ and $\mathcal{R}$
12: Parse $dpk$ as $c$ and $sdpk$
13: $m := \mathsf{PKE.Dec}(sk, c)$
14: $k := H(m, c)$
15: $sdpk' := \mathsf{KDV.Dpk}(pk, \mathcal{R}; k)$
16: if $sdpk' \neq sdpk$
17:     return $0$
18: return $1$
DskDerive($mpk, msk, dpk$):
19: Parse $mpk$ as $pk$ and $\mathcal{R}$
20: Parse $dpk$ as $c$ and $sdpk$
21: if $\mathsf{DpkCheck}(mpk, msk, dpk) = 0$
22:     return $\perp$
23: $m := \mathsf{PKE.Dec}(sk, c)$
24: $k := H(m, c)$
25: $dsk := \mathsf{KDV.Dsk}(pk, sk, \mathcal{R}; k)$
26: return $dsk$
Sign($sk, m$):
27: $(com, \mathsf{st}) \leftarrow \mathsf{CID.P}_1(sk)$
28: $ch := H_1(com, m)$
29: $rsp \leftarrow \mathsf{CID.P}_2(\mathsf{st}, com, ch)$
30: $\sigma := (com, ch, rsp)$
31: return $\sigma$
Verify($pk, m, \sigma$):
32: Parse $pk$ as $c$ and $sdpk$ or parse
        $pk$ as $pk||\mathcal{R}$
33: $b \leftarrow \mathsf{CID.V}_2(sdpk/pk, m, \sigma)$
34: return $b$
```

Appendix C: Combined signature and key encapsulation scheme

A combined signature and key encapsulation scheme (CSK) consists of the following PPT algorithms:

- $\mathsf{CSK.Gen}(pp) \rightarrow (pk, sk)$: The key generation algorithm takes the public parameters *pp* as input and outputs a pair of public/secret keys (*pk*, *sk*).
- $\mathsf{CSK.Enc}(pk) \rightarrow (c, k)$: The encapsulation algorithm takes a public key *pk* as input and outputs a ciphertext $c \in C$ and a key $k \in K$.
- $\mathsf{CSK.Dec}(sk, c) \rightarrow k/\perp$: The decapsulation algorithm takes a secret key *sk* and a ciphertext *c* as input, and outputs a key *k* or a rejection symbol $\perp$.
- $\mathsf{CSK.Sig}(sk, m) \rightarrow \sigma$: The signing algorithm takes a secret key *sk* and a message *m* as input, and outputs a signature σ.
- $\mathsf{CSK.Ver}(pk, m, \sigma) \rightarrow 0/1$: The verification algorithm takes a public key *pk*, a message *m* and a signature σ as input, and outputs 1 or 0.

Appendix C.1: Security

We say that a CSK scheme is jointly secure if it is both IND-CCA &CMA and EUF-CMA &CCA secure. The IND-CCA &CMA denotes indistinguishability of the KEM component under an adaptive chosen ciphertext attack in the presence of an additional signing oracle. The EUF-CMA &CCA denotes existential unforgeability of the signature component under an adaptive chosen message attack in the presence of an additional decapsulation oracle (Paterson et al. 2011).

Appendix C.2: Key-privacy of CSK

There is no explicit definition of key-privacy of CSK has been proposed in prior work, we denote it by IK-CCA &CMA property, capture the indistinguishability of keys under chosen-ciphertext attack in the presence of an additional signature oracle.

Appendix C.3: A framework to construct CSK

We construct a jointly secure and key private CSK from a canonical identification protocol CID with IMP-KOA security and HVZK, and an encryption scheme PKE with IND-CPA security and IK-CPA privacy that shares the same key generation algorithm KeyGen.

Let $\mathsf{CID} = (\mathsf{CID.Gen}, \mathsf{CID.P}, \mathsf{CID.V})$, $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$, then a CSK scheme $\mathsf{CSK} = (\mathsf{CSK.Gen}, \mathsf{CSK.Enc}, \mathsf{CSK.Dec}, \mathsf{CSK.Sig}, \mathsf{CSK.Ver})$ is constructed as Algorithm 3.

