You Can Sign but Not Decrypt: Hierarchical Integrated Encryption and Signature

Min Zhang[1,2,3,4,5], Binbin Tu[1,2,3,4,5], and Yu Chen[1,2,3,4,5](✉)

[1] School of Cyber Science and Technology, Shandong University, Qingdao 266237, China
{zm_min,tubinbin}@mail.sdu.edu.cn, yuchen@sdu.edu.cn
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao 266237, China
[4] Quancheng Laboratory, Jinan 250103, China
[5] Shandong Institute of Block-chain, Jinan 250101, China

Abstract. Recently, Chen et al. (ASIACRYPT 2021) introduced a notion called hierarchical integrated signature and encryption (HISE), which provides a new principle for combining public key schemes. It uses a single public key for both signature and encryption schemes, and one can derive a decryption key from the signing key but not vice versa. Whereas, they left the dual notion where the signing key can be derived from the decryption key as an open problem.

In this paper, we resolve the problem by formalizing the notion called hierarchical integrated encryption and signature (HIES). Similar to HISE, it features a unique public key for both encryption and signature components and has a two-level key derivation mechanism, but reverses the hierarchy between signing key and decryption key, i.e. one can derive a signing key from the decryption key but not vice versa. This property enables secure delegation of signing capacity in the public key reuse setting. We present a generic construction of HIES from constrained identity-based encryption. Furthermore, we instantiate our generic HIES construction and implement it. The experimental result demonstrates that our HIES scheme is comparable to the best Cartesian product combined public-key scheme in terms of efficiency, and is superior in having richer functionality as well as retaining merits of key reuse.

Keywords: Hierarchical integrated signature and encryption · Hierarchical identity-based encryption · Key delegation

1 Introduction

Combined usage of public key schemes is a practically relevant topic in the context of public key cryptography, especially combining public key encryption (PKE) and signature schemes. In many real-word applications, the two primitives are commonly used in combination to guarantee confidentiality and authenticity simultaneously, such as secure communication software (like PGP [2], WhatsApp [5]) and privacy-preserving cryptocurrency (like Zether [10], PGC [13]).

Typically, there are two principles for combining these two schemes, *key separation* and *key reuse*, each of which has its own strengths and weaknesses. Key separation, which means using two independent key pairs for two schemes, supports secure escrow[1] for both signing key and decryption key, while the key management and certificate costs[2] are doubled. Key reuse, which means using a unique key pair for both PKE and signature schemes, can reduce key management and certificate costs, but it does not support secure key escrow, and its joint security is not immediate.

Recently, Chen et al. [14] proposed a new notion called *hierarchical integrated signature and encryption* (HISE), which strikes a sweet balance between key separation and key reuse. It employs a single public key for both encryption and signature schemes, and allows one to derive a decryption key from signing key. This feature gives HISE advantages that (i) key management and certificate costs are reduced by half and (ii) secure delegation of decryption capacity is admitted.

Nevertheless, since the signing key is regarded as the master key in HISE, it is not applicable to some scenarios such as where one wants to delegate his signing capacity while retaining his decryption capacity. Chen et al. remarked that it is possible to consider a dual version of HISE, and it could be useful in scenarios where decryption capability is a first priority. However, they did not give the formal definition, construction and applications of it, and left it as an open problem. Therefore, the motivation for this work is two-fold: (i) find a proper key usage strategy for scenarios where key management costs are desired to be cheap, and signature delegation is needed; (ii) solve the open problem left in [14], and complete the key usage strategies.

1.1 Our Contributions

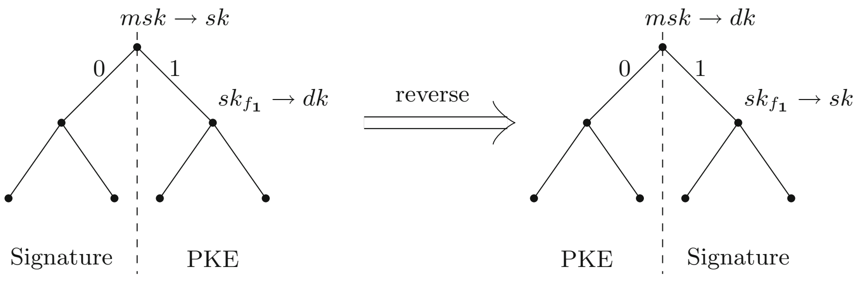
In this work, we resolve the open problem in [14] and our contributions can be summarized as follows:

Formal Definition of HIES. We start off by formalizing the definition and the joint security of the dual notion of HISE, called hierarchical integrated encryption and signature (HIES). It allows one to derive a signing key from the decryption key, such that secure delegation of signature capacity is allowed. In terms of joint security, the PKE component is IND-CCA secure even when the adversary is given the signing key and the signature component is EUF-CMA secure in the presence of an additional decryption oracle.

[1] Key escrow means that the owner delegates his decryption/signing capacity to the escrow agent simply through sharing his decryption/signing key with the agent.

[2] A public key certificate which signed by a certificate authority (CA) is an electronic document used to validate the public key. Its costs include but not limited to registration, issuing, storage, transmission, verification, and building/recurring fees.

Generic Construction from CIBE. We present a generic construction of HIES from constrained identity-based encryption (CIBE) and give a rigorous proof of its joint security.



(a) HISE from CIBE (b) HIES from CIBE

Fig. 1. HISE and HIES from CIBE

Our generic construction is inspired by HISE from CIBE. We notice that CIBE inherently implies a binary tree, where the root node is served as Private Key Generator (PKG) who possesses the master secret key, and each leaf node is viewed as a user, specified by an ID, who owns an ID-based private key. Indeed, each ID can be interpreted as identifying a unique path from root node to corresponding leaf node. We refer the reader to Sect. 2.3 for formal definition of CIBE. As for HISE from CIBE (shown as Fig. 1(a)), users each forms a CIBE binary tree, employs the master secret key of the root node as the signing key, and lets the secret key of its right child node be the decryption key, i.e. $sk_{f_{\mathbf{1}}}$, the secret key for prefix predicate $f_{\mathbf{1}}$, from which all secret keys for ID prefixed with "1" can be derived (we use $sk_{f_{\mathbf{v}}}$ to denote the constrained secret key for prefix predicate $f_{\mathbf{v}}$, where $f_{\mathbf{v}}(\text{ID}) = 1$ iff ID prefixed with $\mathbf{v}$). Thus, the whole tree is divided into two parts. The left one containing IDs prefixed with "0" is used for PKE component and the right one containing IDs prefixed with "1" is used for signature component. Based on above observations, we naturally get a construction of HIES by switching the roles the two secret keys play (shown in Fig. 1(b)).

Extensions. We propose three extensions with different purposes. The first one is for flexible delegation, with which the user is able to delegate his/her decryption and signing capacities separately to different entities. It is actually the combination of HISE and HIES. The second is for limited delegation, with which the user can limit the decryption or signing capacity given to the escrow. The last one is for finegrained delegation, which is designed to generate keys labeled by time or identifier information. We believe these extensions is useful in scenarios where delegation is not straightforward.

Applications. We give several scenarios where HIES is useful. *The first one* is a concern reported in [7]. In a confidential payment system like Zether [10] and PGC [13], which currently is equipped with a key reuse mechanism, if the signing key needs to be revoked or rotated, then all encrypted assets of an account need to be transferred to a new account, which leads to high overhead. While there is no such trouble if HIES is used, in which one can derive time labeled signing keys. *The second one* is the following scenario discussed in Viafirma [4]. In a company, the president needs to deal with multifarious documents everyday, including but not limited to commercial contracts, applications for the procurement of goods and so on. It is quite necessary to delegate his signing right to assistant presidents so that they can help settle documents which are less important. Meanwhile, the president may require keeping the decryption key secret for the security of some confidential business documents. Many similar scenarios where signature delegation is needed widely appear in other institutions, such as schools and government departments [1,3].

