Hierarchical Integrated Signature and Encryption

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

Yu Chen [*] Qiang Tang [†] Yuyu Wang [‡]

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.

[*] School of Cyber Science and Technology, Shandong University. Email: yuchen.prc@gmail.com

[†] University of Sydney. Email: qtang84@gmail.com

[‡] University of Electronic Science and Technology of China. Email: wangyuyu@uestc.edu.cn

Contents

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.[1]

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[2]), 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 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

[1] 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.

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

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.[3]

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

[3] 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.

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 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

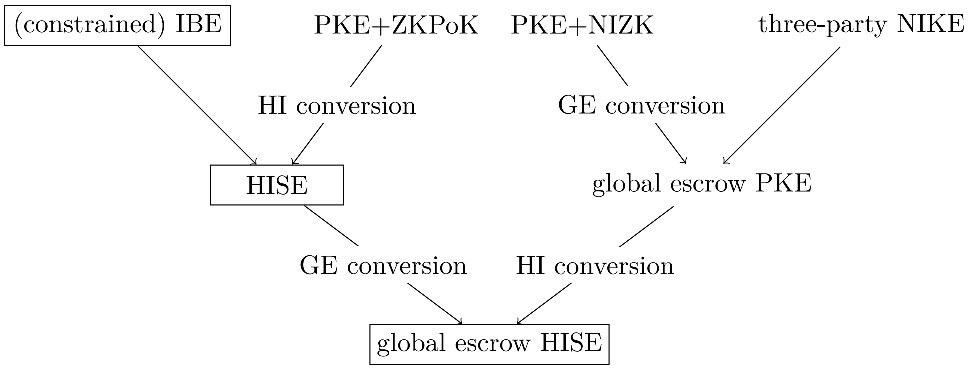


Figure 1: Technology roadmap of global escrow HISE. The rectangles denote our newly introduced cryptographic schemes.

share his decryption key with an authority for audit purpose or delegating costly decryption operations[4] 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.[5] 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.

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

[4] 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.

[5] 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.

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.[6] 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. For convenience and to fix notation, we recall these definitions in Appendix A. 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.

[6] 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.

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.
- $\mathsf{KeyGen}(pp)$: 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:

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

$\mathcal{O}_{\mathsf{ext}}$ denotes the key extraction oracle, which on input id returns $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$. $\mathcal{O}_{\mathsf{constrain}}$ denotes the key constrain oracle, which on input f returns $sk_f \leftarrow \mathsf{Constrain}(msk, f)$. $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{ext}}$ with id^* or query $\mathcal{O}_{\mathsf{constrain}}$ 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. See Appendix B.2 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 $\Pr[\mathsf{Vrfy}(pk, \widetilde{m}, \sigma) = 1] \geq 1 - \mathsf{negl}(\lambda)$, 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 3.1 (Joint Security for HISE)**.** HISE is jointly secure if its encryption and signature components satisfy the following security notions. Hereafter, let $\mathcal{O}_{\mathsf{sign}}$ be the signing oracle that on input $\widetilde{m} \in \widetilde{M}$ returns $\sigma \leftarrow \mathsf{Sign}(sk, \widetilde{m})$, and $\mathcal{O}_{\mathsf{dec}}$ 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:

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

$\mathcal{A}$ has unrestricted access to $\mathcal{O}_{\mathsf{sign}}$, but is not allowed to query $\mathcal{O}_{\mathsf{dec}}$ 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:

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

The set $\mathcal{Q}$ records queries to $\mathcal{O}_{\mathsf{sign}}$. 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 3.1. 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 $\mathcal{O}_{\mathsf{dec}}$. 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 two generic constructions of HISE.

4.1 HISE from Constrained IBE

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 $pp_{\text{cibe}} \leftarrow \text{CIBE}.\mathsf{Setup}(1^\lambda)$, $pp_{\text{ots}} \leftarrow \text{OTS}.\mathsf{Setup}(1^\lambda)$, outputs $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$. 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 $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$, runs $\text{CIBE}.\mathsf{KeyGen}(pp_{\text{cibe}})$ to generate (mpk, msk), outputs public key $pk = mpk$ and secret key $sk = msk$.

- $\mathsf{Derive}(sk)$: parses sk as msk, runs $sk_{f_{\mathbf{v}}} \leftarrow \text{CIBE}.\mathsf{Constrain}(msk, f_{\mathbf{v}})$ 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 $(ovk, osk) \leftarrow \text{OTS}.\mathsf{KeyGen}(pp_{\text{ots}})$. sets $id = 1||ovk$, computes $c_{\text{cibe}} \leftarrow \text{CIBE}.\mathsf{Enc}(mpk, id, m)$, $\sigma \leftarrow \mathsf{OTS}.\mathsf{Sign}(osk, c_{\text{cibe}})$, then outputs $c = (ovk, c_{\text{cibe}}, \sigma)$.

- $\mathsf{Dec}(dk, c)$: parses $dk = sk_{f_{\mathbf{v}}}$ and $c = (ovk, c_{\text{cibe}}, \sigma)$. The decryption algorithm first checks if $\text{OTS}.\mathsf{Vrfy}(ovk, c_{\text{cibe}}, \sigma) = 1$, if not outputs $\perp$, else sets $id = 1||ovk$ and computes $sk_{id} \leftarrow \text{CIBE}.\mathsf{Derive}(sk_{f_{\mathbf{v}}}, id)$, outputs $m \leftarrow \text{CIBE}.\mathsf{Dec}(sk_{id}, c_{\text{cibe}})$.

- $\mathsf{Sign}(sk, \widetilde{m})$: parses sk as msk, computes $sk_{id} \leftarrow \text{CIBE}.\mathsf{Extract}(msk, id)$ 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 $c_{\text{cibe}} \leftarrow \text{CIBE}.\mathsf{Enc}(mpk, id, m)$, outputs "1" if $\text{CIBE}.\mathsf{Dec}(sk_{id}, c_{\text{cibe}}) = m$ and "0" otherwise.

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

Theorem 4.1. *If the constrained IBE scheme is IND-CPA secure and the OTS scheme is strong EUF-CMA secure, then the above HISE construction is jointly secure.*

We prove this theorem via the following two lemmas.

Lemma 4.2. *If the constrained IBE scheme is OW-CPA secure, then the signature component is EUF-CMA secure even in the presence of decryption key.*

Proof. Let $\mathcal{A}$ be an adversary against the EUF-CMA security of the signature component, we show how to build an adversary $\mathcal{B}$ breaking the assumed OW-CPA security of CIBE. Given (pp_{cibe}, mpk), $\mathcal{B}$ simulates $\mathcal{A}$'s challenger as below.

- Setup: $\mathcal{B}$ sets $pk = mpk$, generates $pp_{\text{ots}} \leftarrow \text{OTS}.\mathsf{Setup}(1^\lambda)$, sets $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$, queries its constrained oracle for prefix predicate $f_{\mathbf{v}}$ (where $\mathbf{v} = 1$) and obtains constrained secret key $sk_{f_{\mathbf{v}}}$, sets $dk = sk_{f_{\mathbf{v}}}$, and sends (pp, pk, dk) to $\mathcal{A}$.

- Signing query: Upon receiving a signing query $\langle \widetilde{m} \rangle$, $\mathcal{B}$ queries its extraction oracle for identity $id = 0||\widetilde{m}$, and obtains $sk_{id} \leftarrow \text{CIBE}.\mathsf{Extract}(msk, id)$, then $\mathcal{B}$ sends $\sigma = sk_{id}$ to $\mathcal{A}$.

- Forgery: $\mathcal{A}$ outputs $(\widetilde{m}^*, \sigma^*)$ as forgery. At this point, $\mathcal{B}$ submits $id^* = 0||\widetilde{m}^*$ as the target identity, and receives back $c^*_{\text{cibe}} \leftarrow \text{CIBE}.\mathsf{Enc}(mpk, id^*, m)$ for a random plaintext $m \xleftarrow{\text{R}} M$. Finally, $\mathcal{B}$ parses σ^* as sk_{id^*}, and outputs $m' \leftarrow \text{CIBE}.\mathsf{Dec}(sk_{id^*}, c^*)$. $\mathcal{B}$ wins if $m' = m$.

Clearly, $\mathcal{B}$'s simulation is perfect. If $\mathcal{A}$ breaks the assumed EUF-CMA security with some probability, then according to the definition of verification algorithm, $\mathcal{B}$ succeeds in breaking the OW-CPA security of CIBE with the same probability. This proves Lemma 4.2. □

Lemma 4.3. *If the OTS scheme is strongly EUF-CMA secure and the constrained IBE scheme is selective-identity IND-CPA secure, then the encryption component is IND-CCA secure in the presence of a signing oracle.*

Proof. The proof follows closely to that of Theorem in [BCHK07]. We proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ wins in Game i.

Game 0. This is the standard security experiment for the encryption component. $\mathcal{CH}$ interacts with $\mathcal{A}$ as below.

- Setup: $\mathcal{CH}$ runs $pp_{\text{cibe}} \leftarrow \text{CIBE}.\mathsf{Setup}(1^\lambda)$, $pp_{\text{ots}} \leftarrow \text{OTS}.\mathsf{Setup}(1^\lambda)$, sets $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$, generates $(mpk, msk) \leftarrow \text{CIBE}.\mathsf{KeyGen}(pp_{\text{cibe}})$, sets $pk = mpk$ and $sk = msk$, then sends (pp, pk) to $\mathcal{A}$.

- Signing query: Upon receiving a signing query $\langle \widetilde{m} \rangle$, $\mathcal{CH}$ computes and returns $\sigma \leftarrow \text{HISE}.\mathsf{Sign}(sk, \widetilde{m})$.

- Decryption query: Upon receiving a decryption query $\langle c \rangle$, $\mathcal{CH}$ first parses $c = (ovk, c_{\text{bte}}, \sigma)$, if $\text{OTS}.\mathsf{Vrfy}(ovk, c_{\text{cibe}}, \sigma) = 1$ returns $\perp$, else derives $sk_{id} \leftarrow \text{CIBE}.\mathsf{Extract}(msk, id)$ for $id = 1||ovk$, outputs $m \leftarrow \text{CIBE}.\mathsf{Dec}(sk_{id}, c_{\text{cibe}})$.

- Challenge: $\mathcal{A}$ submits two messages (m_0, m_1). $\mathcal{CH}$ picks a random bit $b \xleftarrow{\text{R}} \{0,1\}$, then generates the challenge ciphertext as follows: generates a fresh OTS keypair $(ovk^*, osk^*) \leftarrow \text{OTS}.\mathsf{KeyGen}(pp_{\text{ots}})$, then sets $id^* = 1||ovk^*$, computes $c^*_{\text{cibe}} \leftarrow \text{CIBE}.\mathsf{Enc}(mpk, id^*, m_b)$, $\sigma^* \leftarrow \text{OTS}.\mathsf{Sign}(osk^*, c^*_{\text{cibe}})$, then sends $c^* = (ovk^*, c^*_{\text{cibe}}, \sigma^*)$ to $\mathcal{A}$. After seeing the challenge ciphertext c^*, $\mathcal{A}$ can still query the signing oracle and decryption oracle, except that the decryption query for c^* is not allowed.

- Guess: Finally, $\mathcal{A}$ outputs a bit b' and wins $b = b'$. According to the definition, we have:

$$\mathsf{Adv}_{\mathcal{A}} = |\Pr[S_0] - 1/2|$$

Game 1. Same as Game 0 except $\mathcal{CH}$ generates the OTS keypair $(ovk^*, osk^*) \leftarrow \text{OTS}.\mathsf{KeyGen}(pp_{\text{ots}})$ associated to the challenge ciphertext in the setup stage. This modification is only conceptual, thus we have:

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

Game 2. Same as Game 1 except that $\mathcal{CH}$ directly aborts when answering decryption queries if one of the following two events happens:

1. E_1: $\mathcal{A}$ makes a decryption query for $c = (ovk^*, c_{\text{cibe}}, \sigma)$ in Phase 1 such that $\text{OTS}.\mathsf{Vrfy}(ovk^*, c_{\text{cibe}}, \sigma) = 1$.

2. E_2: $\mathcal{A}$ makes a decryption query for $c = (ovk^*, c^*_{\text{cibe}}, \sigma)$ in Phase 2 such that $\text{OTS}.\mathsf{Vrfy}(ovk^*, c^*_{\text{cibe}}, \sigma) = 1$ and $\sigma \neq \sigma^*$.

Let $E = E_1 \vee E_2$. Conditioned on E never happens, Game 1 and Game 2 are identical. By the difference lemma, we have:

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

Evidently, a decryption query triggering E translates to a successful forgery against OTS. By the assumed strong EUF-CMA security of OTS, we conclude that $\Pr[E] = \mathsf{negl}(\lambda)$ for any PPT adversary.