Proof Based on existing results, the KEM part of the CSK scheme is IND-CCA and IK-CCA secure in the random oracle model. Assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ breaking the IND-CCA & CMA and IK-CCA & CMA security of the KR-KDS scheme. Then one can construct a PPT adversary $\mathcal{B}$, breaking the IND-CCA security of the KEM scheme. $\mathcal{B}$ can simulate O_{Sig} with lazy sampling of O_{H_1} and the algorithm Sim in the Definition 3. Since the CID is HVZK, the simulation can only be distinguished by $\mathcal{A}$ with negligible probability. Thus, we prove the Claim. □

Claim 2 *Let* CID *be an IMP-POA secure CID protocol with HVZK. Let* PKE *be a OW-CPA secure PKE scheme. Let G, H, and* H_1 *be random oracle. Then the CSK scheme* CSK *is EUF & CMA secure in the random oracle model.*

Algorithm 3 CSK

```
CSK.Gen(GP):
1:  $(pk, sk) \leftarrow \mathsf{CID/PKE.Gen(GP)}$
2:  return $(pk, sk)$
CSK.Enc(pk):
3:  $r \xleftarrow{\$} R$
4:  $c :=\leftarrow \mathsf{PKE.Enc}(pk, r; G(r))$
5:  $k := H(r, c)$
6:  return $(c, k)$
CSK.Dec(sk, c):
7:  $r \leftarrow \mathsf{PKE.Dec}(sk, c)$
8:  if $r = \perp$
9:    return $\perp$
10: end
11: $k = H(r, c)$
12: return $k$
CSK.Sig(sk, m):
13: $com \leftarrow \mathsf{CID.V}_1 pk$
14: $ch = H_1(m||com)$
15: $res \leftarrow \mathsf{CID.P}_2(sk, com, ch, m)$
16: $\sigma := (com, ch, resp)$
17: return $\sigma$
CSK.Ver(pk, m, σ):
18: if $ch = H_1(m||com) \wedge \mathsf{CID.V}_2(pk, com, ch, res) = 1$
19:   return 1
20: else
21:   return 0
22: end
```

Claim 1 *Let* CID *be an IMP-POA secure CID protocol with HVZK. Let* PKE *be an IND-CPA and IK-CPA secure PKE scheme. Let G, H, and* H_1 *be random oracle. Then the CSK scheme* CSK *is IND-CCA & CMA and IK-CCA & CMA secure in the random oracle model.*

Proof Based on existing results, the SIG part of the CSK scheme is EUF-CMA secure. Assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ breaking the EUF & CMA security of the KR-KDS scheme. Then one can construct a PPT adversary $\mathcal{B}$, breaking the EUF-CMA security of the SIG scheme. $\mathcal{B}$ can simulate O_{Dec} with lazy sampling of O_G and O_h in the Definition 3. Since the PKE is OW-CPA secure, the simulation can only be distinguished by $\mathcal{A}$ with negligible probability. Thus, we prove the Claim. □

Appendix D: The KS-KDS instantiation on the lattice

We prove our lattice-based KDV is re-linkable and leakage-resistant based on the properties of algorithms TrapGen, DeleRight, and DeleLeft (Gentry et al. 2008).

For any $dpk = \mathbf{F}$, we construct the Rel algorithm as follow: it takes as input $\mathbf{F} = [\mathbf{F_1}|\mathbf{F_2}]$, randomly chooses $\mathbf{R} \in \{-1,1\}^{m\times m}$, lets $\mathbf{B} = \mathbf{F_1}$ and $\mathbf{A} = \mathbf{F_2} - \mathbf{BR}$, then returns $(\mathbf{A}, \mathbf{B}||\mathbf{R})$. In addition, let DeleRight and DeleLeft are trapdoor delegation algorithms which are defined by Liu et al. (2020). We can construct a PPT algorithm $\mathsf{Alt} = (\mathsf{Alt}_1, \mathsf{Alt}_2)$ as follows:

- $\mathsf{Alt}_1(pp, \mathbf{A}) \to (\mathbf{B}||\mathbf{R}, \mathbf{T_B})$: it takes as inputs the public parameters pp and a public key $\mathbf{A}$. It runs $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(pp)$ and randomly chooses $\mathbf{R} \in \{-1,1\}^{m\times m}$. Then returns $(\mathbf{B}||\mathbf{R}, \mathbf{T_B})$.