Indeed, signature delegation, also known as *Proxy Signature* which was first introduced by Mambo et al. [28], has numerous applications, such as distributed systems [29], mobile agent [26] and electronic commerce [16]. Various schemes and extensions of it were proposed during the last few decades [8,12,22–24,34]. In contrast to these schemes, HIES not only considers delegating signing right, but also combines an additional PKE scheme without increasing the size of public key, yielding a scheme with richer functionality. In general, HIES is suitable for the scenarios where low key management costs are desired, while the signing key is not permanent, or the signature delegation is needed.

Instantiation and Implementation. We instantiate our HIES and implement it with 128-bit security. The performance of our HIES scheme is comparable to the best Cartesian product combined public-key scheme [30] in terms of efficiency, and is superior in having richer functionality as well as retaining merits of key reuse.

1.2 Related Works

Here we briefly review the works related to combined usage of public key schemes.

Key Separation. It is the folklore principle for combining PKE and signature schemes, which indicates using two independent key pairs for two public key schemes. Paterson et al. formalized it via the notion of "Cartesian product" combined public key scheme (CP-CPK) [30], which means using arbitrary encryption and signature schemes as components, and combining two key pairs into one simply through concatenating the public/private keys of the component schemes. They pointed out that CP-CPK provides a benchmark by which other constructions can be judged, so we use it as a baseline.

Key Reuse. The first work to formally study the security of key reuse was by Haber and Pinkas [20]. They introduced the concept of a combined public key (CPK) scheme, where an encryption scheme and a signature scheme are combined. CPK preserves the existing algorithms of sign, verify, encrypt and decrypt, while the two key generation algorithms are modified into a single algorithm, which outputs two key pairs for PKE and signature components respectively, with the key pairs no longer necessarily being independent. In addition, they formalized the joint security of CPK scheme, i.e., the encryption component is IND-CCA secure even in the presence of an additional signing oracle, and the signature component is EUF-CMA secure even in the presence of an additional decryption oracle. Integrated signature and encryption (ISE) is an extreme case of CPK. It uses an identical key pair for both PKE and signature components, which in turn makes it not support key delegation. In subsequent works, both Coron et al. [15] and Komano et al. [25] considered building ISE from trapdoor permutations in the random oracle model. Paterson et al. [30] gave an ISE construction from identity-based encryption.

Hierarchical Integrated Signature and Encryption. It is a new notion presented by Chen et al. in [14]. HISE employs a unique public key for both PKE and signature components, and serves the signing key as the master secret key from which the decryption key can be derived. Thus, HIES supports secure delegation of decryption power and achieves stronger joint security than ISE, that is, the encryption component is IND-CCA secure even in the presence of an additional signing oracle, while the signature component is EUF-CMA secure even in the presence of the decryption key.

Our notion is dual to HISE, where the hierarchy between signing key and decryption key is reversed. It completes the last piece of the key usage strategy puzzle, as shown in Fig. 2. We use index e to indicate keys for PKE component and s to signature component.

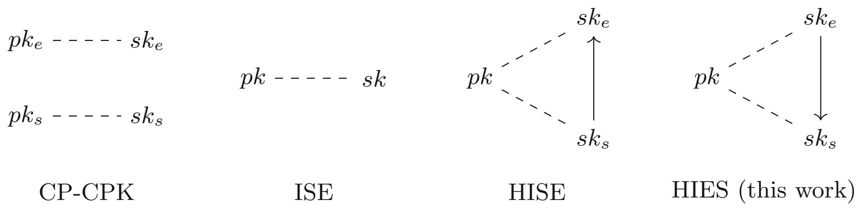


Fig. 2. Different key usage strategies

2 Preliminaries

Notations. We use $m \xleftarrow{R} M$ to denote that m is sampled uniformly at random from a set M and $y \leftarrow A(x)$ to denote the algorithm A that on input x outputs y. We use the abbreviation PPT to indicate probabilistic polynomial-time. We denote by $\mathsf{negl}(\lambda)$ a negligible function in λ. Let tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ denote the descriptions of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of the same prime order p, and g_1, g_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is the bilinear map.

2.1 Public Key Encryption

Definition 1. *A public key encryption (PKE) scheme consists of four polynomial-time algorithms:*

– $\mathsf{Setup}(1^\lambda)$*: on input a security parameter* λ*, outputs public parameters* pp*, including the descriptions of plaintext space* $\mathcal{M}$*, ciphertext space* $\mathcal{C}$*, and randomness space* $\mathcal{R}$*.*
– $\mathsf{KeyGen}(pp)$*: on input public parameters* pp*, outputs a public encryption key* ek *and a secret decryption key* dk*.*
– $\mathsf{Enc}(ek, m)$*: on input an encryption key* ek *and a plaintext* m*, outputs a ciphertext* c*.*
– $\mathsf{Dec}(dk, c)$*: on input a decryption key* dk *and a ciphertext* c*, outputs a plaintext* m *or a special reject symbol* $\perp$ *denoting failure. This algorithm is typically deterministic.*

Correctness. For any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(ek, dk) \leftarrow \mathsf{KeyGen}(pp)$, any $m \in \mathcal{M}$ and any $c \leftarrow \mathsf{Enc}(ek, m)$, it holds that $\mathsf{Dec}(dk, c) = m$.

Security. Let $\mathcal{O}_{\mathsf{dec}}$ be a decryption oracle that on input a ciphertext, outputs a plaintext. A public key encryption scheme is IND-CCA secure if for any PPT adversary $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\lambda)$ such that:

$$\Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (ek, dk) \leftarrow \mathsf{KeyGen}(pp); \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}}(pp, ek); \\ \beta \xleftarrow{\mathrm{R}} \{0,1\}, c^* \leftarrow \mathsf{Enc}(ek, m_\beta); \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}}(c^*); \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

$\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{dec}}$ for c^* in the guess stage. The IND-CPA security can be defined similarly by denying the decryption oracle.

2.2 Digital Signature

Definition 2. *A digital signature scheme consists of four polynomial-time algorithms:*

– $\mathsf{Setup}(1^\lambda)$*: on input the security parameter* λ*, outputs public parameters* pp*, including the descriptions of message space* $\mathcal{M}$ *and signature space* Σ*.*
– $\mathsf{KeyGen}(pp)$*: on input* pp*, outputs a public verification key* vk *and a secret signing key* sk*.*
– $\mathsf{Sign}(sk, m)$*: on input a signing key* sk *and a message* m*, outputs a signature* σ*.*
– $\mathsf{Vrfy}(vk, m, \sigma)$*: on input a verification key* vk*, a message* m*, and a signature* σ*, outputs a bit* b*, with* $b = 1$ *meaning valid and* $b = 0$ *meaning invalid.*

Correctness. For any $(vk, sk) \leftarrow \mathsf{KeyGen}(pp)$, any $m \in \mathcal{M}$ and any $\sigma \leftarrow \mathsf{Sign}(sk, m)$, it holds that $\mathsf{Vrfy}(pk, m, \sigma) = 1$.

