Hierarchical Integrated Signature and Encryption

(or: Key Separation vs. Key Reuse: Enjoy the Best of Both Worlds)

Yu Chen[1,2,3], Qiang Tang[4], Yuyu Wang[5](✉)

[1] School of Cyber Science and Technology, Shandong University, Qingdao 266237, China
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao 266237, China
yuchen@sdu.edu.cn
[4] School of Computer Science, University of Sydney, Sydney, Australia
qiang.tang@sydney.edu.au
[5] University of Electronic Science and Technology of China, Chengdu, China
wangyuyu@uestc.edu.cn

Abstract. In this work, we introduce the notion of hierarchical integrated signature and encryption (HISE), wherein a single public key is used for both signature and encryption, and one can derive a secret key used only for decryption from the signing key, which enables secure delegation of decryption capability. HISE enjoys the benefit of key reuse, and admits individual key escrow. We present two generic constructions of HISE. One is from (constrained) identity-based encryption. The other is from uniform one-way function, public-key encryption, and general-purpose public-coin zero-knowledge proof of knowledge. To further attain global key escrow, we take a little detour to revisit global escrow PKE, an object both of independent interest and with many applications. We formalize the syntax and security model of global escrow PKE, and provide two generic constructions. The first embodies a generic approach to compile any PKE into one with global escrow property. The second establishes a connection between three-party non-interactive key exchange and global escrow PKE. Combining the results developed above, we obtain HISE schemes that support both individual and global key escrow.

We instantiate our generic constructions of (global escrow) HISE and implement all the resulting concrete schemes for 128-bit security. Our schemes have performance that is comparable to the best Cartesian product combined public-key scheme, and exhibit advantages in terms of richer functionality and public key reuse. As a byproduct, we obtain a new global escrow PKE scheme that is $12 - 30\times$ faster than the best prior work, which might be of independent interest.

1 Introduction

Public-key encryption (PKE) and digital signature are widely used in combination in many real-world applications, where the former is used to protect data

confidentiality, and the latter is used to provide authenticity. For example, in secure communication applications such as PGP [PGP], supposing that Alice wants to send an email to Bob in a secure and authenticated manner, she first encrypts the email under Bob's public-key, and then signs the ciphertext using her signing key. In privacy-preserving cryptocurrencies such as Zether [BAZB20], to generate a confidential transaction, a sender account encrypts the transfer amount under the public keys of both the sender account and receiver account, and then signs the transaction using his secret spending key.

When using PKE and signature schemes simultaneously, we require joint security, i.e., their respective security properties are retained in the presence of additional oracles (if there is any, e.g., signing oracle and decryption oracle). The reason is that although PKE and signature schemes might have been proven to be secure individually, they may undermine each other if their respective keys are related. Typically, there are two principals for combining PKE and signature.

Key separation vs. key reuse. The *key separation* principal is an engineering folklore that dictates using different keypairs for different cryptographic operations, which is best illustrated by the "Cartesian product" combined public-key (CPK) scheme: each user independently generates a keypair (ek, dk) for PKE and a keypair (vk, sk) for digital signature, concatenates the two keypairs into one, and then uses appropriate component of the compound key for each operation. Key separation allows one to flexibly choose and combine the off-the-shelf PKE and signature schemes, and the joint security follows readily from the independence of the two keypairs. However, it has an obvious shortcoming that the key size and the complexity of key management are doubled.[6]

In contrast, the *key reuse* principal is using identical keypair, e.g., for both PKE and signatures, and we refer to such cryptosystem as integrated signature and encryption (ISE). To avoid triviality, the keypair should be non-splittable, namely, it cannot be broken into two pieces for different operations respectively.

As advocated by Paterson et al. [PSST11], adopting key reuse principal is beneficial, since it can reduce key storage requirements, reduce the number of certificates needed (which in turn reduces the certificate cost[7]), and reduce the footprint of cryptographic code and development effort. These savings could be vital in constrained environments such as embedded systems and low-end smart card applications. For instance, the globally-deployed EMV standard for authenticating credit and debit card transactions uses the same keypair for both encryption and signature precisely for these reasons (see [EMV11, Sec. 7]). Other real world

[6] One may attempt to include the encryption key ek and verification key vk into one certificate in order to keep the certificate cost unchanged. Unfortunately this theoretically possible solution is not standard-compliant. X.509v3 as per RFC 5280 [X50] only allows a single `subjectPublicKeyInfo` field. If one wants to add more than one public key into this field, new syntax or parsing rule are needed, which would require major changes to implementations and relevant libraries. In contrast, key reuse is readily supported by X.509v3 via the `keyUsage` field.

[7] Certificate costs include but not limit to registration, issuing, storage, transmission, verification, and building/recurring fees.

instances embracing key reuse include identity management solution provider Ping Identity [Pin] and RFC 4055. We highlight that the key reuse principal also helps to simplify the design of high-level protocols. Notably, most known privacy-preserving cryptocurrencies in the account model [NVV18, BAZB20, CMTA20] either explicitly or implicitly use ISE as a core building block, which enables a clean security notion and simple constructions.

Nevertheless, key reuse is not without its issues. In an ISE scheme, the reuse of a single keypair may hinder the individual security of the PKE or the signature scheme, (consider the textbook RSA cryptosystem as a simple example and see [DLP+12] for a more sophisticated example at the protocol level). Therefore, joint security of ISE is not immediate and a rigorous proof is always needed.

Also, Haber and Pinkas [HP01] pointed out that secret keys may require different levels of protection, which becomes out of reach when sticking to key reuse principal. A more puzzling issue, as we elaborate next, is that rigid adherence to key reuse principal introduces hurdles on applications that require key escrow.

Delegation of decryption capability. In privacy-preserving applications enabled by PKE, a user may want to delegate his decryption capability to an agent for key recovery or usability purpose, while an authority (law-enforcement agencies as well as other organizations) may want to acquire decryption capability of users for compliance purpose. This is where key escrow comes into play. In general, there are two types of key escrow mechanisms.

The *individual key escrow* means that the user simply shares his decryption key with the escrow agent. Such delegation of decryption capability is of "one-to-one" flavor, and under the control of each individual user. The *global key escrow* means that the escrow agent has a single "master" key to decrypt any ciphertext of any user. Such delegation of decryption capability is of "all-to-one" flavor. We note that individual key escrow implies a naive solution to global key escrow by having the agent maintain a big database of all individual decryption keys. However, this naive solution comes with two deficiencies: (i) the complexity of key management grows linearly with the number of keys, which severely limits scalability, and thus being inadequate for large-scale applications; (ii) collecting a large number of valid decryption keys could be difficult to conduct in practice.

Conflict between key reuse and key escrow. In the context of combined usage of PKE and signature, the original joint security is insufficient to enable individual key escrow, and strong joint security is needed. This is because now the adversary is directly given *the decryption key*, instead of just a decryption oracle (as we still want to ensure integrity even if escrow agent is corrupted). Clearly, the ISE schemes adhering to key reuse strategy fail to meet strong joint security as the same secret key is used for both decryption and signing, and consequently individual key escrow is insecure since a corrupted escrow agent is able sign on behalf of the user, a basic violation of the concept of digital signing [Ros] (and cannot be applied to many settings such as anonymous cryptocurrency).

From the above discussion, we are facing a dilemma between key reuse that brings performance benefit and key separation that supports key escrow mechanism. We are thus motivated to ask the following intriguing questions:

Can we enable individual key escrow mechanism while retaining the merits of key reuse? And, can we further support global key escrow mechanism?

1.1 Our Contributions

We answer the above questions affirmatively and have the following results.

Hierarchical integrated signature and encryption. In an ISE scheme, a single keypair is used for both encryption and signature, thus the exposure of decryption key will completely compromise the security of signature. A closer look indicates that if there is a hierarchy between the signing key and decryption key, then stronger joint security becomes possible. We put forth a new notion called hierarchical integrated signature and encryption (HISE). In an HISE scheme, a single public key is used for both encryption and signature verification; the signing key plays the role of "master" secret key, namely, one can derive a decryption key from the signing key but not vice versa. This two-level hierarchy key derivation structure *hits a sweet balance* between key separation and key reuse, and thus allows us to enjoy the best of both worlds. It not only admits individual key escrow mechanism and classified protection of signing key and decryption key, but also retains the benefit of key reuse strategy.[8]

We specify a strong joint security model for HISE schemes by capturing multifaceted attacks in the joint sense. For confidentiality, we stipulate that the PKE component satisfies indistinguishability against chosen-ciphertext attacks (IND-CCA) even the adversary is provided with unrestricted access to a signing oracle. For authenticity, we stipulate that the signature component satisfies existentially unforgeability against chosen-message attacks (EUF-CMA) even the adversary is *directly given the associated decryption key*. We then present two generic constructions of HISE schemes.

HISE from (constrained) IBE. Our first construction is inspired by the elegant ISE construction due to Paterson et al. [PSST11]. In their construction, they apply the Naor transform [BF03] and the tag-based version of the Canetti-Halevi-Katz (CHK) transform [BCHK07] to an identity-based encryption (IBE) scheme simultaneously, yielding a signature component and a PKE component in one shot. The two components share the same keypair, i.e., the master keypair of the underlying IBE. Note that signatures in the signature component derived from the Naor transform are private keys for messages (playing the role of identities), while these private keys can decrypt ciphertexts in the PKE component derived from the CHK transform. To attain joint security, they use a bit prefix in the identity space to provide a domain separation between the identities used for encoding messages and the identities used as tags. However, ISE schemes from

[8] As briefly elaborated before, the advantage of key reuse strategy mostly resides in the fact that one public key is used for both encryption and verification.

IBE do not directly lend themselves to HISE schemes, as the master secret key of IBE plays the roles of *both the signing key and decryption key*.

We resolve this problem by introducing a new notion called *constrained IBE* (see Section 2.2 for definition and construction) as our starting point. In a constrained IBE one can derive constrained keys sk_f for $f \in \mathcal{F}$ from the master secret key, where $\mathcal{F}$ is a predicate family defined over identity space, e.g., a family of prefix predicates. A constrained key sk_f enables the decryption of ciphertexts encrypted under id if and only if $f(id) = 1$. We are now ready to sketch our HISE construction from any constrained IBE that supports prefix predicates, which is in turn implied by binary tree encryption (BTE) [CHK03]. Suppose the identity space I of the underlying constrained IBE is $\{0,1\}^{\ell+1}$, we use bit prefix to partition I to two disjoint sets, say, I_0 starting with bit 0 and I_1 starting with bit 1. The key generation algorithm first generates a master keypair (mpk, msk) of the constrained IBE, sets mpk as the public key and msk as the secret key, and derives a constrained key sk_{f_1} from msk, where $f_1(id) = 1$ iff $id \in I_1$. Thanks to the properties of constrained IBE, sk_{f_1} can decrypt all ciphertexts encrypted under identities in I_1, and thus could serve as the decryption key. We then build the signature component from the constrained IBE via the Naor transform by encoding messages into I_1, and build the encryption component from the constrained IBE and one-time signature via the CHK transform by using identities from I_1 as tags. The security of constrained IBE implies that the signature component remains secure even in the presence of the decryption key. In this way, we obtain HISE with strong joint security in the standard model.