Claim 4.4. *If the constrained IBE scheme is selective-identity IND-CPA secure, then no PPT adversary has non-negligible advantage in Game 2.*

Proof. Let $\mathcal{B}$ be an adversary against the selective-identity IND-CPA security of the constrained IBE scheme. Given pp_{cibe}, $\mathcal{B}$ simulates $\mathcal{A}$'s challenger in Game 2 as below:

Setup: $\mathcal{B}$ generates $pp_{\text{ots}} \leftarrow \text{OTS}.\mathsf{Setup}(1^\lambda)$, $(ovk^*, osk^*) \leftarrow \text{OTS}.\mathsf{KeyGen}(pp_{\text{ots}})$, then commits $id^* = 1||ovk^*$ to its own challenger as the target identity and receives back mpk. $\mathcal{B}$ sends $pp = (pp_{\text{cibe}}, pp_{\text{ots}})$ and $pk = mpk$ to $\mathcal{A}$.

Signing query: Upon receiving a signing query $\langle \widetilde{m} \rangle$, $\mathcal{B}$ queries its extraction oracle for identity $id = 0||\widetilde{m}$ to obtain sk_{id}, then sends it to $\mathcal{A}$ as a signature for $\widetilde{m}$. Since we always have $id \neq id^*$ due to different prefix, $\mathcal{B}$ simulates the signing oracle perfectly.

Decryption query: Upon receiving a decryption query $\langle c \rangle$ where $c = (ovk, c_{\text{cibe}}, \sigma)$, if E_1 happens, $\mathcal{B}$ aborts, else $\mathcal{B}$ proceeds as below: if $\text{OTS}.\mathsf{Vrfy}(ovk, c_{\text{cibe}}, \sigma) = 0$ then returns $\perp$, else queries its decryption oracle with $\langle 1||ovk, c_{\text{cibe}} \rangle$, and forwards the reply to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits two messages (m_0, m_1). At this point, $\mathcal{B}$ sends (m_0, m_1) to its own challenger and receives back c^*_{cibe}, which is an encryption of m_b under the target identity $id^* = 1||ovk^*$. $\mathcal{B}$ computes $\sigma^* \leftarrow \text{OTS}.\mathsf{Sign}(osk^*, c^*_{\text{cibe}})$, sends $c^* = (ovk^*, c^*_{\text{cibe}}, \sigma^*)$ to $\mathcal{A}$ as the challenge ciphertext.

Guess: After receiving c^*, $\mathcal{A}$ may continue to query the signing and decryption oracles under the restriction that it must not query the decryption oracle with c^*. If E_2 happens, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ responds in the same way as it did in Phase 1. Finally, $\mathcal{A}$ submits a guess b' which $\mathcal{B}$ outputs as its guess.

Clearly, $\mathcal{B}$'s simulation is perfect. If c^*_{cibe} is a constrained IBE ciphertext of m_b, then c^* is also an HISE ciphertext of m_b. Thus, $\mathcal{B}$ succeeds in breaking the selective-identity IND-CPA security with the same advantage as $\mathcal{A}$ wins in Game 2. This proves Claim 4.4, namely, $|\Pr[S_2] - 1/2| = \mathsf{negl}(\lambda)$. □

Putting all of the above together, Lemma 4.3 follows immediately. □

4.2 HISE from IBE

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. Let the identity space of IBE be $\{0,1\}^{\ell+1}$, we build an HISE scheme with message space $\{0,1\}^\ell$ as follows.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, runs $pp_{\text{ibe}} \leftarrow \text{IBE}.\mathsf{Setup}(1^\lambda)$, outputs $pp = (pp_{\text{ibe}}, id^*)$, where $id^* = 1^{\ell+1}$.
- $\mathsf{KeyGen}(pp)$: parses $pp = (pp_{\text{ibe}}, id^*)$, runs $(mpk, msk) \leftarrow \text{IBE}.\mathsf{KeyGen}(pp_{\text{ibe}})$, outputs public key $pk = mpk$ and secret key $sk = msk$.
- $\mathsf{Derive}(sk)$: parses sk as msk, outputs $dk \leftarrow \text{IBE}.\mathsf{Extract}(msk, id^*)$.
- $\mathsf{Enc}(pk, m)$: parses $pk = mpk$, outputs $c \leftarrow \text{IBE}.\mathsf{Enc}(mpk, id^*, m)$.
- $\mathsf{Dec}(dk, c)$: parses $dk = sk_{id^*}$, outputs $m \leftarrow \text{IBE}.\mathsf{Dec}(sk_{id^*}, c)$.
- $\mathsf{Sign}(sk, \widetilde{m})$: parses $sk = msk$, computes $sk_{id} \leftarrow \text{IBE}.\mathsf{Extract}(msk, id)$ 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 $c \leftarrow \text{IBE}.\mathsf{Enc}(mpk, id, m)$, outputs "1" if $\text{IBE}.\mathsf{Dec}(sk_{id}, c) = m$ and "0" otherwise.

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

Theorem 4.5. *If the IBE scheme is IND-CPA secure, then the above HISE construction is jointly secure in the sense that the signature component is EUF-CMA secure and the PKE component is IND-CPA secure.*

We prove this theorem via the following two lemmas.

Lemma 4.6. *If the IBE scheme is OW-CPA secure, then the signature component is EUF-CMA secure even in the presence of decryption key.*

Proof. Let $\mathcal{A}$ be an adversary against the EUF-CMA security of the signature component, we show how to build an adversary $\mathcal{B}$ breaking the assumed OW-CPA security of IBE. Given (pp_{ibe}, mpk), $\mathcal{B}$ simulates $\mathcal{A}$'s challenger as below.

Setup: $\mathcal{B}$ sets $pp = (pp_{\text{ibe}}, id^* = 1^{\ell+1})$, $pk = mpk$, queries its extraction oracle for id^* to obtain $sk_{id^*} \leftarrow \text{IBE}.\mathsf{Extract}(msk, id^*)$, sets $dk = sk_{id^*}$, and sends (pp, pk, dk) to $\mathcal{A}$.

Signing query: Upon receiving a signing query $\langle \tilde{m} \rangle$, $\mathcal{B}$ queries its extraction oracle for identity $id = 0||\tilde{m}$, and obtains $sk_{id} \leftarrow \text{IBE}.\mathsf{Extract}(msk, id)$, then $\mathcal{B}$ sends $\sigma = sk_{id}$ to $\mathcal{A}$.

Forgery: $\mathcal{A}$ outputs $(\tilde{m}^*, \sigma^*)$ as forgery. At this point, $\mathcal{B}$ outputs $id^* = 0||\tilde{m}^*$ as the target identity, and receives back $c^* \leftarrow \text{IBE}.\mathsf{Enc}(mpk, id^*, m)$ for some randomly chosen plaintext $m \in M$. Finally, $\mathcal{B}$ parses σ^* as sk_{id^*}, and outputs $m' \leftarrow \text{IBE}.\mathsf{Dec}(sk_{id^*}, c^*)$. $\mathcal{B}$ wins if $m' = m$.

Clearly, $\mathcal{B}$'s simulation is perfect. If $\mathcal{A}$ breaks the assumed EUF-CMA security with some probability, then according to the definition of verification algorithm, $\mathcal{B}$ succeeds in breaking the OW-CPA security of IBE with the same probability. This proves Lemma 4.6. □

Lemma 4.7. *If the IBE scheme is selective-identity IND-CPA secure, then the encryption component is IND-CPA secure in the presence of a signing oracle.*

Proof. Let $\mathcal{A}$ be an adversary against the IND-CPA security of the encryption component, we show how to build an adversary $\mathcal{B}$ breaking the assumed selective-identity IND-CPA security of IBE. Given pp_{ibe}, $\mathcal{B}$ simulates $\mathcal{A}$'s challenger as below.

Setup: $\mathcal{B}$ sets $id^* = 1^{\ell+1}$, submits id^* to its own challenger as the target identity and receives back mpk. $\mathcal{B}$ sets $pp = (pp_{\text{ibe}}, id^*)$ and $pk = mpk$, then sends (pp, pk) to $\mathcal{A}$.

Extraction query: Upon receiving a extraction query $\langle id \rangle$, $\mathcal{B}$ queries its extraction oracle for id and forwards the result to $\mathcal{A}$.

Signing query: Upon receiving a signing query $\langle \tilde{m} \rangle$, $\mathcal{B}$ queries it extraction oracle for $id = 0||\tilde{m}$ and forwards the result to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits (m_0, m_1) to $\mathcal{B}$. $\mathcal{B}$ forwards (m_0, m_1) to its own challenger and receives back $c^* \leftarrow \text{IBE}.\mathsf{Enc}(mpk, id^*, m_b)$ for a random bit b, then sends c^* to $\mathcal{A}$ as the challenge. Finally, $\mathcal{A}$ outputs a guess b' for b and wins if $b' = b$.

Clearly, $\mathcal{B}$'s simulation is perfect. If $\mathcal{A}$ breaks the assumed IND-CPA security with some probability, then $\mathcal{B}$ breaks the selective-identity IND-CPA security of IBE with the same probability. This proves Lemma 4.7. □

The PKE component of above HISE construction is IND-CPA secure. We can enhance it to IND-CCA security by applying the Fujisaki-Okamoto transformation [FO99] with random oracle heuristic.

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 $pp_{\text{pke}} \leftarrow \text{PKE}.\mathsf{Setup}(1^\lambda)$, $pp_{\text{zkpok}} \leftarrow \text{ZKPoK}.\mathsf{Setup}(1^\lambda)$, picks a uniform OWF $\mathsf{F} : \{0,1\}^n \rightarrow \{0,1\}^\ell$, outputs $pp = (pp_{\text{pke}}, pp_{\text{zkpok}}, \mathsf{F})$.

- $\mathsf{KeyGen}(pp)$: parses $pp = (pp_{\text{pke}}, pp_{\text{zkpok}}, \mathsf{F})$, picks $sk \xleftarrow{\text{R}} \{0,1\}^n$, computes $r \leftarrow \mathsf{F}(sk)$, runs $(ek, dk) \leftarrow \text{PKE}.\mathsf{KeyGen}(pp_{\text{pke}}; r)$, 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 $(ek, dk) \leftarrow \text{PKE}.\mathsf{KeyGen}(pp_{\text{pke}}; r)$, 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 the algorithm that outputs the first outcome of PKE.KeyGen, say pk. G and F induce an $\mathcal{NP}$ relation R_{key} over $PK \times \{0,1\}^n$ defined as below.

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

  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 R_{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 $\widetilde{m}$.

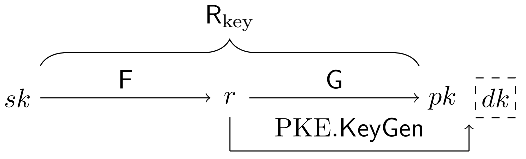


Figure 2: The hierarchical key structure

- $\mathsf{Vrfy}(pk, \widetilde{m}, \sigma)$: 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. R_{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 NIZKPoK.$\mathsf{Prove}(pk, sk, \tilde{m})$ and the verifying procedure as 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 5.1. *The above HISE construction is jointly secure assuming the security of its building blocks and modeling* H *as a random oracle.*

We prove this theorem by proving its encryption component and signature component are secure in the joint sense. In all the security proofs hereafter, the challenger $\mathcal{CH}$ emulates the random oracle queries by maintaining an initially empty list T, which is used to track random oracle queries. The adversary may interleave its hash and signing queries arbitrarily. We make a few simplifying assumptions without any loss of generality. First, we assume that the adversary makes any given hash query only once. When a signature (a, z) on a message $\widetilde{m}$ is given to $\mathcal{A}$, we also include the value $e := \mathsf{H}(a, \widetilde{m})$. We may therefore assume that the adversary never queries $\mathsf{H}(a, \widetilde{m})$ after receiving such a signature.

Lemma 5.2. *The encryption component is IND-CCA secure in the joint sense if the underlying PKE is IND-CCA secure,* F *is uniform, ZKPoK is honest-verifier zero-knowledge and* H *is a random oracle.*

Proof. The main difficulty of reducing the IND-CCA security of the encryption component to the underlying PKE lies in that the reduction algorithm has to answer signing queries without sk. We solve this problem by having the reduction algorithm simulate the singing oracle by utilizing the zero-knowledge property of ZKPoK and the programmability of H. We prove this lemma via a sequence of games. Let S_i be the event that $\mathcal{A}$ wins in Game i.

Game 0. This is the standard joint IND-CCA experiment for PKE. Let $\mathcal{A}$ be an adversary against the encryption component. $\mathcal{CH}$ interacts with an adversary $\mathcal{A}$ as below.

Setup: $\mathcal{CH}$ runs $pp \leftarrow \mathsf{Setup}(1^\lambda)$ to generate public parameters, that is, runs $pp_{\text{pke}} \leftarrow \text{PKE}.\mathsf{Setup}(1^\lambda)$, $pp_{\text{zkpok}} \leftarrow \text{ZKPoK}.\mathsf{Setup}(1^\lambda)$, picks a uniform OWF $\mathsf{F} : \{0,1\}^n \to \{0,1\}^\ell$, sets $pp = (pp_{\text{pke}}, pp_{\text{zkpok}}, \mathsf{F})$, then picks $sk \xleftarrow{\text{R}} \{0,1\}^n$, computes $r \leftarrow \mathsf{F}(sk)$, $(ek, dk) \leftarrow \text{PKE}.\mathsf{KeyGen}(r)$, sets $pk = ek$, and sends (pp, pk) to $\mathcal{A}$.