Security. Let $\mathcal{O}_{\mathsf{sign}}$ be a signing oracle that on input a message, outputs a signature. A digital signature scheme is EUF-CMA secure if for any PPT adversary $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\lambda)$ such that:

$$\Pr\left[\begin{array}{c} \mathsf{Vrfy}(vk, m^*, \sigma^*) = 1 \\ \wedge\ m^* \notin \mathcal{Q} \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (vk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}}(pp, vk); \end{array}\right] \leq \mathsf{negl}(\lambda).$$

The set $\mathcal{Q}$ records queries to $\mathcal{O}_{\mathsf{sign}}$. The *strong* EUF-CMA security can be defined similarly by asking $\mathcal{A}$ to output a fresh valid message-signature tuple. The one-time signature can also be defined similarly by restricting the adversary to access $\mathcal{O}_{\mathsf{sign}}$ only once.

2.3 Constrained Identity-Based Encryption

We recall the definition of constrained IBE introduced by Chen et al. [14] below.

Definition 3. *A constrained identity-based encryption (CIBE) scheme consists of seven polynomial-time algorithms:*

- $\mathsf{Setup}(1^\lambda)$*: on input a security parameter λ, outputs public parameters pp. Let $\mathcal{F}$ be a family of predicates over identity space $\mathcal{I}$.*
- $\mathsf{KeyGen}(pp)$*: on input public parameters pp, outputs a master public key mpk and a master secret key msk.*
- $\mathsf{Extract}(msk, id)$*: on input a master secret key msk and an identity $id \in \mathcal{I}$, outputs a user secret key sk_{id}.*
- $\mathsf{Constrain}(msk, f)$*: on input a master secret key msk and a predicate $f \in \mathcal{F}$, outputs a constrained secret key sk_f.*
- $\mathsf{Derive}(sk_f, id)$*: on input a constrained secret key sk_f and an identity $id \in \mathcal{I}$, outputs a user secret key sk_{id} if $f(id) = 1$ or $\perp$ otherwise.*
- $\mathsf{Enc}(mpk, id, m)$*: on input mpk, an identity $id \in \mathcal{I}$, and a message m, outputs a ciphertext c.*
- $\mathsf{Dec}(sk_{id}, c)$*: on input a user secret key sk_{id} and a ciphertext c, outputs a message m or a special reject symbol $\perp$ denoting failure.*

Correctness. For any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$, any identity $id \in \mathcal{I}$, any $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$, any message m, and any $c \leftarrow \mathsf{Enc}(mpk, id, m)$, it always holds that $\mathsf{Dec}(sk_{id}, c) = m$. Besides, for any $f \in \mathcal{F}$ such that $f(id) = 1$, the outputs of $\mathsf{Extract}(msk, id)$ and $\mathsf{Derive}(sk_f, id)$ have the same distribution.

Security. Let $\mathcal{O}_{\mathsf{extract}}$ be an oracle of $\mathsf{Extract}$ that on input an identity id outputs sk_{id}. Let $\mathcal{O}_{\mathsf{constrain}}$ be an oracle of $\mathsf{Constrain}$ that on input a predicate f outputs sk_f. A CIBE scheme is IND-CPA secure, if for all PPT adversary $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\lambda)$ suth that:

$$\Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (mpk, msk) \leftarrow \mathsf{KeyGen}(pp); \\ (id^*, (m_0, m_1)) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{extract}}, \mathcal{O}_{\mathsf{constrain}}}(pp, mpk); \\ \beta \xleftarrow{R} \{0,1\}, c^* \leftarrow \mathsf{Enc}(mpk, id^*, m_\beta); \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{extract}}, \mathcal{O}_{\mathsf{constrain}}}(c^*); \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

$\mathcal{A}$ is not allowed to query the $\mathcal{O}_{\mathsf{extract}}$ with id^* or query the $\mathcal{O}_{\mathsf{constrain}}$ with f such that $f(id^*) = 1$. Meanwhile, two weaker security notions can be defined similarly. One is OW-CPA security, in which the adversary is required to recover the plaintext from a random ciphertext. The other is selective-identity IND-CPA security, in which the adversary must commit ahead of time (non-adaptively) to the identity it intends to attack before seeing the mpk.

3 Hierarchical Integrated Encryption and Signature

3.1 Definition of HIES

As mentioned in the introduction, HIES allows one to derive the signing key from the decryption key, which is opposite to HISE. Next, we give a self-contained description of the formal definition of HIES.

Definition 4. *A hierarchical integrated encryption and signature (HIES) scheme is defined by seven polynomial-time algorithms:*

- $\mathsf{Setup}(1^\lambda)$*: on input a security parameter* λ*, outputs public parameters* pp *including the description of plaintext space* $\mathcal{M}$ *and message space* $\widetilde{\mathcal{M}}$.
- $\mathsf{KeyGen}(pp)$*: on input public parameters* pp*, outputs a public key* pk *and a decryption key* dk*. Here,* dk *serves as a master secret key, which can be used to derive signing key.*
- $\mathsf{Derive}(dk)$*: on input a decryption key* dk*, outputs a signing key* sk.
- $\mathsf{Enc}(pk, m)$*: on input a public key* pk *and a plaintext* $m \in \mathcal{M}$*, outputs a ciphertext* c.
- $\mathsf{Dec}(dk, c)$*: on input a decryption key* dk *and a ciphertext* c*, outputs a plaintext* m *or a special reject symbol* $\perp$ *denoting failure.*
- $\mathsf{Sign}(sk, \widetilde{m})$*: on input a signing key* sk *and a message* $\widetilde{m} \in \widetilde{\mathcal{M}}$*, outputs a signature* σ.
- $\mathsf{Vrfy}(pk, \widetilde{m}, \sigma)$*: on input a public key* pk*, a message* $\widetilde{m}$*, and a signature* σ*, outputs a bit* b*, with* $b = 1$ *meaning valid and* $b = 0$ *meaning invalid.*

Correctness. The correctness of HIES is divided into two parts, the correctness of PKE and signature components: (i) the PKE component satisfies correctness if for any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(pk, dk) \leftarrow \mathsf{KeyGen}(pp)$, any $m \in \mathcal{M}$ and any $c \leftarrow \mathsf{Enc}(pk, m)$, it holds that $\mathsf{Dec}(dk, c) = m$; (ii) the signature component satisfies correctness if for any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(pk, dk) \leftarrow \mathsf{KeyGen}(pp)$, any $sk \leftarrow \mathsf{Derive}(dk)$, any $\widetilde{m} \in \widetilde{\mathcal{M}}$ and any $\sigma \leftarrow \mathsf{Sign}(sk, \widetilde{m})$, it holds that $\mathsf{Very}(pk, \widetilde{m}, \sigma) = 1$.

Security. (Joint security) The joint security for HIES needs to be considered from two aspects as well. The PKE component requires to satisfy IND-CCA security in the presence of a signing key and the signature component requires to satisfy EUF-CMA security in the presence of a decryption oracle. Let $\mathcal{O}_{\mathsf{dec}}$ be the decryption oracle and $\mathcal{O}_{\mathsf{sign}}$ be the signing oracle. The formal security notion is defined as below.

Definition 5. *HIES is joint secure if its encryption and signature components satisfy the following security notions:*

(i) The PKE component is IND-CCA secure in the presence of a signing key, if for any PPT adversary $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\lambda)$ *such that:*

$$\Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, dk) \leftarrow \mathsf{KeyGen}(pp); \\ sk \leftarrow \mathsf{Derive}(dk); \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}}(pp, pk, sk); \\ \beta \xleftarrow{R} \{0,1\}, c^* \leftarrow \mathsf{Enc}(pk, m_\beta); \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}}(c^*); \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

$\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{dec}}$ with c^* in the guess stage.

