Two-Round PAKE from Approximate SPH and Instantiations from Lattices

Jiang Zhang[1] and Yu Yu[2,1,3]

[1] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[3] Westone Cryptologic Research Center, Beijing 100070, China
Email: jiangzhang09@gmail.com,yuyuathk@gmail.com

Abstract. Password-based authenticated key exchange (PAKE) enables two users with shared low-entropy passwords to establish cryptographically strong session keys over insecure networks. At Asiacrypt 2009, Katz and Vaikuntanathan showed a generic *three-round* PAKE based on any CCA-secure PKE with associated approximate smooth projective hashing (ASPH), which helps to obtain the first PAKE from lattices. In this paper, we give a framework for constructing PAKE from CCA-secure PKE with associated ASPH, which uses only *two-round* messages by carefully exploiting a *splittable* property of the underlying PKE and its associated *non-adaptive* ASPH. We also give a *splittable* PKE with associated *non-adaptive* ASPH based on the LWE assumption, which finally allows to instantiate our two-round PAKE framework from lattices.

1 Introduction

As one of the most fundamental and widely used cryptographic primitives, key exchange (KE) dates back to the seminal work of Diffie and Hellman (DH) [25], and it enables users to establish a session key via public exchanges of messages. Due to lack of authentication, the original DH protocol, and key exchange in general, only ensures two users to share a secure session key in presence of passive eavesdroppers, and it is insecure against an active adversary who has full control of all communication. To overcome this issue, authenticated key exchange (AKE) enables each user to authenticate the identities of others with the help of some pre-shared information, and thus provides the additional guarantee that only the intended users can access the session key. Typically, the shared information can be either a high-entropy cryptographic key (such as a secret key for symmetric-key encryption, or a public key for digital signature) or a low-entropy password. After decades of development, the community has witnessed great success in designing AKE based on high-entropy cryptographic keys, even in the setting of lattices [52,60,7]. However, people rarely make full use of the large character set in forming passwords and many tend to pick easily memorizable ones from a relatively small dictionary. AKEs based on high-entropy cryptographic keys usually do not apply to the case where only low-entropy passwords are available.

Indeed, as shown in [34,39], it can be trivially insecure to use a low-entropy password as a cryptographic key.

Informally, a secure password-based AKE (PAKE) should resist *off-line dictionary attacks* in which the adversary tries to determine the correct password using only information obtained during previous protocol executions, and limit the adversary to the trivial *on-line attacks* where the adversaries simply run the protocol with honest users using (a bounded number of) password trials. Formal security models for PAKE were developed in [10,17]. Later, many provably secure PAKE protocols based on various hardness assumptions were proposed, where the research mainly falls into two lines:[4] the first line starts from the work of Bellovin and Merritt [12], followed by plenty of excellent work focusing on PAKE in the random oracle/ideal cipher models and aiming at achieving the highest possible levels of performance [10,17,47,18]; the second line dates back to the work of Katz, Ostrovsky and Yung [39], from which Gennaro and Lindell [31] abstracted out a generic PAKE framework (in the CRS model) based on smooth projective hash (SPH) functions [23]. This line of research devoted to seeking more efficient PAKE in the standard model [22,3,37,41,13,2].

As noted in [40], none of the above PAKEs can be instantiated from lattices. In particular, it is an open problem [54] to instantiate SPH functions [23] on lattice assumptions. Despite the great success in lattice-based cryptography in the past decade, little progress was made on lattice-based PAKE until the work of Katz and Vaikuntanathan [40]. Concretely, they [40] introduced the notion of approximate smooth projective hashing (ASPH) so as to be instantiatable from lattices, and plugged it into an adapted version of the GL-framework [31] to yield the first lattice-based PAKE by using only three-round messages in the standard model (just like the counterparts in [39,31]). Up until now (seven years after the publication of [40]), the Katz-Vaikuntanathan PAKE remained the most efficient lattice-based PAKE to the best of our knowledge. This raises the following questions: *is it possible to construct more efficient PAKE from lattices (e.g., a PAKE with less message rounds/communication overheads), and does there exist other generic PAKE framework that fits better with lattices?*

1.1 Our Contribution

In this paper, we first give a new PAKE framework (also in the CRS model) from PKE with associated ASPH, which uses only *two-round* messages. We mainly benefit from two useful features of the underlying primitives: 1) the PKE is *splittable*, which informally requires that each ciphertext of the PKE scheme consists of two relatively independent parts, where the first part is designed for realizing the "functionality" of encryption, while the second part helps to achieve CCA-security; and 2) the ASPH is *non-adaptive* [41], i.e., the projection function only depends on the hash key, and the smoothness property holds even when the ciphertext depends on the projection key. By carefully exploiting the above

[4] Please refer to Section 1.3 for other related works.

features, we overcome several obstacles (e.g., the "approximate correctness" of ASPH) to obtain a generic two-round PAKE in the standard model.

We also propose a concrete construction of splittable PKE with associated non-adaptive ASPH from learning with errors (LWE). Note that the PKEs with associated SPH (based on either DDH or decisional linear assumptions) in [41] can be used to instantiate our framework, but the only known lattice-based PKE with associated ASPH in [40] does not satisfy our requirements. We achieve our goal by first developing an adaptive smoothing lemma for q-ary lattices, and then combining it with several recent techniques. Technically, the lemma is needed for achieving the strong smoothness of our *non-adaptive* ASPH, and may be of independent interest. As in [41], our PKE construction relies on simulation-sound non-interactive zero-knowledge (SS-NIZK) proofs, and thus, in general, is computationally inefficient. Fortunately, we can construct an efficient SS-NIZK from lattices in the random oracle model,[5] and finally obtain an efficient two-round lattice-based PAKE in the random oracle model, which is at least $O(\log n)$ times more efficient in the communication overhead than the three-round lattice-based PAKE (in the standard model) [40].

1.2 Our Techniques

We begin with the GL-framework [31] from CCA-secure public-key encryption (PKE) with associated smooth projective hash (SPH) functions. Informally, the SPH for a PKE scheme is a keyed hash function which maps a ciphertext-plaintext pair into a hash value, and can be computed in two ways: either using the hash key hk or using a projection key hp (which can be efficiently determined from hk and a targeted ciphertext c). The GL-framework for PAKE roughly relies on the following two properties of SPH:

Correctness: if c is an encryption of the password pw using randomness r, then the hash value $\mathrm{H}_{\mathsf{hk}}(c, pw) = \mathsf{Hash}(\mathsf{hp}, (c, pw), r)$, where both functions H and Hash can be efficiently computed from the respective inputs.

Smoothness: if c is not an encryption of pw, the value $\mathrm{H}_{\mathsf{hk}}(c, pw)$ is statistically close to uniform given hp, c and pw (over the random choice of hk).

Specifically, the GL-framework for PAKE has three-round messages: 1) the client computes an encryption c_1 of the password pw using randomness r_1, and sends c_1 to the server; 2) the server randomly chooses a hash key hk_2, computes a projection key hp_2 (from hk_2 and c_1) together with an encryption c_2 of the password pw using randomness r_2, and sends (hp_2, c_2) to the client; 3) the client sends a projection key hp_1 corresponding to a randomly chosen hash key hk_1 and c_2. After exchanging the above three messages, both users can compute the same session key $sk = \mathrm{H}_{\mathsf{hk}_1}(c_2, pw) \oplus \mathsf{Hash}(\mathsf{hp}_2, (c_1, pw), r_1) = \mathsf{Hash}(\mathsf{hp}_1, (c_2, pw), r_2) \oplus \mathrm{H}_{\mathsf{hk}_2}(c_1, pw)$ by the correctness of the SPH. Note that if the PKE scheme is CCA-secure, no user can obtain useful information about the password held by the

[5] We leave it as an open problem to directly construct an SS-NIZK from lattice problems in the standard model.

other user from the received ciphertext. Thus, if the client (resp., the server) does not hold the correct password pw, his view is independent from the "session key" computed by the server (resp., the client) by the smoothness of the SPH. We stress that the above discussion is very informal and omits many details. For example, a verification key vk should be sent in the first message such that the client can generate a signature σ on the protocol transcripts in the third message (and thus the total communication cost is determined by $|\mathsf{hp}| + |c| + |vk| + |\sigma|$).