We remark that if one does not insist on joint security in the standard model, then it is not necessary to resort to the CHK transform to achieve CCA security. As a result, a much simpler construction of HISE can be built from any IBE. The construction is similar to the one from constrained IBE, except that I_1 shrinks to a single identity fixed in the public parameters, and the encryption component is obtained by applying the IBE-to-PKE degradation and the Fujisaki-Okamoto transformation [FO99] sequentially.

HISE from PKE and ZKPoK. Our second construction is from PKE and zero-knowledge proof of knowledge (ZKPoK). At the heart of it is a novel hierarchical key derivation mechanism. Roughly speaking, the key generation algorithm consists of two steps: (1) choosing a random bit string as the signing key, and then map it to random coins via a uniform one-way function (OWF) F (a OWF that outputs uniform bits when input uniform bits); (2) feeding the resulting random coins to the key generation algorithm of PKE, yielding a keypair. The public key serves as both the encryption key and verification key. The encryption component is exactly the underlying PKE. In this way, the decryption key can be easily derived from the signing key, but not vice versa. The merit of the above hierarchical key derivation mechanism is that it endows great flexibility of the underlying PKE schemes, and thus is of particular interest for application scenarios where it is desirable to upgrade the PKE in use to HISE in a seamless way. However, it also gives rise to a technical challenge: how to design a signature scheme with *an unstructured bit string* as the signing key, which should remain

secure even in the presence of partial leakage, say, the decryption key. We show that if the function G from random coins to public key induced by the key generation algorithm is target-collision resistant, then the composed function $\mathsf{G} \circ \mathsf{F}$ from signing key to public key is one-way even with respect to arbitrary leakage of the intermediate random coins, let alone the decryption key. Therefore, we can overcome the aforementioned difficulty by leveraging public-coin ZKPoK. A signature is a non-interactive zero-knowledge proof of the signing key, incorporating a message to be signed. This construction essentially embodies a generic approach of converting any PKE to HISE with the help of ZKPoK (we refer to it as the HI conversion hereafter).

We note that the high-level idea of using OWF and ZKPoK to build signatures had appeared in previous works [CDG+17, KKW18], but our usage of this technique is *qualitatively* different. Prior works focus on building a standalone signature scheme: the public key is simply an image $y = \mathsf{F}(x)$ of a OWF F and secret key x. In our construction, we aim to add signature functionality to existing PKE schemes, yielding HISE schemes with strong joint security. To do so, the public key is set as the output of secret key via a function composed of a OWF and the PKE's key generation algorithm. Careful analysis of the minimal requirements on the OWF and key generation algorithm, as well as the HISE construction we propose, are new to this work.

Supporting global key escrow. We then turn to the problem of equipping HISE with global key escrow mechanism. To make our techniques more general, we first take a little detour to revisit the topic in the setting of PKE.

Global escrow PKE. In global escrow PKE there is an escrow agent holding a global escrow decryption key that can decrypt ciphertexts encrypted under any public key. The state of the art of global escrow PKE is less satisfactory, which is long overdue for formal definition and efficient construction. So far, the only known practical scheme based on standard assumption is the escrow ElGamal PKE proposed by Boneh and Franklin [BF03] from bilinear maps.

At first glance, it seems that global escrow PKE can be trivially built from broadcast encryption by having the receiver set include the real intended receiver and the escrow agent. However, the idea does not work since the sender in broadcast encryption is always assumed to be honest, while in the context of global escrow PKE the sender could be *malicious* (e.g. generate ciphertexts dishonestly) especially if he has the incentive to evade the oversight of escrow agent. To capture such misbehaving, we introduce the "consistency" notion to enforce the decryption results of any ill-formed ciphertexts yielded by the receiver's decryption key and the global escrow decryption key to be identical. We then propose two generic constructions of global escrow PKE.

Our first construction is based on PKE and non-interactive zero-knowledge proof (NIZK) (see Section 6.1 for details). The escrow agent generates a keypair (pk_γ, sk_γ), then publishes pk_γ as public parameters, and uses sk_γ as the global escrow decryption key. To generate a ciphertext for the receiver holding public key pk_β, the sender encrypts the plaintext under pk_β and pk_γ respectively, and then appends a NIZK proof for the validity of encryption. To decrypt a

ciphertext, the receiver (resp. escrow agent) first checks if the proof is valid, and then decrypts with secret key sk_β (resp. sk_γ) if so or returns $\perp$ otherwise. The main purpose of using NIZK is to guarantee the consistency of decryption results yielded by the receiver's decryption key and global escrow decryption key, while a bonus is that the resulting global escrow PKE automatically satisfies CCA security. This construction can be interpreted as a novel usage of the celebrated Naor-Yung paradigm [NY90], which indicates that any PKE can be upgraded to support global escrow with the help of NIZK (we refer to it as the GE conversion hereafter).

Our second construction is based on three-party non-interactive key exchange (NIKE) (see Section 6.2 for details). Same as our first construction, the escrow agent generates a keypair (pk_γ, sk_γ), publishes pk_γ as part of public parameters, and uses sk_γ as the global escrow decryption key. To generate a ciphertext for the receiver holding public key pk_β, the sender generates a random keypair (pk_α, sk_α), and then runs the three-party NIKE *in his head* to compute a shared key among $(pk_\alpha, pk_\beta, pk_\gamma)$. The final ciphertext consists of pk_α and a symmetric encryption of plaintext under the shared key. To decrypt, the receiver (resp. escrow agent) uses secret key sk_β (resp. sk_γ) to compute the shared key among $(pk_\alpha, pk_\beta, pk_\gamma)$, and then decrypts the symmetric part. This construction suggests a generic approach of converting three-party NIKE to global escrow PKE, uncovering a connection between two seemingly unrelated notions. More interestingly, we show that the construction still works by relying on a relaxed version of three-party NIKE, leading to the most efficient global escrow PKE to date (outperforms prior scheme [BF03] in speed by a factor $12 - 30\times$), which might be of independent interest.

Global escrow HISE. Now, we are ready to construct HISE that supports global key escrow mechanism that we dub "global escrow HISE". In a global escrow HISE, the escrow agent is capable of decrypting any ciphertext under any public key with a succinct global escrow decryption key, while the security of the signature component retains even in the presence of the associated individual decryption key and the global escrow decryption key. Combining the results developed above, we obtain two paths of building global escrow HISE from different starting points. One is to apply the Naor-Yung like transform (GE conversion) to any HISE, and the other is to add hierarchy key derivation structure (HI conversion) to any global escrow PKE meeting the mild requirement described above. Figure 1 depicts the technology roadmap for the constructions of global escrow HISE.

Applications of (global escrow) HISE. Besides the merit of compact public key sizes, (global escrow) HISE also helps to reduce the key management complexity and simplify the design and analysis of high-level protocols. In general, they are suitable for scenarios that simultaneously require privacy, authenticity and key escrow. Below, we give several illustrative usages.

Usage of HISE. In privacy-preserving cryptocurrencies such as Zether [BAZB20], a user may need to share his decryption key with an authority for audit purpose

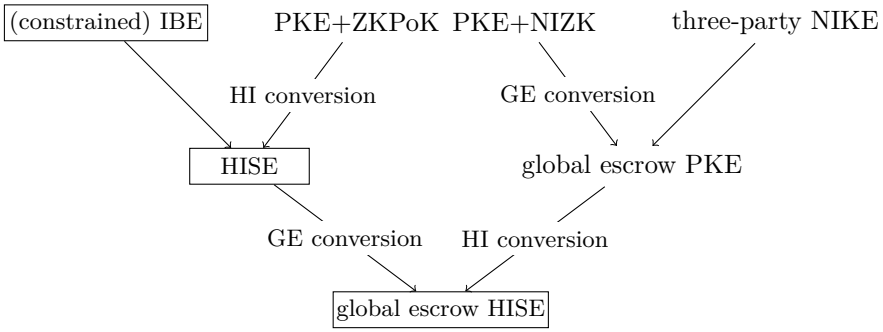

Fig. 1: Technology roadmap of global escrow HISE. The rectangles denote our newly introduced cryptographic schemes.

or delegating costly decryption operations[9] to a service. Currently, Zether is equipped with ISE and thus does not support individual key escrow. In another case, a PGP user may be required to handover his decryption key to an authority on demand for compliance purpose.[10] For the time being, PGP adopts key separation and thus naturally supports individual key escrow, but each user has to maintain at least two public key certificates. In either case, the user wants to guarantee that his signing capability remains exclusive. By deploying HISE, not only the systems can benefit from key reuse, but also the user can safely escrow his decryption key to a third party without worrying the security of signature being breached (e.g. in the cryptocurrency setting, even the auditing authority with decryption key cannot spend user's coin).

Usage of global escrow HISE. Enterprise applications such as Slack get increasing adoption for large-scale collaborative working, and thus has raised the demand for secure internal communication which may contain proprietary information. The employer may have the right to get access to all private communications as in traditional work emails [vox], or might be obliged to possess "super" decryption capability for various reasons such as archival purpose, litigation-related eDiscovery, or detection of malware. On the other side, the employees need to be assured that even a malicious administrator of the "super" key cannot slander them by forging signatures for fabricated communications. Global escrow HISE is perfectly suitable for these cases. By playing the role of escrow agent, the authority is able to conduct large-scale supervision efficiently with the global escrow decryption key, but unable to violate users' exclusive signing capability.

[9] A bunch of recent privacy-preserving cryptocurrencies [NVV18, BAZB20, CMTA20] employ lifted ElGamal like PKE schemes, and thus decryption operations require computing the discrete logarithm, which is time consuming.

[10] The government of the United Kingdom requires any PGP user to give the police both his private key and his passphrase on demand. Failure to comply is a criminal offense, punishable by a jail term of two years.

Instantiation, implementation and evaluation. We instantiate our generic constructions of (global escrow) HISE and implement all the resulting concrete schemes for 128-bit security. We choose the Cartesian product CPK built from the best available encryption and signature schemes as benchmark. Our (global escrow) HISE schemes have performance that is comparable to the Cartesian product CPK scheme, while exhibiting advantages in terms of richer functionality for escrow and compact key sizes. Moreover, we report the most efficient global escrow PKE known to date ($12 - 30\times$ faster than prior scheme), which is interesting in its own right. Our implementation is released on Github: https://github.com/yuchen1024/HISE. We summarize experimental results in Section 8.