(ii) The signature component is EUF-CMA secure in the presence of a decryption oracle, if for all PPT adversary $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\lambda)$ *such that:*

$$\Pr\left[\begin{array}{c} \mathsf{Vrfy}(pk, m^*, \sigma^*) = 1 \\ \wedge m^* \notin \mathcal{Q} \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, dk) \leftarrow \mathsf{KeyGen}(pp); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}, \mathcal{O}_{\mathsf{sign}}}(pp, pk); \end{array}\right] \leq \mathsf{negl}(\lambda).$$

The set $\mathcal{Q}$ records queries to $\mathcal{O}_{\mathsf{sign}}$.

3.2 HIES from Constrained IBE

In this section, we give a generic construction of HIES from constrained identity-based encryption. Let CIBE be a constrained IBE scheme and OTS be a strong one-time signature scheme, then an HIES scheme can be created as Fig. 3. We assume the identity space of CIBE is $\mathcal{I} = \{0,1\}^{\ell+1}$, and the verification space of OTS is $\{0,1\}^\ell$.

The correctness of the scheme follows directly from the correctness of CIBE and OTS. The joint security of the HIES scheme is formalized as below.

```
Setup(1^λ) :
    $pp_{\text{cibe}} \leftarrow \text{CIBE.}\mathsf{Setup}(1^\lambda)$
    $pp_{\text{ots}} \leftarrow \text{OTS.}\mathsf{Setup}(1^\lambda)$
    Return $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$

KeyGen(pp) :
    Parse $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$
    $(mpk, msk) \leftarrow \text{CIBE.}\mathsf{KeyGen}(pp_{\text{cibe}})$
    Return $(pk, dk) = (mpk, msk)$

Derive(dk) :
    Parse $dk = msk$
    $sk_{f_1} \leftarrow \text{CIBE.}\mathsf{Constrain}(msk, f_1)$
    $(f_1(id) = 1 \text{ iff } id[1] = 1)$
    Return $sk = sk_{f_1}$

Enc(pk, m) :
    Parse $pk = mpk$
    $(ovk, osk) \leftarrow \text{OTS.}\mathsf{KeyGen}(pp_{\text{ots}})$
    Set $id = 0||ovk$
    $c_{\text{cibe}} \leftarrow \text{CIBE.}\mathsf{Enc}(mpk, id, m)$
    $\sigma_{\text{ots}} \leftarrow \text{OTS.}\mathsf{Sign}(osk, c_{\text{cibe}})$
    Return $c = (ovk, c_{\text{cibe}}, \sigma_{\text{ots}})$

Dec(dk, c) :
    Parse $c = (ovk, c_{\text{cibe}}, \sigma_{\text{ots}})$
    If $\text{OTS.}\mathsf{Vrfy}(ovk, c_{\text{cibe}}, \sigma_{\text{ots}}) \neq 1$
    return $\perp$
    Parse $dk = msk$
    Set $id = 0||ovk$
    $sk_{id} \leftarrow \text{CIBE.}\mathsf{Extract}(msk, id)$
    $m \leftarrow \text{CIBE.}\mathsf{Dec}(sk_{id}, c_{\text{cibe}})$
    Return $m$

Sign(sk, $\widetilde{m}$) :
    Parse $sk = sk_{f_1}$
    Set $id = 1||\widetilde{m}$
    $sk_{id} \leftarrow \text{CIBE.}\mathsf{Derive}(sk_{f_1}, id)$
    Return $\sigma = sk_{id}$

Vrfy(pk, $\widetilde{m}$, σ) :
    Parse $pk = mpk$ and $\sigma = sk_{id}$
    Set $id = 1||\widetilde{m}$
    $m \xleftarrow{R} M$
    $c_{\text{cibe}} \leftarrow \text{CIBE.}\mathsf{Enc}(mpk, id, m)$
    If $\text{CIBE.}\mathsf{Dec}(c_{\text{cibe}}, sk_{id}) = m$
    Return 1, else return 0
```

Fig. 3. A generic construction of HIES from CIBE

Theorem 1. *Assume CIBE satisfies IND-CPA security and OTS satisfies strong EUF-CMA security, then the HIES scheme constructed as Fig. 3 satisfies joint security.*

This theorem comes straightforwardly from two lemmas.

Lemma 1. *If CIBE scheme is OW-CPA secure, then the signature component is EUF-CMA secure in the presence of the decryption oracle.*

Proof. If there exists a PPT adversary $\mathcal{A}$ against the signature component, we can construct a PPT adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine and attacks the CIBE. $\mathcal{B}$ is given public parameters pp_{cibe}, public key mpk and access to $\mathcal{O}_{\mathsf{extract}}$ and $\mathcal{O}_{\mathsf{constrain}}$ by its own challenger $\mathcal{CH}_{\text{cibe}}$, then it simulates $\mathcal{A}$'s challenger $\mathcal{CH}_{\text{sign}}$ as below.

- Setup: $\mathcal{B}$ runs $pp_{\text{ots}} \leftarrow \text{OTS.}\mathsf{Setup}(1^\lambda)$, sets $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$ and $pk = mpk$, then sends (pp, pk) to $\mathcal{A}$.
- Signing query: when $\mathcal{A}$ requests a signature on message $\widetilde{m}$, $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{extract}}$ with identity $id = 1||\widetilde{m}$ to obtain sk_{id}, outputs $\sigma = sk_{id}$.
- Decryption query: when $\mathcal{A}$ requests the plaintext of a ciphertext c, $\mathcal{B}$ first parses c as $(ovk, c_{\text{cibe}}, \sigma_{\text{ots}})$, then checks whether $\text{OTS.}\mathsf{Vrfy}(ovk, c_{\text{cibe}}, \sigma_{\text{ots}}) =$

1, and returns $\perp$ if not; else it queries $\mathcal{O}_{\mathsf{extract}}$ for $id = 0||ovk$ to obtain sk_{id} and returns the plaintext $m \leftarrow$ CIBE.$\mathsf{Dec}(sk_{id}, c_{\text{cibe}})$ to $\mathcal{A}$.
- Forgery: when $\mathcal{A}$ outputs a forged message-signature pair $(\widetilde{m}^*, \sigma^*)$, $\mathcal{B}$ first submits $id^* = 1||\widetilde{m}$ as the target identity to $\mathcal{CH}_{\text{cibe}}$ and receives back $c^*_{cibe} \leftarrow$ CIBE.$\mathsf{Enc}(mpk, id^*, m)$ for a random plaintext $m \xleftarrow{\text{R}} \mathcal{M}$, then it parses $\sigma^* = sk_{id^*}$, and computes $m' \leftarrow$ CIBE.$\mathsf{Dec}(sk_{id^*}, c^*_{cibe})$. $\mathcal{B}$ wins if $m' = m$.

The view of $\mathcal{A}$ when it interacts with $\mathcal{B}$ is identical to the view of $\mathcal{A}$ interacting with a real challenger, which implies the simulation of $\mathcal{B}$ is perfect. If no PPT adversary $\mathcal{B}$ has non-negligible probability to break the OW-CPA security of the CIBE scheme, then no PPT adversary $\mathcal{A}$ has non-negligible probability to break the EUF-CMA security of signature component. This proves Lemma 1.

Lemma 2. *If the OTS scheme satisfies strong EUF-CMA security and the CIBE scheme satisfies selective-identity IND-CPA security, then the encryption component PKE satisfies IND-CCA security even in the presence of signing key.*

Proof. Consider following games. Let $\mathcal{A}$ be an adversary against the PKE component and S_i be the event that $\mathcal{A}$ wins in Game i.

Game 0. This is the standard IND-CCA security experiment for PKE component in the presence of a signing key, $\mathcal{CH}_{\text{pke}}$ interacts with $\mathcal{A}$ as below.