Random oracle query: $\mathcal{A}$ can adaptively make random oracle queries. Upon receiving a random oracle query $\langle a, \widetilde{m}\rangle$, $\mathcal{CH}$ picks a random $e \xleftarrow{\text{R}} \Omega$, then records $(a, \widetilde{m}, e)$ into T.

Signing query: $\mathcal{A}$ can adaptively make signing queries. Upon receiving a signing query $\langle \widetilde{m}\rangle$, $\mathcal{CH}$ responds with $\sigma \leftarrow \mathsf{Sign}(sk, \widetilde{m})$.

Decryption query: $\mathcal{A}$ can adaptively make decryption queries. Upon receiving a decryption query $\langle c\rangle$, $\mathcal{CH}$ responds with $m \leftarrow \mathsf{Dec}(dk, c)$.

Challenge: $\mathcal{A}$ submits (m_0, m_1) to $\mathcal{CH}$. $\mathcal{CH}$ picks a random bit $b \xleftarrow{\text{R}} \{0,1\}$, sends $c^* \leftarrow \mathsf{Enc}(pk, m_b)$ to $\mathcal{A}$ as the challenge. $\mathcal{A}$ can still make signing and decryption queries after receiving the challenge ciphertext c^*, but the decryption query for c^* is forbidden.

Guess: $\mathcal{A}$ outputs a bit b' and wins if its guess $b' = b$.

According to the definition, we have:

$$\mathsf{Adv}_{\mathcal{A}} = |\Pr[S_0] - 1/2|$$

Game 1. Same as Game 0 except that $\mathcal{CH}$ now emulates signing oracle by programming the random oracle H and utilizing the zero-knowledge property of ZKPoK, rather than using sk.

Signing query: Upon receiving a signing query on $\widetilde{m}$, $\mathcal{CH}$ invokes $\mathcal{S}$ to obtain a transcript (a, e, z). If $(a, \widetilde{m})$ has been previously queried for hash value, $\mathcal{CH}$ aborts; otherwise, $\mathcal{CH}$ sets $e := \mathsf{H}(a, \widetilde{m})$, records $(a, \widetilde{m}, e)$ to the T list, and responds with $\sigma = (a, z)$.

We argue that $|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{negl}(\lambda)$, which is proved by the claim below.

Claim 5.3. *For any PPT adversary $\mathcal{A}$, its advantage in Game 0 and Game 1 are negligibly close assuming the statistical honest verifier zero-knowledge property of ZKPoK.*

Proof. Conditioned on $\mathcal{CH}$ does not abort, the claim holds obviously. This is because that for any PPT adversary $\mathcal{A}$, its views in Game 0 and Game 1 are indistinguishable. This follows from the HVZK property and a standard hybrid argument on Q_s signing queries. We then bound the probability that $\mathcal{CH}$ aborts. By the HVZK property, the distribution of a (the first element of $\mathcal{S}$'s output) distributes uniformly over A. Thus, the probability that $\mathcal{CH}$ aborts when answering the i-th signing query is at most $Q_h/|A|$. Applying the union bound, we conclude that the probability that $\mathcal{CH}$ aborts when handling Q_s signing queries is upper bounded by $Q_s Q_h/|A|$, which is negligible in λ. By the difference lemma, Claim 5.3 immediately follows. □

Remark 5.1. In the above claim, the standard HVZK could be relaxed to computational sense as long as the distribution of first message of $\mathcal{S}$, say a, is still close to uniform. This suffices to guarantee that $\mathcal{CH}$ emulates the signing oracle successfully with overwhelming probability.

Game 2. Same as Game 1 except the following modifications:

- Setup: $\mathcal{CH}$ runs $\mathsf{Setup}(1^\lambda)$ to generate $pp = (pp_{\text{pke}}, pp_{\text{zkpok}}, \mathsf{F})$, picks $sk \xleftarrow{\text{R}} \{0,1\}^n$, samples $r \xleftarrow{\text{R}} \{0,1\}^\ell$, $(ek, dk) \leftarrow \text{PKE}.\mathsf{KeyGen}(r)$, sets $ek = pk$, and then sends (pp, pk) to $\mathcal{A}$.

Since sk is uniformly chosen from $\{0,1\}^n$, by the uniformity of F the two distributions $r \xleftarrow{\text{R}} \{0,1\}^\ell$ and $r \leftarrow \mathsf{F}(sk)$ are statistically close. In both Game 1 and Game 2, $\mathcal{A}$'s view is only determined by r. Thereby, $|\Pr[S_2] - \Pr[S_1]| = \mathsf{negl}(\lambda)$.

We then argue that $\Pr[S_2] = \mathsf{negl}(\lambda)$, which is proved by the claim below.

Claim 5.4. *If the PKE scheme is IND-CCA secure, then no PPT adversary has non-negligible advantage in Game 2.*

Proof. Suppose there is an adversary $\mathcal{A}$ that has some non-negligible advantage in Game 2, we build an algorithm $\mathcal{B}$ that breaks the IND-CCA security of PKE with the same advantage. Given (pp_{pke}, ek) of PKE, $\mathcal{B}$ emulates $\mathcal{A}$'s challenger in Game 2 as follows:

Setup: $\mathcal{B}$ runs ZKPoK.$\mathsf{Setup}(1^\lambda)$ to generate pp_{zkpok}, picks a uniform one-way function F from $\{0,1\}^n$ to $\{0,1\}^\ell$, sets $pp = (pp_{\text{pke}}, pp_{\text{zkpok}}, \mathsf{F})$, $pk = ek$, then sends (pp, pk) to $\mathcal{A}$.

Random oracle query: $\mathcal{B}$ simulates the random oracle the same way as $\mathcal{CH}$ does in Game 2.

Signing query: $\mathcal{B}$ simulates the signing oracle the same way as $\mathcal{CH}$ does in Game 2.

Decryption query: Upon receiving decryption query for ciphertext c, $\mathcal{B}$ queries its decryption oracle and forwards the response back to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits (m_0, m_1). $\mathcal{B}$ submits (m_0, m_1) to its own challenger and receives back the challenge ciphertext c^*, then sends c^* to $\mathcal{A}$. After seeing c^*, $\mathcal{A}$ can continue to query the signing and decryption oracles except that decryption query for c^* is not allowed.

Guess: Eventually, $\mathcal{A}$ outputs its guess b'. $\mathcal{B}$ forwards b' to its own challenger.

Clearly, $\mathcal{B}$'s simulation is perfect. If c^* is a PKE ciphertext of m_b, then c^* is also an HISE ciphertext of m_b. Thus, $\mathcal{B}$ succeeds in breaking the IND-CCA security with the same advantage as $\mathcal{A}$ wins in Game 2. This proves the Claim 5.4. □

Putting all of the above together, Lemma 5.2 immediately follows. □

Lemma 5.5. *The signature component is EUF-CMA secure in the joint sense if F is one-way, G is target-collision resistant, and ZKPoK satisfies the PoK and HVZK property.*

We prove this lemma by establishing the following results.

Proposition 5.6. *If F is one-way, G is target-collision resistant, then $\mathsf{G} \circ \mathsf{F}$ from $\{0,1\}^n \to PK$ is leakage-resilient one-way w.r.t. $\mathcal{H} = \{h \circ \mathsf{F}\}$, where h could be any efficiently computable functions with domain R.*

$$\begin{tikzcd}
sk \arrow[r, "\mathsf{F}", "\text{one-way}"'] & r \arrow[r, "\mathsf{G}", "\text{target-collision resistant}"'] \arrow[dr, "h", "\text{leakage query}"'] & pk \\
 & & \boxed{h(r)} \text{ leakage on } sk
\end{tikzcd}$$

Figure 3: The leakage-resilient one-way function $\mathsf{G} \circ \mathsf{F}$ w.r.t. $\mathcal{H} = \{h \circ \mathsf{F}\}$

Proof. We proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ wins in Game i.

Game 0. This is the standard leakage-resilient one-wayness security experiment for $\mathsf{G} \circ \mathsf{F}$. $\mathcal{CH}$ interacts with $\mathcal{A}$ as below:

Setup and challenge: $\mathcal{CH}$ picks $sk \xleftarrow{\text{R}} \{0,1\}^n$, computes $r \leftarrow \mathsf{F}(sk)$, $pk \leftarrow \mathsf{G}(r)$, sends $(\mathsf{G} \circ \mathsf{F}, pk)$ to $\mathcal{A}$ as the challenge.

Leakage query: $\mathcal{A}$ may adaptively make leakage queries on sk. The leakage functions could be arbitrary functions of the form $h \circ \mathsf{F}$, with the only constraint that h is efficiently computable. In other words, $\mathcal{A}$ can obtain any efficiently computable leakage about the internal state r. Upon receiving a query $\langle h \circ \mathsf{F} \rangle$, $\mathcal{CH}$ responds with $h(r)$.

Attack: $\mathcal{A}$ outputs sk' and wins if $\mathsf{G} \circ \mathsf{F}(sk') = pk$.

According to the definition of Game 0, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr[S_0]$$

Game 1. Same as Game 0 except that the solution sk' such that $\mathsf{G} \circ \mathsf{F}(sk') = pk$ but $\mathsf{F}(sk') \neq r$ is not considered to be successful. Let E be the event that $\mathcal{A}$ outputs such a solution. Clearly, conditioned on E does not happen, $\mathcal{A}$'s view in Game 0 and Game 1 are identical. By the difference lemma, we have:

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[E]$$

Claim 5.7. *Assuming the target-collision resistance of* G, $\Pr[E]$ *is negligible in* λ.

Proof. Suppose $\mathcal{B}$ is an adversary against the target-collision resistance of G. Given G and $r \xleftarrow{\text{R}} R$, $\mathcal{B}$ simulates $\mathcal{A}$'s challenger in Game 1. $\mathcal{B}$ computes $pk \leftarrow \mathsf{G}(r)$, sends $(\mathsf{F} \circ \mathsf{G}, pk)$ to $\mathcal{A}$. $\mathcal{B}$ responds all leakage queries correctly with r. Finally, $\mathcal{A}$ outputs sk' and $\mathcal{B}$ sends $r' \leftarrow \mathsf{F}(sk')$ to its own challenger. Since the distributions $r \xleftarrow{\text{R}} R$ and $r \leftarrow \mathsf{F}(sk)$ for $sk \xleftarrow{\text{R}} \{0,1\}^n$ are identical, $\mathcal{B}$'s simulation is prefect. If E happens with some non-negligible probability, $\mathcal{B}$ breaks the assumed target-collision resistance of G with the same probability since $r' \neq r \wedge \mathsf{G}(r') = \mathsf{G}(r)$. The claim immediately follows. □

Claim 5.8. $\Pr[S_1]$ *is negligible in* λ *assuming the one-wayness of* F.

Proof. Let $\mathcal{A}$ be a PPT adversary that has non-negligible advantage in Game 1, we build a PPT algorithm $\mathcal{B}$ breaking the assumed one-wayness of F as below. Given r where $r \leftarrow \mathsf{F}(sk)$ for a randomly chosen sk, $\mathcal{B}$ simulates $\mathcal{A}$'s challenger in Game 1, with the aim to find a preimage of r under F.

Setup and challenge: $\mathcal{B}$ computes $pk \leftarrow \mathsf{G}(r)$, sends $(\mathsf{F} \circ \mathsf{G}, pk)$ to $\mathcal{A}$.

Leakage query: Upon receiving a leakage query $\langle h \circ \mathsf{F} \rangle$, $\mathcal{B}$ responds with $h(r)$.

Attack: $\mathcal{A}$ outputs a preimage sk', $\mathcal{B}$ forwards it to its own challenger.

According to the definition of Game 1. If $\mathcal{A}$ wins, we must have $\mathsf{G}(\mathsf{F}(sk')) = pk$ and $\mathsf{F}(sk') = r$. Thus $\mathcal{B}$ breaks the one-wayness of F with the same advantage as $\mathcal{A}$ wins in Game 1. This proves Claim 5.8. □

Putting all of the above together, we have $\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr[S_0] = \mathsf{negl}(\lambda)$. This proves Proposition 5.6. □

Corollary 5.9. *The relation* R_{key} *induced by* $\mathsf{G} \circ \mathsf{F}$ *is one-way even given leakage* dk.

Proof. Let K be the function that on input $r \in R$ outputs the second component of $\mathsf{KeyGen}(r)$, say dk. Clearly, $\mathsf{K} \circ \mathsf{F}$ is an admissible leakage function on sk, resulting leakage dk. This corollary thus immediately follows from Proposition 5.6. □

Proof. We are now ready to prove Lemma 5.5. The security proof is in spirit similar to that of signature scheme from identification protocol [Kat10, Theorem 8.1]. The difference here is that we treat the Sigma protocol as a ZKPoK for a leakage-resilient one-way relation, rather than a canonical identification protocol. This change brings us some conceptual advantages in that it avoids the need for explicit rewinding in the security reduction, thus simplifying the proof. We proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ wins in Game i.