Clearly, a lattice-based PAKE is immediate if a PKE with associated SPH could be obtained from lattice assumptions. However, the literature [54] suggests that it is highly non-trivial, if not impossible, to instantiate SPH from lattices. Instead, Katz and Vaikuntanathan [40] provided a solution from a weaker notion of SPH—Approximate SPH (ASPH), which weakens both the correctness and smoothness properties of the SPH notion in [31]. First, ASPH only provides "approximate correctness" in the sense that $\mathrm{H}_{\mathsf{hk}}(c, pw)$ and $\mathsf{Hash}(\mathsf{hp}, (c, pw), r)$ may differ at a few positions when parsed as bit-strings. Second, the smoothness property of ASPH only holds for some (c, pw) that pw is not equal to the decryption of c, and hence leaves a gap that there exists (c, pw) for which ASPH provides neither correctness nor smoothness guarantee. This relaxation is necessary for instantiating ASPH on lattices, since in the lattice setting there is no clear boundary between "c is an encryption of pw" and "c is not an encryption of pw", which is actually one of the main difficulties for realizing SPH from lattices.

Thus, if one directly plugs ASPH into the GL-framework [31], neither the correctness nor the security of the resulting PAKE is guaranteed. Because both users may not compute the same session key, and the adversary may break the protocol by exploiting the (inherent) gap introduced by ASPH. The authors [40] fixed the issues by relying on error correcting codes (ECC) and the robustness of the GL-framework [31]. Specifically, in addition to sending a projection key hp_1, the client also randomly chooses a session key sk, computes $tk = \mathrm{H}_{\mathsf{hk}_1}(c_2, pw) \oplus \mathsf{Hash}(\mathsf{hp}_2, (c_1, pw), r_1)$, and appends $\Delta = tk \oplus \mathsf{ECC}(sk)$ to the third message (i.e., tk is used as a masking key to deliver sk to the server), where ECC and ECC^{-1} are the corresponding encoding and decoding algorithms. After receiving the third message, the server can compute the session key $sk' = \mathsf{ECC}^{-1}(tk' \oplus \Delta)$, where $tk' = \mathsf{Hash}(\mathsf{hp}_1, (c_2, pw), r_2) \oplus \mathrm{H}_{\mathsf{hk}_2}(c_1, pw)$. By the "approximate correctness" of the ASPH, we know that $tk' \oplus \Delta$ is not far from the codeword $\mathsf{ECC}(sk)$. Thus, both users can obtain the same session key $sk = sk'$ by the correctness of an appropriately chosen ECC, which finally allows [40] to obtain a three-round PAKE from PKE with associated ASPH.

However, the techniques of [40] are not enough to obtain a two-round PAKE (in particular, they cannot be applied into the PAKE framework [41]) due to the following two main reasons. First, the ASPH in [40] is *adaptive* (i.e., the projection key hp depends on the ciphertext c, and the smoothness only holds when c is independent of hp), which seems to inherently require at least three-round messages [31,41]. Second, the strategy of delivering a random session key to deal with the "approximate correctness" of ASPH can only be applied when one user (e.g., the client) obtained the masking key tk, and may be vulnerable to active attacks (e.g., modifications) because of the loose relation between the marking part (namely, Δ) and other protocol messages. This is not a problem for the GL-framework [31], since it had three-round messages and used one-time signatures, which allows the authors of [40] to simply send Δ in the third message and tightly bind it with other protocol messages by incorporating it into the one-time signature. Nevertheless, the above useful features are not available in the more efficient PAKE framework [41].

In order to get a two-round PAKE from PKE with associated ASPH, we strengthen the underlying primitive with several reasonable properties. First, we require that the ASPH is non-adaptive, i.e., the projection function only depends on the hash key, and the smoothness property holds even when the ciphertext c depends on hp. Second, we require that the underlying PKE is splittable. Informally, this property says that a ciphertext $c = (u, v)$ of the PKE scheme can be "independently" computed by two functions (f, g), where $u = f(pk, pw, \cdots)$ mainly takes a plaintext pw as input and plays the role of "encrypting" pw, while $v = g(pk, \mathsf{label}, \cdots)$ mainly takes a label as input and plays the role of providing non-malleability for CCA-security.[6] Third, we require that the hash value of the ASPH is determined by the hash key hk, the first part u of the ciphertext $c = (u, v)$, as well as the password pw. At a high level, the first enhancement allows us to safely compute the masking key tk after receiving the first message, while the second and third enhancements enable us to leverage the non-malleability of the underlying CCA-secure PKE scheme to tightly bind the masking part Δ with other protocol messages. Concretely, we let the client to send the projection hash key hp_1 together with the ciphertext c_1 in a single message, and let the server compute the masking key tk immediately after it has obtained the first part $u_2 = f(pk, pw, \cdots)$ of the ciphertext $c_2 = (u_2, v_2)$, and compute the second part $v_2 = g(pk, \mathsf{label}, \cdots)$ with a label consisting of $\mathsf{hp}_1, c_1, \mathsf{hp}_2, u_2$ and $\Delta = tk \oplus sk$ for some randomly chosen session key sk. The protocol ends with a message $(\mathsf{hp}_2, c_2, \Delta)$ sent by the server to the client. A high level overview of our two-round PAKE framework is given in Fig. 1.

Note that the PKEs with associated SPH in [41] can be used to instantiate our two-round PAKE framework, but the only known lattice-based PKE with associated ASPH [40] does not satisfy our requirements. Actually, it is highly non-trivial to realize non-adaptive ASPH from lattices. One of the main reason is that the smoothness should hold even when the ciphertext c is adversarially chosen and dependent on the projection key hp (and thus is stronger than that in [40]), which gives the adversary an ability to obtain non-trivial information about the secret hash key hk, and makes the above (inherent) gap introduced by the ASPH notion more problematic. In order to ensure the stronger smoothness property, we first develop an adaptive smoothing lemma for q-ary lattices, which may be of independent interest. Then, we combine it with several other techniques [51,57,40,32,49] to achieve our goal. As in [41], our PKE is computationally inefficient due to the use of simulation-sound non-interactive zero-knowledge (SS-NIZK) proofs. However, we can obtain an efficient SS-NIZK

[6] Similar properties were also considered for identity-based encryptions [61,4].

from lattices in the random oracle model, and finally get an efficient lattice-based PAKE. Despite the less message rounds, our PAKE (in the random oracle model) is also at least $O(\log n)$ times more efficient in the communication overhead than the one in [40], because they used the correlated products technique [56] and signatures. Specifically, the communication cost of [40] is determined by $|vk| + |c| + |\mathsf{hp}|$, where vk is the verification key of signatures (which usually consists of matrices on lattices [59]), c is the ciphertext of the underlying PKE scheme and hp is the projective hashing key. Since [40] used the correlated products technique [56] (which introduces an expansion factor n w.r.t. the basic CPA-secure PKE scheme) to achieve CCA-secure PKE, their communication cost is dominated by $|c|$ (which is at least $O(\log n)$ times larger than $|\mathsf{hp}|$ when setting $k = O(n)$ or $\ell = O(n)$ in our notation). Since our framework does not use signatures, the communication cost is mainly determined by $|c| + |\mathsf{hp}|$. Although the use of Stern-like ZK introduces an $\omega(\log n)$ expansion factor, the ciphertext c of our PKE scheme is still $n/\omega(\log n)$ times shorter than that of [40]. Thus, the communication cost of our PAKE is now dominated by $|\mathsf{hp}|$, which is asymptotically the same as that in [40]. This is why we can (only) save a factor of $O(\log n)$ in the total communication cost. Note that one can also use our PKE with ASPH to instantiate the three-round PAKE framework in [40] with improved efficiency, but currently there seems no other way to do it significantly better even in the random oracle model.

1.3 Other Related Work and Discussions

Gong et al. [35] first considered the problem of resisting off-line attacks in the "PKI model" where the server also has a public key in addition to a password. A formal treatment on this model was provided by Halevi and Krawczyk [38]. At CRYPTO 1993, Bellovin and Merritt [12] considered the setting where only a password is shared between users, and proposed a PAKE with heuristic security arguments. Formal security models for PAKE were provided in [10,17]. Goldreich and Lindell [34] showed a PAKE solution in the plain model, which does not support concurrent executions of the protocol by the same user. As a special case of secure multiparty computations, PAKEs supporting concurrent executions in the plain model were studied in [9,36,20]. All the protocols in [34,9,36,20] are inefficient in terms of both computation and communication. In the setting where all users share a common reference string, Katz et al. [39] provided a practical three-round PAKE based on the DDH assumption, which was later generalized and abstracted out by Gennaro and Lindell [31] to obtain a PAKE framework from PKE with associated SPH [23]. Canetti et al. [22] considered the security of PAKE within the framework of universal composability (UC) [19], and showed that an extension of the KOY/GL protocol was secure in the UC model.