1.2 Related Works

Combined usage of PKE and signature. Key separation is a conventional wisdom originated from real-world practice. Haber and Pinkas [HP01] investigate this security engineering folklore and initiate a formal study of key reuse. They introduce the notion of combined public key (CPK) scheme, which is a combination of a signature and encryption scheme: the existing algorithms of sign, verify, encrypt and decrypt are preserved, while the two key generation algorithms are modified into a single algorithm. This algorithm outputs two keypairs for signing and encryption operations respectively, with the keypairs no longer necessarily being independent. They also formalize the joint security of CPK scheme, i.e., 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 an additional decryption oracle. Finally, they show that various well-known concrete schemes are jointly secure when their keys are partially shared. As an extreme case of CPK scheme, ISE scheme uses a single non-splittable keypair for both signature and encryption. Degabriele et al. [DLP+12] find a theoretical attack for the RSA-based ISE scheme in EMV standard version 4.1. Coron et al. [CJNP02] and Komano and Ohta [KO03] build ISE from trapdoor permutations in the random oracle model. Paterson et al. [PSST11] give an elegant construction of ISE from identity-based encryption.

In contrast to ISE, HISE is equipped with a two-level hierarchy key structure, i.e., the signing key plays the role of master secret key, and one can derive a decryption key from the signing key. The joint security of HISE stipulates that the signature component is EUF-CMA secure even in the presence of a decryption key, which is strictly stronger than that of ISE.

Key escrow. We now briefly survey existing works on key escrow in the context of public-key encryption. As aforementioned, there are two types of key escrow: *individual key escrow* and *global key escrow*. While individual key escrow is straightforwards, global key escrow appears to be harder to attain. The earlier solutions to global key escrow are not satisfactory. They either rely on tamper-resistant devices, or require the escrow agent to get involved in interactive computations at an undesirable level. Paillier and Yung [PY99] propose a solution called self-escrowed public-key infrastructure, which requires that the

relation between secret key and public key is trapdoorness. Such stringent requirement greatly limits the choice of possible candidates, and so far the only known realization of SE-PKI is based on a non-standard assumption. Until 2003, Boneh and Franklin [BF03] give the first practical scheme called escrow ElGamal based on standard assumption. Nevertheless, formal definition and generic constructions of global escrow PKE are still missing.

To our knowledge, the only work in the literature that considers key reuse and key escrow together is due to Verheul [Ver01]. Verheul considers the problem of supporting non-repudiation and individual key escrow in the single public key setting, and proposes a candidate scheme from the XTR subgroup. The author gives an indication of security, but is not aware of more rigorous security proof.[11] Therefore, this problem remains open. In this work, we resolve this open problem by proposing a new cryptographic primitive called HISE and giving efficient and provably secure constructions.

2 Preliminaries

We use the standard definitions of bilinear maps, SKE, PKE, signature, IBE, zero-knowledge proof systems, as well as non-interactive key exchange protocols. The definition of one-way functions has appeared previously, while the definition and construction of constrained IBE schemes are new. Since they are central to our work, we include their formal definitions as below.

2.1 One-Way Function

A function $\mathsf{F} : X \to Y$ is one-way if it is efficiently computable and hard-to-invert on average. Let $\mathcal{H}$ be a family of leakage functions defined over domain X. F is leakage-resilient one-way [DHLW10] w.r.t. $\mathcal{H}$ if the one-wayness remains in the presence of leakage $h(x)$, where x is the preimage and h could be any function from $\mathcal{H}$. If $\mathsf{F}(x)$ is uniform over Y when $x \xleftarrow{R} X$, we say that F is uniform.

2.2 Constrained Identity-Based Encryption

We introduce a new notion called constrained IBE. In a nutshell, a constrained IBE is an IBE in which master secret key allows efficient delegation with respect to a family of predicates over identity space. Formally, a constrained IBE consists of the following PPT 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 I.

[11] Our perspective is that a security reduction from Verheul's scheme to standard hardness problem is unlikely to be forthcoming, since it is difficult to emulate the decryption key for the adversary against the signature component.

- $\mathsf{KeyGen}(pp)$: 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 $id \in I$, 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 $id \in I$, outputs a user secret key sk_{id} if $f(id) = 1$ or $\perp$ otherwise.
- $\mathsf{Enc}(mpk, id, m)$: on input mpk, an identity $id \in I$, 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 $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$, any identity $id \in I$, any $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$, any message m, and any $c \leftarrow \mathsf{Enc}(mpk, id, m)$, it 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. Roughly speaking, a secure constrained IBE should ensure the secrecy of plaintexts encrypted by id as long as id has not been queried for user secret key or related constrained secret key. We formally define IND-CPA security for constrained IBE as below. Let $\mathcal{A}$ be an adversary against the IND-CPA security of constrained IBE and define its advantage in the following experiment:

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

$\mathcal{O}_{\mathsf{ext}}(\cdot)$ denotes the key extraction oracle, which on input id returns $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$. $\mathcal{O}_{\mathsf{constrain}}(\cdot)$ denotes the key constrain oracle, which on input f returns $sk_f \leftarrow \mathsf{Constrain}(msk, f)$. $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{ext}}(\cdot)$ with id^* or query $\mathcal{O}_{\mathsf{constrain}}(\cdot)$ with f such that $f(id^*) = 1$. A constrained IBE is IND-CPA secure if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the above security experiment. 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 is asked to specify the target identity id^* before seeing mpk.

We present a generic construction of constrained IBE for prefix predicates from BTE. Please see the full version for the details.

3 Definition of HISE

An HISE scheme consists of the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, outputs public parameters pp. We assume that pp includes the description of plaintext space M and message space $\widetilde{M}$.
- $\mathsf{KeyGen}(pp)$: on input pp, outputs a secret key sk and a public key pk. Here, sk serves as a master secret key, which can be used to derive decryption key.
- $\mathsf{Derive}(sk)$: on input a secret key sk, outputs a decryption key dk.
- $\mathsf{Enc}(pk, m)$: on input a public key pk and a plaintext $m \in M$, 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 secret key sk and a message $\widetilde{m} \in \widetilde{M}$, 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. For the PKE component, we require that for any $m \in M$, it holds that $\Pr[\mathsf{Dec}(dk, c) = m] \geq 1 - \mathsf{negl}(\lambda)$, where the probability is taken over the choice of $pp \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, $dk \leftarrow \mathsf{Derive}(sk)$, and $c \leftarrow \mathsf{Enc}(pk, m)$. For the signature component, we require that for any $\widetilde{m} \in \widetilde{M}$, it holds that $\Pr[\mathsf{Vrfy}(pk, \tilde{m}, \sigma) = 1] \geq 1 - \mathsf{negl}(\lambda)$, where the probability is taken over the choice of $pp \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, $\sigma \leftarrow \mathsf{Sign}(sk, \widetilde{m})$, and the random coins used by Vrfy.

The joint security of HISE stipulates that the PKE component is IND-CCA secure even in the presence of a signing oracle, while the signature component is EUF-CMA secure in the presence of the decryption key. The formal security notion is defined as below.

Definition 1 (Joint Security for HISE). *HISE is jointly secure if its encryption and signature components satisfy the following security notions. Hereafter, let $\mathcal{O}_{\mathsf{sign}}(\cdot)$ be the signing oracle that on input $\widetilde{m} \in \widetilde{M}$ returns $\sigma \leftarrow \mathsf{Sign}(sk, \widetilde{m})$, and $\mathcal{O}_{\mathsf{dec}}(\cdot)$ be the decryption oracle that on input c returns $m \leftarrow \mathsf{Dec}(dk, c)$.*

IND-CCA security in the presence of a signing oracle. *Let $\mathcal{A}$ be an adversary against the PKE component and define its advantage as:*

$$\Pr\left[b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}(\cdot), \mathcal{O}_{\mathsf{sign}}(\cdot)}(pp, pk); \\ b \xleftarrow{\mathrm{R}} \{0,1\}, c^* \leftarrow \mathsf{Enc}(pk, m_b); \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}(\cdot), \mathcal{O}_{\mathsf{sign}}(\cdot)}(c^*); \end{array}\right] - \frac{1}{2}.$$

$\mathcal{A}$ has unrestricted access to $\mathcal{O}_{\mathsf{sign}}(\cdot)$, but is not allowed to query $\mathcal{O}_{\mathsf{dec}}(\cdot)$ with c^* in Phase 2. The PKE component is IND-CCA secure in the joint sense if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the above security experiment.

EUF-CMA security in the presence of a decryption key. *Let $\mathcal{A}$ be an adversary against the signature component and define its advantage as:*

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

The set $\mathcal{Q}$ records queries to $\mathcal{O}_{\mathsf{sign}}(\cdot)$. The signature component is EUF-CMA secure in the joint sense if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the above security experiment.

Remark 1. The security notion of HISE is strictly stronger than that of ISE in the sense that the signature component remains secure even when the adversary learns the entire decryption key rather than only has access to $\mathcal{O}_{\mathsf{dec}}(\cdot)$. This strengthening is crucial for applications that require secure delegation of decryption capability. We then discuss possible weakening of joint security. It is well-known that homomorphism denies CCA security. Thus, when homomorphic property is more desirable, we can instead only require the PKE component to be CPA-secure. We refer to the corresponding security as weak joint security.

Towards a modular design, the PKE component can be defined as key encapsulation mechanism. We omit the formal definition here for straightforwardness.

Global escrow extension. If an HISE scheme further satisfies global escrow property, we refer to it as global escrow HISE. In global escrow HISE, the setup algorithm additionally outputs a escrow decryption key edk, and there is an alternative decryption algorithm enabled by edk, whose decryption results of any ciphertext are identical to those obtained by applying normal decryption algorithm with the decryption key of intended receiver. The joint security stipulates that the encryption component remains secure in the presence of a signing oracle, and the signature component is secure even in the presence of the decryption key and escrow decryption key. We omit the formal definition here for its straightforwardness.

4 HISE from Constrained Identity-Based Encryption

In this section, we present a generic construction of HISE. Given a constrained IBE for prefix predicates (cf. definition in Section 2.2) and a strong one-time signature (OTS), we create an HISE scheme as below.

- $\mathsf{Setup}(1^\lambda)$: runs $pp_{\mathrm{cibe}} \leftarrow \mathrm{CIBE}.\mathsf{Setup}(1^\lambda)$, $pp_{\mathrm{ots}} \leftarrow \mathrm{OTS}.\mathsf{Setup}(1^\lambda)$, outputs $pp = (pp_{\mathrm{cibe}}, pp_{\mathrm{ots}})$. We assume the identity space of constrained IBE is $\{0,1\}^{\ell+1}$, and the verification space of OTS is $\{0,1\}^\ell$.
- $\mathsf{KeyGen}(pp)$: on input $pp = (pp_{\mathrm{cibe}}, pp_{\mathrm{ots}})$, runs $\mathrm{CIBE}.\mathsf{KeyGen}(pp_{\mathrm{cibe}})$ to generate (mpk, msk), outputs public key $pk = mpk$ and secret key $sk = msk$.
- $\mathsf{Derive}(sk)$: parses sk as msk, runs $sk_{f_{\mathbf{v}}} \leftarrow \mathrm{CIBE}.\mathsf{Constrain}(msk, f_{\mathbf{v}})$ where $\mathbf{v} = 1$ and $f_{\mathbf{v}}(id) = 1$ iff $id[1] = 1$, outputs $dk = sk_{f_{\mathbf{v}}}$.