Game 0. This is the standard security experiment for the signature component of HISE. We make the following simplified assumptions without any loss of generality. We require that if $\mathcal{A}$ outputs the forgery $\sigma^* = (a^*, z^*)$ on a message $\widetilde{m}^*$, then $\mathcal{A}$ must have previously asked the hash query $\mathsf{H}(a^*, \widetilde{m}^*)$. Let Q_h (resp. Q_s) be a polynomial upper bound on the number of hash queries (resp. signing queries) made by $\mathcal{A}$.

Setup: $\mathcal{CH}$ runs $\mathsf{Setup}(1^\lambda)$ to generate public parameters, that is, runs $pp_{\text{pke}} \leftarrow \text{PKE}.\mathsf{Setup}(1^\lambda)$, $pp_{\text{zkpok}} \leftarrow \text{ZKPoK}.\mathsf{Setup}(1^\lambda)$, picks a uniform one-way function $\mathsf{F} : \{0,1\}^n \to \{0,1\}^\ell$, sets $pp = (pp_{\text{pke}}, pp_{\text{zkpok}}, \mathsf{F})$; then picks $sk \xleftarrow{\text{R}} \{0,1\}^n$, computes $r \leftarrow \mathsf{F}(sk)$, $(ek, dk) \leftarrow \text{PKE}.\mathsf{KeyGen}(pp_{\text{pke}}; r)$, sets $pk = ek$, then sends (pp, pk, dk) to $\mathcal{A}$.

Random oracle query: $\mathcal{A}$ can adaptively make random oracle queries. Upon receiving a random query $\langle a, \widetilde{m} \rangle$, $\mathcal{CH}$ picks a random $e \xleftarrow{\text{R}} \Omega$, then records $(a, \widetilde{m}, e)$ into T.

Signing query: $\mathcal{A}$ can adaptively make signing queries. Upon receiving a signing query $\langle \widetilde{m} \rangle$, $\mathcal{CH}$ responds with $\sigma \leftarrow \mathsf{Sign}(sk, \widetilde{m})$.

Forgery: $\mathcal{A}$ finally outputs $(\widetilde{m}^*, \sigma^*)$. Let $\mathcal{Q} = \{m_i\}_{i \in Q_s}$ be the set of all queried messages. $\mathcal{A}$ wins if $m^* \notin \mathcal{Q}$ and $\mathsf{Vrfy}(vk, m^*, \sigma^*) = 1$.

Game 1. Same as Game 0 except that $\mathcal{CH}$ guesses a random index $j \xleftarrow{\text{R}} \{1, \ldots, Q_h\}$. This represents a guess to the index of the special hash query that will be explicitly made by $\mathcal{A}$, which corresponds to the forgery.[7] $\mathcal{CH}$ will abort if the guess is wrong, i.e., $\langle a_j, m_j \rangle \neq \langle a^*, m^* \rangle$. The random guess for j is totally hidden from $\mathcal{A}$, thus the probability that $\mathcal{CH}$ does not abort is exactly $1/Q_h$. We therefore have:

$$\Pr[S_1] = \Pr[S_0]/Q_h$$

Game 2. Same as Game 1 except that $\mathcal{CH}$ handles the signing queries by invoking zero-knowledge simulator $\mathcal{S}$ for Π.

Signing query: Upon receiving a signing query $\langle \widetilde{m} \rangle$, $\mathcal{CH}$ invokes $\mathcal{S}$ to obtain a transcript (a, e, z). If $\langle a, \widetilde{m} \rangle$ has been previously queried for hash value, $\mathcal{CH}$ aborts. Otherwise, $\mathcal{CH}$ sets $e := \mathsf{H}(a, \widetilde{m})$, records $(a, \widetilde{m}, e)$ to the list T, and responds with $\sigma = (a, z)$.

By the same claim we make in Claim 5.3, we conclude that $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{negl}(\lambda)$. It remains to calculate $\Pr[S_2]$. We have the following claim.

Claim 5.10. *For any PPT adversary $\mathcal{A}$, its advantage in Game 2 is negligible assuming the proof of knowledge of ZKPoK and one-wayness of R_{key} w.r.t. decryption key as leakage of signing key.*

Proof. We prove this claim by showing that an adversary $\mathcal{A}$ having non-negligible advantage in Game 2 implies an adversary $\mathcal{E}^{\mathcal{B}}$ also having non-negligible advantage against one-wayness of R_{key} w.r.t. decryption key as the leakage. Given public parameters of R_{key} (which includes pp_{pke} and the description of F) and a randomly sampled instance pk, $\mathcal{E}$ invokes $\mathcal{B}$ as a subroutine to emulate $\mathcal{A}$'s challenger in Game 2, with the aim to find a preimage sk of pk such that $(pk, sk) \in \mathsf{R}_{\text{key}}$. Figure 4 depicts the high-level idea that underlies the formal security proof.

Setup: $\mathcal{B}$ runs $\text{ZKPoK}.\mathsf{Setup}(1^\lambda)$ to generate pp_{zkpok}, sets $pp = (pp_{\text{pke}}, pp_{\text{zkpok}}, \mathsf{F})$. $\mathcal{B}$ also makes a leakage query of the form $\mathsf{K} \circ \mathsf{F}$ (as described in Corollary 5.9) to its own challenger and receives back dk. $\mathcal{B}$ sends (pp, pk, dk) to $\mathcal{A}$.

Random oracle query: $\mathcal{B}$ handles the random oracle queries in the same way as $\mathcal{CH}$ does in Game 2 except that when handling the j-th random oracle query $\langle a_j, \widetilde{m}_j \rangle$, $\mathcal{B}$ invokes a verifier V by playing the role of prover in ZKPoK. According to the definition of Game 2, we have $\langle a_j, \widetilde{m}_j \rangle = \langle a^*, \widetilde{m}^* \rangle$. $\mathcal{B}$ sends a^* to V as the 1st-move message, and receives V's response e^*. $\mathcal{B}$ then sets $e^* := \mathsf{H}(a^*, \widetilde{m}^*)$, and records $(a^*, \widetilde{m}^*, e^*)$ to the list T.

Signing query: $\mathcal{B}$ handles the signing queries in the same way as $\mathcal{CH}$ does in Game 2.

Forgery: When $\mathcal{A}$ outputs a forgery $(\sigma^* = (a^*, z^*), \widetilde{m}^*)$, $\mathcal{B}$ sends z^* to V as the 3rd-move message.

Clearly, $\mathcal{B}$'s simulation is perfect. If $\mathcal{A}$ succeeds with advantage $\varepsilon(\lambda)$, $\mathcal{B}$ convinces V to accept the proof with the same probability $\varepsilon(\lambda)$. According to the proof of knowledge property, $\mathcal{E}$ can thus extract a witness sk' such that $(pk, sk') \in \mathsf{R}_{\text{key}}$ via accessing $\mathcal{B}$ in a black-box manner with probability at least $\varepsilon(\lambda) - \mu(\lambda)$. If $\varepsilon(\lambda)$ is non-negligible in λ, the probability that $\mathcal{E}^{\mathcal{B}}$ outputs sk' such that $(pk, sk') \in \mathsf{R}_{\text{key}}$ is also non-negligible. This contradicts to the assumed leakage-resilient one-wayness of R_{key}, and thus proves Claim 5.10. □

[7] The reduction holds if $\mathcal{A}$ always make explicit random oracle queries. To guarantee this, we can only hope for standard unforgeability, not strong unforgeability.

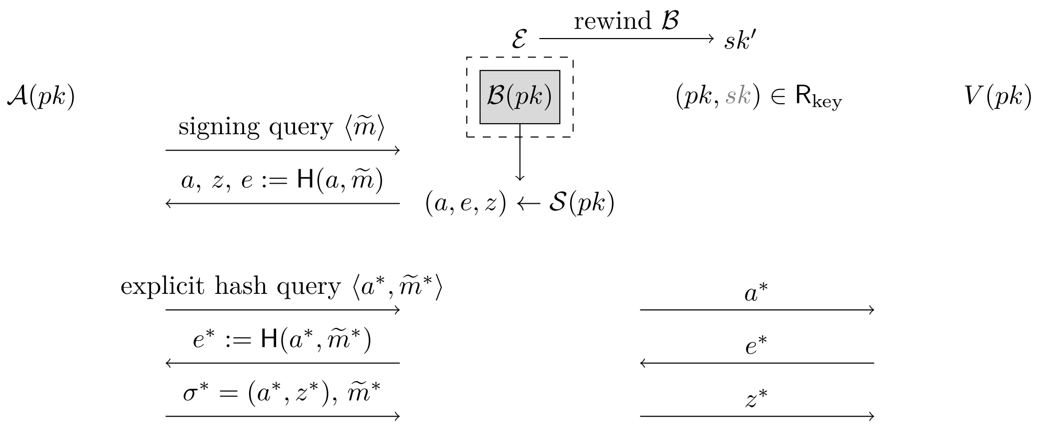

$\mathcal{E}$
rewind $\mathcal{B}$
sk'
$\mathcal{A}(pk)$
$\mathcal{B}(pk)$
$(pk, sk) \in \mathsf{R}_{\text{key}}$
$V(pk)$
signing query $\langle \widetilde{m} \rangle$
$a,\ z,\ e := \mathsf{H}(a, \widetilde{m})$
$(a, e, z) \leftarrow \mathcal{S}(pk)$
explicit hash query $\langle a^*, \widetilde{m}^* \rangle$
$e^* := \mathsf{H}(a^*, \widetilde{m}^*)$
$\sigma^* = (a^*, z^*),\ \widetilde{m}^*$
a^*
e^*
z^*

Figure 4: High-level idea of reduction

Putting all of the above together, Lemma 5.5 immediately follows. □

Remark 5.2. We note that the starting 3-round public-coin ZKPoK can be relaxed in two ways: (1) the proof of knowledge property can be relaxed to argument of knowledge property; (2) 3-round can be generalized to $(2k+1)$-round for any integer $k \geq 1$; in the generalized setting, we can use the BCS transform [BCS16] in the place of the Fiat-Shamir transform. The above relaxations greatly enrich the choices of the underlying zero-knowledge proof systems.

Remark 5.3. The above construction actually tells us that one can upgrade any PKE scheme that has been deployed (satisfying mild requirement) to an HISE scheme. However, users have to run the key generation algorithm of HISE from scratch to generate a new public key and the corresponding signing key and decryption key. It would be more desirable if the upgrade does not ask users to switch to using a new public key, since changing public key would be far more cumbersome (involves using new certificate and updating public-key related stuffs). A solution to this problem is using uniform trapdoor one-way function (TDF) to replace uniform one-way function. Along the ordinary PKE key generation step, each user also picks a uniform TDF, then records the trapdoor as well as the randomness used for key generation. At some point in the future, the user can upgrade to HISE by computing a preimage of the randomness as the signing key, keeping the public key unchanged. In this way, the upgrade is almost costless and perfect smooth.

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 and Section 5, 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 $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Dec}')$. The KeyGen, Enc, and Dec algorithms 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.

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

Here, $\mathcal{O}_{\mathsf{dec}}$ is the decryption oracle. $\mathcal{A}$ can make polynomial number of decryption queries with the restriction that $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{dec}}$ 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 $\mathcal{O}_{\mathsf{dec}}$ in Phase 1 (resp. denying access to $\mathcal{O}_{\mathsf{dec}}$).

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 the escrow agent generates a keypair (pk_γ, sk_γ) himself when building up the system , 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 $pp_{\text{pke}} \leftarrow \text{PKE}.\mathsf{Setup}(1^\lambda)$, $pp_{\text{nizk}} \leftarrow \text{NIZK}.\mathsf{Setup}(1^\lambda)$, $crs \leftarrow \text{NIZK}.\mathsf{CRSGen}(pp_{\text{nizk}})$, computes $(pk_\gamma, sk_\gamma) \leftarrow \text{PKE}.\mathsf{KeyGen}(pp_{\text{pke}})$, 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 check if c is a valid encryption under pk by running $\text{NIZK}.\mathsf{Verify}(crs, (pk, c_1, c_2), \pi)$; 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 6.1. *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 6.1. 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 $L_{pk}^{\text{kem}} = \{c \mid \exists r \text{ s.t. } (c, k) = \mathsf{Encaps}(pk; r)\}$, 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:

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

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.

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

Here, $\mathcal{O}_{\mathsf{decaps}}$ 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 $\mathcal{O}_{\mathsf{decaps}}$.