Relations to [40,41]. The works [40,41] due to Katz and Vaikuntanathan are most related to our work. First, the ASPH notion in this paper is stronger than that in [40]. In particular, the PKE with associated ASPH in [40] cannot be used to instantiate our framework. Our PAKE framework with less message rounds is obtained by strengthening the underlying primitives with several useful and achievable features, which provide us a better way to handle lattice assumptions. Besides, our PKE with associated SPH can be used to instantiate the PAKE framework in [40] (with improved efficiency). Second, our ASPH notion is much weaker than the SPH in [41], which means that our PKE with associated ASPH cannot be used to instantiate the PAKE framework in [41]. In fact, it is still an open problem to construct PKE with associated SPH from lattices, and we still do not know how to instantiate the efficient one-round PAKE framework [41] with lattice assumptions (recall that our PAKE has two-round messages). Third, our PAKE framework is inspired by [40,41], and thus shares some similarities to the latter. However, as discussed above, there are technical differences among the underlying primitives used in the three papers, and several new ideas/techniques are needed to obtain a two-round PAKE from lattices.

1.4 Roadmap

After some preliminaries in Section 2, we propose a generic two-round PAKE from splittable PKE with associated ASPH in Section 3. In Section 4, we give some backgrounds together with a new technical lemma on lattices. We construct a concrete splittable PKE with associated ASPH from lattices in Section 5.

2 Preliminaries

2.1 Notation

Let κ be the natural security parameter. By $\log_2$ (resp. log) we denote the logarithm with base 2 (resp. the natural logarithm). A function $f(n)$ is negligible, denoted by $\mathrm{negl}(n)$, if for every positive c, we have $f(n) < n^{-c}$ for all sufficiently large n. A probability is said to be overwhelming if it is $1-\mathrm{negl}(n)$. The notation $\leftarrow_r$ denotes randomly choosing elements from some distribution (or the uniform distribution over some finite set). If a random variable x follows some distribution D, we denote it by $x \backsim D$. For any strings $x, y \in \{0,1\}^\ell$, denote $\mathsf{Ham}(x, y)$ as the hamming distance of x and y.

By $\mathbb{R}$ (resp. $\mathbb{Z}$) we denote the set of real numbers (resp. integers). Vectors are used in the column form and denoted by bold lower-case letters (e.g., $\mathbf{x}$). Matrices are treated as the sets of column vectors and denoted by bold capital letters (e.g., $\mathbf{X}$). The concatenation of the columns of $\mathbf{X} \in \mathbb{R}^{n\times m}$ followed by the columns of $\mathbf{Y} \in \mathbb{R}^{n\times m'}$ is denoted as $(\mathbf{X}\|\mathbf{Y}) \in \mathbb{R}^{n\times(m+m')}$. By $\|\cdot\|$ and $\|\cdot\|_\infty$ we denote the l_2 and l_∞ norm, respectively. The largest singular value of a matrix $\mathbf{X}$ is $s_1(\mathbf{X}) = \max_{\mathbf{u}} \|\mathbf{X}\mathbf{u}\|$, where the maximum is taken over all unit vectors $\mathbf{u}$.

2.2 Security Model for PAKE

We recall the security model for password-based authenticated key exchange (PAKE) in [10,39,41]. Formally, the protocol relies on a setup assumption that a common reference string (CRS) and other public parameters are established (possibly by a trusted third party) before any execution of the protocol. Let $\mathcal{U}$ be the set of protocol users. For every distinct $A, B \in \mathcal{U}$, users A and B share a password $pw_{A,B}$. We assume that each $pw_{A,B}$ is chosen independently and uniformly from some dictionary set $\mathcal{D}$ for simplicity. Each user $A \in \mathcal{U}$ is allowed to execute the protocol multiple times with different partners, which is modeled by allowing A to have an unlimited number of *instances* with which to execute the protocol. Denote instance i of A as Π_A^i. An instance is for one-time use only and it is associated with the following variables that are initialized to $\perp$ or 0:

- $\mathsf{sid}_A^i, \mathsf{pid}_A^i$ and sk_A^i denote the *session id, parter id*, and *session key* for instance Π_A^i. The session *id* consists of the (ordered) concatenation of all messages sent and received by Π_A^i; while the partner *id* specifies the user with whom Π_A^i believes it is interacting;
- acc_A^i and term_A^i are boolean variables denoting whether instance Π_A^i has accepted or terminated, respectively.

For any user $A, B \in \mathcal{U}$, instances Π_A^i and Π_B^j are *partnered* if $\mathsf{sid}_A^i = \mathsf{sid}_B^j \neq \perp$, $\mathsf{pid}_A^i = B$ and $\mathsf{pid}_B^j = A$. We say that a PAKE protocol is *correct* if instances Π_A^i and Π_B^j are partnered, then we have that $\mathrm{acc}_A^i = \mathrm{acc}_B^j = 1$ and $\mathsf{sk}_A^i = \mathsf{sk}_B^j \neq \perp$ hold (with overwhelming probability).

Adversarial abilities. The adversary $\mathcal{A}$ is a probabilistic polynomial time (PPT) algorithm with full control over all communication channels between users. In particular, $\mathcal{A}$ can intercept all messages, read them all, and remove or modify any desired messages as well as inject its own messages. $\mathcal{A}$ is also allowed to obtain the session key of an instance, which models possible leakage of session keys. These abilities are formalized by allowing the adversary to interact with the various instances via access to the following oracles:

- $\mathsf{Send}(A, i, \mathsf{msg})$: This sends message msg to instance Π_A^i. After receiving msg, instance Π_A^i runs according to the protocol specification, and updates its states as appropriate. Finally, this oracle returns the message output by Π_A^i to the adversary. We stress that the adversary can prompt an unused instance Π_A^i to execute the protocol with partner B by querying $\mathsf{Send}(A, i, B)$, and obtain the first protocol message output by Π_A^i.
- $\mathsf{Execute}(A, i, B, j)$: If both instances Π_A^i and Π_B^j have not yet been used, this oracle executes the protocol between Π_A^i and Π_B^j, updates their states as appropriate, and returns the transcript of this execution to the adversary.
- $\mathsf{Reveal}(A, i)$: This oracle returns the session key sk_A^i to the adversary if it has been generated (i.e., $\mathsf{sk}_A^i \neq \perp$).
- $\mathsf{Test}(A, i)$: This oracle chooses a random bit $b \leftarrow_r \{0, 1\}$. If $b = 0$, it returns a key chosen uniformly at random; if $b = 1$, it returns the session key sk_A^i of instance Π_A^i. The adversary is only allowed to query this oracle once.

Definition 1 (Freshness). *We say that an instance Π_A^i is fresh if the following conditions hold:*

– *the adversary $\mathcal{A}$ did not make a* $\mathsf{Reveal}(A,i)$ *query to instance* Π_A^i*;*
– *the adversary $\mathcal{A}$ did not make a* $\mathsf{Reveal}(B,j)$ *query to instance* Π_B^j*, where instances* Π_A^i *and* Π_B^j *are partnered;*

Security Game. The security of a PAKE protocol is defined via the following game. The adversary $\mathcal{A}$ makes any sequence of queries to the oracles above, so long as only one $\mathsf{Test}(A,i)$ query is made to a fresh instance Π_A^i, with $\mathrm{acc}_A^i = 1$ at the time of this query. The game ends when $\mathcal{A}$ outputs a guess b' for b. We say $\mathcal{A}$ wins the game if its guess is correct, so that $b' = b$. The advantage $\mathbf{Adv}_{\Pi,\mathcal{A}}$ of adversary $\mathcal{A}$ in attacking a PAKE protocol Π is defined as $|2 \cdot \Pr[b' = b] - 1|$.