- Setup: $\mathcal{CH}_{\text{pke}}$ runs $pp_{\text{cibe}} \leftarrow$ CIBE.$\mathsf{Setup}(1^\lambda)$ and $pp_{\text{ots}} \leftarrow$ OTS.$\mathsf{Setup}(1^\lambda)$, sets $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$, then runs $(mpk, msk) \leftarrow$ CIBE.$\mathsf{KeyGen}(pp_{\text{cibe}})$, sets $pk = mpk$ and $dk = msk$, runs $sk \leftarrow \mathsf{Derive}(dk)$, and gives (pp, pk, sk) to $\mathcal{A}$.
- Decryption query: upon receiving a ciphertext c, $\mathcal{CH}_{\text{pke}}$ first parses $c = (ovk, c_{\text{cibe}}, \sigma)$, checks if OTS.$\mathsf{Vrfy}(ovk, c_{\text{cibe}}, \sigma) = 1$, outputs $\perp$ if not; else sets $id = 0||ovk$, parses $dk = msk$, runs $sk_{id} \leftarrow$ CIBE.$\mathsf{Extract}(msk, id)$ and outputs $m \leftarrow$ CIBE.$\mathsf{Dec}(sk_{id}, c_{\text{cibe}})$.
- Challenge: $\mathcal{A}$ outputs a pair of messages (m_0,m_1). $\mathcal{CH}_{\text{pke}}$ chooses a random bit $b \xleftarrow{\text{R}} \{0,1\}$, runs $(ovk^*, osk^*) \leftarrow$ OTS.$\mathsf{KeyGen}(pp_{\text{ots}})$, sets $id^* = 0||ovk^*$, computes $c^*_{\text{cibe}} \leftarrow$ CIBE.$\mathsf{Enc}(mpk, id^*, m_b)$, and $\sigma^* \leftarrow$ OTS.$\mathsf{Sign}(osk^*, c^*_{\text{cibe}})$, outputs $c^* = (ovk^*, c^*_{\text{cibe}}, \sigma^*)$ to $\mathcal{A}$. Then $\mathcal{A}$ can continue to query the decryption oracle, but it is not allowed to query for c^*.
- Guess: Eventually, $\mathcal{A}$ outputs a bit b'. $\mathcal{A}$ wins if $b' = b$.

Game 1. Same as Game 0 except that $\mathcal{CH}_{\text{pke}}$ generates the OTS keypair $(ovk^*, osk^*) \leftarrow$ OTS.$\mathsf{KeyGen}(pp_{\text{ots}})$ in the setup stage rather than in the challenge stage. The modification is only conceptual and does not affect the advantage of $\mathcal{A}$, so we have:

$$\Pr[S_1] = \Pr[S_0] .$$

Game 2. Same as Game 1 except that the experiment directly aborts when one of following two events happens:

E_1: in phase 1, $\mathcal{A}$ queries the decryption oracle with $c = (ovk^*, c_{\text{cibe}}, \sigma)$ such that OTS.$\mathsf{Vrfy}(ovk^*, c_{\text{cibe}}, \sigma) = 1$.

E_2: in phase 2, $\mathcal{A}$ queries the decryption oracle with $c = (ovk^*, c^*_{\text{cibe}}, \sigma)$ such that OTS.$\mathsf{Vrfy}(ovk^*, c^*_{\text{cibe}}, \sigma) = 1$ and $\sigma \neq \sigma^*$.

Let E be the event that E_1 or E_2 happens, then we have (Game 1 $\wedge \neg E$) = (Game 2 $\wedge \neg E$). According to the difference lemma, we have:

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[E].$$

Actually, the two events mean a successful attack on the OTS, while the strong EUF-CMA security of OTS ensures that for any PPT $\mathcal{A}$, it holds that $\Pr[E] = \mathsf{negl}(\lambda)$.

Claim 1. *If the CIBE scheme is selective-identity IND-CPA secure, then for any PPT adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\lambda)$ such that:*

$$\left|\Pr[S_2] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

Proof. Let $\mathcal{B}$ be an adversary against CIBE scheme. It is given public parameters pp_{cibe} and access to $\mathcal{O}_{\mathsf{extract}}$ and $\mathcal{O}_{\mathsf{constrain}}$ by its own challenger $\mathcal{CH}_{\text{cibe}}$. $\mathcal{B}$ simulates $\mathcal{A}$'s challenger as below.

- Setup: $\mathcal{B}$ runs $pp_{\text{ots}} \leftarrow$ OTS.$\mathsf{Setup}(1^\lambda)$, $(ovk^*, osk^*) \leftarrow$ OTS.$\mathsf{KeyGen}(pp_{\text{ots}})$, sets $id^* = 0||ovk^*$, then commits to id^* and sends the commitment to its own challenger $\mathcal{CH}_{\text{cibe}}$ as the target identity and receives back public key $pk = mpk$. Next, $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{constrain}}$ with f_1, and obtains the signing key $sk = sk_{f_1}$. $\mathcal{B}$ gives $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$, $pk = mpk$ and $sk = sk_{f_1}$ to $\mathcal{A}$.
- Decryption query: When $\mathcal{A}$ queries for a ciphertext $c = (ovk, c_{\text{cibe}}, \sigma)$, $\mathcal{B}$ first checks whether OTS.$\mathsf{Vrfy}(ovk, c_{\text{cibe}}, \sigma) = 1$, if not, outputs $\perp$; else if $ovk = ovk^*$ which means event E_1 happens, $\mathcal{B}$ aborts; otherwise it must have $ovk \neq ovk^*$, $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{extract}}$ with $id = 0||ovk$ to obtain sk_{id}, and outputs $m \leftarrow$ CIBE.$\mathsf{Dec}(sk_{id}, c_{\text{cibe}})$.
- Challenge: $\mathcal{A}$ submits two messages (m_0, m_1) to $\mathcal{B}$. $\mathcal{B}$ sends the two messages to its own challenger and receives back a ciphertext c^*_{cibe} which is a ciphertext of m_b under the target identity $id^* = 0||ovk^*$. $\mathcal{B}$ proceeds to compute a signature σ^* on c^*_{cibe}, then sends $c^* = (ovk^*, c^*_{\text{cibe}}, \sigma^*)$ to $\mathcal{A}$.
- Guess: Upon receiving c^*, $\mathcal{A}$ continues to query decryption oracle but is not allowed to query it with c^*. If E_2 happens, $\mathcal{B}$ aborts. Else it answers the query as before. Finally, $\mathcal{A}$ outputs b', and $\mathcal{B}$ uses b' as its own guess.

The view of $\mathcal{A}$ when it interacts with $\mathcal{B}$ is identical to the view of $\mathcal{A}$ in experiment Game 2 which implies the simulation of $\mathcal{B}$ is perfect. Due to the selective-identity IND-CPA security of CIBE, the advantage of $\mathcal{A}$ wins in Game 2 is negligible. This proves Claim 1.

Therefore, the proof of Lemma 2 is completed.

Remark 1. We strengthen PKE component to IND-CCA security via the Canetti-Halevi-Katz (CHK) transform [11] with the help of a one-time signature. To enhance the efficiency, we can get rid of OTS and use an $id_0 = 0^{\ell+1}$ as a fixed target identity for encryption, then apply the Fujisaki-Okamoto transformation [17] to achieve the IND-CCA security in the random oracle model.

4 Further Discussion

In this section, we discuss three simple extensions of HIES for different delegation purposes and each of them is in the public key reuse setting. The key observation is that the prefix predicates in a constrained IBE can be assigned different and specific meanings.