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

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 the public parameters pp, we define a collection of $\mathcal{NP}$ languages indexed by pk. Let $L_{pk} = \{(c_{\text{kem}}, c_{\text{ske}}) \mid \exists m, r_1, r_2 \text{ s.t. } (c_{\text{kem}}, k) = \text{KEM}.\mathsf{Encaps}(pk; r_1) \wedge c_{\text{ske}} = \text{SKE}.\mathsf{Enc}(k, m; r_2)\}$, which represents all valid ciphertexts encrypted under pk. It is easy to see that for any ciphertext no matter whether $c_{\text{kem}} \in L_{pk}^{\text{kem}}$ or not, the consistency of global escrow KEM guarantees that the decapsulation results are identical, and so are the final decryption results.

Theorem 6.2. *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 (see definition in Appendix A.8). 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 $pp_{\text{nike}} \leftarrow \text{NIKE}.\mathsf{Setup}(1^\lambda)$ and $(pk_\gamma, sk_\gamma) \leftarrow \text{NIKE}.\mathsf{KeyGen}(pp_{\text{nike}})$, outputs public parameters $pp = (pp_{\text{nike}}, pk_\gamma)$ and sets the global escrow decryption key $edk = sk_\gamma$.
- $\mathsf{KeyGen}(pp)$: parses $pp = (pp_{\text{nike}}, pk_\gamma)$, runs $\text{NIKE}.\mathsf{KeyGen}(pp_{\text{nike}})$ to generate a keypair (pk, sk).
- $\mathsf{Encaps}(pk)$: parses $pk = pk_\beta$, the sender runs $\text{NIKE}.\mathsf{KeyGen}(pp_{\text{nike}})$ 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 6.3. *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).*

Proof. We give the proof for the case of IND-CCA security, which carries over to the case of IND-CPA security. If there is an adversary $\mathcal{A}$ that breaks the IND-CCA security of the escrow KEM, we build an adversary $\mathcal{B}$ that breaks the CKS-light security of the three-party NIKE in the DKR setting with the same advantage. $\mathcal{B}$ interacts with $\mathcal{A}$ by simulating its challenger in the IND-CCA experiment.

Setup: Given pp_{nike}, $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{regH}}$ three times and receives back $(pk_\alpha, pk_\beta, pk_\gamma)$ and k^*, where k^* is either k_S where $S = (pk_\alpha, pk_\beta, pk_\gamma)$ or a random key. $\mathcal{B}$ sets $pp = (pp_{\text{nike}}, pk_\gamma)$, $pk = pk_\beta$, then sends (pp, pk) to $\mathcal{A}$.

Challenge: $\mathcal{B}$ sets $c^* = (pk_\alpha, pk_\beta)$, and sends (c^*, k^*) to $\mathcal{A}$.

Decapsulation query: $\mathcal{A}$ may adaptively make decapsulation queries. Upon receiving a decapsulation query $c \neq c^*$, if $c \notin L_{pk_\beta}$, $\mathcal{B}$ directly rejects according to the definition of Decaps. Otherwise, $\mathcal{B}$ first makes a corrupt user registration query with the first element of c, say pk, then makes a corrupt reveal query with $(pk, pk_\beta, pk_\gamma)$. Note that the restriction $c \neq c^*$ ensures that $(pk, pk_\beta, pk_\gamma) \neq S$, thus the corrupt reveal queries made by $\mathcal{B}$ are always permissible. After receiving the response k from its own challenger, $\mathcal{B}$ forwards it to $\mathcal{A}$.

Guess: Finally, $\mathcal{A}$ outputs its guess b' for b and $\mathcal{B}$ forwards it to its own challenger.

Clearly, $\mathcal{B}$'s emulation is perfect. If $k^* = k_S$, then k^* is the session key encapsulated by ciphertext pk_α under pk_β. If k^* is a random key, then k^* is also a random session key. Therefore, $\mathcal{B}$ succeeds with the same advantage as $\mathcal{A}$. This proves the theorem. □

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. As we discussed in Section A.8, 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 asymmetric 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 Type-III 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 4.2 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 Two Instantiations of HISE

7.1.1 HISE from IBE

We instantiate our first generic HISE construction (presented in Section 4.2) by choosing Boneh-Franklin IBE with asymmetric pairing (recall in Appendix A.5.1) 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 $g_1 \xleftarrow{R} \mathbb{G}_1$, 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 $sk \xleftarrow{R} \mathbb{Z}_p$, 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 $r \xleftarrow{R} \mathbb{Z}_p$, 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 $r \xleftarrow{R} \mathbb{Z}_p$, outputs "1" if $e(pk, \mathsf{H}(0||\tilde{m}))^r = e(g_1^r, \sigma)$ and "0" otherwise.

Remark 7.1. 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][8], 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]).

7.1.2 HISE from PKE and ZKPoK

We instantiate our second generic construction of HISE (presented in Section 5) from the following building blocks, yielding HISE scheme 2.

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 $r \xleftarrow{\text{R}} \mathbb{Z}_p$ 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 Appendix B.3 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, R_{key} defined by $\mathsf{G} \circ \mathsf{F}$ is unlikely to be an algebraic relation. As a consequence, it is difficult to prove R_{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. such as ZKBoo and its variants [GMO16, CDG+17, KKW18], Ligero and its improved version [AHIV17, RBZ20], Bulletproof [BBB+18], zk-STARK [BBHR18], Aurora [BCR+19], Spartan [Set20] and its extensions [SL20]. In our instantiation, we choose Spartan [Set20]. We convert the proved relation R_{key} into R1CS format using xJsnark [KPS18]; the number of R1CS constraints of is roughly $680,000 \approx 2^{20}$.

With the above building blocks, HISE scheme 2 is as below.

- $\mathsf{Setup}(1^\lambda)$: 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 $pp_{\text{nizkpok}} \leftarrow \mathsf{NIZKPoK.Setup}(1^\lambda)$, and outputs $pp = (\mathsf{F}, pp_{\text{nizkpok}})$. The plaintext space is $M = \mathbb{G}$. The message space is $\widetilde{M} = \{0,1\}^*$.
- $\mathsf{KeyGen}(pp)$: on input $pp = (\mathsf{F}, pp_{\text{nizkpok}})$, picks $sk \xleftarrow{\text{R}} \{0,1\}^n$, 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 $r \xleftarrow{\text{R}} \mathbb{Z}_p$, 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 $\sigma \leftarrow \text{NIZKPoK}.\mathsf{Prove}(pk, sk, \tilde{m})$.
- $\mathsf{Vrfy}(pk, \tilde{m}, \sigma)$: on input pk, $\tilde{m}$ and σ, outputs $b \leftarrow \text{NIZKPoK}.\mathsf{Verify}(pk, \sigma, \tilde{m})$.

[8] 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 Two Instantiations of Global Escrow HISE

As depicted in Figure 1 in the introduction part, there are two paths to build global escrow HISE. We present one instantiation per path as below.

7.2.1 Global Escrow HISE via the GE Conversion

Our first construction is along the path enabled by the GE conversion. Starting from the HISE scheme presented in Section 7.1.1, we compile it into a global escrow one by applying the twisted Naor-Yung transform [BMV16], yielding global escrow HISE scheme 1.

7.3 Instantiation of Global Escrow HISE (via GE conversion)

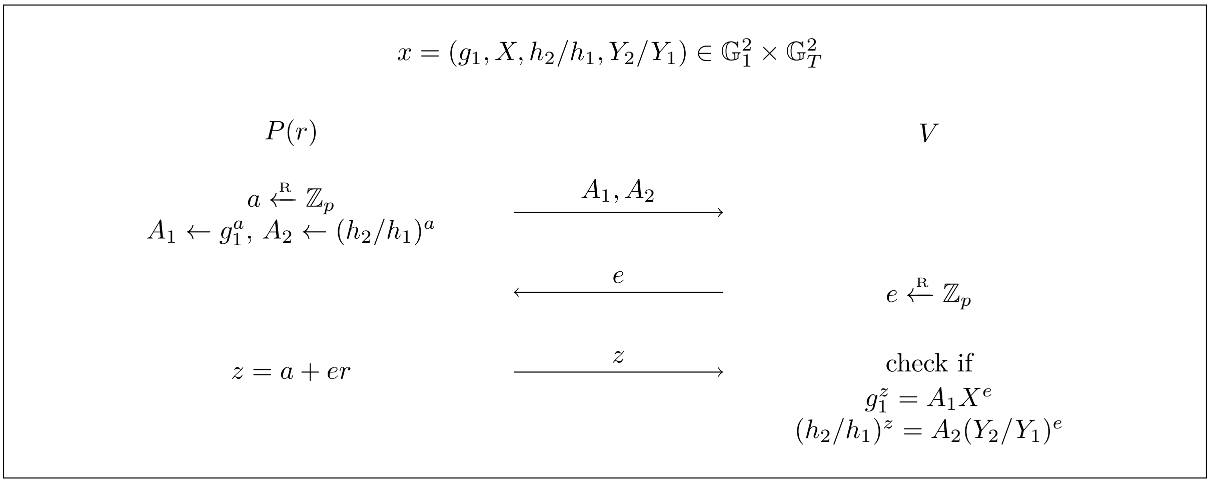
We describe global escrow HISE scheme 1 as follows. Its KeyGen, Derive, Sign and Vrfy algorithms are the same as those of the starting HISE. We describe its Setup, Enc, Dec and Dec' algorithms as below.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathsf{BLGroupGen}(1^\lambda)$, sets $id^* = 1^{\ell+1}$, samples $s \xleftarrow{\text{R}} \mathbb{Z}_p$, computes $epk \leftarrow g_1^s \in \mathbb{G}_1$, $edk \leftarrow \mathsf{H}(id^*)^s \in \mathbb{G}_2$, picks cryptographic hash functions $\mathsf{H} : \{0,1\}^{\ell+1} \to \mathbb{G}_2$, $\tilde{\mathsf{H}} : \mathbb{G}_1^3 \times \mathbb{G}_T^3 \to \mathbb{Z}_p$, outputs global escrow decryption key edk and public parameters pp that include epk and the descriptions of bilinear groups and hash functions.

- $\mathsf{Enc}(pk, m)$: on input pk and $m \in \mathbb{G}_T$, picks $r \xleftarrow{\text{R}} \mathbb{Z}_p$, computes a "double encryption" of m under pk and epk, i.e., $X = g_1^r \in \mathbb{G}_1$, $Y_1 = h_1^r \cdot m \in \mathbb{G}_T$ and $Y_2 = h_2^r \cdot m \in \mathbb{G}_T$, where $h_1 = e(pk, \mathsf{H}(id^*))$, $h_2 = e(epk, \mathsf{H}(id^*))$, then generates a (simulation-sound) NIZK proof π for the fact that (X, Y_1, Y_2) is ciphertext encrypting the same plaintext with shared randomness; this is equivalent to showing $\log_{g_1} X = \log_{h_2/h_1} Y_2/Y_1$. Such proof can be generated by applying the Fiat-Shamir transform [FS86, FKMV12] to the Sigma protocol (described in Figure 5), with the challenge defined as $e := \tilde{\mathsf{H}}(x||A_1||A_2)$ through the application of the random oracle $\tilde{\mathsf{H}}$, yielding a simulation-sound NIZK proof $\pi = (A_1, A_2, z)$ in the random oracle model. In this way, the final ciphertext $c = (X, Y_1, Y_2, \pi)$ consists of 6 group elements.

- $\mathsf{Dec}(dk, c)$: on input dk and $c = (X, Y_1, Y_2)$, first runs NIZK.$\mathsf{Verify}(x, \pi)$ to perform the consistency check; if the proof is invalid then outputs $\perp$. Otherwise, outputs IBE.$\mathsf{Dec}(dk, (X, Y_1))$.

- $\mathsf{Dec}'(edk, c)$: on input edk and $c = (X, Y_1, Y_2)$, first runs NIZK.$\mathsf{Verify}(x, \pi)$ to perform the consistency check; if the proof is invalid then outputs $\perp$. Otherwise, outputs IBE.$\mathsf{Dec}(edk, (X, Y_2))$.



$$x = (g_1, X, h_2/h_1, Y_2/Y_1) \in \mathbb{G}_1^2 \times \mathbb{G}_T^2$$

$P(r)$

V

$a \xleftarrow{\text{R}} \mathbb{Z}_p$

$A_1 \leftarrow g_1^a$, $A_2 \leftarrow (h_2/h_1)^a$

$\xrightarrow{A_1, A_2}$

$\xleftarrow{e}$

$e \xleftarrow{\text{R}} \mathbb{Z}_p$

$z = a + er$

$\xrightarrow{z}$

check if

$g_1^z = A_1 X^e$

$(h_2/h_1)^z = A_2 (Y_2/Y_1)^e$

Figure 5: Sigma protocol for discrete logarithm equality in two isomorphic groups

We realize global escrow HISE scheme 1 atop pairing-friendly curve `bls12-381`.

7.3.1 Global Escrow HISE via the HI Conversion

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.