We say that an *on-line attack* happens when the adversary makes one of the following queries to some instance Π_A^i: $\mathsf{Send}(A,i,*)$, $\mathsf{Reveal}(A,i)$ or $\mathsf{Test}(A,i)$. In particular, the $\mathsf{Execute}$ queries are not counted as on-line attacks. Since the size of the password dictionary is small, a PPT adversary can always win by trying all password one-by-one in an on-line attack. The number $Q(\kappa)$ of on-line attacks represents a bound on the number of passwords the adversary could have tested in an on-line fashion. Informally, a PAKE protocol is secure if online password guessing attacks are already the best strategy (for all PPT adversaries).

Definition 2 (Security). *We say that a PAKE protocol Π is secure if for all dictionary $\mathcal{D}$ and for all PPT adversaries $\mathcal{A}$ making at most $Q(\kappa)$ on-line attacks, it holds that* $\mathbf{Adv}_{\Pi,\mathcal{A}}(\kappa) \leq Q(\kappa)/|\mathcal{D}| + \mathrm{negl}(\kappa)$.

3 PAKE from Splittable PKE with Associated ASPH

In this section, we give a new PAKE framework which only has two-round messages. We begin with the definition of splittable PKE with associated ASPH.

3.1 Public-Key Encryption

A (labeled) public-key encryption (PKE) with plaintext-space $\mathcal{P}$ consists of three PPT algorithms $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm KeyGen takes the security parameter κ as input, outputs a public key pk and a secret key sk, denoted as $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. The encryption algorithm Enc takes pk, a string $\mathsf{label} \in \{0,1\}^*$, and a plaintext $pw \in \mathcal{P}$ as inputs,[7] with an internal coin flipping r, outputs a ciphertext c, which is denoted as $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{label}, pw; r)$, or $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{label}, pw)$ in brief. The deterministic algorithm Dec takes sk and c as inputs, and produces as output a plaintext pw or $\perp$, which is denoted as $pw \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{label}, c)$.

For correctness, we require that for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, any $\mathsf{label} \in \{0,1\}^*$, any plaintext pw and $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{label}, pw)$, the equation $\mathsf{Dec}(\mathsf{sk}, \mathsf{label}, c) = pw$ holds with overwhelming probability. For security, consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

[7] The notation 'pw' stands for password, and we keep several other commonly used notations such as 'm' and 'w' for latter use on lattices.

Setup. The challenger $\mathcal{C}$ first computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. Then, it gives the public key pk to $\mathcal{A}$, and keeps the secret key sk to itself.

Phase 1. The adversary $\mathcal{A}$ can make a number of decryption queries on any pair (label, c), and $\mathcal{C}$ returns $pw \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{label}, c)$ to $\mathcal{A}$ accordingly.

Challenge. At some time, $\mathcal{A}$ outputs two equal-length plaintexts $pw_0, pw_1 \in \mathcal{P}$ and a $\mathsf{label}^* \in \{0,1\}^*$. The challenger $\mathcal{C}$ chooses a random bit $b^* \leftarrow_r \{0,1\}$, and returns the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{label}^*, pw_{b^*})$ to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ can make more decryption queries on any $(\mathsf{label}, c) \neq (\mathsf{label}^*, c^*)$, the challenger $\mathcal{C}$ responds as in Phase 1.

Guess. Finally, $\mathcal{A}$ outputs a guess $b \in \{0,1\}$.

The adversary $\mathcal{A}$ wins the game if $b = b^*$. The advantage of $\mathcal{A}$ in the above game is defined as $\mathrm{Adv}^{\text{ind-cca}}_{\mathcal{PKE},\mathcal{A}}(1^\kappa) \stackrel{\text{def}}{=} |\Pr[b = b^*] - \frac{1}{2}|$.

Definition 3 (IND-CCA). *We say that a PKE scheme $\mathcal{PKE}$ is* CCA*-secure if for any PPT adversary $\mathcal{A}$, its advantage $\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PKE},\mathcal{A}}(1^\kappa)$ is negligible in κ.*

Informally, the splittable property of a PKE scheme $\mathcal{PKE}$ requires that the encryption algorithm can be split into two functions.

Definition 4 (Splittable PKE). *A labeled CCA-secure* PKE *scheme $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is splittable if there exists a pair of two efficiently computable functions (f, g) such that the followings hold:*

1. *for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, string $\mathsf{label} \in \{0,1\}^*$, plaintext $pw \in \mathcal{P}$ and randomness $r \in \{0,1\}^*$, we have $c = (u, v) = \mathsf{Enc}(\mathsf{pk}, \mathsf{label}, pw; r)$, where $u = f(\mathsf{pk}, pw, r)$ and $v = g(\mathsf{pk}, \mathsf{label}, pw, r)$. Moreover, the first part u of the ciphertext $c = (u, v)$ fixes the plaintext pw in the sense that for any v' and $\mathsf{label}' \in \{0,1\}^*$, the probability that $\mathsf{Dec}(\mathsf{sk}, \mathsf{label}', (u, v')) \notin \{\perp, pw\}$ is negligible in κ over the random choices of sk and r;*
2. *the security of $\mathcal{PKE}$ still holds in a CCA game with modified challenge phase: the adversary $\mathcal{A}$ first submits two equal-length plaintexts $pw_0, pw_1 \in \mathcal{P}$. Then, the challenger $\mathcal{C}$ chooses a random bit $b^* \leftarrow_r \{0,1\}$, randomness $r^* \leftarrow_r \{0,1\}^*$, and returns $u^* = f(\mathsf{pk}, pw_{b^*}, r^*)$ to $\mathcal{A}$. Upon receiving u^*, $\mathcal{A}$ outputs a string $\mathsf{label} \in \{0,1\}^*$. Finally, $\mathcal{C}$ computes $v^* = g(\mathsf{pk}, \mathsf{label}, pw_{b^*}, r^*)$, and returns the challenge ciphertext $c^* = (u^*, v^*)$ to $\mathcal{A}$;*

Definition 4 captures the "splittable" property in both the functionality and the security of the PKE scheme. In particular, the modified CCA game allows the adversary to see the first part u^* of c^* and then adaptively determine label to form the complete challenge ciphertext $c^* = (u^*, v^*)$. We note that similar properties had been used in the context of identity-based encryption (IBE) [61,4], where one part of the ciphertext is defined as a function of the plaintext, and the other part is a function of the user identity. By applying generic transformations such as the CHK technique [21] from IBE (with certain property) to PKE, it is promising to get a splittable PKE such that the g function simply outputs a tag or a signature which can be used to publicly verify the validity of the whole ciphertext. Finally, we stress that the notion of splittable PKE is not our main goal, but rather a crucial intermediate step to reaching two-round PAKE.

3.2 Approximate Smooth Projective Hash Functions

Smooth projective hash (SPH) functions were first introduced by Cramer and Shoup [23] for achieving CCA-secure PKEs. Later, several works [31,41] extended the notion for PAKE. Here, we tailor the definition of approximate SPH (ASPH) in [40] to our application. Formally, let $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a splittable PKE scheme with respect to functions (f, g), and let $\mathcal{P}$ be an efficiently recognizable plaintext space of $\mathcal{PKE}$. As in [40], we require that $\mathcal{PKE}$ defines a notion of ciphertext validity in the sense that the validity of a label-ciphertext pair (label, c) with respect to any public key pk can be efficiently determined using pk alone, and all honestly generated ciphertexts are valid. We also assume that given a valid ciphertext c, one can easily parse $c = (u, v)$ as the outputs of (f, g). Now, fix a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, and let C_{pk} denote the set of valid label-ciphertexts with respect to pk. Define sets X, L and $\bar{L}$ as follows:

$$
\begin{aligned}
X &= \{(\mathsf{label}, c, pw) \mid (\mathsf{label}, c) \in C_{\mathsf{pk}}; pw \in \mathcal{P}\} \\
L &= \{(\mathsf{label}, c, pw) \in X \mid \mathsf{label} \in \{0,1\}^*; c = \mathsf{Enc}(\mathsf{pk}, \mathsf{label}, pw)\} \\
\bar{L} &= \{(\mathsf{label}, c, pw) \in X \mid \mathsf{label} \in \{0,1\}^*; pw = \mathsf{Dec}(\mathsf{sk}, \mathsf{label}, c)\}
\end{aligned}
$$

By the definitions, for any ciphertext c and $\mathsf{label} \in \{0,1\}^*$, there is at most a single plaintext $pw \in \mathcal{P}$ such that $(\mathsf{label}, c, pw) \in \bar{L}$.