- $\mathsf{Alt}_2(pp, \mathbf{A}, \mathbf{B}||\mathbf{R}, \mathbf{T_B}) \to (\mathbf{F}, \mathbf{T_F})$: it takes as input $(\mathbf{B}||\mathbf{R}, \mathbf{T_B})$ and calculates $\mathbf{F} = [\mathbf{B}|\mathbf{BR}+\mathbf{A}]$ and $\mathbf{T_F} \leftarrow \mathsf{DeleLeft}(\mathbf{B}, \mathbf{A}, \mathbf{T_B}, \mathbf{BR}+\mathbf{A}, pp)$ and returns $\mathbf{T_F}$.

Appendix E: Proofs

Appendix E.1: Theorem.1

Case 1 According to the definitions of EUF and IMP-KOA, we have

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{IMP-KOA}}_{\mathsf{CID},\mathcal{B}}(pp)\\
&\quad= \Pr[\mathsf{V}_2(spk, com^*, ch^*, rsp^*) = 1]\\
&\quad\geq \Pr[\mathsf{Verify}(mpk, m^*, \sigma^*) = 1, ch'^* = ch^*]\\
&\quad\geq \Pr[\mathsf{Verify}(mpk, m^*, \sigma^*) = 1, ch'^* = ch^*, (com^*||m^*, ch'^*) \in L_{H_1}]\\
&\quad= \Pr[ch'^* = ch^* | \mathsf{Verify}(mpk, m^*, \sigma^*) = 1, (com^*||m^*, ch'^*) \in L_{H_1}]\cdot\\
&\qquad \Pr[\mathsf{Verify}(mpk, m^*, \sigma^*) = 1, (com^*||m^*, ch'^*) \in L_{H_1}],
\end{aligned}$$

and

$$\begin{aligned}
&\Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1, pk^* = mpk]\\
&\quad\leq \Pr[\mathsf{Verify}(mpk, m^*, \sigma^*) = 1] + q_{Sig}\cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp),
\end{aligned}$$

where q_{Sig} is the number of $\mathcal{A}$ queries to O_{Sig}. If $\mathsf{Verify}(mpk, m^*, \sigma^*) = 1, ch'^* = H_1(com^*, m^*)$. Moreover,

$$\begin{aligned}
\Pr[\mathsf{Verify}(mpk, m^*, \sigma^*) = 1] = &\Pr[\mathsf{Verify}(mpk, m^*, \sigma^*) = 1,\\
&(com^*||m^*, ch'^*) \in L_{H_1}]\\
&+ \Pr[\mathsf{Verify}(mpk, m^*, \sigma^*) = 1,\\
&(com^*||m^*, ch'^*) \notin L_{H_1}].
\end{aligned}$$

Due to the randomness of O_{H_1},

$$\begin{aligned}
&\Pr[\mathsf{Verify}(mpk, m^*, \sigma^*) = 1, (com^*||m^*, ch'^*) \notin L_{H_1}]\\
&\quad= \Pr[\mathsf{Verify}(mpk, m^*, \sigma^*) = 1, H_1(com^*||m^*)\\
&\quad= ch'^*, (com^*||m^*, ch'^*) \notin L_{H_1}]\\
&\quad\leq \Pr[H_1(com^*||m^*)\\
&\quad= ch'^*, (com^*||m^*, ch'^*) \notin L_{H_1}] = \frac{1}{|CH|},
\end{aligned}$$

where $|CH|$ is the size of H_1's range. Since O_{H_1} is simulated by $\mathcal{B}$ as above,

$$\begin{aligned}
\Pr[ch = ch^* | \mathsf{Verify}(mpk, m^*, \sigma^*) = 1, (com^*||m^*, ch)\\
\in L_{H_1}] \geq \frac{1}{q_{H_1}},
\end{aligned}$$

where q_{H_1} is the number of $\mathcal{A}$'s queries to O_{H_1}.

Combining these results, we have:

$$\begin{aligned}
&\Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1, pk^* = mpk] \leq q_{H_1}\cdot \mathsf{Adv}^{\mathsf{IMP-POA}}_{\mathsf{CID},\mathcal{B}}(\mathrm{pp}) +\\
&\quad q_{Sig}\cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp) + \frac{1}{|CH|}.