7.4 Instantiation of Global Escrow HISE (via HI conversion)

We describe global escrow HISE scheme 2 as below.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e) \leftarrow \mathsf{BLGroupGen}(1^\lambda)$, picks $edk \overset{\text{R}}{\leftarrow} \mathbb{Z}_p$, computes $pk_\gamma^1 \leftarrow g_1^{edk} \in \mathbb{G}_1$, $pk_\gamma^2 \leftarrow g_2^{edk} \in \mathbb{G}_2$, picks a uniform one-way function $\mathsf{F} : \{0,1\}^n \to \{0,1\}^\ell$, runs $pp_{\text{nizkpok}} \leftarrow \mathsf{NIZKPoK.Setup}(1^\lambda)$, outputs $pp = (\mathsf{F}, pp_{\text{nizkpok}}, epk = (pk_\gamma^1, pk_\gamma^2))$ and edk. The plaintext space is $M = \mathbb{G}$. The message space is $\widetilde{M} = \{0,1\}^*$.

- $\mathsf{KeyGen}(pp)$: on input $pp = (\mathsf{F}, pp_{\text{nizkpok}}, epk)$, picks $sk \overset{\text{R}}{\leftarrow} \{0,1\}^n$, computes $pk \leftarrow g_1^{\mathsf{F}(sk)} \in \mathbb{G}_1$.

- $\mathsf{Derive}(sk)$: on input sk, outputs $dk \leftarrow \mathsf{F}(sk) \in \mathbb{Z}_p$.

- $\mathsf{Enc}(pk, m)$: on input $pk = pk_\beta$ and $m \in \mathbb{G}_T$, picks $dk_\alpha \overset{\text{R}}{\leftarrow} \mathbb{Z}_p$, computes $X \leftarrow g_2^{dk_\alpha} \in \mathbb{G}_2$, $k \leftarrow e(pk_\beta, pk_\gamma^2)^{dk_\alpha}$, $Y \leftarrow k \cdot m \in \mathbb{G}_T$, outputs $c = (X, Y)$.

- $\mathsf{Dec}(dk, c)$: on input $dk = dk_\beta$ and $c = (X, Y)$, outputs $m \leftarrow Y / e(pk_\gamma^1, X)^{dk_\beta}$.

- $\mathsf{Dec}'(edk, c)$: on input $edk = dk_\gamma$ and $c = (X, Y)$, outputs $m \leftarrow Y / e(pk_\beta, X)^{dk_\gamma}$.

- $\mathsf{Sign}(sk, \tilde{m})$: computes $\sigma \leftarrow \text{NIZKPoK}.\mathsf{Prove}(pk, sk, \tilde{m})$, where pk is the instance, sk is the witness, $\tilde{m}$ is treated as the auxiliary input. The number of constraints of proved relation is roughly 2^{20}.

- $\mathsf{Vrfy}(pk, \tilde{m}, \sigma)$: outputs $b \leftarrow \text{NIZKPoK}.\mathsf{Verify}(pk, \sigma, \tilde{m})$.

We realize global escrow HISE scheme 2 atop pairing-friendly curve `bls12-381` and use the same uniform one-way function and NIZKPoK as specified in Section 7.1.2.

The joint security of the above two schemes follows from the fact that the signing key is independent of the global escrow decryption key.

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.

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]	✓	✓	✗	✗	×2
ISE [PSST11]	✗	✗	✗	✓	×1
HISE	✓	✓	✗	✓	×1
global escrow HISE	✓	✓	✓	✓	×1

For certificate cost, ×1 (resp. ×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 gurantee.

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′	$\|pk\|$	$\|sk\|$	$\|c\|$	$\|\sigma\|$
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[9] 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.

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 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 $\mathsf{R}_{\mathsf{key}}$. 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.

[9] 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 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′	$\|pp\|$	$\|edk\|$	$\|pk\|$	$\|sk\|$	$\|c\|$
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.

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 Appendix B.1) 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).

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)[10] 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.

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 notion 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.

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

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 *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015*, pages 1370–1381. ACM, 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 *Advances in Cryptology - CRYPTO 2018*, volume 10993 of *Lecture Notes in Computer Science*, pages 643–673. Springer, 2018.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*, pages 2087–2104, 2017.

[AMPR19] Navid Alamati, Hart Montgomery, Sikhar Patranabis, and Arnab Roy. Minicrypt primitives with algebraic structure and applications. In *Advances in Cryptology - EUROCRYPT 2019*, volume 11477 of *Lecture Notes in Computer Science*, pages 55–82. Springer, 2019.

[BAZB20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *Financial Cryptography and Data Security - FC 2020*, volume 12059, pages 423–443. Springer, 2020.

[BBB$^+$18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018*, pages 315–334, 2018.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. 2018. http://eprint.iacr.org/2018/046.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology - EUROCRYPT 2016*, pages 327–357, 2016.

[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.

[BCR$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *Advances in Cryptology - EUROCRYPT 2019*, pages 103–128, 2019.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60. Springer, 2016.

[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 *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532, 2001.

[BMV16] Silvio Biagioni, Daniel Masny, and Daniele Venturi. Naor-yung paradigm with shared randomness and applications. In *Security and Cryptography for Networks - 10th International Conference, SCN 2016*, volume 9841 of *Lecture Notes in Computer Science*, pages 62–80. Springer, 2016.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. 2002. http://eprint.iacr.org/2002/080.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology - ASIACRYPT 2013*, volume 8270 of *LNCS*, pages 280–300. Springer, 2013.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Advances in Cryptology - CRYPTO 2014*, pages 480–499, 2014.

[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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*, pages 1825–1842, 2017.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, 2003.

[CJNP02] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. Universal padding schemes for RSA. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 226–241. Springer, 2002.

[CKS08] David Cash, Eike Kiltz, and Victor Shoup. The twin diffie-hellman problem and applications. In *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 127–145. Springer, 2008.

[CMTA20] Yu Chen, Xuecheng Ma, Cong Tang, and Man Ho Au. PGC: Pretty Good Confidential Transaction System with Auditability. In *The 25th European Symposium on Research in Computer Security, ESORICS 2020*, 2020.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology - EUROCRYPT 2002*, pages 45–64, 2002.

[Dam] Ivan Damgård. On sigma protocols. http://www.cs.au.dk/~ivan/Sigma.pdf.

[DH76] Whitefield Diffie and Martin E. Hellman. New directions in cryptograpgy. *IEEE Transactions on Infomation Theory*, 22(6):644–654, 1976.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51th Annual IEEE Symposium on Foundations of Computer Science, 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, *Topics in Cryptology - CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 116–135. Springer, 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.

[FHKP13] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In *16th International Conference on Practice and Theory in Public-Key Cryptography - PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, 2013.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *Progress in Cryptology - INDOCRYPT 2012*, pages 60–79, 2012.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology - CRYPTO 1999*, volume 1666 of *LNCS*, pages 537–554, 1999.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO 1986*, pages 186–194, 1986.

[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.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *25th USENIX Security Symposium, USENIX Security 2016*, pages 1069–1083, 2016.

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

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008*, pages 197–206. ACM, 2008.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2322 of *LNCS*, pages 466–481. Springer, 2002.

[HP01] Stuart Haber and Benny Pinkas. Securely combining public-key cryptosystems. In *Proceedings of the 8th ACM Conference on Computer and Communications Security, CCS 2001*, pages 215–224. ACM, 2001.

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

[Kat10] Jonathan Katz. *Digital Signatures*. Springer US, 2010.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018*, pages 525–537. ACM, 2018.

[KO03] Yuichi Komano and Kazuo Ohta. Efficient universal padding techniques for multiplicative trapdoor one-way permutation. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 366–382. Springer, 2003.

[KPS18] Ahmed E. Kosba, Charalampos Papamanthou, and Elaine Shi. xjsnark: A framework for efficient verifiable computation. In *2018 IEEE Symposium on Security and Privacy, SP 2018*, pages 944–961. IEEE Computer Society, 2018.

[NVV18] Neha Narula, Willy Vasquez, and Madars Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018*, pages 65–80, 2018.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing, STOC 1990*, pages 427–437. ACM, 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 *Advances in Cryptology - ASIACRYPT 2011*, pages 161–178, 2011.

[PY99] Pascal Paillier and Moti Yung. Self-escrowed public-key infrastructures. In *Information Security and Cryptology - ICISC 1999*, volume 1787 of *Lecture Notes in Computer Science*, pages 257–268. Springer, 1999.

[RBZ20] Carmit Hazay Muthuramakrishnan Venkitasubramaniam Tiancheng Xie Rishabh Bhadauria, Zhiyong Fang and Yupeng Zhang. Ligero++: A new optimized sublinear iop. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS 2020*, 2020.

[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 *Advances in Cryptology - CRYPTO 2020*, volume 12172 of *Lecture Notes in Computer Science*, pages 704–737. Springer, 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.

[SL20] Srinath Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zksnarks. Cryptology ePrint Archive, Report 2020/1275, 2020. https://eprint.iacr.org/2020/1275.

[Ver01] Eric R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 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 *Advances in Cryptology - EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 1998.

[YY99] Adam L. Young and Moti Yung. Auto-recoverable cryptosystems with faster initialization and the escrow hierarchy. In *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC 1999*, volume 1560 of *Lecture Notes in Computer Science*, pages 306–314. Springer, 1999.

A Review of Standard Cryptographic Primitives

We provide the standard definitions of bilinear maps, PKE schemes, digital signature schemes, IBE schemes, BTE schemes, zero-knowledge proof systems, as well as non-interactive key exchange protocols.

A.1 Bilinear Maps

Let $\mathsf{BLGroupGen}$ be a PPT algorithm that on input 1^λ return a description $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of the same prime order $p = \Theta(2^\lambda)$, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$. Following the standard terminology, we refer to e as *pairing*. Pairings fall into three basic types:

- Type-I: $\mathbb{G}_1 = \mathbb{G}_2$;
- Type-II: $\mathbb{G}_1 \neq \mathbb{G}_2$ but there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$;
- Type-III: $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

As summarized in [AGH15], Type-I called “symmetric” is typically how schemes are described and proven secure in the literature, since it is simpler and the complexity assumptions can be weaker; Type-II and Type-III called “asymmetric”, in which Type-III is typically the most efficient choice for implementation. Next, we recall the decisional bilinear Diffie-Hellman (DBDH) assumption in bilinear groups equipped with asymmetric pairing.

Definition A.1 (DBDH Assumption)**.** The DBDH assumption holds if for any PPT adversary, we have:

$$|\Pr[\mathcal{A}(g_1^a, g_1^b, g_2^c, e(g_1, g_2)^{abc}) = 1] - \Pr[\mathcal{A}(g_1^a, g_1^b, g_2^c, e(g_1, g_2)^z) = 1]| \leq \mathsf{negl}(\lambda)$$

where the probability is over the randomness of $\mathsf{BLGroupGen}(1^\lambda)$, $\mathcal{A}$'s random tape, and the random choices of $a, b, c, z \xleftarrow{\text{R}} \mathbb{Z}_p$.

By augmenting the above tuples with an additional element g_2^a, we obtain a slight stronger assumption known as the co-DBDH assumption, which stipulates that the distributions of $(g_1^a, g_1^b, g_2^a, g_2^c, e(g_1, g_2)^{abc})$ and $(g_1^a, g_1^b, g_2^a, g_2^c, e(g_1, g_2)^z)$ are computationally indistinguishable.

A.2 Symmetric Key Encryption

An SKE scheme consists of four polynomial-time algorithms as follows.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, outputs public parameters pp. We assume pp also includes the descriptions of plaintext space M, key space K, and ciphertext space C.
- $\mathsf{KeyGen}(pp)$: on input pp, outputs a symmetric key k.
- $\mathsf{Enc}(k, m)$: on input k and a plaintext m, outputs a ciphertext c.
- $\mathsf{Dec}(k, c)$: on input k and a ciphertext c, outputs a plaintext m or a special reject symbol $\perp$ denoting failure.

Correctness. For any $\lambda \in \mathbb{N}$ and any $m \in M$, it holds that $\Pr[\mathsf{Dec}(k, c) = m] \geq 1 - \mathsf{negl}(\lambda)$, where the probability is taken over the choice $pp \leftarrow \mathsf{Setup}(1^\lambda)$, $k \leftarrow \mathsf{KeyGen}(pp)$, and $c \leftarrow \mathsf{Enc}(k, m)$.

IND-CCA security. Let $\mathcal{A}$ be an adversary against SKE and define its advantage as:

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

An SKE scheme is IND-CCA secure if no stateful PPT adversary $\mathcal{A}$ has non-negligible advantage in the above security experiment. The IND-CPA security can be defined similarly by denying access to $\mathcal{O}_{\mathsf{dec}}$.

A.3 Public Key Encryption

A PKE scheme consists of four polynomial-time algorithms as follows.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, outputs public parameters pp. We assume pp also includes the descriptions of plaintext space M, ciphertext space C, and randomness space R.
- $\mathsf{KeyGen}(pp)$: on input pp, outputs a (public) encryption key ek and a (private) decryption key dk.
- $\mathsf{Enc}(ek, m)$: on input an encryption key ek and a plaintext m, outputs a ciphertext c. When emphasizing the randomness r used for encryption, we write this as $c \leftarrow \mathsf{Enc}(ek, m; r)$.
- $\mathsf{Dec}(dk, c)$: on input a decryption key dk and a ciphertext c, outputs a plaintext m or a special reject symbol $\perp$ denoting failure. This algorithm is typically deterministic.