Definition 5 (ϵ-approximate SPH). *An ϵ-approximate SPH function is defined by a sampling algorithm that, given a public key* pk *of $\mathcal{PKE}$, outputs* $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk}\in K}, S, \mathsf{Proj} : K \to S)$ *such that*

- *There are efficient algorithms for (1) sampling a hash key* $\mathsf{hk} \leftarrow_r K$*, (2) computing* $\mathrm{H}_{\mathsf{hk}}(x) = \mathrm{H}_{\mathsf{hk}}(u, pw)$ *for all* $\mathsf{hk} \in K$ *and* $x = (\mathsf{label}, (u, v), pw) \in X$*,*[8] *and (3) computing* $\mathsf{hp} = \mathsf{Proj}(\mathsf{hk})$ *for all* $\mathsf{hk} \in K$*.*
- *For all* $x = (\mathsf{label}, (u, v), pw) \in L$ *and randomness r such that* $u = f(\mathsf{pk}, pw, r)$ *and* $v = g(\mathsf{pk}, \mathsf{label}, pw, r)$*, there exists an efficient algorithm computing the value* $\mathsf{Hash}(\mathsf{hp}, x, r) = \mathsf{Hash}(\mathsf{hp}, (u, pw), r)$*, and satisfies* $\Pr[\mathsf{Ham}(\mathrm{H}_{\mathsf{hk}}(u, pw), \mathsf{Hash}(\mathsf{hp}, (u, pw), r)) \geq \epsilon \cdot \ell] = \mathrm{negl}(\kappa)$ *over the choice of* $\mathsf{hk} \leftarrow_r K$*.*
- *For any (even unbounded) function* $h : S \to X \backslash \bar{L}$, $\mathsf{hk} \leftarrow_r K$, $\mathsf{hp} = \mathsf{Proj}(\mathsf{hk})$, $x = h(\mathsf{hp})$ *and* $\rho \leftarrow_r \{0,1\}^\ell$*, the statistical distance between* $(\mathsf{hp}, \mathrm{H}_{\mathsf{hk}}(x))$ *and* (hp, ρ) *is negligible in the security parameter κ.*

Compared to the ASPH notion in [40], our ASPH notion in Definition 5 mainly has three modifications: 1) the projection function only depends on the hash key; 2) the value $\mathrm{H}_{\mathsf{hk}}(x) = \mathrm{H}_{\mathsf{hk}}(u, pw)$ is determined by the hash key hk, the first part u of the ciphertext $c = (u, v)$, as well as the plaintext pw (i.e., it is independent from the pair (label, v)); and 3) the smoothness property holds even for adaptive choice of $x = h(\mathsf{hp}) \notin \bar{L}$. Looking ahead, the first modification allows us to achieve PAKE with two-round messages, whereas the last two are needed

[8] For all $x = (\mathsf{label}, (u, v), pw) \in X$, we slightly abuse the notation $\mathrm{H}_{\mathsf{hk}}(x) = \mathrm{H}_{\mathsf{hk}}(u, pw)$ by omitting (label, v) from its inputs. Similarly, the notation $\mathsf{Hash}(\mathsf{hp}, x, r) = \mathsf{Hash}(\mathsf{hp}, (u, pw), r)$ will be used later.

Common reference string: pk		
Client A (pw)		Server B (pw)
$r_1 \leftarrow_r \{0,1\}^*$	$\xrightarrow{A, \mathsf{hp}_1, c_1 = (u_1, v_1)}$	$r_2 \leftarrow_r \{0,1\}^*$
$\mathsf{hk}_1 \leftarrow_r K$		$\mathsf{hk}_2 \leftarrow_r K$
$\mathsf{hp}_1 = \mathsf{Proj}(\mathsf{hk}_1)$		$sk \leftarrow_r \{0,1\}^\kappa$
$\mathsf{label}_1 := A\|B\|\mathsf{hp}_1$		$\mathsf{hp}_2 = \mathsf{Proj}(\mathsf{hk}_2)$
$u_1 = f(\mathsf{pk}, pw, r_1)$		$u_2 = f(\mathsf{pk}, pw, r_2)$
$v_1 = g(\mathsf{pk}, \mathsf{label}_1, pw, r_1)$	$\xleftarrow{\mathsf{hp}_2, c_2 = (u_2, v_2), \Delta}$	$tk = \mathsf{Hash}(\mathsf{hp}_1, (u_2, pw), r_2)$
		$\oplus \mathrm{H}_{\mathsf{hk}_2}(u_1, pw)$
$tk' = \mathrm{H}_{\mathsf{hk}_1}(u_2, pw)$		$\Delta = tk \oplus \mathsf{ECC}(sk)$
$\oplus \mathsf{Hash}(\mathsf{hp}_2, (u_1, pw), r_1)$		$\mathsf{label}_2 := A\|B\|\mathsf{hp}_1\|c_1\|\mathsf{hp}_2\|\Delta$
$sk = \mathsf{ECC}^{-1}(tk' \oplus \Delta)$		$v_2 = g(\mathsf{pk}, \mathsf{label}_2, pw, r_2)$

Fig. 1. PAKE from splittable PKE with ASPH

for proving the security of the resulting PAKE. One can check that the PKEs with associated SPH (based on either DDH or decisional linear assumptions) in [41] satisfy Definition 5 with $\epsilon = 0$ (under certain choices of f and g). We will construct a splittable PKE with associated ASPH from lattices in Section 5.

3.3 A Framework for Two-Round PAKE

Let $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a splittable PKE scheme with respect to functions (f, g). Let $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk} \in K}, S, \mathsf{Proj} : K \to S)$ be the associated ϵ-approximate SPH for some $\epsilon \in (0, 1/2)$. Let the session key space be $\{0,1\}^\kappa$, where κ is the security parameter. Let $\mathsf{ECC} : \{0,1\}^\kappa \to \{0,1\}^\ell$ be an error-correcting code which can correct 2ϵ-fraction of errors, and let $\mathsf{ECC}^{-1} : \{0,1\}^\ell \to \{0,1\}^\kappa$ be the decoding algorithm. We assume that for uniformly distributed $\rho \in \{0,1\}^\ell$, the distribution of $w = \mathsf{ECC}^{-1}(\rho)$ conditioned on $w \neq \perp$ is uniform over $\{0,1\}^\kappa$. A high-level overview of our PAKE is given in Fig. 1.

Public parameters. The public parameter consists of a public key pk of the scheme $\mathcal{PKE}$, which can be generated by a trusted third party using $\mathsf{KeyGen}(1^\kappa)$. No users in the system need to know the secret key corresponding to pk.

Protocol Execution. Consider an execution of the protocol between a client A and a server B holding a shared password $pw \in \mathcal{D} \subset \mathcal{P}$, where $\mathcal{D}$ is the set of valid passwords in the system. First, A chooses random coins $r_1 \leftarrow_r \{0,1\}^*$ for encryption, a hash key $\mathsf{hk}_1 \leftarrow_r K$ for the ASPH, and computes the projection key $\mathsf{hp}_1 = \mathsf{Proj}(\mathsf{hk}_1)$. Then, it defines $\mathsf{label}_1 := A\|B\|\mathsf{hp}_1$, and computes $(u_1, v_1) = \mathsf{Enc}(\mathsf{pk}, \mathsf{label}_1, pw; r_1)$, where $u_1 = f(\mathsf{pk}, pw, r_1)$ and $v_1 = g(\mathsf{pk}, \mathsf{label}_1, pw, r_1)$. Finally, A sends $(A, \mathsf{hp}_1, c_1 = (u_1, v_1))$ to the server B.

Upon receiving $(A, \mathsf{hp}_1, c_1 = (u_1, v_1))$ from the client A, the server B checks if c_1 is a valid ciphertext with respect to pk and $\mathsf{label}_1 := A\|B\|\mathsf{hp}_1$.[9] If not, B rejects and aborts. Otherwise, B chooses random coins $r_2 \leftarrow_r \{0,1\}^*$ for encryption, a hash key $\mathsf{hk}_2 \leftarrow_r K$ for the ASPH, and a random session key $sk \leftarrow_r \{0,1\}^\kappa$. Then, it computes $\mathsf{hp}_2 = \mathsf{Proj}(\mathsf{hk}_2), u_2 = f(\mathsf{pk}, pw, r_2), tk = \mathsf{Hash}(\mathsf{hp}_1, (u_2, pw), r_2) \oplus \mathrm{H}_{\mathsf{hk}_2}(u_1, pw)$, and $\Delta = tk \oplus \mathsf{ECC}(sk)$. Finally, let $\mathsf{label}_2 := A\|B\|\mathsf{hp}_1\|c_1\|\mathsf{hp}_2\|\Delta$, the server B computes $v_2 = g(\mathsf{pk}, \mathsf{label}_2, pw, r_2)$, and sends the message $(\mathsf{hp}_2, c_2 = (u_2, v_2), \Delta)$ to the client A.