- $\mathsf{Enc}(pk, m)$: parses $pk = mpk$. The encryption algorithm runs $(ovk, osk) \leftarrow \mathrm{OTS.KeyGen}(pp_{\mathrm{ots}})$. sets $id = 1||ovk$, computes $c_{\mathrm{cibe}} \leftarrow \mathrm{CIBE.Enc}(mpk, id, m)$, $\sigma \leftarrow \mathrm{OTS.Sign}(osk, c_{\mathrm{cibe}})$, then outputs $c = (ovk, c_{\mathrm{cibe}}, \sigma)$.
- $\mathsf{Dec}(dk, c)$: parses $dk = sk_{f_{\mathsf{v}}}$ and $c = (ovk, c_{\mathrm{cibe}}, \sigma)$. The decryption algorithm first checks if $\mathrm{OTS.Vrfy}(ovk, c_{\mathrm{cibe}}, \sigma) = 1$, if not outputs $\perp$, else sets $id = 1||ovk$ and computes $sk_{id} \leftarrow \mathrm{CIBE.Derive}(sk_{f_{\mathsf{v}}}, id)$, outputs $m \leftarrow \mathrm{CIBE.Dec}(sk_{id}, c_{\mathrm{cibe}})$.
- $\mathsf{Sign}(sk, \widetilde{m})$: parses sk as msk, computes $sk_{id} \leftarrow \mathrm{CIBE.Extract}(msk, id)$ where $id = 0||\widetilde{m}$, outputs $\sigma = sk_{id}$.
- $\mathsf{Vrfy}(pk, \sigma, \widetilde{m})$: parses pk as mpk, σ as sk_{id} for $id = 0||\widetilde{m}$, picks a random plaintext $m \in M$, computes $c_{\mathrm{cibe}} \leftarrow \mathrm{CIBE.Enc}(mpk, id, m)$, outputs "1" if $\mathrm{CIBE.Dec}(sk_{id}, c_{\mathrm{cibe}}) = m$ and "0" otherwise.

Correctness follows from that of constrained IBE and OTS. For security, we have the following theorem.

Theorem 1. *If the constrained IBE scheme is IND-CPA secure and the OTS scheme is strong EUF-CMA secure, then the HISE construction is jointly secure.*

Due to space limit, we defer the security proof to the full version.

Remark 2. The above generic construction from constrained IBE enjoys joint security in the standard model. So far, we only know how to build constrained IBE for prefix predicates from BTE [CHK03]. However, in existing constructions of BTE the size of secret key and ciphertext and encryption/decryption efficiency are all linear in ℓ, which are inefficient. We leave more efficient constructions of BTE and constrained IBE as an interesting open problem.

In applications where the encryption component only has to be IND-CPA secure, or one is willing to accept IND-CCA security in the random oracle model, we have a simpler and more efficient construction of HISE from any IBE. We defer the details to the full version.

5 HISE from PKE and ZKPoK

In this section, we present a generic construction of HISE from a PKE scheme and a 3-round public-coin ZKPoK protocol. At the heart of our construction is a novel mechanism what we called hierarchical key derivation. The high-level idea is to pick a random bit string as secret key sk, then derive an encryption/decryption keypair (ek, dk) of PKE in a deterministic manner. The encryption key ek is used for both encrypting plaintexts and verifying signatures, and hence will be denoted by pk. The decryption key is only used for decrypting. The secret key sk is used for signing messages and deriving the decryption key dk. The key derivation should be one-way, namely, one can derive the decryption key from the signing key, but not vice versa. Thus, the signing key acts as master secret key. Let the randomness space R of PKE's key generation algorithm be $\{0,1\}^{\ell}$, we describe the generic construction as below.

- $\mathsf{Setup}(1^\lambda)$: runs $pp_{\mathrm{pke}} \leftarrow \mathrm{PKE}.\mathsf{Setup}(1^\lambda)$, $pp_{\mathrm{zkpok}} \leftarrow \mathrm{ZKPoK}.\mathsf{Setup}(1^\lambda)$, picks a uniform OWF $\mathsf{F} : \{0,1\}^n \to \{0,1\}^\ell$, outputs $pp = (pp_{\mathrm{pke}}, pp_{\mathrm{zkpok}}, \mathsf{F})$.
- $\mathsf{KeyGen}(pp)$: parses $pp = (pp_{\mathrm{pke}}, pp_{\mathrm{zkpok}}, \mathsf{F})$, picks $sk \xleftarrow{\mathrm{R}} \{0,1\}^n$, computes $r \leftarrow \mathsf{F}(sk)$, runs $(ek, dk) \leftarrow \mathrm{PKE}.\mathsf{KeyGen}(pp_{\mathrm{pke}}; r)$, outputs public key $pk = ek$ and secret key sk. Let PK be the public key space.
- $\mathsf{Derive}(sk)$: this algorithm is exactly a part of KeyGen, i.e., on input sk, computes $r \leftarrow \mathsf{F}(sk)$, runs $(ek, dk) \leftarrow \mathrm{PKE}.\mathsf{KeyGen}(pp_{\mathrm{pke}}; r)$, outputs the resulting decryption key dk.
- $\mathsf{Enc}(pk, m)$ and $\mathsf{Dec}(dk, c)$ are same as those of the underlying PKE.
- $\mathsf{Sign}(sk, \widetilde{m})$: Let G be $\mathrm{PKE}.\mathsf{KeyGen}_1$, i.e., the algorithm that outputs the first outcome pk of $\mathrm{PKE}.\mathsf{KeyGen}$. G and F induce an $\mathcal{NP}$ relation $\mathsf{R}_{\mathrm{key}}$ over $PK \times \{0,1\}^n$ defined as below.

$$\mathsf{R}_{\mathrm{key}} = \{(pk, sk) \mid pk = \mathsf{G}(\mathsf{F}(sk))\} \tag{1}$$

  We are thus able to build a signature scheme with sk as the signing key and pk as the verification key from a three-round public-coin ZKPoK for $\mathsf{R}_{\mathrm{key}}$.
  1. Run the prover algorithm $P(sk)$ with randomness α to sample a random element a from the initial message space A. We assume that $|A|$ is exponential in λ.
  2. Hash a with the message $\widetilde{m}$ to be signed into the challenge, i.e., $e \leftarrow \mathsf{H}(a, \widetilde{m})$. Here, H is a cryptographic hash function, which is modeled as a random oracle.
  3. Run the prover algorithm $P(sk, \alpha, e)$ to generate a response z.

  Finally, outputs the signature $\sigma = (a, z)$ for $\widetilde{m}$.

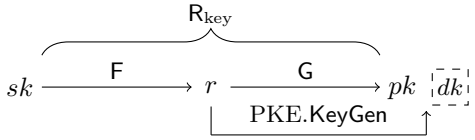

Fig. 2: The hierarchical key structure

- $\mathsf{Vrfy}(pk, \widetilde{m}, \sigma)$: on input a public key pk, a message $\widetilde{m}$ and a signature $\sigma = (a, z)$, first recovers the challenge $e \leftarrow \mathsf{H}(a, \widetilde{m})$, then runs the verifier's verification algorithm $V(a, e, z)$ to decide if (a, e, z) is an accepting transcript w.r.t. $\mathsf{R}_{\mathrm{key}}$.

In the above construction, the signature generation follows the same routine of crushing the ZKPoK into a non-interactive one via Fiat-Shamir heuristic. Thus, we can simplify the syntax of the construction by describing the signing procedure as $\mathrm{NIZKPoK}.\mathsf{Prove}(pk, sk, \tilde{m})$ and the verifying procedure as $\mathrm{NIZKPoK}.\mathsf{Verify}(pk, \tilde{m}, \sigma)$, where pk serves as the instance, sk serves as the witness, $\tilde{m}$ is treated as auxiliary input, and σ serves as the proof.

The correctness of the above construction follows from those of the underlying PKE and ZKPoK. For the security, we have the following theorem.

Theorem 2. *The above HISE construction is jointly secure assuming the security of its building blocks and modeling* H *as a random oracle.*

Due to space limit, we defer the security proof to the full version.

6 Global Escrow PKE

As discussed in the introduction, HISE naturally supports individual key escrow mechanism, but may not satisfy global key escrow property. To investigate how to further support global escrow mechanism for HISE in a general manner, next we make a little detour to revisit the topic of global escrow PKE, with focus on formal definition and generic construction. The obtained results can be used in a mixed way with the results in Section 4, yielding global escrow HISE.

Global escrow PKE is an extension of PKE. In global escrow PKE, there is a single global escrow decryption key that enables the decryption of ciphertexts encrypted under any public key. Such scheme enables government intelligence and law enforcement agencies to reveal encrypted information without the knowledge or consent of users. Formally, a global escrow PKE consists of five polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Dec}')$. KeyGen, Enc, and Dec are the same as those of ordinary PKE. The Setup algorithm outputs an additional escrow decryption key, while Dec' can decrypt ciphertexts under any public key using this escrow decryption key.

- $\mathsf{Setup}(1^\lambda)$: on input the security parameter λ, outputs global public parameters pp and a global escrow decryption key edk. This algorithm is run by a trusted party.
- $\mathsf{Dec}'(edk, c)$: on input an escrow decryption key edk and a ciphertext c, outputs a plaintext m or a special reject symbol $\perp$ denoting failure.

In most applications of global escrow PKE, the escrow agent needs to know the public key of the intended receiver. Therefore, we assume that the public key of the intended receiver is always provided in the clear from ciphertext.

Correctness. For all $m \in M$, we have $\Pr[\mathsf{Dec}(sk, c) = m = \mathsf{Dec}'(edk, c)] \geq 1 - \mathsf{negl}(\lambda)$, where the probability is taken over the choice of $(pp, edk) \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, and $c \leftarrow \mathsf{Enc}(pk, m)$.

Consistency. The definition of correctness stipulates that the decryption results of the receiver and the escrow agent are identical when the ciphertexts are honestly generated. In applications of escrow PKE, the sender may generate the ciphertexts dishonestly to evade supervision. Therefore, in addition to correctness, we also need to consider the notion of consistency for global escrow PKE. The intuition is that the decryption results of the receiver and the escrow agent are still identical when the ciphertexts are dishonestly generated. Fix pp, we define a collection of $\mathcal{NP}$ languages indexed public key, namely,

$L_{pk} = \{c \mid \exists m, r \text{ s.t. } c = \mathsf{Enc}(pk, m; r)\}$, which represents the set of all valid ciphertexts encrypted under pk. We are now ready to formally define consistency. For an adversary $\mathcal{A}$ against consistency, we define its advantage function as:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} c \notin L_{pk} \wedge \\ \mathsf{Dec}(sk, c) \neq \mathsf{Dec}'(edk, c) \end{array} : \begin{array}{l} (pp, edk) \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ c \leftarrow \mathcal{A}(pp, pk); \end{array}\right].$$

A global escrow PKE is computationally (resp. statistically) consistent if no PPT (resp. unbounded) adversary has non-negligible advantage in the above experiment.