Flexible Delegation. One delegation function is insufficient sometimes, such as the cases when the president wants to give his signing right to his assistants and give his decryption right to vice president. Thus, it is attractive to give a more flexible notion that enables the secret key owner to delegate these two types of authorities to different entities. The key technique is equalizing two secret keys by deriving them both from the master secret key as shown in Fig. 4(a). It is evident that the extended version satisfies united joint security as long as the two agents are not in collusion, namely the PKE/signature component is IND-CCA secure even when the adversary is given the signing/decryption key.

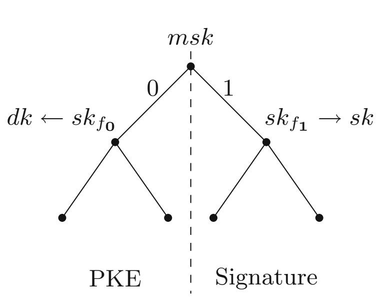
Limited Delegation. In the signature proxy function introduced by Mambo et al. [27], the full delegation (giving the full original secret key to the proxy signer) requests the proxy is authentic, since the proxy signer has the ability to sign any message and the proxy signature is indistinguishable from the created by the original signer. The decryption proxy suffers the similar discomfort if the decryption key is disclosed. In order to limit the decryption and signing capacity of proxy, we consider an extension which supports partial delegation. It divides the decryption/signing capacity into two parts so that the original user can retain the higher power while delegating partial power to agents as shown in Fig. 4(b).

Finegrained Delegation. Giving the prefix predicates with more specific meanings such as the ID (identifier information such as email address) of a person or the number of a department, more finegrained delegation keys can be derived.

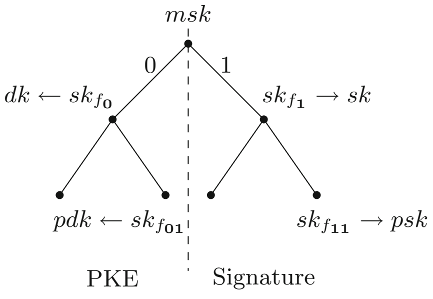
5 Instantiation and Implementation

5.1 Instantiation of HIES

Towards efficient realizations, we choose the hierarchical IBE (cf. Appendix A.1) rather than the constrained IBE to instantiate our HIES scheme, where the security can be similarly demonstrated. By choosing Boneh-Boyen two-level hierarchical IBE scheme (BB_1-IBE, cf. Appendix A.2), we instantiate our HIES scheme as below.



(a) Flexible delegation



msk
0
1
$dk \leftarrow sk_{f_0}$
$sk_{f_1} \rightarrow sk$
$pdk \leftarrow sk_{f_{01}}$
$sk_{f_{11}} \rightarrow psk$
PKE
Signature

(b) Limited delegation

Fig. 4. Extensions of HIES from constrained IBE

- $\mathsf{Setup}(1^\lambda)$: on input the security parameter λ, generates an asymmetric pairing tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$, picks two collision resistant hash functions $\mathsf{H}_j : \{0,1\}^* \to \mathbb{G}_2$ for $j = 1, 2$, sets $\mathsf{ID}_0 = 0^n$ and $\mathsf{ID}_1 = 1^n$ with $n = \Theta(\lambda)$. The public parameter $pp = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e), \mathsf{H}_1, \mathsf{H}_2, \mathsf{ID}_0, \mathsf{ID}_1)$. The plaintext space is $\mathbb{G}_T$. The message space is $\{0,1\}^*$.
- $\mathsf{KeyGen}(pp)$: on input the public parameters pp, picks a random $\alpha \in \mathbb{Z}_p$, sets $f_1 = g_1^\alpha$ and $f_2 = g_2^\alpha$, sets public key $mpk = f_1 = g_1^\alpha$ and decryption key $dk = msk = f_2 = g_2^\alpha$.
- $\mathsf{Derive}(dk)$: on input the decryption key dk, picks a random $r' \in \mathbb{Z}_p$, computes $d'_0 = dk \cdot \mathsf{H}_1(\mathsf{ID}_1)^{r'}$ and $d'_1 = g_1^{r'}$, outputs $sk = (d'_0, d'_1) \in (\mathbb{G}_2, \mathbb{G}_1)$.
- $\mathsf{Enc}(pk, m)$: on input the public key $pk = mpk$ and a plaintext $m \in \mathbb{G}_T$, firstly picks a random $s \in \mathbb{Z}_p$ and computes $A = e(f_1, g_2)^s \cdot m$, $B = g_1^s$ and $C_1 = \mathsf{H}_1(\mathsf{ID}_0)^s$, outputs $c = (A, B, C_1) \in (\mathbb{G}_T, \mathbb{G}_1, \mathbb{G}_2)$.
- $\mathsf{Dec}(dk, c)$: on input $dk = f_2 = g_2^\alpha$ and a ciphertext $c = (A, B, C_1)$, picks a random $r'' \in \mathbb{Z}_p$, computes $d''_0 = dk \cdot \mathsf{H}_1(\mathsf{ID}_0)^{r''}$ and $d''_1 = g_1^{r''}$, outputs $A \cdot e(d''_1, C_1)/e(B, d''_0) = m$.
- $\mathsf{Sign}(sk, \widetilde{m})$: on input a signing key $sk = sk_{\mathsf{ID}_1} = (d'_0, d'_1)$ and a message $\widetilde{m} \in \{0,1\}^*$, first picks a random $r \in \mathbb{Z}_p$, computes $d_0 = d'_0 \cdot \mathsf{H}_2(\widetilde{m})^r$, $d_1 = d'_1$ and $d_2 = g_1^r$, outputs $\sigma = sk_{\mathsf{ID}} = (d_0, d_1, d_2) \in (\mathbb{G}_2, \mathbb{G}_1, \mathbb{G}_1)$.
- $\mathsf{Vrfy}(pk, \widetilde{m}, \sigma)$: on input the public key $pk = mpk = f_1$, a message $\widetilde{m} \in \{0,1\}^*$ and a signature $\sigma = sk_{\mathsf{ID}} = (d_0, d_1, d_2)$, outputs 1 if following equation holds, otherwise outputs 0.

$$e(f_1, g_2) \cdot e(d_1, \mathsf{H}_1(\mathsf{ID}_1)) \cdot e(d_2, \mathsf{H}_2(\widetilde{m})) = e(g_1, d_0).$$

Remark 2. We simplify the Vrfy algorithm based on the fact that if σ is a valid signature for $\widetilde{m}$, then it can be used as the secret key for user $\mathsf{ID} = \langle \mathsf{ID}_1, \widetilde{m} \rangle$ to successfully decrypt any ciphertext $c = (A, B, C_1, C_2)$ for any plaintext m encrypted by mpk via BB_1-IBE. Specifically, for any randomness s, it always holds that $e(f_1, g_2)^s \cdot e(d_1, C_1) \cdot e(d_2, C_2) = e(B, d_0)$.

5.2 Implementation

Since our HIES is a newly proposed notion, there is no similar schemes can be used to judge its performance. As mentioned before (see Sect. 1.2), the "Cartesian product" construction of combined public key (CP-CPK) scheme introduced by Paterson et al. [30], which uses arbitrary public key encryption and signature schemes as components, provides a benchmark for any bespoke construction. Thus, we build a concrete CP-CPK scheme by choosing the most efficient encryption and signature schemes, i.e. ElGamal PKE and Schnorr signature and use it as a baseline.