After receiving $(\mathsf{hp}_2, c_2 = (u_2, v_2), \Delta)$ from the server B, the client A checks if c_2 is a valid ciphertext with respect to pk and $\mathsf{label}_2 := A\|B\|\mathsf{hp}_1\|c_1\|\mathsf{hp}_2\|\Delta$. If not, A rejects and aborts. Otherwise, A computes $tk' = \mathrm{H}_{\mathsf{hk}_1}(u_2, pw) \oplus \mathsf{Hash}(\mathsf{hp}_2, (u_1, pw), r_1)$, and decodes to obtain $sk = \mathsf{ECC}^{-1}(tk' \oplus \Delta)$. If $sk = \perp$ (i.e., an error occurs during decoding), A rejects and aborts. Otherwise, A accepts $sk \in \{0,1\}^\kappa$ as the shared session key. This completes the description of our protocol.

In the following, we say that a user (or an instance of a user) accepts an incoming message msg as *a valid protocol message* if no abort happens during the computations after receiving msg. Note that a client/server will only obtain a session key when he accepts a received message as a valid protocol message.

Correctness. It suffices to show that honestly users can obtain the same session key $sk \in \{0,1\}^\kappa$ with overwhelming probability. First, all honestly generated ciphertexts are valid. Second, $\mathrm{H}_{\mathsf{hk}_1}(u_2, pw) \oplus \mathsf{Hash}(\mathsf{hp}_1, (u_2, pw), r_2) \in \{0,1\}^\ell$ has at most ϵ-fraction non-zeros by the ϵ-approximate correctness of the ASPH. Similarly, $\mathsf{Hash}(\mathsf{hp}_2, (u_1, pw), r_1) \oplus \mathrm{H}_{\mathsf{hk}_2}(u_1, pw) \in \{0,1\}^\ell$ has at most ϵ-fraction non-zeros. Thus, $tk' \oplus tk$ has at most 2ϵ-fraction non-zeros. Since ECC can correct 2ϵ-fraction of errors by assumption, we have that $sk = \mathsf{ECC}^{-1}(tk' \oplus tk \oplus \mathsf{ECC}(sk))$ holds. This completes the correctness argument.

Security. We now show that the above PAKE is secure. Formally,

Theorem 1. *If $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a splittable CCA-secure PKE scheme associated with an ϵ-approximate SPH $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk} \in K}, S, \mathsf{Proj} : K \to S)$, and $\mathsf{ECC} : \{0,1\}^\kappa \to \{0,1\}^\ell$ is an error-correcting code which can correct 2ϵ-fraction of errors, then the above protocol is a secure PAKE.*

Before giving the proof, we first give some intuitions. Without loss of generality we assume $0 \in \mathcal{P} \backslash \mathcal{D}$ (i.e., 0 is not a valid password in the system). First, by the CCA-security of the PKE scheme $\mathcal{PKE}$, the adversary cannot obtain any useful information of the real password pw via the $\mathsf{Execute}$ query (i.e., by eavesdropping on a protocol execution). In particular, it is computationally indistinguishable for the adversary if the encryption of pw is replaced by an encryption of 0 in answering the $\mathsf{Execute}$ queries. Since $0 \notin \mathcal{D}$, by the smoothness of the ASPH we have that the session keys corresponding to the instances used in the $\mathsf{Execute}$ queries are indistinguishable from uniform in the adversary's view.

[9] Recall that the validity of a ciphertext can be efficiently determined using pk alone.

Second, if the adversary simply relays the messages between honest instances, the proof is the same for the $\mathsf{Execute}$ queries. In case that the adversary modifies the message (i.e., the label-ciphertext pair) output by some instance, then one can use the decryption oracle provided by the CCA-security to decrypt the modified ciphertext, and check if the decrypted result pw' is equal to the real password pw. For $pw' = pw$ the attack is immediately considered successful (note that this will only increase the advantage of the adversary). By the CCA-security of $\mathcal{PKE}$ and the fact that pw is uniformly chosen from $\mathcal{D}$ at random, we have $\Pr[pw' = pw]$ is at most $1/|\mathcal{D}|$. Thus, for $Q(\kappa)$ times on-line attacks, this will only increase the adversary's advantage by at most $Q(\kappa)/|\mathcal{D}|$. Otherwise (i.e., $pw' \neq pw$) we again have that the corresponding session key is indistinguishable from uniform in the adversary's view by the smoothness of the ASPH.

Proof. We now formally prove Theorem 1 via a sequence of games from G_0 to G_{10}, where G_0 is the real security game, and G_{10} is a random game with uniformly chosen session keys. The security is established by showing that the adversary's advantage in game G_0 and G_{10} will differ at most $Q(\kappa)/|\mathcal{D}|+\text{negl}(\kappa)$. Let $\mathbf{Adv}_{\mathcal{A},i}(\kappa)$ be the adversary $\mathcal{A}$'s advantage in game G_i.

Game G_0: This game is the real security game as defined in Section 2.2, where all the oracle queries are honestly answered following the protocol specification.

Game G_1: This game is similar to game G_0 except that in answering each $\mathsf{Execute}$ query the value tk' is directly computed using the corresponding hash keys hk_1 and hk_2, i.e., $tk' = \mathrm{H}_{\mathsf{hk}_1}(u_2, pw) \oplus \mathrm{H}_{\mathsf{hk}_2}(u_1, pw)$.

Lemma 1. *Let $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk}\in K}, S, \mathsf{Proj} : K \to S)$ be an ϵ-approximate SPH, and $\mathsf{ECC} : \{0,1\}^\kappa \to \{0,1\}^\ell$ be an error-correcting code which can correct 2ϵ-fraction of errors, then $|\,\mathbf{Adv}_{\mathcal{A},1}(\kappa) - \mathbf{Adv}_{\mathcal{A},0}(\kappa)| \leq \text{negl}(\kappa)$.*

Proof. Since the simulator knows both hk_1 and hk_2, this lemma follows from the approximate correctness of the ASPH and the correctness of the ECC. □

Game G_2: This game is similar to game G_1 except that the ciphertext c_1 is replaced with an encryption of $0 \notin \mathcal{D}$ in answering each $\mathsf{Execute}$ query.

Lemma 2. *If $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a CCA-secure scheme, then we have that $|\,\mathbf{Adv}_{\mathcal{A},2}(\kappa) - \mathbf{Adv}_{\mathcal{A},1}(\kappa)| \leq \text{negl}(\kappa)$.*

Proof. Since the adversary $\mathcal{A}$ can only make polynomial times $\mathsf{Execute}$ queries, it is enough to consider that $\mathcal{A}$ only makes a single $\mathsf{Execute}$ query by a standard hybrid argument. In this case, the only difference between game G_1 and G_2 is that the encryption of pw is replaced by an encryption of $0 \notin \mathcal{D}$. We now show that any PPT adversary $\mathcal{A}$ that distinguishes the two games with non-negligible advantage can be directly transformed into an algorithm $\mathcal{B}$ that breaks the CCA-security of the underlying $\mathcal{PKE}$ scheme with the same advantage.

Formally, given a challenge public key pk, the algorithm $\mathcal{B}$ sets pk as the CRS of the protocol, and interacts with $\mathcal{A}$ as in game G_1. When $\mathcal{B}$ has to answer the adversary's $\mathsf{Execute}(A, i, B, j)$ query, it first randomly chooses a hash key $\mathsf{hk}_1 \leftarrow_r K$ for the ASPH, and computes the projection key $\mathsf{hp}_1 = \mathsf{Proj}(\mathsf{hk}_1)$. Then, $\mathcal{B}$ submits two plaintexts $(pw, 0)$ and $\mathsf{label}_1 := A\|B\|\mathsf{hp}_1$ to its own challenger, and obtains a challenge ciphertext c_1^*. Finally, $\mathcal{B}$ uses c_1^* to form the answer of the $\mathsf{Execute}(A, i, B, j)$ query, and returns whatever $\mathcal{A}$ outputs as its own guess.