Security. Let $\mathcal{A}$ be an adversary against global escrow PKE and define its advantage in the following experiment.

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[b = b' : \begin{array}{l} (pp, edk) \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}(\cdot)}(pp, pk); \\ b \xleftarrow{\mathrm{R}} \{0,1\}, c^* \leftarrow \mathsf{Enc}(pk, m_b); \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}(\cdot)}(pp, pk, c^*); \end{array}\right] - \frac{1}{2}.$$

Here, $\mathcal{O}_{\mathsf{dec}}(\cdot)$ is the decryption oracle. $\mathcal{A}$ can make polynomial number of decryption queries with the restriction that $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{dec}}(\cdot)$ with c^* in Phase 2. A global escrow PKE scheme is IND-CCA secure if no PPT adversary has non-negligible advantage in the above experiment. We can define IND-CCA1 security (resp. IND-CPA security) similarly by only giving $\mathcal{A}$ access to $\mathcal{O}_{\mathsf{dec}}(\cdot)$ in Phase 1 (resp. denying access to $\mathcal{O}_{\mathsf{dec}}(\cdot)$).

6.1 Global Escrow PKE from PKE and NIZK

At first glance, it seems that global escrow PKE is trivially implied by broadcast encryption by having the receiver set include the public keys of the intended receiver and the escrow agent. However, the consistency of this construction is not guaranteed since broadcast encryption always assume that the sender generates ciphertexts honestly.

Next, we show how to make any PKE scheme satisfy global escrow property by leveraging NIZK. The idea is that when building up the system the escrow agent generates a keypair (pk_γ, sk_γ) himself, and then includes his public key pk_γ in the public parameters and uses the secret key sk_γ as escrow decryption key. To send an encrypted message to receiver with public key pk, the sender encrypts the same plaintext m twice under pk and pk_γ independently, then appends a NIZK proof for the consistency of encryption. To decrypt the ciphertext, both the receiver and the escrow agent first check the correctness of NIZK proof, then decrypts the corresponding part using their secret keys. Our construction coincides with the celebrated Naor-Yung double encryption paradigm for chosen-ciphertext security. In the Naor-Yung paradigm, the two public keys belong to the receiver, and the NIZK proof is used to achieve CCA security. In

our case, one public key belongs to the receiver, the other key belongs to the escrow agent, and the NIZK proof is used to the ensure that the escrow agent has the same decryption capability as the receiver. Our construction is somewhat dual to previous solutions [YY98, YY99, PY99]. Rather than providing a proof of key recoverability to CA when registering public key, our construction provides a proof of correct encryption each time when generating ciphertexts. The advantage of our construction is that it removes the need of recoverability certificate entirely, and efficient zero-knowledge proof is relatively easy to design for most PKE schemes. Moreover, if we aim for CCA security, then the added zero-knowledge proofs do not constitute extra overhead.

For completeness, we present our construction as below.

- $\mathsf{Setup}(1^\lambda)$: runs $pp_{\text{pke}} \leftarrow \text{PKE.}\mathsf{Setup}(1^\lambda)$, $(pk_\gamma, sk_\gamma) \leftarrow \text{PKE.}\mathsf{KeyGen}(pp_{\text{pke}})$, $pp_{\text{nizk}} \leftarrow \text{NIZK.}\mathsf{Setup}(1^\lambda)$, generates $crs \leftarrow \text{NIZK.}\mathsf{CRSGen}(pp_{\text{nizk}})$, outputs $pp = (pp_{\text{pke}}, pp_{\text{nizk}}, crs, pk_\gamma)$ and $edk = sk_\gamma$.
- $\mathsf{KeyGen}(pp)$: parses $pp = (pp_{\text{pke}}, pp_{\text{nizk}}, crs, epk)$, then outputs $(pk, dk) \leftarrow \text{PKE.}\mathsf{KeyGen}(pp_{\text{pke}})$.
- $\mathsf{Enc}(pk, m)$: picks two random coins r_1 and r_2 independently, computes $c_1 \leftarrow \text{PKE.}\mathsf{Enc}(pk, m; r_1)$ and $c_2 \leftarrow \text{PKE.}\mathsf{Enc}(pk_\gamma, m; r_2)$, then generates $\pi \leftarrow \text{NIZK.}\mathsf{Prove}(crs, (pk, c_1, c_2), (r_1, r_2, m))$, outputs $c = (pk, c_1, c_2, \pi)$. Here, π is a proof for (c_1, c_2) being encryptions of the same plaintext under pk and pk_γ, i.e., $(pk, c_1, c_2) \in L_{pk}$, where L_{pk} is defined as below:

$$L_{pk} = \{(pk, c_1, c_2) \mid \exists m, r_1, r_2 \text{ s.t.}$$
$$c_1 = \text{PKE.}\mathsf{Enc}(pk, m; r_1) \wedge c_2 = \text{PKE.}\mathsf{Enc}(pk_\gamma, m; r_2)\}$$

- $\mathsf{Dec}(sk, c)$: on input a decryption key sk and a ciphertext $c = (pk, c_1, c_2, \pi)$, first runs $\text{NIZK.}\mathsf{Verify}(crs, (pk, c_1, c_2), \pi)$ to check if c is a valid encryption under pk; if the check fails then returns $\perp$, else returns $m \leftarrow \text{PKE.}\mathsf{Dec}(dk, c_1)$.
- $\mathsf{Dec}'(edk, c)$: on input a global escrow decryption key $edk = sk_\gamma$ and a ciphertext $c = (pk, c_1, c_2, \pi)$, first checks if c is a valid encryption under pk_γ by running $\text{NIZK.}\mathsf{Verify}(crs, (pk_\gamma, c_1, c_2), \pi)$; if the check fails then returns $\perp$, else returns $m \leftarrow \text{PKE.}\mathsf{Dec}(sk_\gamma, c_2)$.

The correctness follows from that of PKE and NIZK, and the consistency holds based on the adaptive soundness of the underlying NIZK. For the security, we have the following theorem.

Theorem 3. *The above construction of global escrow PKE is CCA1-secure (resp. CCA-secure) if the underlying PKE is CPA-secure and the NIZK is adaptively secure (resp. simulation sound adaptive secure).*

Proof. The security proofs are very similar to those for Naor-Yung construction [NY90] and Sahai construction [Sah99]. We omit the details here.

Remark 3. The above generic construction encrypts the plaintext twice independently under the public keys of the intended receiver and the escrow agent. When the underlying PKE satisfies a mild property called "randomness fusion", we can safely reuse the random coins and apply twisted Naor-Yung transform [BMV16], leading to improvements in terms of both efficiency and bandwidth.

6.2 Global Escrow PKE from Three-party NIKE and SKE

In this section, we present another generic construction of global escrow PKE from three-party NIKE and SKE. This construction follows the KEM-DEM paradigm. We start by defining the notion of global escrow KEM by adapting KEM to the escrow setting. A global escrow KEM consists of five polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps}, \mathsf{Decaps}')$. The KeyGen, Encaps, and Decaps algorithms are same as those of an ordinary KEM. The Setup algorithm outputs an additional escrow decryption key, while Decaps' decapsulates ciphertexts using this escrow decryption key.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, outputs global public parameters pp and a global escrow decryption key edk. This algorithm is run by a trusted party. We assume that pp includes the description of session key space K.
- $\mathsf{Decaps}'(edk, c)$: on input a global escrow decryption key edk and a ciphertext c, outputs a session key k or a special reject symbol $\perp$ denoting failure.

Correctness. We require that $\Pr[\mathsf{Decaps}(sk, c) = k = \mathsf{Decaps}'(edk, c)] \geq 1 - \mathsf{negl}(\lambda)$, where the probability is taken over the choice of $(pp, edk) \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, and $(c, k) \leftarrow \mathsf{Encaps}(pk)$.

Consistency. Analogous to the setting of global escrow PKE, we also need to consider the notion of consistency for global escrow KEM. Fix pp, we define a collection of $\mathcal{NP}$ languages indexed by pk. Let $L_{pk}^{\mathrm{kem}} = \{c \mid \exists r \text{ s.t. } (c, k) = \mathsf{Encaps}(pk; r)\}$, which represents all valid ciphertexts encapsulated under pk. We are now ready to define consistency. For an adversary $\mathcal{A}$ against consistency, we define its advantage function as:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} c \notin L_{pk}^{\mathrm{kem}} \wedge \\ \mathsf{Decap}(sk, c) \neq \mathsf{Decap}'(edk, c) \end{array} : \begin{array}{l} (pp, edk) \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ c \leftarrow \mathcal{A}(pp, pk); \end{array}\right].$$

We say that a global escrow KEM is computationally (resp. statistically) consistent if no PPT (resp. unbounded) adversary has non-negligible advantage in the above experiment.

Security. Let $\mathcal{A}$ be an adversary against global escrow KEM and define its advantage in the following experiment.

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[b = b' : \begin{array}{l} (pp, edk) \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ (c^*, k_0^*) \leftarrow \mathsf{Encaps}(pk), k_1^* \leftarrow K; \\ b \xleftarrow{\mathrm{R}} \{0, 1\}; \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{decaps}}(\cdot)}(pp, pk, c^*, k_b^*); \end{array}\right] - \frac{1}{2}.$$

Here, $\mathcal{O}_{\mathsf{decaps}}(\cdot)$ denotes the decapsulation oracle. $\mathcal{A}$ can make polynomial number of such queries with the restriction that $c \neq c^*$, and the challenger responds with $k \leftarrow \mathsf{Decaps}(sk, c)$. A global escrow KEM is IND-CCA secure if no PPT adversary has non-negligible advantage in the above experiment. A global escrow KEM is IND-CPA secure if no PPT adversary has non-negligible advantage in the same experiment but denying access to $\mathcal{O}_{\mathsf{decaps}}(\cdot)$.

6.2.1 Global Escrow PKE from Global Escrow KEM and SKE

We build global escrow PKE from global escrow KEM and SKE as below.