\end{aligned}$$

Case 2 According to the definitions of EUF and IMP-KOA, we have

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{IMP-KOA}}_{\mathsf{CID},\mathcal{C}}(pp) = \Pr[\mathsf{V}_2(dspk, com^*, ch^*, rsp^*) = 1]\\
&\quad\geq \Pr[\mathsf{Verify}(pk^*, m^*, (com^*, ch'^*, rsp^*))) = 1, pk^* = (c^*, dspk), ch'^* = ch^*],
\end{aligned}$$

and

$$\Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1, pk^* \in L_{Dpk}] \\ \leq \Pr[\mathsf{Verify}(pk^*, m^*, \sigma^*) = 1, pk^* \in L_{Dpk}] + q_{Sig} \cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp) + \mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{CID},\mathcal{A}}(pp).$$

If $pk^* = (c^*, dspk) \in L_{Dpk}$, there exists $(m^*||c^*, k^*) \in L_H$ such that $dpk^* = \mathsf{KDV.Dpk}(spk, \mathcal{R}; k^*)$ and $c^* := \mathsf{PKE.Enc}(epk, m^*; G(m^*))$. Therefore,

$$\Pr[pk^* = (c^*, dspk) | pk^* \in L_{Dpk}] \geq \frac{1}{q_H}.$$

In addition,

$$\Pr[ch = ch^* | \mathsf{Verify}(pk^*, m^*, \sigma^*) = 1, (com^*||m^*, ch) \in L_{H_1}] \geq \frac{1}{q_{H_1}},$$

where q_H is the number of $\mathcal{A}$'s queries to O_H and q_{H_1} is the number of $\mathcal{A}$'s queries to O_{H_1}.

Similar to the analysis in *Case 1*, we can get

$$\Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1, pk^* \in L_{Dpk}] \leq q_{H_1} q_H \cdot \mathsf{Adv}^{\mathsf{IMP-POA}}_{\mathsf{CID},\mathcal{C}}(\mathrm{pp}) + \\ q_{Sig} \cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp) + \mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{CID},\mathcal{A}}(pp) + \frac{1}{|CH|},$$

Put these results together,

$$\Pr[\mathsf{G}^{\mathsf{EUF}}_{\mathsf{KS-KDS},\mathcal{A}}(pp) \to 1] \leq q_{H_1} \cdot \mathsf{Adv}^{\mathsf{IMP-POA}}_{\mathsf{CID},\mathcal{B}}(\mathrm{pp}) + q_{H_1} q_H \cdot \mathsf{Adv}^{\mathsf{IMP-POA}}_{\mathsf{CID},\mathcal{C}}(pp) + \\ 2q_{Sig} \cdot \cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}}(pp) + \mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{CID},\mathcal{A}}(pp) + \frac{2}{|CH|}.$$

Appendix E.2: Theorem.2

Proof Assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ breaking the SEUF security of the KS-KDS scheme. Then one can construct a PPT adversary $\mathcal{B}$ (or $\mathcal{C}$), breaking the IMP-KOA security of the CID scheme, in the same way as Theorem 1 except that O_H and O_{Dsk} are simulated as follows:

- $O_H(m, c)$:
  - For each $\mathcal{A}$'s query (m, c) to O_H, if $c = \mathsf{PKE.Enc}(epk, m; G(m))$, $\mathcal{B}$ runs $(k, aux) \leftarrow \mathsf{Alt}_1(spk, \mathcal{R})$ and $(sdpk, dsk) = \mathsf{Alt}_2(spk, \mathcal{R}, k, aux)$. Let $dpk := (c, sdpk)$, $L_H = L_H \cup \{(m||c, k)\}$ and $L_{Dsk'} = L_{Dsk'} \cup \{(dpk, dsk)\}$, where $L_{Dsk'}$ is initialized as a empty set at beginning. If $c \neq \mathsf{PKE.Enc}(epk, m; G(m))$ and there exists $(m||c, k) \in L_H$, it returns k. Otherwise, it randomly picks k and sets $L_H = L_H \cup \{(m||c, k)\}$. Then, it returns k
  - $\mathcal{C}$ picks $i^* \xleftarrow{\$} [1, q_H]$. For each $\mathcal{A}$'s i-th $(i \neq i^*)$ query (m, c) to O_H, $\mathcal{C}$ simulates O_H like $\mathcal{B}$. If $i = i^*$, O_H is simulated by $\mathcal{C}$ in the same way as in the previous proof of Theorem 1.