Note that if c_1^* is an encryption of pw, then $\mathcal{B}$ exactly simulates the attack environment of game G_1 for adversary $\mathcal{A}$, else it simulates the attack environment of G_2 for $\mathcal{A}$. Thus, if $\mathcal{A}$ can distinguish G_1 and G_2 with non-negligible advantage, then $\mathcal{B}$ can break the CCA-security of $\mathcal{PKE}$ with the same advantage. □

Game G_3 This game is similar to game G_2 except that in answering each $\mathsf{Execute}$ query: 1) the value tk is directly computed by using the corresponding hash keys hk_1 and hk_2, i.e., $tk = \mathrm{H}_{\mathsf{hk}_1}(u_2, pw) \oplus \mathrm{H}_{\mathsf{hk}_2}(u_1, pw)$; 2) the ciphertext c_2 is replaced with an encryption of $0 \notin \mathcal{D}$.

Lemma 3. *If $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a splittable CCA-secure scheme, $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk} \in K}, S, \mathsf{Proj} : K \to S)$ is an ϵ-approximate SPH, and $\mathsf{ECC} : \{0,1\}^\kappa \to \{0,1\}^\ell$ is an error-correcting code which can correct 2ϵ-fraction of errors, then we have that $|\,\mathbf{Adv}_{\mathcal{A},3}(\kappa) - \mathbf{Adv}_{\mathcal{A},2}(\kappa)| \le \mathrm{negl}(\kappa)$.*

Proof. This lemma can be shown by using a sequence of games similar to that from G_0 to G_2 except the modified CCA-security game considered in Definition 4 is used instead of the standard CCA-security game, we omit the details. □

Game G_4 This game is similar to game G_3 except that a random session key $\mathsf{sk}_A^i = \mathsf{sk}_B^j$ is set for both Π_A^i and Π_B^j in answering each $\mathsf{Execute}(A, i, B, j)$ query.

Lemma 4. *If $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk} \in K}, S, \mathsf{Proj} : K \to S)$ is an ϵ-approximate SPH, then we have that $|\,\mathbf{Adv}_{\mathcal{A},4}(\kappa) - \mathbf{Adv}_{\mathcal{A},3}(\kappa)| \le \mathrm{negl}(\kappa)$.*

Proof. Since both ciphertexts $c_1 = (u_1, v_1)$ and $c_2 = (u_2, v_2)$ in answering each $\mathsf{Execute}(A, i, B, j)$ query are encryptions of $0 \notin \mathcal{D}$, the value $tk' = tk = \mathrm{H}_{\mathsf{hk}_1}(u_2, pw) \oplus \mathrm{H}_{\mathsf{hk}_2}(u_1, pw)$ is statistically close to uniform by the smoothness of the ASPH. Thus, the masking part $\Delta = tk \oplus \mathsf{ECC}(sk)$ in answering each $\mathsf{Execute}(A, i, B, j)$ query statistically hides $sk \in \{0,1\}^\kappa$ from the adversary $\mathcal{A}$. Since $sk \in \{0,1\}^\kappa$ is uniformly random, the modification in game G_4 can only introduce a negligible statistical difference. Since $\mathcal{A}$ can only make polynomial times $\mathsf{Execute}$ queries, this lemma follows by a standard hybrid argument. □

Game G_5 This game is similar to game G_4 except that the simulator generates the CRS pk by running $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, and keeps sk private.

Lemma 5. $\mathbf{Adv}_{\mathcal{A},5}(\kappa) = \mathbf{Adv}_{\mathcal{A},4}(\kappa)$.

Proof. This lemma follows from the fact that the modification from game G_4 to G_5 is just conceptual. □

Before continuing, we divide the adversary's Send query into three types depending on the message which may be sent as part of the protocol:

- $\mathsf{Send}_0(A, i, B)$: the adversary prompts an unused instance Π_A^i to execute the protocol with partner B. This oracle updates $\mathsf{pid}_A^i = B$, and returns the message $\mathsf{msg}_1 = (A, \mathsf{hp}_1, c_1)$ output by Π_A^i to the adversary.
- $\mathsf{Send}_1(B, j, (A, \mathsf{hp}_1, c_1))$: the adversary sends message $\mathsf{msg}_1 = (A, \mathsf{hp}_1, c_1)$ to an unused instance Π_B^j. This oracle updates $(\mathsf{pid}_B^j, \mathsf{sk}_B^j, \mathrm{acc}_B^j, \mathrm{term}_B^j)$ as appropriate, and returns the message $\mathsf{msg}_2 = (\mathsf{hp}_2,\, c_2, \Delta)$ output by Π_B^j to the adversary (only if Π_B^j accepts msg_1 as a valid protocol message).
- $\mathsf{Send}_2(A, i, (\mathsf{hp}_2, c_2, \Delta))$: the adversary sends message $\mathsf{msg}_2 = (\mathsf{hp}_2, c_2, \Delta)$ to instance Π_A^i. This oracle updates $(\mathsf{sk}_B^j, \mathrm{acc}_B^j, \mathrm{term}_B^j)$ as appropriate.

Game G_6 This game is similar to game G_5 except that each $\mathsf{Send}_1(B, j, \mathsf{msg}_1' = (A', \mathsf{hp}_1', c_1'))$ query is handled as follows:

- If msg_1' was output by a previous $\mathsf{Send}_0(A', *, B)$ query, the simulator $\mathcal{C}$ performs exactly as in game G_5;
- Otherwise, let $\mathsf{label}_1' := A' \| B \| \mathsf{hp}_1'$, and distinguish the following two cases:
  - If c_1' is not a valid ciphertext with respect to pk and label_1', the simulator $\mathcal{C}$ rejects this query;
  - Else, $\mathcal{C}$ decrypts $(\mathsf{label}_1', c_1')$ using the secret key sk corresponding to pk, and let pw' be the decryption result. If pw' is equal to the real password pw shared by A and B (i.e., $pw' = pw$), the simulator $\mathcal{C}$ declares that $\mathcal{A}$ succeeds, and terminates the experiment. Otherwise, $\mathcal{C}$ answers this query as in game G_5 but sets the session key sk_B^j for instance Π_B^j by using an independently and uniformly chosen element from $\{0,1\}^\kappa$.

Lemma 6. *If $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk} \in K}, S, \mathsf{Proj} : K \to S)$ is an ϵ-approximate SPH, then we have that $\mathbf{Adv}_{\mathcal{A},5}(\kappa) \le \mathbf{Adv}_{\mathcal{A},6}(\kappa) + \mathrm{negl}(\kappa)$.*

Proof. We only have to consider the case that $\mathsf{msg}_1' = (A', \mathsf{hp}_1', c_1')$ was not output by any previous $\mathsf{Send}_0(A', *, B)$ query and c_1' is a valid cipertext with respect to pk and label_1' (note that B will always reject invalid ciphertexts in the real run of the protocol). Since $\mathcal{C}$ knows the secret key sk corresponding to pk in both game G_5 and G_6, it can always decrypt $(\mathsf{label}_1', c_1')$ to obtain the decryption result pw'. Obviously, the modification for the case $pw' = pw$ can only increase the advantage of the adversary $\mathcal{A}$. As for the case $pw' \neq pw$, we have $(\mathsf{label}_1', c_1', pw) \notin \bar{L}$. By the smoothness of the underlying ASPH (in Definition 5), the masking part $\Delta = tk \oplus \mathsf{ECC}(sk)$ output by Π_B^j statistically hides $sk \in \{0,1\}^\kappa$ from the adversary $\mathcal{A}$ with knowledge of $\mathsf{hp}_2 = \mathsf{Proj}(\mathsf{hk}_2)$ (because tk has a term $\mathrm{H}_{\mathsf{hk}_2}(u_1', pw)$ for $c_1' = (u_1', v_1')$ and $\mathsf{hk}_2 \leftarrow_r K$). Using the fact that sk is essentially uniformly chosen from $\{0,1\}^\kappa$, we have that the modification for the case $pw' \neq pw$ in game G_6 can only introduce a negligible statistical difference. In all, we have that $\mathbf{Adv}_{\mathcal{A},5}(\kappa) \le \mathbf{Adv}_{\mathcal{A},6}(\kappa) + \mathrm{negl}(\kappa)$. □