- $\mathsf{Setup}(1^\lambda)$: runs $(pp_{\mathrm{kem}}, edk) \leftarrow \mathrm{KEM}.\mathsf{Setup}(1^\lambda)$, $pp_{\mathrm{ske}} \leftarrow \mathrm{SKE}.\mathsf{Setup}(1^\lambda)$, outputs $pp = (pp_{\mathrm{kem}}, pp_{\mathrm{ske}})$ and edk.
- $\mathsf{KeyGen}(pp)$: parses public parameters $pp = (pp_{\mathrm{kem}}, pp_{\mathrm{ske}})$, outputs $(pk, sk) \leftarrow \mathrm{KEM}.\mathsf{KeyGen}(pp_{\mathrm{kem}})$.
- $\mathsf{Enc}(pk, m)$: computes $(c_{\mathrm{kem}}, k) \leftarrow \mathrm{KEM}.\mathsf{Encaps}(pk)$, $c_{\mathrm{ske}} \leftarrow \mathrm{SKE}.\mathsf{Enc}(k, m)$, outputs $c = (c_{\mathrm{kem}}, c_{\mathrm{ske}})$.
- $\mathsf{Dec}(sk, c)$: parses $c = (c_{\mathrm{kem}}, c_{\mathrm{ske}})$, computes $k \leftarrow \mathrm{KEM}.\mathsf{Decaps}(sk, c_{\mathrm{ske}})$; if $k = \perp$ outputs $\perp$, else outputs $m \leftarrow \mathrm{SKE}.\mathsf{Dec}(k, c_{\mathrm{ske}})$.
- $\mathsf{Dec}'(edk, c)$: parses $c = (c_{\mathrm{kem}}, c_{\mathrm{ske}})$, computes $k \leftarrow \mathrm{KEM}.\mathsf{Decaps}'(edk, c_{\mathrm{ske}})$; if $k = \perp$ outputs $\perp$, else outputs $m \leftarrow \mathrm{SKE}.\mathsf{Dec}(k, c_{\mathrm{ske}})$.

The correctness follows from that of global escrow KEM and SKE. We analyze the consistency requirement as below. The above construction follows the KEM-DEM approach. Fix pp, we define a collection of $\mathcal{NP}$ languages indexed by pk. Let $L_{pk} = \{(c_{\mathrm{kem}}, c_{\mathrm{ske}}) \mid \exists m, r_1, r_2 \text{ s.t. } (c_{\mathrm{kem}}, k) = \mathrm{KEM}.\mathsf{Encaps}(pk; r_1) \wedge c_{\mathrm{ske}} = \mathrm{SKE}.\mathsf{Enc}(k, m; r_2)\}$. It is easy to see that no matter whether $c_{\mathrm{kem}} \in L_{pk}^{\mathrm{kem}}$ or not, the consistency of global escrow KEM guarantees that the decapsulation results are identical, and so are the final decryption results.

Theorem 4. *The above construction is IND-CPA secure (resp. IND-CCA secure) if the underlying global escrow KEM is IND-CPA secure (resp. IND-CCA secure) and the SKE is IND-CPA secure (resp. IND-CCA secure).*

Proof. The security proof is similar to that of PKE from the KEM-DEM methodology. We omit the details here.

6.2.2 Global Escrow KEM from Three-Party NIKE

We present a generic construction of global escrow KEM from three-party NIKE. The high-level idea is that the escrow agent generates a keypair (pk_γ, sk_γ), then publishes pk_γ as part of the public parameters and keeps sk_γ to itself. To send a ciphertext to the receiver with public key $pk = pk_\beta$, the sender generates a random keypair (pk_α, sk_α), then runs the three-party NIKE in his head to derive a shared key for $\{pk_\alpha, pk_\beta, pk_\gamma\}$, and finishes encapsulation by setting pk_α as the ciphertext and the shared key as the session key. According to the functionality and security of NIKE, both the escrow agent and the receiver can derive the same session key, which is pseudorandom in any PPT adversary's view. The construction is as below.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, runs $pp_{\mathrm{nike}} \leftarrow \mathrm{NIKE}.\mathsf{Setup}(1^\lambda)$ and $(pk_\gamma, sk_\gamma) \leftarrow \mathrm{NIKE}.\mathsf{KeyGen}(pp_{\mathrm{nike}})$, outputs public parameters $pp = (pp_{\mathrm{nike}}, pk_\gamma)$ and sets the global escrow decryption key $edk = sk_\gamma$.
- $\mathsf{KeyGen}(pp)$: parses $pp = (pp_{\mathrm{nike}}, pk_\gamma)$, runs $\mathrm{NIKE}.\mathsf{KeyGen}(pp_{\mathrm{nike}})$ to generate a keypair (pk, sk).

- $\mathsf{Encaps}(pk)$: parses $pk = pk_\beta$, the sender runs NIKE.$\mathsf{KeyGen}(pp_{\text{nike}})$ to generate a random keypair (pk_α, sk_α), sets $S = \{pk_\alpha, pk_\beta, pk_\gamma\}$, computes $k_S \leftarrow \mathsf{ShareKey}(sk_\alpha, S)$, outputs ciphertext $c = (pk_\alpha, pk_\beta)$ and session key $k = k_S$. The language for valid encapsulation is: $L_{pk}^{\text{KEM}} = \{(pk_\alpha, pk) \mid pk_\alpha \in PK\}$.
- $\mathsf{Decaps}(sk, c)$: on input a secret key $sk = sk_\beta$ and a ciphertext $c = (pk_\alpha, pk_\beta)$, first sets $S = \{pk_\alpha, pk_\beta, pk_\gamma\}$, then computes $k_S \leftarrow \mathsf{ShareKey}(sk_\beta, S)$ and outputs session key $k = k_S$.
- $\mathsf{Decaps}'(edk, c)$: on input $edk = sk_\gamma$ and a ciphertext $c = (pk_\alpha, pk_\beta)$, sets $S = \{pk_\alpha, pk_\beta, pk_\gamma\}$, then computes $k_S \leftarrow \mathsf{ShareKey}(sk_\gamma, S)$ and outputs session key $k = k_S$.

The correctness and consistency of global escrow KEM follow from those of the underlying three-party NIKE. For security, we have the following theorem.

Theorem 5. *If the three-party NIKE is CKS-light secure in the HKR setting (resp. in the DKR setting), then the resulting global escrow KEM is IND-CPA secure (resp. IND-CCA secure).*

Due to space limit, we defer the security proof to the full version.

6.2.3 Relaxation of Three-Party NIKE

We note that the above construction of global escrow KEM does not require the full power of three-party NIKE. In fact, a relaxed version suffices for our purpose, a.k.a., there are three types of public keys in the system (say Type-A, Type-B and Type-C), and the shared key can be agreed upon if the three participants hold different types of public keys. When building global escrow KEM, we can set user's public key as Type-A, the temporary public key as Type-B (serves as the ciphertext), and the escrow agent's public key as Type-C (serves as part of the public parameters). This relaxation increases the space of the underlying protocols that can be used, and hence can potentially lead to more efficient construction of global escrow PKE. Next, we show how to build an efficient global escrow KEM from a relaxed version of Joux's protocol [Jou04] to exemplify the power of this conceptual insight.

As noticed by [GPS08, AGH15], there is a huge gap in pairing-based cryptography: schemes are usually presented in the academic literature via symmetric pairing because it is simpler and the complexity assumptions can be weaker, while schemes are preferable to be implemented via asymmetric pairing (notably Type-III pairing) since it is the most efficient choice in terms of bandwidth and computation time. Such gap also occurs in our case. The original Joux's protocol is based on symmetric pairing and cannot be easily adapted to the setting of asymmetric pairing. Consequently, it does not lend itself to an efficient global escrow KEM. We fill this gap by observing that the relaxed version of Joux's protocol described above can be realized using Type-III pairing under the co-DBDH assumption. Towards minimizing the public key size of the resulting global escrow KEM, we adapt the original Joux's protocol by designating Type-A public key of the form $g_1^b \in \mathbb{G}_1$, Type-B public key of the form $g_2^c \in \mathbb{G}_2$, and Type-C

public key of the form $(g_1^a, g_2^a) \in \mathbb{G}_1 \times \mathbb{G}_2$. This yields a global escrow KEM (and hence a global escrow PKE) from asymmetric pairing based on the co-DBDH assumption. See Section 8 for comparison with the only known prior work called escrow ElGamal PKE [BF03].

7 Instantiations

In this section, we present instantiations of our two generic HISE constructions (described in Section 4 and 5) and two generic global escrow HISE constructions (yielded by mixing the general approaches for building HISE and global escrow PKE). We limit ourselves to discrete-log/pairing-based realizations since factoring-based and lattice-based realizations suffer from large key size.

7.1 Instantiation of HISE from IBE

We instantitate our first HISE construction (presented in Section 4) by choosing Boneh-Franklin IBE with asymmetric pairing as the underlying IBE scheme, yielding HISE scheme 1 as below.

- $\mathsf{Setup}(1^\lambda)$: runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e) \leftarrow \mathsf{BLGroupGen}(1^\lambda)$, picks $g_1 \xleftarrow{\mathrm{R}} \mathbb{G}_1$, sets $id^* = 1^{\ell+1}$, outputs $pp = id^*$. We assume that pp also includes the descriptions of bilinear groups and a hash function $\mathsf{H} : \{0,1\}^{\ell+1} \to \mathbb{G}_2$.
- $\mathsf{KeyGen}(pp)$: on input $pp = id^*$, picks $sk \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, computes $pk = g_1^{sk} \in \mathbb{G}_1$.
- $\mathsf{Derive}(sk)$: on input sk, outputs $dk = \mathsf{H}(id^*)^{sk} \in \mathbb{G}_2$.
- $\mathsf{Enc}(pk, m)$: on input pk and $m \in \mathbb{G}_T$, picks $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, computes $c_1 \leftarrow g_1^r \in \mathbb{G}_1$ and $c_2 \leftarrow e(pk, \mathsf{H}(id^*))^r \cdot m$, outputs $c = (c_1, c_2)$.
- $\mathsf{Dec}(dk, c)$: on input dk and c, outputs $m = c_2 / e(c_1, dk)$.
- $\mathsf{Sign}(sk, \tilde{m})$: on input sk and $\tilde{m} \in \{0,1\}^\ell$, outputs $\sigma = \mathsf{H}(0||\tilde{m})^{sk} \in \mathbb{G}_2$.
- $\mathsf{Vrfy}(pk, \tilde{m}, \sigma)$: picks $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, outputs "1" if $e(pk, \mathsf{H}(0||\tilde{m}))^r = e(g_1^r, \sigma)$ and "0" otherwise.

Remark 4. HISE scheme 1 is obtained by faithfully applying the generic transform to the Boneh-Franklin IBE. We note that in this case the Vrfy algorithm could be simplified by directly checking if $e(pk, \mathsf{H}(0||\tilde{m})) = e(g_1, \sigma)$, the resulting the signature component is exactly the Boneh-Lynn-Shacham signature [BLS01] from the asymmetric pairing.

We realize HISE scheme 1 atop pairing-friendly curve `bls12-381` with 128-bit security level [SKSW20][12], in which $|\mathbb{G}_1| = 48$ bytes, $|\mathbb{G}_2| = 96$ bytes, $|\mathbb{Z}_p| = 32$ bytes, and $|\mathbb{G}_T| = 191$ bytes (by exploiting compression techniques [RS08]).

[12] Recent security evaluations show that the security level of `bls12-381` is close to but less than 128-bit. As curves of 128-bit security level are currently the most widely used, `BLS12-381` and `BN462` are recommended in the memo [SKSW20] in order to have a more efficient and a more prudent option respectively.