- $O_{Dsk}(dpk)$: For each query dpk, if $dpk = (c, sdpk) \in L_{Dpk}$, $\mathcal{B}/\mathcal{C}$ checks the query list L_H. If there exists $(m||c, k) \in L_H$ such that $c = \mathsf{PKE.Enc}(pk, m; G(m))$ and $sdpk = \mathsf{KDV.Dpk}(spk, \mathcal{R}; k)$, it finds $(sdpk, dsk) \in L_{Dsk'}$ and lets $L_{Dsk} = L_{Dsk} \cup \{dsk\}$, where L_{Dsk} is initialized as $\emptyset$. It returns dsk. Otherwise, it returns $\perp$.

In the above simulation, $\mathcal{B}$ and $\mathcal{C}$ use the algorithm Alt to generate valid derived key pairs in advance. Thus, they can simulates O_{Dsk} without using ssk. According to the leakage-resistance of KDV, the simulation of O_{Dsk} can be distinguished by the adversary $\mathcal{A}$ with negligible probability. Then, similar to the analysis in the previous proof of Theorem 1, we can demonstrate the the KS-KDS scheme is SEUF security in the random oracle model. ☐

Appendix E.3: Theorem.3

Proof Assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ that breaks the UNL security of the KS-KDS scheme. We can construct a PPT adversary $\mathcal{B}$, breaking the IK-CPA security of the PKE scheme, as follows:

- *Setup Phase* The IK-CPA challenger generates $(epk_0, esk_0) \leftarrow \mathsf{PKE.Gen}(pp)$ and (epk_1, esk_1)

$\leftarrow \mathsf{PKE.Gen}(pp)$. $\mathcal{B}$ receives (epk_0, epk_1) and public parameters pp. $\mathcal{B}$ extracts a deterministic polynomial-time verifiable relation $\mathcal{R} := \{(pk, sk) | (pk, sk) \leftarrow \mathsf{CID.Gen}(pp)\}$ and calculates a derived key pair $(sdpk^*, sdsk^*)$. It runs $\mathsf{Rel}(sdpk^*, \mathcal{R})$ to generate (spk_0, k_0^*) and (spk_1, k_1^*), where $spk_0 \neq spk_1$. Let $mpk_0 = spk_0 || epk_0 || \mathcal{R}$, $mpk_1 = spk_1 || epk_1 || \mathcal{R}$. It sends mpk_0 and mpk_1 to $\mathcal{A}$. The query lists L_H,L_G,L_{H_1},$L_{Dpk,i}$ and $L_{Sig,i}$ are initialized as empty sets, where $i \in \{0, 1\}$.

- *Challenge Phase* The challenger picks $b \xleftarrow{\$} \{0, 1\}$ and $\mathcal{B}$ randomly chooses $m^* \in M$.The challenge ciphertext $c^* \leftarrow \mathsf{PKE.Enc}(epk_b, m^*)$ is sent to $\mathcal{B}$. Let $dpk^* := (c^*, sdpk^*)$. $\mathcal{B}$ sends dpk^* to $\mathcal{A}$.
- *Query Phase* O_G, O_H, O_{H_1} O_{Dpk} and O_{Sig} are simulated as follows:
  - $O_G(m)$: For each $\mathcal{A}$'s query m to O_G, if there exist $(m, r) \in L_G$, it returns r. Otherwise, it randomly picks r and sets $L_G = L_G \cup \{(m, r)\}$; then, returns r.
  - $O_H(m, c)$: For each $\mathcal{A}$'s query $(m,\ c)$ to O_H, if there exist $(m||c, k) \in L_H$, it returns k. Otherwise, it randomly picks k and sets $L_H = L_H \cup \{(m||c, k)\}$; then, returns r.