Correctness. For any $\lambda \in \mathbb{N}$ and 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 $pp \leftarrow \mathsf{Setup}(1^\lambda)$, $(ek, dk) \leftarrow \mathsf{KeyGen}(pp)$, and $c \leftarrow \mathsf{Enc}(ek, m)$.

IND-CCA security. Let $\mathcal{A}$ be an adversary against PKE and define its advantage as:

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

A PKE scheme is IND-CCA secure if no stateful PPT adversary $\mathcal{A}$ has non-negligible advantage in the above security experiment. The IND-CPA security can be defined similarly by denying the decryption oracle.

A.4 Digital Signature

A signature scheme consists of four polynomial-time algorithms as follows.

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

Correctness. For any $\lambda \in \mathbb{N}$ and any $m \in M$, it holds that $\Pr[\mathsf{Very}(pk, m, \sigma)] = 1 - \mathsf{negl}(\lambda)$, where the probability is taken over the choice $pp \leftarrow \mathsf{Setup}(1^\lambda)$, $(vk, sk) \leftarrow \mathsf{KeyGen}(pp)$, and $\sigma \leftarrow \mathsf{Sign}(sk, m)$.

EUF-CMA security. Let $\mathcal{A}$ be an adversary against signature component and define its advantage as:

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

The set $\mathcal{Q}$ records queries to $\mathcal{O}_{\mathsf{sign}}$. A signature is EUF-CMA secure if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the above security experiment. The *strong* EUF-CMA security can be defined similarly by asking $\mathcal{A}$ to output a fresh valid message-signature tuple.

A.5 Identity-Based Encryption

Formally, an IBE scheme [BF03] 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 descriptions of identity space I, plaintext space M and ciphertext space C.

- $\mathsf{KeyGen}(pp)$: on input public parameters pp, outputs a master public key mpk and a master secret key msk.

- $\mathsf{Extract}(msk, id)$: on input msk and an identity $id \in I$, outputs a secret key sk_{id} for id.

- $\mathsf{Enc}(mpk, id, m)$: on input mpk, an identity $id \in I$, and a plaintext $m \in M$, outputs a ciphertext $c \in C$.

- $\mathsf{Dec}(sk_{id}, c)$: on input a secret key sk_{id} for identity id and a ciphertext $c \in C$, outputs a plaintext $m \in M$ or a distinguished reject symbol $\perp$ indicating that c is invalid.

Correctness. For any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$, any identity $id \in I$ and secret key $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$, and any plaintext $m \in M$, we have $m = \mathsf{Dec}(sk_{id}, \mathsf{Enc}(mpk, id, m))$.

Security. Let $\mathcal{A}$ be an adversary against the IND-CPA security of IBE and define its advantage in the following experiment:

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

$\mathcal{O}_{\mathsf{ext}}$ denotes the key extraction oracle, which on input id returns $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$. $\mathcal{A}$ can query $\mathcal{O}_{\mathsf{ext}}$ with any identity but id^*. An 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 message from a random ciphertext. The other one is selective-identity IND-CPA security, in which the adversary is asked to specify the target identity id^* in advance, before mpk is published.

A.5.1 Boneh-Franklin IBE Scheme

We recall the Boneh-Franklin IBE scheme with asymmetric pairings [BF03] as below.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e) \leftarrow \mathsf{BLGroupGen}(1^\lambda)$, picks a cryptographic hash function $\mathsf{H} : \{0,1\}^\ell \rightarrow \mathbb{G}_2$. The public parameters pp includes the descriptions of bilinear groups and H. The identity space is $I = \{0,1\}^\ell$. The plaintext space is $M = \mathbb{G}_T$. The ciphertext space is $C = \mathbb{G}_1 \times \mathbb{G}_T$.

- $\mathsf{KeyGen}(pp)$: on input pp, picks $msk \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, computes $mpk \leftarrow g_1^{msk} \in \mathbb{G}_1$.

- $\mathsf{Extract}(msk, id)$: on input msk and an identity $id \in \{0,1\}^\ell$, outputs $sk_{id} \leftarrow \mathsf{H}(id)^{msk} \in \mathbb{G}_2$.

- $\mathsf{Enc}(mpk, id, m)$: on input mpk, an identity $id \in \{0,1\}^\ell$ and a plaintext $m \in \mathbb{G}_T$, picks $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, computes $c_1 \leftarrow g_1^r$, $k \leftarrow e(mpk, \mathsf{H}(id))^r$, $c_2 = k \cdot m$, outputs $c = (c_1, c_2)$.

- $\mathsf{Dec}(sk_{id}, c)$: on input sk_{id} and $c = (c_1, c_2)$, computes $k \leftarrow e(c_1, sk_{id})$, outputs $m \leftarrow c_2 \cdot k^{-1}$.

Boneh-Franklin IBE is IND-CPA secure based on the DBDH assumption by modeling H as a random oracle.

A.6 Binary Tree Encryption

Canetti et al. [CHK03] defined a relaxed variant of hierarchical identity-based encryption (HIBE) [GS02, HL02] called binary tree encryption (BTE). The only difference between HIBE and BTE is that in the former the hierarchy tree can have arbitrary degree, and a child of node id at level j is labeled (id, id_{j+1}) for an arbitrary id_{j+1}, whereas in the latter the hierarchy tree is a complete binary one, and the children of a node id at level j are labeled $id||0$ and $id||1$. We describe the formal definition of BTE as below. Our definition is slightly different but actually equivalent to the original one in [CHK03]. Formally, a BTE scheme consists of the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: on input a security parameter λ and a value ℓ representing the depth of the tree, outputs public parameters pp. The identity space I is $\{0,1\}^{\leq \ell}$, which represents the set of all binary strings whose length is less or equal than ℓ.
- $\mathsf{KeyGen}(pp)$: on input public parameters pp, outputs a master public key mpk and a master secret key msk. Alternatively, msk can be written as sk_ϵ, meaning the secret key for the root node.
- $\mathsf{Delegate}(sk_{id}, b)$: on input a secret key sk_{id} for node $id \in \{0,1\}^{<\ell}$ and a bit $b \in \{0,1\}$, outputs a secret key $sk_{id||b}$. This algorithm is used to delegate secret keys along the hierarchy. Note that msk is essentially the secret key at depth 0, an efficient secret key extraction algorithm $\mathsf{Extract}$ is thus off-the-shelf, which can be defined by calling this algorithm iteratively.
- $\mathsf{Enc}(mpk, id, m)$: on input mpk, a node $id \in \{0,1\}^{\leq \ell}$, and a plaintext m, outputs a ciphertext c.
- $\mathsf{Dec}(sk_{id}, c)$: on input a secret key sk_{id} for node $id \in \{0,1\}^{\leq \ell}$, its secret key sk_{id}, and a ciphertext c, outputs a plaintext m or a distinguished reject symbol $\perp$ indicating c is invalid.

Correctness. For any $pp \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, any $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$, any identity $id \in \{0,1\}^{\leq \ell}$, any secret key $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$, and any plaintext m, we have $m = \mathsf{Dec}(sk_{id}, \mathsf{Enc}(mpk, id, m))$.

Security. Roughly speaking, a secure BTE should ensure the secrecy of ciphertexts encrypted by id even if the secret keys of other identities (as long as they are not ancestors of id) are exposed. We formally define IND-CPA security for BTE as below. Let $\mathcal{A}$ be an adversary against the IND-CPA security of BTE and define its advantage in the following experiment:

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

$\mathcal{O}_{\mathsf{ext}}$ denotes key extraction oracle, which on input identity id returns $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$. $\mathcal{A}$ can query $\mathcal{O}_{\mathsf{ext}}$ at any point but id^* and its ancestors. A BTE 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 message from a random ciphertext. The other one is selective-identity IND-CPA security, in which the adversary is asked to commit the challenge identity id^* even before seeing mpk.

A.7 Zero-Knowledge Protocols

We begin with the definition of interactive proof systems.

Definition A.2 (Interactive Proof System). An interactive proof system is a two-party protocol in which a prover can convince a verifier in an interactive manner that some statement is true without revealing any knowledge about why it holds. A round consists of a message sent from one party to the other. Towards uttermost generality, we define an additional setup algorithm Setup, which is executed once and for all by a possibly trusted party. Formally, an interactive proof system consists of three PPT algorithms (Setup, P, V) as below.

- $\mathsf{Setup}(1^\lambda)$: on input the security parameter λ, outputs public parameters pp. Let $\mathsf{R}_{pp} \subseteq X \times W$ be an $\mathcal{NP}$ relation indexed by pp. We say $w \in W$ is a witness for a statement x iff $(x, w) \in \mathsf{R}_{pp}$. R_{pp} naturally defines a family of public-parameters-dependent $\mathcal{NP}$ languages:

  $$L_{pp} = \{x \mid \exists w \in W \text{ s.t. } (x, w) \in \mathsf{R}_{pp}\}$$

  We will drop the subscript pp occasionally when the context is clear.

- P and V are a pair of interactive algorithms, which both take pp as implicit input and the statement x as common input. We use the notation $tr \leftarrow \langle P(x), V(y)\rangle$ to denote the transcript of an execution between P and V, where P has input x and V has input y. We write $\langle P(x), V(y)\rangle = b$ depending on whether V accepts, $b = 1$, or rejects, $b = 0$. When the context is clear, we will also slightly abuse the notation of $\langle P(x), V(y)\rangle$ to denote V's view (View_V) in the interaction, which consists of V's input tape, random tape and incoming messages sent by P. If all messages sent from V are chosen uniformly at random and independent of P's message, we say the interactive proof system is public-coin.

An interactive proof system is called zero-knowledge proof of knowledge (ZKPoK) if it satisfies the following three properties:

Completeness. For any $(x, w) \in \mathsf{R}_{pp}$ where $pp \leftarrow \mathsf{Setup}(1^\lambda)$, it holds that:

$$\Pr[\langle P(x, w), V(x)\rangle = 1] = 1$$

Proof of knowledge. This property is an enhancement of soundness. Formally, for $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and any prover P^*, there exists an expected PPT extractor $\mathcal{E}$, such that for all x, if $\Pr[\langle P^*(x), V(x)\rangle = 1] \geq \varepsilon(\lambda)$, then $\Pr[(x, w) \in \mathsf{R}_{pp} : \mathcal{E}^{P^*}(x) = w] \geq \varepsilon(\lambda) - \mu(\lambda)$, where $\mu(\lambda)$ is a negligible function in λ. If the existence of $\mathcal{E}$ relies on additional computational assumptions, proof of knowledge is weakened to argument of knowledge.

Statistical zero-knowledge. For any malicious PPT V^*, there exists an expected PPT simulator $\mathcal{S}$ such that for $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and any $(x, w) \in \mathsf{R}_{pp}$, we have:

$$\langle P(x, w), V^*(x)\rangle \approx_s \mathcal{S}(x)$$

Statistical zero-knowledge can be strengthened to perfect (resp. weakened to computational) zero-knowledge by requiring that the real views and simulated views are identically (resp. computationally) indistinguishable.

Definition A.3 (Sigma Protocol (Σ-protocol) [Dam])**.** An interactive proof system is called a Sigma protocol if it follows the following communication pattern (3-round public-coin):

1. (Commit) P sends a first message a to V;
2. (Challenge) V sends a random challenge e to P;
3. (Response) P replies with a second message z.

and satisfies standard completeness and the variants of soundness and zero-knowledge as below:

n-Special soundness. There exists a PPT extractor that can compute the witness for any x giving n accepting transcripts $\{(a, e_i, z_i)\}_{i\in[n]}$ with the same initial message and distinct challenge e_i. By the general forking lemma [BCC+16, BBB+18], n-special soundness implies proof of knowledge.

Honest-verifier zero-knowledge (HVZK). There exists a PPT simulator $\mathcal{S}$ such that for any $(x, w) \in \mathsf{R}_{pp}$, we have:

$$\langle P(x, w), V(x)\rangle \equiv \mathcal{S}(x)$$

The Fiat-Shamir transform can crush any public-coin interactive proof system into a non-interactive one. Generally, we have the following theorem.

Theorem A.1 (Fiat-Shamir Transform [BR93, FKMV12])**.** *Let (Setup, P, V) be a $(2k+1)$-move public-coin HVZK proof of knowledge, x be the statement, a_i be the prover P's ith round message and e_i be verifier V's ith round challenge, and H be a hash function with range equal to V's challenge space. By setting $e_i = \mathsf{H}(a_1, \ldots, a_i)$ in (Setup, P, V), we obtain $(\mathsf{Setup}, P^{\mathsf{H}}, V^{\mathsf{H}})$, which is a NIZKPoK assuming H is a random oracle.*[11]

[11] To get a unifying syntax of NIZK, one can also interpret the description of H as common reference string.