7.2 Instantiation of HISE from PKE and ZKPoK

Public-key encryption. We choose the ElGamal PKE as the starting PKE scheme. The randomness space R for KeyGen is $\mathbb{Z}_p$. The KeyGen algorithm on input $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ outputs $sk = r$ and $pk = g^r$. Thus, $\mathsf{G} : \mathbb{Z}_p \to \mathbb{G}$ is defined as $r \mapsto g^r$. Clearly, G is injective, and thus it is unconditionally target-collision resistant. We assume that there is a one-to-one mapping from $\{0,1\}^\ell$ to $\mathbb{Z}_p$ for some integer ℓ. Concretely, we choose the elliptic curve `secp256k1` with 128-bit security. We demonstrate the generality of our second HISE construction by providing two more eligible PKE candidates (see the full version for the details).
Uniform one-way function. After fixing $R = \{0,1\}^\ell$, we choose a one-way function H from $\{0,1\}^n$ to $\{0,1\}^\ell$. A popular choice is using hash function like SHA-256, in which the number of AND gates of a single call is about 25000. Motivated by applications in FHE schemes, MPC protocols and SNARKs, recently there is a trend to design lightweight symmetric encryption primitives with a low number of multiplications or a low multiplicative depth. In our instantiation, we choose the POSEIDON-128 hash [GKR+21], whose number of rank-1 constraint satisfiability (R1CS) constraints is roughly 300.
General purpose ZKPoK. Due to the involvement of F, $\mathsf{R}_{\mathrm{key}}$ defined by $\mathsf{G} \circ \mathsf{F}$ is unlikely to be an algebraic relation. As a consequence, it is difficult to prove $\mathsf{R}_{\mathrm{key}}$ using simple Sigma protocols. Our solution is to resort efficient general purpose public-coin ZKPoK protocols. A flurry of recent work on zk-SNARKs with transparent setup offers plenty of candidates, including the backbone protocols that underlie almost all the known zk-SNARKs in the random oracle model. In our instantiation, we choose Spartan [Set20]. We convert the proved relation $\mathsf{R}_{\mathrm{key}}$ into R1CS format using xJsnark [KPS18]; the number of R1CS constraints of is roughly $680,000 \approx 2^{20}$.

We are now ready to instantiate our second HISE construction (presented in Section 5) from the above building blocks, yielding HISE scheme 2.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, runs $(\mathbb{G}, g, p) \leftarrow \mathsf{GroupGen}(1^\lambda)$, picks a uniform one-way function $\mathsf{F} : \{0,1\}^n \to \{0,1\}^\ell$, runs $pp_{\mathrm{nizkpok}} \leftarrow \mathsf{NIZKPoK.Setup}(1^\lambda)$, and outputs $pp = (\mathsf{F}, pp_{\mathrm{nizkpok}})$. The plaintext space is $M = \mathbb{G}$. The message space is $\widetilde{M} = \{0,1\}^*$.
- $\mathsf{KeyGen}(pp)$: on input $pp = (\mathsf{F}, pp_{\mathrm{nizkpok}})$, picks $sk \xleftarrow{\mathrm{R}} \{0,1\}^n$, computes $pk = g^{\mathsf{F}(sk)} \in \mathbb{G}$.
- $\mathsf{Derive}(sk)$: on input sk, outputs $dk \leftarrow \mathsf{F}(sk) \in \mathbb{Z}_p$.
- $\mathsf{Enc}(pk, m)$: on input pk and $m \in \mathbb{G}$, picks $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, computes $X \leftarrow g^r \in \mathbb{G}$, $Y \leftarrow pk^r \cdot m$, outputs $C = (X, Y)$.
- $\mathsf{Dec}(dk, c)$: on input dk and $C = (X, Y)$, outputs $m \leftarrow Y / X^{dk}$.
- $\mathsf{Sign}(sk, \tilde{m})$: computes $\sigma \leftarrow \mathrm{NIZKPoK.Prove}(pk, sk, \tilde{m})$.
- $\mathsf{Vrfy}(pk, \tilde{m}, \sigma)$: on input pk, $\tilde{m}$ and σ, outputs $b \leftarrow \mathrm{NIZKPoK.Verify}(pk, \sigma, \tilde{m})$.

7.3 Two Instantiations of Global Escrow HISE

As depicted in Figure 1 in the introduction part, there are two paths to build global escrow HISE. Our first construction is along the path enabled by the

GE conversion. Starting from the HISE scheme presented in Section 7.1, we compile it into a global escrow one by applying the twisted Naor-Yung transform [BMV16], yielding global escrow HISE scheme 1. Our second construction is along the path enabled by the HI conversion. Starting from the global escrow PKE based a relaxed version of Joux's three-party NIKE (sketched in Section 6.2.3), we add the signing functionality via the HI conversion, yielding global escrow HISE scheme 2. The joint security of the above two schemes follows from the fact that the signing key is independent of the global escrow decryption key.

Due to space limit, we defer the specification global escrow HISE scheme 1/2 to the full version.

8 Comparison and Evaluation

This section compares (global escrow) HISE with CPK and ISE in terms of security and functionality properties, then evaluates our instantiations of (global escrow) HISE and global escrow PKE.

8.1 Comparison of Security and Functionality Properties

Paterson et al. [PSST11] introduce a "Cartesian product" construction of CPK (henceforth CP-CPK for short). The construction uses arbitrary encryption and signature schemes as components, runs the key generation algorithms independently, then concatenates the keypairs of the encryption scheme and signature scheme, and uses the appropriate component of the compound keypair for each operation. CP-CPK best formalizes the principle of key separation, and hence also naturally supports individual key escrow. We choose it as a baseline to judge (global escrow) HISE schemes that use the principle of key reuse.

Table 1 offers a comparision of (global escrow) HISE against previous CP-CPK and ISE in terms of security and functionality properties as well as certificate cost. The results show that HISE supports individual key escrow in the context of key reuse, while global escrow HISE further supports global key escrow. Besides, we highlight that CP-CPK doubles the certificate cost, which should be minimized in practice.

8.2 Efficiency Evaluation of (Global Escrow) HISE

Baseline. We build a concrete CP-CPK scheme atop elliptic curve `secp256k1` with 128-bit security (where $|\mathbb{G}| = 33$ bytes, $|\mathbb{Z}_p| = 32$ bytes) as a baseline. More precisely, we choose ElGamal PKE as the encryption component and Schnorr signature as the signature component, because they are among the most efficient elliptic-curve based cryptosystems with short public keys.

Methodology. We implement the CP-CPK scheme and our (global escrow) HISE instantiations in C++ based on the `mcl` library [Shi]. Parameters of all schemes are set to achieve 128-bit security level. All experiments are carried on

Table 1: Comparison between CP-CPK, ISE, and our (global escrow) HISE

Scheme	strong joint security	individual escrow	global escrow	key reuse	certificate cost
CP-CPK [PSST11]	✓	✓	✗	✗	$\times 2$
ISE [PSST11]	✗	✗	✗	✓	$\times 1$
HISE	✓	✓	✗	✓	$\times 1$
global escrow HISE	✓	✓	✓	✓	$\times 1$

For certificate cost, $\times 1$ (resp. $\times 2$) means the cost associated with one (resp. two) certificate(s). As aforementioned, certificate costs include but not limit to registration, issuing, storage, transmission, verification, and building/recurring fees. Take SSL certificate as an example, one certificate is roughly 1KB, takes roughly 200∼300ms to transmit in WAN setting with 50Mbps network bandwidth and 8ms to verify. The monetary cost for an SSL certificate varies depending on features and business needs. While the cost of an SSL certificate for common usage is \$10∼\$2000/year, the banks and large financial institutions could spend up to \$500,000/year on an SSL certificate with high-level security guranttee.

a MacBook Pro with Intel i7-9750H CPU (2.6GHz) and 16GB of RAM. We view the key size and the associated certificate cost as the primary metric of interest. The experimental results are presented in Table 2. As shown in this table, our (global escrow) HISE schemes have more compact key size than the CP-CPK in both asymptotic and concrete sense. Among the five schemes, global escrow HISE scheme 1 achieves joint security, while the rest schemes achieve weak joint security (the encryption component is CPA-secure).

The ciphertext size of HISE scheme 1 and global escrow HISE scheme 1 and 2 are slightly large. Nevertheless, this is not a big issue since in real-world applications long plaintexts are typically encrypted using hybrid encryption, thereby the overhead of the PKE ciphertext can be greatly amortized. The signature components of (global escrow) HISE scheme 2 are less efficient due to the involvement of general-purpose ZKPoK for large-size circuit describing the composite relation $\mathsf{R}_{\mathrm{key}}$. We hence regard (global escrow) HISE scheme 2 more of theoretical interest for the time being. We leave how to improve the efficiency as an interesting problem. A possible solution is to adapt the techniques of creating efficient NIZK for composite statement [AGM18] to the public-coin setting.

8.3 Comparison of Global Escrow PKE

As a byproduct, we obtain a global escrow PKE, which serves as the starting point of our global escrow HISE 2. Our scheme (see details in the full version) can be viewed as an adaption of Boneh-Franklin escrow ElGamal PKE [BF03, Section 7] to the setting of asymmetric pairing, and hence enjoys much better efficiency. While this may appear straightforward in hindsight, we stress again that the adaptation is non-trivial, which is leaded by our observation that global escrow PKE can be derived from a relaxed version of three-party NIKE (see discussions in Section 6.2.3).

[13] The frontend of a ZK proof system provides means to express statements in high-level language and compile them into low-level representation (e.g., rank 1 constraint system), then invokes a suitable ZK backend.