We implement the concrete CP-CPK scheme atop elliptic curve `secp256k1` with 128-bit security in which $|\mathbb{G}| = 256$ bits and $|\mathbb{Z}_p| = 256$ bits, and implement our HIES scheme atop pairing-friendly curve `bls12-381` with 128-bit security level [32] in which $|\mathbb{G}_1| = 381$ bits, $|\mathbb{G}_2| = 762$ bits, $|\mathbb{Z}_p| = 256$ bits, and $|\mathbb{G}_T| = 1524$ bits (by exploiting compression techniques [31]).

Both of them are implemented in C++ based on the `mcl` library [33], and all the experiments are carried out on a MacBook Pro with Intel i7-9750H CPU (2.6 GHz) and 16 GB of RAM. Our implementation is released on GitHub and is available on https://github.com/yuchen1024/HISE/tree/master/hies. The code follows KEM-DEM paradigm.

Table 1. Efficiency comparison between CP-CPK and our HIES scheme

Functionality	strong joint security	individual escrow	key reuse	certificate costs
CP-CPK	✓	✓	✗	×2
HIES	✓	✓	✓	×1

Sizes (bits)	$\|pk\|$	$\|sk\|$	$\|c\|$	$\|\sigma\|$
CP-CPK	512	512	512	512
HIES	381	762	2667	1524

Efficiency (ms)	KeyGen	Derive	Enc	Dec	Sign	Vrfy
CP-CPK	0.015	$\oslash$	0.118	0.056	0.064	0.120
HIES	0.111	0.116	0.500	0.621	0.117	1.022

In the paradigm of KEM-DEM, we test the efficiency of algorithms of key generation, key derivation, encryption, decryption, signing and verification as well as the sizes of public key, secret key, ciphertext and signature. Symbol $\oslash$ means no corresponding algorithm.

Table 1 offers a comparison of HIES against the previous CP-CPK. In terms of functionality, it shows that HIES is in the public key reuse setting while CP-CPK is not. Moreover, HIES reduces the key management and key certificate costs. In terms of the experimental results, we admit the efficiency of our HIES scheme is slower than CP-CPK, but it is fortunately still considerable.

6 Conclusion

In this work, we resolve the problem left in [14] by formalizing the definition and the joint security of HIES. Similar to HISE, HIES also has a two-level key derive system, but the hierarchy between signing key and decryption key are reversed, thus it enables secure delegation of signature capacity. In addition, we present a generic construction of HIES from constrained identity-based encryption and give a rigorous proof of its joint security. Furthermore, we discuss three simple extensions of HIES for different delegation purposes. In the end, we implement our HIES scheme with 128-bit security. Though the construction here is a straightforward variant of HISE from constrained IBE, we emphasize the theoretical significance of HIES for completing the last piece of the key usage strategy puzzle. We leave the more ingenious and efficient constructions for future work.

Acknowledgements. We thank the anonymous reviewers for their helpful feedback. This work is supported by the National Key Research and Development Program of China (Grant No. 2021YFA1000600), the National Natural Science Foundation of China (Grant No. 62272269), and the Taishan scholar program of Shandong Province.

A Hierarchical Identity-Based Encryption

Hierarchical identity-based encryption (HIBE) is first introduced in [19,21]. We formally describe the definition of HIBE below. In an HIBE scheme, users having a position in the hierarchy, are specified by an ID-tuple $\mathsf{ID} = (I_1, \cdots, I_j)$, where I_i corresponds to the identity at level i.

A.1 Definition of HIBE

Definition 6. *A hierarchical identity-based encryption scheme consists of five polynomial-time algorithms:*

- $\mathsf{Setup}(1^\lambda)$*: on input a security parameter* λ*, outputs public parameters* pp*, including the plaintext space* $\mathcal{M}$*, the ciphertext space* $\mathcal{C}$ *and the identity space* $\mathcal{I}$ *in every level.*
- $\mathsf{KeyGen}(pp)$*: on input the public parameters* pp*, outputs a public key* mpk *and a master secret key* msk *(i.e. root secret in level-*0*).*
- $\mathsf{Extract}(mpk, sk_{\mathsf{ID}}, \langle \mathsf{ID}, I\rangle)$*: on input the public key* mpk*, a secret key for ID-tuple* ID*, and an ID-tuple* $\langle \mathsf{ID}, I\rangle$ *which is a child node of* ID*, outputs* $sk_{\langle \mathsf{ID}, I\rangle}$*.*
- $\mathsf{Enc}(mpk, \mathsf{ID}, m)$*: on input public key* mpk*, the ID-tuple of the intended message recipient* ID *and a message* $m \in \mathcal{M}$*, outputs a ciphertext* $c \in \mathcal{C}$*.*
- $\mathsf{Dec}(sk_{\mathsf{ID}}, c)$*: on input a secret key* sk_{ID} *and a ciphertext* c*, outputs a message* m *or a special reject symbol* $\perp$ *denoting failure.*

Correctness. An HIBE scheme is correct, if encryption algorithm Enc and decryption algorithm Dec satisfy the standard consistency constraint, namely, when sk_{ID} is the secret key generated by the extraction algorithm $\mathsf{Extract}$ for user ID, then for any $m \in \mathcal{M}$ and $c \leftarrow \mathsf{Enc}(mpk, \mathsf{ID}, m)$, it always holds that $\mathsf{Dec}(sk_{\mathsf{ID}}, c) = m$.

Security. Let $\mathcal{O}_{\mathsf{extract}}$ be an oracle of $\mathsf{Extract}$ that on input an ID-tuple ID and outputs sk_{ID}. An HIBE scheme is IND-CPA secure, if for all PPT adversary $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\lambda)$ such that:

$$\Pr\left[b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^{\lambda}); \\ (mpk, msk) \leftarrow \mathsf{KeyGen}(pp); \\ (\mathsf{ID}^*, (m_0, m_1)) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{extract}}}(pp, mpk); \\ b \xleftarrow{R} \{0,1\}, c^* \leftarrow \mathsf{Enc}(mpk, \mathsf{ID}^*, m_b); \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{extract}}}(c^*); \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

In guess stage, $\mathcal{A}$ is not allowed to query the $\mathcal{O}_{\mathsf{extract}}$ for ID^* or the ancestor nodes of it (i.e. IDs which are prefixed with ID^*). Meanwhile, two weaker security notions can be defined similarly. One is OW-CPA security, in which the adversary is required to recover the plaintext from a random ciphertext. The other is selective-identity IND-CPA security, in which the adversary must commit ahead of time (non-adaptively) to the identity it intends to attack before seeing the mpk.

A.2 Boneh-Boyen HIBE Scheme

We review the ℓ-HIBE scheme of Boneh-Boyen ($\mathsf{BB_1}$-IBE) [9] as below. As [6,18] noticed, compared to symmetric pairings, asymmetric pairings yield schemes having more efficiency in terms of both bandwidth and computation time. Therefore, we adjust the original Boneh-Boyen HIBE with asymmetric pairings.