  - $O_{H_1}(com, m)$: For each $\mathcal{A}$'s query $(com,\ m)$ to O_{H_1}, if there exist $(com||m,\ ch) \in L_H$, it returns ch. Otherwise, it randomly picks ch and sets $L_{H_1} = L_{H_1} \cup \{(com, m)\}$; then, returns ch.
  - $O_{Dpk}(dpk \neq dpk^*, i)$: $\mathcal{B}$ parse dpk as c and $sdpk$. If there exists $(m||c, k) \in L_H$ such that $\mathsf{PKE.Enc}(epk_i, m;\ G(m)) = c$ and $\mathsf{KDV.Dpk}(spk_i, \mathcal{R};\ k) = dpk$, it sets $L_{Dpk,i} := L_{Dpk} \cup \{dpk\}$ and returns 1. Otherwise, return 0.
  - $O_{Sig}(pk, m, i)$: If $pk = (c, sdpk) \in L_{Dpk,i}$ or $pk = mpk_i$, $\mathcal{B}$ keeps running $(com, ch, rsp) \leftarrow \mathsf{Sim}(sdpk_i)$ or $(com, ch, rsp) \leftarrow \mathsf{Sim}(spk_i)$ respectively until there does not exist $com' \neq com$ and $ch' \neq ch$ such that $(com'||m, ch) \in L_{H_1}$ and $(com||m, ch') \in L_{H_1}$. It returns $\sigma := (com, ch, rsp)$ and lets $L_{H_1} = L_{H_1} \cup \{(com||m, ch)\}$ and $L_{Sig} = L_{Sig,i} \cup \{(pk, m)\}$. Otherwise, it returns $\perp$.
- *Output Phase.* When $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs b'.

Due to the re-linkability of KDV, spk_i and spk can be distinguished by $\mathcal{A}$ with probability at most $\mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{CID},\mathcal{A}}(pp)$, where $(spk, ssk) \leftarrow \mathsf{CID.Gen}(pp)$. The simulation in *Challenge Phase* is perfect if $\mathcal{A}$ does not query m^* to O_G and $m^*||c^*$ to O_H. We define that $\mathcal{A}$ query m^* to O_G or $m^*||c^*$ to O_H as the event Event. Then we can construct a PPT adversary $\mathcal{C}$ attacking OW-CPA security of PKE such that

$$(q_G + q_H) \cdot \mathsf{Adv}^{\mathsf{OW-CPA}}_{\mathsf{PKE},\mathcal{C}}(pp) \geq \Pr[\mathsf{Event}],$$

where q_H is the number of $\mathcal{A}$'s queries to O_H and q_G is the number of queries to O_G. Using the same proof strategy for Theorem 1, we can get

$$\begin{aligned}\mathsf{Adv}^{\mathsf{UNL}}_{\mathsf{KS-KDS},\mathcal{A}}(\mathsf{GP}) &\leq (q_G + q_H) \cdot \mathsf{Adv}^{\mathsf{OW-CPA}}_{\mathsf{PKE},\mathcal{C}}(\mathsf{GP}) \\ &+ \mathsf{Adv}^{\mathsf{IK-CPA}}_{\mathsf{PKE},\mathcal{B}} + q_{Sig} \cdot \mathsf{Adv}^{\mathsf{HVZK}}_{\mathsf{CID},\mathcal{A}} + 2\mathsf{Adv}^{\mathsf{RLink}}_{\mathsf{CID},\mathcal{A}}(pp),\end{aligned}$$

where q_{Sig} is the number of $\mathcal{A}$'s queries to O_{Sig}. □

Appendix E.4: Theorem.4

Proof Assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ breaking the SUNL security of the KS-KDS scheme. Then one can construct a PPT adversary $\mathcal{B}$, breaking the IK-CPA security of the CID scheme, in the same way as Theorem 3 except that O_H and O_{Dsk} are simulated as follows:

- $O_H(m, c)$: For each $\mathcal{A}$'s query $(m,\ c)$ to O_H, if $c = \mathsf{PKE.Enc}(epk_i, m;\ G(m))$, $\mathcal{B}$ runs $(k, aux) \leftarrow \mathsf{Alt}_1(spk_i, \mathcal{R})$ and $(sdpk, dsk) = \mathsf{Alt}_2(spk_i, \mathcal{R}, k, aux)$. Let $dpk := (c, sdpk)$, $L_H = L_H \cup \{(m||c, k)\}$ and $L_{Dsk',i} = L_{Dsk',i} \cup \{(dpk, dsk)\}$, where $L_{Dsk',i}$ $(i \in \{0, 1\})$ is initialized as a empty set at beginning.