A.8 Non-Interactive Key Exchange

In a NIKE scheme, ℓ parties each post a single message to a public bulletin board. All parties then read the board and any n-size subset users can agree on a shared key that is secret from any outside eavesdropper. The classic Diffie-Hellman key-exchange [DH76] solves the two-party case $n = 2$ based on the DDH assumption. Joux [Jou04] gives the first three-party NIKE protocol using bilinear maps. For the general case where n could be any positive integer, Boneh and Silverberg [BS02] create a scheme from multilinear maps. Boneh and Zhandry [BZ14] show a construction from using indistinguishability obfuscation. Very recently, Alamati et al. [AMPR19] put forward a black-box construction from composable input homomorphic weak PRF.

Cash et al. [CKS08] propose a security model for NIKE scheme in the public key setting, known as the CKS model. The CKS model allows an adversary to obtain honestly generated public keys, but also can register dishonestly generated public keys (for which the adversary need not know the corresponding secret keys). This dishonest key registration (DKR) setting captures realistic PKI where the Certificate Authority (CA) does not demand a proof of knowledge or possession of the secret key when issuing a certificate on a public key. Freire et al. [FHKP13] provide different security models derived from the CKS model and explore the relationships between them. They also consider the security models in the honest key registration (HKR) setting where dishonest key registration queries are disallowed.

We formally define the notion of multiparty NIKE by extending the syntax and the CKS-light security model of two-party NIKE [FHKP13] to the general multiparty case. The essential difference from the standard definition is that we eliminate all identities from the algorithms and allow different users to hold the same public key. A NIKE scheme consists of three polynomial-time algorithms as below.

- $\mathsf{Setup}(1^\lambda, n, \ell)$: on input a security parameter λ and two integers ℓ and n, outputs global public parameters pp. Here, n is the number of users that can derive a shared key, and ℓ is an upper bound on the number of users in the system. When there is no prior-fixed bound for ℓ, we can omit it from the inputs.

- $\mathsf{KeyGen}(pp)$: on input public parameters pp, outputs a keypair (pk, sk). User keeps sk as his secret, and publishes pk to the other users.

- $\mathsf{ShareKey}(sk_i, S)$: on input sk_i, a set S of n public keys (here the public keys are not required to be distinct), users holding $pk_i \in S$ derives the shared key k_S from his secret key and the set S.

Correctness. We require that user holding $pk \in S$ derives the same shared key, i.e., for any set S of n public keys and any $pk_i \in S$, we have:

$$\mathsf{ShareKey}(sk_i, S) = k_S$$

where sk_i is the secret key of pk_i.

Consistency. Note that the correctness requirement only defines the behavior of the $\mathsf{ShareKey}$ algorithm when all elements in S come from the public key space. Here, we introduce consistency to require that if there is only one element in S (say pk_i) does not belong to the public key space, the outputs of $\mathsf{ShareKey}(sk_j, S)$ are still the same for all $j \neq i$. Clearly, this notion is meaningful for $n \geq 3$. We remark that this is a very mild property. All the known n-party NIKE constructions [Jou04, BZ14, AMPR19] satisfy this property.

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

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda, \ell, n); \\ S \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}, \mathcal{O}_{\mathsf{regC}}, \mathcal{O}_{\mathsf{reveal}}}(pp); \\ k_0^* \leftarrow k_S, k_1^* \xleftarrow{\text{R}} K; \\ b \xleftarrow{\text{R}} \{0,1\}; \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regC}}, \mathcal{O}_{\mathsf{reveal}}}(k_b^*); \end{array} \right] - \frac{1}{2}.$$

Here, $\mathcal{O}_{\mathsf{regH}}$ is the honest user registration oracle, capturing that an adversary can observe public keys of honest users. $\mathcal{A}$ makes n such queries. Upon receiving an honest user registration query, the challenger runs KeyGen to generate a keypair (pk, sk), records the tuple (pk, sk) into an initially empty list L_{honest} and returns pk to $\mathcal{A}$. S represents the set of public keys that $\mathcal{A}$ had made for honest user registration queries, which $\mathcal{A}$ would like to be challenged on. $\mathcal{O}_{\mathsf{regC}}$ denotes corrupt user registration oracle, capturing that in real-world the Certificate Authority may not demand a proof of knowledge of the secret key or check if the public key had been registered when issuing a certificate. $\mathcal{A}$ can make polynomial number of such queries, each time with a distinct public key pk. We stress that $\mathcal{A}$ is even allowed to make corrupt user registration query with honest public keys. The challenger records the tuple $(pk, \perp)$ into an initially empty list L_{corrupt}. $\mathcal{O}_{\mathsf{reveal}}$ denotes the corrupt reveal oracle, capturing that the adversary may learn the shared keys of some particular sets of public keys. $\mathcal{A}$ can make polynomial number of such queries, each time with a set of n public keys as long as at least one of the public keys was registered as *corrupt* and the other as *honest*. The challenger runs $\mathsf{ShareKey}$ with a secret key corresponding to one honest public key and returns the result to $\mathcal{A}$. To prevent trivial win, the only constraint is that $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{reveal}}$ with S.

A NIKE scheme is secure in the CKS-light model under the DKR setting if no stateful PPT adversary has non-negligible advantage in the above experiment. A NIKE scheme is secure in the CKS-light model under the HKR setting if no stateful PPT adversary has non-negligible advantage in the same experiment but denying access to the $\mathcal{O}_{\mathsf{regC}}$ and $\mathcal{O}_{\mathsf{reveal}}$ oracles.

A.8.1 Joux's Three-Party NIKE

We recall Joux's three-party NIKE from bilinear maps [Jou04] as below. The original protocol inherently relies on symmetric pairing.

- $\mathsf{Setup}(1^\lambda, 3)$: runs $(\mathbb{G}, \mathbb{G}_T, p, g, e) \leftarrow \mathsf{BLGroupGen}(1^\lambda)$, picks a function H from $\mathbb{G}_T$ to the session key space K, outputs public parameters pp that includes the descriptions of bilinear groups and H.

- $\mathsf{KeyGen}(pp)$: picks $sk \xleftarrow{\text{R}} \mathbb{Z}_q$, computes $pk \leftarrow g^{sk}$, outputs (pk, sk).

- $\mathsf{ShareKey}(sk, S)$: on input sk, a set of public keys $S = \{pk_\alpha, pk_\beta, pk_\gamma\}$, if sk is the secret key of one public key in S, say, sk_γ for pk_γ, then outputs $k_S \leftarrow \mathsf{H}(e(pk_\alpha, pk_\beta)^{sk_\gamma})$, else outputs $\perp$.

Joux's three-party NIKE [Jou04] is secure in the CKS-light model under the HKR setting based on the decisional BDH assumption by setting H as identity function and $K = \mathbb{G}_T$, or based on the computational BDH assumption by setting H as a cryptographic hash function modeled as a random oracle.

Remark A.1. We note that Joux's three-party NIKE inherently relies on symmetric pairing. To adapt Joux's protocol with asymmetric pairing, there are two ready approaches: (i) set the public key as element in $\mathbb{G}_1 \times \mathbb{G}_2$, and derive the shared key by selecting appropriate part of the public key; (ii) set the public key as element in $\mathbb{G}_2$, and derive the shared key by mapping one of the public keys to $\mathbb{G}_1$. The shortcoming of the first approach is larger key size, while the shortcoming of the second approach is that one has to stick to Type-II pairing, whose efficient realizations are rare. Moreover, in either case we have to resort case-tailored ad-hoc assumptions to make the security reduction go through.

B Miscellaneous

B.1 Global Escrow PKE Scheme

For completeness, we describe our newly proposed global escrow PKE from asymmetric pairing, which is implied by a relaxed version of Joux's protocol, as sketched in Section 6.2.3.

- $\mathsf{Setup}(1^\lambda)$: runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathsf{BLGroupGen}(1^\lambda)$, picks $edk \xleftarrow{\text{R}} \mathbb{Z}_p$, computes $pk_\gamma^1 \leftarrow g_1^{edk} \in \mathbb{G}_1$, $pk_\gamma^2 \leftarrow g_2^{edk} \in \mathbb{G}_2$, outputs public parameters $pp = (pk_\gamma^1, pk_\gamma^2)$ and edk. The plaintext space is $M = \mathbb{G}$.

- $\mathsf{KeyGen}(pp)$: on input pp, picks $sk \xleftarrow{\text{R}} \mathbb{Z}_p$, computes $pk \leftarrow g_1^{sk} \in \mathbb{G}_1$.

- $\mathsf{Enc}(pk, m)$: on input $pk = pk_\beta$ and $m \in \mathbb{G}_T$, picks $sk_\alpha \xleftarrow{\text{R}} \mathbb{Z}_p$, computes $X \leftarrow g_2^{sk_\alpha} \in \mathbb{G}_2$, $k \leftarrow e(pk_\beta, pk_\gamma^2)^{sk_\alpha}$, $Y \leftarrow k \cdot m \in \mathbb{G}_T$, outputs $c = (X, Y)$.

- $\mathsf{Dec}(sk, c)$: on input $sk = sk_\beta$ and $c = (X, Y)$, outputs $m \leftarrow Y/e(pk_\gamma^1, X)^{sk_\beta}$.

- $\mathsf{Dec}'(edk, c)$: on input $edk = sk_\gamma$ and $c = (X, Y)$, outputs $m \leftarrow Y/e(pk_\beta, X)^{sk_\gamma}$.

The correctness is obvious. The IND-CPA security is based on the co-DBDH assumption.

B.2 Constrained IBE for Prefix Predicates from BTE

Observe that for any subtree T in a BTE [CHK03], the secret key of its root node serves as a succinct representation of all the secret keys of the nodes on T. This property is reminiscent of the GGM PRF, which implies constrained PRF for prefix predicates [BW13]. We are thus inspired to build constrained IBE for prefix predicates from BTE. The construction is as below.

- $\mathsf{Setup}(1^\lambda)$: runs $pp \leftarrow \text{BTE}.\mathsf{Setup}(1^\lambda, 1^n)$ to generate public parameters. Let $\mathcal{F} = \{f_\mathbf{v}\}_{\mathbf{v} \in \{0,1\}^\ell, \ell \leq n}$ be a family of predicates over identity space $I = \{0,1\}^n$, where $f_\mathbf{v}(id) = 1$ iff $\mathbf{v}$ is a prefix of id.

- $\mathsf{KeyGen}(pp)$: outputs $(mpk, msk) \leftarrow \text{BTE}.\mathsf{KeyGen}(pp)$.

- $\mathsf{Extract}(msk, id)$: outputs $sk_{id} \leftarrow \text{BTE}.\mathsf{Extract}(msk, id)$.

- $\mathsf{Constrain}(msk, f_\mathbf{v})$: runs BTE.$\mathsf{Delegate}$ iteratively to derive $sk_{f_\mathbf{v}}$.

- $\mathsf{Derive}(sk_{f_\mathbf{v}}, id)$: if $\mathbf{v}$ is a prefix of id, then runs BTE.$\mathsf{Delegate}$ iteratively to derive sk_{id}, else outputs $\perp$.

- $\mathsf{Enc}(mpk, id, m)$: outputs $c \leftarrow \text{BTE}.\mathsf{Enc}(mpk, id, m)$.

- $\mathsf{Dec}(sk_{id}, c)$: outputs $m \leftarrow \text{BTE}.\mathsf{Dec}(sk_{id}, c)$.

The correctness and security of the above construction follow straightforwardly from those of the underlying BTE.

B.3 More Eligible PKE Candidates for the Second HISE Construction

We provide two more candidates of PKE schemes to demonstrate the generality of our second generic HISE construction. One candidate is the Cramer-Shoup PKE from hash proof system [CS02]. The randomness space R of KeyGen is $\mathbb{Z}_p \times \mathbb{Z}_p$. The KeyGen algorithm on input randomness $r = (r_1, r_2) \in \mathbb{Z}_p \times \mathbb{Z}_p$, outputs $sk = (r_1, r_2)$ and $pk = g_1^{r_1} g_2^{r_2} \in \mathbb{G}$. Here, $\mathsf{G} : \mathbb{Z}_p \times \mathbb{Z}_p \rightarrow \mathbb{G}$ is defined as $(r_1, r_2) \mapsto g_1^{r_1} g_2^{r_2}$, which is collision-resistant based on the discrete logarithm assumption defined over (g_1, g_2), which is implied by the security of the Cramer-Shoup's PKE. Another candidate is the dual Regev's PKE proposed by Gentry, Peikert, and Vaikuntanathan [GPV08]. The KeyGen algorithm on input randomness $\mathbf{r} \in \{0,1\}^\ell$, outputs secret key $\mathbf{x} = \mathbf{r}$ and public key $\mathbf{u} = \mathbf{A}\mathbf{x}$. Here, $\mathsf{G} : \{0,1\}^\ell \rightarrow \mathbb{Z}_q^n$ is defined as $\mathbf{r} \mapsto \mathbf{A}\mathbf{r}$, which is collision-resistant based on SIS assumption defined by $\mathbf{A}$.