- $\mathsf{Setup}(1^{\lambda})$: on input the security parameter λ, generates an asymmetric pairings tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$, and picks a family of collision resistant hash functions $\mathsf{H}_j : \{0,1\}^* \to \mathbb{G}_2$ for $j \in [0, \ell]$. The public parameters pp include the description of bilinear groups and the hash functions $\{\mathsf{H}_j\}_{j \in [0,\ell]}$. The ID at level-j is $\mathcal{I}^j = (\{0,1\}^*)^j$. The plaintext space is $\mathcal{M} = \mathbb{G}_T$.
- $\mathsf{KeyGen}(pp)$: on input the public parameters pp, picks a random $\alpha \in \mathbb{Z}_p$, sets $f_1 = g_1^{\alpha}$ and $f_2 = g_2^{\alpha}$, sets public key $mpk = f_1 = g_1^{\alpha}$ and master secret key $msk = f_2 = g_2^{\alpha}$.
- $\mathsf{Extract}(mpk, sk_{\mathsf{ID}}, \langle \mathsf{ID}, I \rangle)$: on input the public key mpk, a level-j private key $sk_{\mathsf{ID}} = (d_0, \ldots, d_j) \in \left(\mathbb{G}_2, \mathbb{G}_1^j\right)$ and a level-$(j+1)$ ID-tuple $\langle \mathsf{ID}, I \rangle = (I_1, \ldots, I_j, I_{j+1}) \in (\{0,1\}^*)^{j+1}$, first picks a random $r \in \mathbb{Z}_p$ and outputs

$$sk_{\langle \mathsf{ID}, I \rangle} = (d_0 \cdot \mathsf{H}_{j+1}(I_{j+1})^r, d_1, \ldots, d_j, g_2^r) \in \left(\mathbb{G}_2, \mathbb{G}_1^{j+1}\right)$$

  Note that (1) when ID is an empty set denoted as ϵ, sk_{ID} is exactly the master secret key msk, that is $sk_{\epsilon} = f_2 = g_2^{\alpha}$. (2) all the private keys can be also extracted directly from the master secret key msk through computing $sk_{\langle \mathsf{ID}, I \rangle} = \left(g_2^{\alpha} \cdot \prod_{k=1}^{j+1} \mathsf{H}_k(I_k)^{r_k}, g_1^{r_1}, \ldots, g_1^{r_{j+1}} \right)$ with random elements $r_1, \ldots, r_{j+1} \in \mathbb{Z}_p$.

– $\mathsf{Enc}(mpk, \mathsf{ID}, m)$: on input the public key mpk, an ID-tuple $\mathsf{ID} = (I_1, \ldots, I_j) \in (\{0,1\}^*)^j$ and a message $m \in \mathbb{G}_T$, picks a random $s \in \mathbb{Z}_p$ and outputs

$$C = (e(f_1, g_2)^s \cdot m, g_1^s, \mathsf{H}_1(I_1)^s, \ldots, \mathsf{H}_j(I_j)^s) \in \left(\mathbb{G}_T, \mathbb{G}_1, \mathbb{G}_2^j \right).$$

– $\mathsf{Dec}(sk_{\mathsf{ID}}, c)$: on input a private key $sk_{\mathsf{ID}} = (d_0, d_1, \ldots, d_j)$ and a ciphertext $C = (A, B, C_1, \ldots, C_j)$, outputs

$$A \cdot \frac{\prod_{k=1}^{j} e(d_k, C_k)}{e(B, d_0)} = m.$$

References

1. Government of Canada. https://www.canada.ca/en/shared-services/corporate/transparency/briefing-documents/ministerial-briefing-book/delegation.html
2. PGP. https://www.openpgp.org
3. The University of Iowa. https://opsmanual.uiowa.edu/administrative-financial-and-facilities-policies/facsimile-signatures-and-signature-assignment-2
4. Viafirma. https://www.viafirma.com/blog-xnoccio/en/signature-delegation/
5. WhatsApp. https://www.whatsapp.com
6. Akinyele, J.A., Garman, C., Hohenberger, S.: Automating fast and secure translations from type-i to type-iii pairing schemes. In: ACM CCS 2015, pp. 1370–1381 (2015)
7. Alimi, P.: On the use of pedersen commitments for confidential payments. https://research.nccgroup.com/2021/06/15/on-the-use-of-pedersen-commitments-for-confidential-payments/
8. Boldyreva, A.: Secure proxy signature scheme for delegation of signing rights (2003). http://eprint.iacr.org/2003/096/
9. Boneh, D., Boyen, X.: Efficient selective-id secure identity based encryption without random oracles. Cryptology ePrint Archive, Report 2004/172 (2004). https://ia.cr/2004/172
10. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: towards privacy in a smart contract world. In: Bonneau, J., Heninger, N. (eds.) FC 2020. LNCS, vol. 12059, pp. 423–443. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51280-4_23
11. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_16
12. Cao, F., Cao, Z.: A secure identity-based multi-proxy signature scheme. Comput. Electr. Eng. **35**(1), 86–95 (2009)
13. Chen, Y., Ma, X., Tang, C., Au, M.H.: PGC: pretty good confidential transaction system with auditability. In: ESORICS 2020, pp. 591–610 (2020)
14. Chen, Y., Tang, Q., Wang, Y.: Hierarchical integrated signature and encryption. Cryptology ePrint Archive, Report 2021/1237 (2021). https://ia.cr/2021/1237
15. Coron, J.-S., Joye, M., Naccache, D., Paillier, P.: Universal padding schemes for RSA. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 226–241. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_15

16. Dai, J.Z., Yang, X.H., Dong, J.X.: Designated-receiver proxy signature scheme for electronic commerce. In: SMC 2003 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483) (2003)
17. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_34
18. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discret. Appl. Math. **16**, 3113–3121 (2008)
19. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_34
20. Haber, S., Pinkas, B.: Securely combining public-key cryptosystems. In: ACM CCS 2001, pp. 215–224 (2001)
21. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_31
22. Huang, X., Mu, Y., Susilo, W., Zhang, F., Chen, X.: A short proxy signature scheme: efficient authentication in the ubiquitous world. In: Enokido, T., Yan, L., Xiao, B., Kim, D., Dai, Y., Yang, L.T. (eds.) EUC 2005. LNCS, vol. 3823, pp. 480–489. Springer, Heidelberg (2005). https://doi.org/10.1007/11596042_50
23. Huang, X., Susilo, W., Mu, Y., Wu, W.: Proxy signature without random oracles. In: Cao, J., Stojmenovic, I., Jia, X., Das, S.K. (eds.) MSN 2006. LNCS, vol. 4325, pp. 473–484. Springer, Heidelberg (2006). https://doi.org/10.1007/11943952_40
24. Kim, S., Park, S., Won, D.: Proxy signatures, revisited. In: Han, Y., Okamoto, T., Qing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 223–232. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0028478
25. Komano, Y., Ohta, K.: Efficient universal padding techniques for multiplicative trapdoor one-way permutation. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 366–382. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_22
26. Lee, B., Kim, H., Kim, K.: Secure mobile agent using strong non-designated proxy signature. In: Varadharajan, V., Mu, Y. (eds.) ACISP 2001. LNCS, vol. 2119, pp. 474–486. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47719-5_37
27. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures: delegation of the power to sign messages. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **79**(9), 1338–1354 (1996)
28. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: Proceedings of the 3rd ACM Conference on Computer and Communications Security, pp. 48–57. CCS 1996, Association for Computing Machinery, New York, NY, USA (1996)
29. Neuman, B.: Proxy-based authorization and accounting for distributed systems. In: 1993 Proceedings. The 13th International Conference on Distributed Computing Systems, pp. 283–291 (1993)
30. Paterson, K.G., Schuldt, J.C.N., Stam, M., Thomson, S.: On the joint security of encryption and signature, revisited. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 161–178. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_9
31. Rubin, K., Silverberg, A.: Compression in finite fields and torus-based cryptography. SIAM J. Comput. **37**(5), 1401–1428 (2008)

32. Sakemi, Y., Kobayashi, T., Saito, T., Wahby, R.S.: Pairing-Friendly Curves. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-09, Internet Engineering Task Force (2020). https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-pairing-friendly-curves-09
33. Shigeo, M.: A portable and fast pairing-based cryptography library. https://github.com/herumi/mcl
34. Shim, K.A.: Short designated verifier proxy signatures. Comput. Electr. Eng. **37**(2), 180–186 (2011)