- $O_{Dsk}(dpk, i)$: For each query (dpk, i), if $dpk = (c, sdpk) \in L_{Dpk,i}$, $\mathcal{B}$ checks the query list L_H. If there exists $(m||c, k) \in L_H$ such that $c = \mathsf{PKE.Enc}(epk_i, m;\ G(m))$ and $sdpk = \mathsf{KDV.Dpk}(spk_i, \mathcal{R};\ k)$, it finds $(sdpk, dsk) \in L_{Dsk',i}$ and lets $L_{Dsk,i} = L_{Dsk,i} \cup \{dsk\}$, where $L_{Dsk,i}$ is initialized as $\emptyset$. It returns dsk. Otherwise, it returns $\perp$.

In the above simulation, $\mathcal{B}$ uses the algorithm Alt to generate valid derived key pairs in advance. Thus, it can simulates $O_{Dsk,i}$ without using ssk_i. Then, similar to the analysis in the previous proofs of Theorems 2 and 3, we can demonstrate the the KS-KDS scheme is SUNL in the random oracle model. □

Appendix E.5: Theorem.5

Proof Sketch: assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ breaking the (S)EUF security of the KR-KDS scheme. Then one can construct a PPT adversary $\mathcal{B}$ or $\mathcal{C}$, breaking the IMP-KOA security of the CID scheme. The simulations of oracles are similar to the proof of Theorem 1 (Theorem 2). Additionally, O_G and

O_H both are simulated by lazy sampling because $\mathcal{B}$ does not have the secret key of PKE. $\mathcal{B}$ also simulates O_{Dpk} in the same way as in the proof of Theorem 3. Thus, it simulates O_{Dpk} without using the secret key of PKE. Since G, H and H_1 are different random oracles, these simulations can only be distinguished with negligible probability by $\mathcal{A}$ even in the case of key reusing. Thus, we can demonstrate the KR-KDS scheme is (S)EUF secure in the random oracle model. □

Appendix E.6: Theorem.6

Proof Sketch: assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ breaking the (S)UNL security of the KR-KDS scheme. Then one can construct a PPT adversary $\mathcal{B}$ or $\mathcal{C}$, breaking the IK-CPA security of the PKE scheme. The simulations of oracles are similar to the proof of Theorem 3 (Theorem 4). Since G, H and H_1 are different random oracles, these simulations can only be distinguished with negligible probability by $\mathcal{A}$ even in the case of key reusing. Thus, we can demonstrate the KR-KDS scheme is (S)UNL secure in the random oracle model. □

Acknowledgements
We would like to thank the anonymous reviewers and editors for detailed comments and useful feedback. This work was supported by the National Key Research and Development Program of China (Grant No. 2022YFB2702701).

Author Contributions
WRD, LZY and LXH completed the drafted manuscripts of the paper. ZZF and WKP participated in problem discussions. All authors read and approved the final manuscripts.

Funding
National Key Research and Development Program of China (Grant No. 2022YFB2702701).

Declarations

Competing interests
The authors declare no competing interests.