Table 2: Efficiency comparison of CPK and our proposed (global escrow) HISE schemes

Scheme	efficiency (ms) [# exp, #pairing]							sizes (bytes) [# $\mathbb{G}$, # $\mathbb{Z}_p$]			
	KGen	Sign	Vrfy	Enc	Dec	Der	Dec′	$\lvert pk\rvert$	$\lvert sk\rvert$	$\lvert c\rvert$	$\lvert\sigma\rvert$
CP-CPK	0.015	0.064	0.120	0.118	0.056	⊘	⊘	66	64	66	65
	[2, 0]	[1, 0]	[2, 0]	[2, 0]	[1, 0]	⊘	⊘	$2\mathbb{G}$	$2\mathbb{Z}_p$	$2\mathbb{G}$	$[\mathbb{G}, \mathbb{Z}_p]$
HISE scheme 1	0.057	0.148	0.733	0.569	0.364	0.148	⊘	48	32	239	96
	[1, 0]	[1, 0]	[0, 2]	[2, 1]	[0, 1]	[1, 0]	⊘	$\mathbb{G}_1$	$\mathbb{Z}_p$	$[\mathbb{G}_1, \mathbb{G}_T]$	$\mathbb{G}_2$
HISE scheme 2	0.058	3.5s	250	0.115	0.056	0.0004	⊘	33	32	66	40K
	[1, 0]	N/A	N/A	[2, 0]	[1, 0]	N/A	⊘	$\mathbb{G}$	$\mathbb{Z}_p$	$2\mathbb{G}$	N/A
global escrow	0.057	0.148	0.733	1.462	1.505	0.148	1.505	48	32	701	96
HISE scheme 1	[1, 0]	[1, 0]	[0, 2]	[5, 2]	[4, 1]	[1, 0]	[4, 1]	$\mathbb{G}_1$	$\mathbb{Z}_p$	$[2\mathbb{G}_1, 3\mathbb{G}_T, \mathbb{Z}_p]$	$\mathbb{G}_2$
global escrow	0.057	3.5s	250	0.629	0.531	0.0004	0.532	48	32	287	40K
HISE scheme 2	[1, 0]	N/A	N/A	[2, 1]	[1, 1]	N/A	[1, 1]	$\mathbb{G}_1$	$\mathbb{Z}_p$	$[\mathbb{G}_2, \mathbb{G}_T]$	N/A

Performance of Cartesian product CPK and (global escrow) HISE schemes with 128-bit security level. $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ refers to asymmetric pairing groups. $\mathbb{G}$ refers to ordinary elliptic group. We report times for setup, key generation, signing, verification, key derivation, encryption, and (escrow) decryption, as well as the sizes of public key pk, secret key sk, ciphertext c and signature σ, and ignore the size of public parameters and group operations in the interests of space. The symbol ⊘ indicates that there is no corresponding algorithm. The symbol N/A indicates that the efficiency (or bandwidth) is hard to measure by algebra operations (or elements). At the time of this writing, the frontend tool[13] for Spartan [Set20] is not available, and hence we estimate the costs of signing/verification operations and signature size of (global escrow) HISE scheme 2 using the cost model provided by the authors, and mark the figures with gray color.

We build escrow ElGamal PKE on supersingular curve `ss-1536` [SKSW20] (where $|\mathbb{G}| = 193$ bytes, $|\mathbb{G}_T| = 192$ bytes, $|\mathbb{Z}_p| = 32$ bytes)[14] based on the `relic` library [AGM+]. We implement our global escrow PKE atop pairing-friendly curve `bls12-381`. To attain the same security level, our scheme could operate in elliptic groups defined on much smaller base field than the case of escrow ElGamal PKE. The comparison results in Table 3 show that our scheme outperforms escrow ElGamal PKE in all parameters, in particularly, being several orders of magnitude faster in terms of speed.

Table 3: Comparison of escrow ElGamal PKE [BF03] and our global escrow PKE

Scheme	efficiency (ms) [# exp, #pairing]					sizes (bytes) [# $\mathbb{G}$, # $\mathbb{Z}_p$]				
	Setup	KGen	Enc	Dec	Dec′	$\lvert pp\rvert$	$\lvert edk\rvert$	$\lvert pk\rvert$	$\lvert sk\rvert$	$\lvert c\rvert$
Boneh-Franklin	2.879	2.014	8.723	6.654	6.745	386	32	193	32	385
escrow ElGamal PKE	[2, 0]	[1, 0]	[2, 1]	[1, 1]	[1, 1]	$2\mathbb{G}$	$\mathbb{Z}_p$	$\mathbb{G}$	$\mathbb{Z}_p$	$[\mathbb{G}, \mathbb{G}_T]$
our proposed	0.243	0.058	0.680	0.579	0.586	288	32	48	32	287
global escrow PKE	[4, 0]	[1, 0]	[2, 1]	[1, 1]	[1, 1]	$[2\mathbb{G}_1, 2\mathbb{G}_2]$	$\mathbb{Z}_p$	$\mathbb{G}_1$	$\mathbb{Z}_p$	$[\mathbb{G}_2, \mathbb{G}_T]$

Performance of global escrow PKE schemes with 128-bit security level. $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ refers to asymmetric pairing groups. $(\mathbb{G}, \mathbb{G}_T)$ refers to symmetric pairing groups. We report times for setup, key generation, encryption, and (escrow) decryption, as well as the sizes of public parameters pp, global escrow decryption key edk, public key pk, secret key sk, and ciphertext c.

[14] So far, `ss-1536` is the only reported pairing-friendly curve with 128-bit security that supports Weil pairing.

9 Conclusion

Key reuse and key escrow are two broad issues arising from practical applications of cryptography. In this work, we investigated the interdiscipline of these two contradictory objects, an important but much-overlooked problem in prior work, aiming to enjoying the best of both worlds. We introduced a new notion called HISE featuring a novel two-level key derivation structure, which hits a sweet balance between key separation and key reuse. HISE not only admits individual key escrow, but also retains the benefit of key reuse. We then gave a black-box construction from (constrained) IBE, as well as a non-black-box construction from uniform OWF, PKE, and ZKPoK. To further attain global key escrow, we initiated a systematic study of global escrow PKE, which is long overdue for formal definition and efficient construction. We provided rigorous security notions and two generic constructions. The first uncovers a new application of the Naor-Yung paradigm. The second establishes an interesting connection to the three-party NIKE, and leads to the most efficient global escrow PKE to date. By mixing the results developed above, we suggested two paths for building global escrow HISE. The concrete (global escrow) HISE schemes instantiated from our generic constructions have competitive performance to the best CP-CPK scheme, and exhibit advantages in terms of richer functionality and public key reuse.

On the theoretical side our work resolves the problems left open in prior works [Ver01, PSST11], of reconciling the conflict between key reuse and key escrow. On the practical side our work serves as a developer guide for integrated usage of signature and encryption.

Finally, we remark that it is possible to consider a dual version of HISE, in which the hierarchy between signing key and decryption key are reversed. Such dual HISE could be useful in scenarios where decryption capability is a first priority. We leave the construction and application of dual HISE as an interesting problem.

Acknowledgments. We would like to thank the anonymous reviewers for their valuable comments on this paper. We thank Ren Zhang and Weiran Liu for helpful discussions. We thank Zhi Hu, Changan Zhao and Shiping Cai for help on implementation of pairing-based cryptography. We thank Xiangling Zhang for help on the test of certificate cost.

Yu Chen is supported by National Natural Science Foundation of China (Grant No. 61772522, No. 61932019), Shandong Provincial Key Research and Development Program (Major Scientific and Technological Innovation Project under Grant No. 2019JZZY010133), and Shandong Key Research and Development Program (Grant No. 2020ZLYS09). Yuyu Wang is supported by the National Natural Science Foundation for Young Scientists of China (Grant No. 62002049) and the Fundamental Research Funds for the Central Universities (Grant No. ZYGX2020J017).

References

[AGH15] Joseph A. Akinyele, Christina Garman, and Susan Hohenberger. Automating fast and secure translations from type-i to type-iii pairing schemes. In *ACM CCS 2015*, pages 1370–1381, 2015.

[AGM+] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient Library for Cryptography. https://github.com/relic-toolkit/relic.

[AGM18] Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In *CRYPTO 2018*, pages 643–673, 2018.

[BAZB20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *FC 2020*, pages 423–443, 2020.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computation*, 36(5):1301–1328, 2007.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal on Computation*, 32:586–615, 2003.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT 2001*, pages 514–532, 2001.

[BMV16] Silvio Biagioni, Daniel Masny, and Daniele Venturi. Naor-yung paradigm with shared randomness and applications. In *SCN 2016*, pages 62–80, 2016.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *ACM CCS 2017*, pages 1825–1842, 2017.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT 2003*, pages 255–271, 2003.

[CJNP02] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. Universal padding schemes for RSA. In *CRYPTO 2002*, volume 2442, pages 226–241, 2002.

[CMTA20] Yu Chen, Xuecheng Ma, Cong Tang, and Man Ho Au. PGC: Pretty Good Confidential Transaction System with Auditability. In *ESORICS 2020*, pages 591–610, 2020.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS 2010*, pages 511–520, 2010.

[DLP+12] Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, Nigel P. Smart, and Mario Strefler. On the joint security of encryption and signature in EMV. In Orr Dunkelman, editor, *CT-RSA 2012*, pages 116–135, 2012.

[EMV11] EMV Co. EMV Book 2 - Security and Key Management -Version 4.3, 2011. https://www.emvco.com/wp-content/uploads/2017/05/EMV_v4.3_Book_2_Security_and_Key_Management_20120607061923900.pdf.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO 1999*, pages 537–554, 1999.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge

proof systems. In *USENIX Security 21*, 2021.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discret. Appl. Math.*, 156(16):3113–3121, 2008.

[HP01] Stuart Haber and Benny Pinkas. Securely combining public-key cryptosystems. In *ACM CCS 2001*, pages 215–224, 2001.

[Jou04] Antoine Joux. A one round protocol for tripartite diffie-hellman. *J. Cryptology*, 17(4):263–276, 2004.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *ACM CCS 2018*, pages 525–537, 2018.

[KO03] Yuichi Komano and Kazuo Ohta. Efficient universal padding techniques for multiplicative trapdoor one-way permutation. In *CRYPTO 2003*, pages 366–382, 2003.

[KPS18] Ahmed E. Kosba, Charalampos Papamanthou, and Elaine Shi. xjsnark: A framework for efficient verifiable computation. In *IEEE S&P 2018*, pages 944–961, 2018.

[NVV18] Neha Narula, Willy Vasquez, and Madars Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In *USENIX NSDI 2018*, pages 65–80, 2018.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC 1990*, pages 427–437, 1990.

[PGP] PGP. https://www.openpgp.org.

[Pin] Ping identity. http://www.pingidentity.com.

[PSST11] Kenneth G. Paterson, Jacob C. N. Schuldt, Martijn Stam, and Susan Thomson. On the joint security of encryption and signature, revisited. In *ASIACRYPT 2011*, pages 161–178, 2011.

[PY99] Pascal Paillier and Moti Yung. Self-escrowed public-key infrastructures. In *ICISC 1999*, pages 257–268, 1999.

[Ros] David E. Ross. PGP: Backdoors and key escrow. https://www.rossde.com/PGP/pgp_backdoor.html.

[RS08] Karl Rubin and Alice Silverberg. Compression in finite fields and torus-based cryptography. *SIAM J. Comput.*, 37(5):1401–1428, 2008.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS 1999*, pages 543–553. ACM, 1999.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *CRYPTO 2020*, pages 704–737, 2020.

[Shi] Mitsunari Shigeo. A portable and fast pairing-based cryptography library. https://github.com/herumi/mcl.

[SKSW20] Yumi Sakemi, Tetsutaro Kobayashi, Tsunekazu Saito, and Riad S. Wahby. 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.

[Ver01] Eric R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In *EUROCRYPT 2001*, pages 195–210, 2001.

[vox] https://www.vox.com/recode/2020/1/24/21079275/slack-private-messages-privacy-law-enforcement-lawsuit.

[X50] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. https://tools.ietf.org/html/rfc5280.

[YY98] Adam L. Young and Moti Yung. Auto-recoverable auto-certifiable cryptosystems. In *EUROCRYPT 1998*, pages 17–31, 1998.

[YY99] Adam L. Young and Moti Yung. Auto-recoverable cryptosystems with faster initialization and the escrow hierarchy. In *PKC 1999*, pages 306–314, 1999.