Received: 29 December 2023 Accepted: 11 March 2024
Published online: 30 May 2024

References

Alwen J, Peikert C (2009) Generating shorter bases for hard random lattices. In: STACS. LIPIcs, vol 3, pp 75–86
Boyen X (2010) Lattice mixing and vanishing trapdoors: a framework for fully secure short signatures and more. In: Proceedings of the 13th international conference on practice and theory in public key cryptography—PKC'10. LNCS 6056, pp 499–517
Chen Y, Tang Q, Wang Y (2021) Hierarchical integrated signature and encryption:(or: Key separation vs. key reuse: enjoy the best of both worlds). In: 27th international conference on theory and application of cryptology and information security, ASIACRYPT 2021. Springer Science and Business Media Deutschland GmbH, pp 514–543
Courtois NT, Mercer R (2017) Stealth address and key management techniques in blockchain systems. In: Proceedings of the 3rd international conference on information systems security and privacy—volume 1: ICISSP, pp 559–566
Cramer R, Shoup V (1998) A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack
Degabriele JP, Lehmann A, Paterson KG, Smart NP, Strefler M (2012) On the joint security of encryption and signature in EMV. In: Cryptographers' track at the RSA conference. Springer, pp 116–135
Feng C, Tan L, Xiao H, Yu K, Qi X, Wen Z, Jiang Y (2020) PDKSAP: perfected double-key stealth address protocol without temporary key leakage in blockchain. In: 2020 IEEE/CIC international conference on communications in China (ICCC Workshops). IEEE, pp 151–155
Fuchsbauer G, Plouviez A, Seurin Y (2020) Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, pp 63–95
Gentry C, Peikert C, Vaikuntanathan V (2008) Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on theory of computing, pp 197–206
Gilad Y, Hemo R, Micali S, Vlachos G, Zeldovich N (2017) Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th symposium on operating systems principles, pp 51–68
Haber S, Pinkas B (2001) Securely combining public-key cryptosystems. In: Proceedings of the 8th ACM conference on computer and communications security, pp 215–224
Liu Z, Nguyen K, Yang G, Wang H, Wong DS (2019a) A lattice-based linkable ring signature supporting stealth addresses. In: European symposium on research in computer security. Springer, pp 726–746
Liu Z, Yang G, Wong DS, Nguyen K, Wang H (2019b) Key-insulated and privacy-preserving signature scheme with publicly derived public key. In: 2019 IEEE European symposium on security and privacy (EuroS &P). IEEE, pp 215–230
Liu W, Liu Z, Nguyen K, Yang G, Yu Y (2020) A lattice-based key-insulated and privacy-preserving signature scheme with publicly derived public key. In: European symposium on research in computer security. Springer, pp 357–377
Lyubashevsky V (2012) Lattice signatures without trapdoors. In: EUROCRYPT. Lecture notes in computer science, vol 7237, pp 738–755
Micciancio D, Peikert C (2012) Trapdoors for lattices: simpler, tighter, faster, smaller. In: Advances in cryptology—EUROCRYPT'12. LNCS 7237. Springer, pp 700–718
Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system. Consulted
Naor M, Reingold O (1997) Number-theoretic constructions of efficient pseudo-random functions. In: 38th Annual symposium on foundations of computer science. Proceedings
Noether S, Mackenzie A (2016) Ring confidential transactions. Ledger 1:1–18
Paterson KG, Schuldt JC, Stam M, Thomson S (2011) On the joint security of encryption and signature, revisited. In: International conference on the theory and application of cryptology and information security. Springer, pp 161–178
Pointcheval D, Stern J (1996) Security proofs for signature schemes. In: EURO-CRYPT 96, pp 387–398
Regev O (2009) On lattices, learning with errors, random linear codes, and cryptography. J ACM (JACM) 56(6):1–40
Saberhagen NV (2012) CryptoNote v 1.0
Saberhagen NV (2013) CryptoNote v 2.0
Sasson EB, Chiesa A, Garman C, Green M, Miers I, Tromer E, Virza M (2014) Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE symposium on security and privacy. IEEE, pp 459–474
Tsiounis Y, Yung M (1998) On the security of ElGamal based encryption
Vasco MIG, Hess F, Steinwandt R (2008) Combined (identity-based) public key schemes. IACR Cryptol. ePrint Arch. 2008:466
Wang X, Zhu C, Liu Z (2024) A universally composable linkable ring signature supporting stealth addresses. Mathematics 12(3):491
Wood G (2014) Ethereum: a secure decentralised generalised transaction ledger. Ethereum Proj Yellow Pap 151(2014):1–32
Wood G (2016) Polkadot: vision for a heterogeneous multi-chain framework. White Pap 21(2327):4662

Wuille P, Nick J, Ruffing T (2021) BIP 340: Schnorr signatures for secp256k1. https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki

Yu G (2020) Blockchain stealth address schemes. IACR Cryptol. ePrint Arch. 2020:548

Zhu C, Wang X, Liu Z (2023) Universally composable key-insulated and privacy-preserving signature scheme with publicly derived public key. In: Inscrypt'23

Publisher's Note