Game G_7 This game is similar to game G_6 except that each $\mathsf{Send}_2(A, i, \mathsf{msg}'_2 = (\mathsf{hp}'_2, c'_2, \Delta'))$ query is handled as follows: let $\mathsf{msg}_1 = (A, \mathsf{hp}_1, c_1)$ be the message output by a previous $\mathsf{Send}_0(A, i, B)$ query (note that such a query must exist),

– If msg'_2 was output by a previous $\mathsf{Send}_1(B, j, \mathsf{msg}_1)$ query, the simulator $\mathcal{C}$ performs as in game G_6 except that $\mathcal{C}$ computes tk' directly using the corresponding hash keys hk_1 and hk_2, and sets the session key $\mathsf{sk}^i_A = \mathsf{sk}^j_B$;
– Otherwise, let $\mathsf{label}'_2 := A\|B\|\mathsf{hp}_1\|c_1\|\mathsf{hp}'_2\|\Delta'$, and distinguish the following two cases:
  - If c'_2 is not a valid ciphertext with respect to pk and label'_2, the simulator $\mathcal{C}$ rejects this query;
  - Else, $\mathcal{C}$ decrypts $(\mathsf{label}'_2, c'_2)$ using the secret key sk corresponding to pk, and let pw' be the decryption result. If $pw' = pw$, the simulator $\mathcal{C}$ declares that $\mathcal{A}$ succeeds, and terminates the experiment. Otherwise, $\mathcal{C}$ performs the computations on behalf of Π^i_A as in game G_6. If Π^i_A accepts msg'_2 as a valid protocol message, $\mathcal{C}$ sets the session key sk^i_A for instance Π^i_A by using an independently and uniformly chosen element from $\{0,1\}^\kappa$ (note that Π^i_A might reject msg'_2 if the decoding algorithm returns $\perp$, and thus no session key is generated in this case, i.e., $\mathrm{acc}^i_A = 0$ and $\mathsf{sk}^i_A = \perp$).

Lemma 7. *If $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk}\in K}, S, \mathsf{Proj} : K \to S)$ is an ϵ-approximate SPH, and $\mathsf{ECC} : \{0,1\}^\kappa \to \{0,1\}^\ell$ is an error-correcting code which can correct 2ϵ-fraction of errors, then $\mathbf{Adv}_{\mathcal{A},6}(\kappa) \le \mathbf{Adv}_{\mathcal{A},7}(\kappa) + \mathrm{negl}(\kappa)$.*

Proof. First, if both msg_1 and msg'_2 were output by previous oracle queries, then the simulator $\mathcal{C}$ knows the corresponding hash keys hk_1 and hk_2 needed for computing tk', and it is just a conceptual modification to compute tk' using $(\mathsf{hk}_1, \mathsf{hk}_2)$ and set $\mathsf{sk}^i_A = \mathsf{sk}^j_B$. Second, as discussed in the proof of Lemma 6, $\mathcal{C}$ knows the secret key sk corresponding to pk in both game G_6 and G_7, it can always decrypt $(\mathsf{label}'_2, c'_2)$ to obtain the decryption result pw'. Obviously, the modification for the case $pw' = pw$ can only increase the advantage of the adversary $\mathcal{A}$. Moreover, if $pw' \ne pw$, we have $(\mathsf{label}'_2, c'_2, pw) \notin \bar{L}$. By the smoothness of the ASPH, the value $tk' \in \{0,1\}^\ell$ computed by Π^i_A is statistically close to uniform over $\{0,1\}^\ell$ (because tk' has a term $\mathrm{H}_{\mathsf{hk}_1}(u'_2, pw)$ for $c'_2 = (u'_2, v'_2)$). By our assumption on ECC^{-1}, if $sk = \mathsf{ECC}^{-1}(tk' \oplus \Delta') \ne \perp$, then it is statistically close to uniform over $\{0,1\}^\kappa$. Thus, the modification for the case $pw' \ne pw$ in game G_6 can only introduce a negligible statistical difference. In all, we can have that $\mathbf{Adv}_{\mathcal{A},6}(\kappa) \le \mathbf{Adv}_{\mathcal{A},7}(\kappa) + \mathrm{negl}(\kappa)$ holds. □

Game G_8 This game is similar to game G_7 except that the ciphertext c_1 is replaced with an encryption of $0 \notin \mathcal{D}$ in answering each $\mathsf{Send}_0(A, i, B)$ query.

Lemma 8. *If $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a CCA-secure scheme, we have that $|\,\mathbf{Adv}_{\mathcal{A},8}(\kappa) - \mathbf{Adv}_{\mathcal{A},7}(\kappa)| \le \mathrm{negl}(\kappa)$.*

Proof. By a standard hybrid argument, it is enough to consider that $\mathcal{A}$ only makes a single $\mathsf{Send}_0(A, i, B)$ query. In this case, the only difference between game G_8 and G_7 is that the encryption of pw is replaced with an encryption of $0 \notin \mathcal{D}$. We now show that any PPT adversary $\mathcal{A}$ that distinguishes the two games with non-negligible advantage can be directly transformed into an algorithm $\mathcal{B}$ that breaks the CCA-security of the underlying $\mathcal{PKE}$ scheme.

Formally, given a challenge public key pk, the algorithm $\mathcal{B}$ sets pk as the CRS of the protocol, and simulates the attack environment for $\mathcal{A}$ as in game G_7. When $\mathcal{B}$ has to answer the adversary's $\mathsf{Send}_0(A, i, B)$ query, it first randomly chooses a hash key $\mathsf{hk}_1 \leftarrow_r K$ for the ASPH, and computes the projection key $\mathsf{hp}_1 = \mathsf{Proj}(\mathsf{hk}_1)$. Then, $\mathcal{B}$ submits two plaintexts $(pw, 0)$ and $\mathsf{label}_1 := A\|B\|\mathsf{hp}_1$ to its own challenger, and obtains a challenge ciphertext c_1^*. Finally, $\mathcal{B}$ sends $(A, \mathsf{hp}_1, c_1^*)$ to the adversary $\mathcal{A}$. When $\mathcal{B}$ has to decrypt some valid label-ciphertext pair $(\mathsf{label}_1', c_1') \neq (\mathsf{label}_1, c_1^*)$, it submits $(\mathsf{label}_1', c_1')$ to its own CCA-security challenger for decryption. At some time, the adversary $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$, $\mathcal{B}$ outputs b as its own guess.

Note that if c_1^* is an encryption of pw, then $\mathcal{B}$ exactly simulates the attack environment of game G_7 for adversary $\mathcal{A}$, else it simulates the attack environment of game G_8 for $\mathcal{A}$. Thus, if $\mathcal{A}$ can distinguish game G_7 and G_8 with non-negligible advantage, then $\mathcal{B}$ can break the CCA-security of the PKE scheme $\mathcal{PKE}$ with the same advantage, which completes the proof. □

Game G_9 This game is similar to game G_8 except that each $\mathsf{Send}_1(B, j, \mathsf{msg}_1' = (A', \mathsf{hp}_1', c_1'))$ query is handled as follows:

- If msg_1' was output by a previous $\mathsf{Send}_0(A', *, B)$ query, the simulator $\mathcal{C}$ performs as in game G_8 except that it computes tk directly using the corresponding hash keys $(\mathsf{hk}_1, \mathsf{hk}_2)$, and sets the session key sk_B^j for instance Π_B^j by using an independently and uniformly chosen element from $\{0, 1\}^\kappa$;
- Otherwise, $\mathcal{C}$ performs exactly as in game G_8.

Lemma 9. *If $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0, 1\}^\ell\}_{\mathsf{hk} \in K}, S, \mathsf{Proj} : K \to S)$ is an ϵ-approximate SPH, and $\mathsf{ECC} : \{0, 1\}^\kappa \to \{0, 1\}^\ell$ is an error-correcting code which can correct 2ϵ-fraction of errors, then $|\,\mathbf{Adv}_{\mathcal{A},9}(\kappa) - \mathbf{Adv}_{\mathcal{A},8}(\kappa)| \leq \mathrm{negl}(\kappa)$.*

Proof. Note that if msg_1' was output by a previous $\mathsf{Send}_0(A', *, B)$ query, then we have that 1) the simulator $\mathcal{C}$ knows the corresponding hash keys $(\mathsf{hk}_1, \mathsf{hk}_2)$ and 2) $c_1' = (u_1', v_1')$ is an encryption of $0 \notin \mathcal{D}$. In other words, $\mathcal{C}$ can directly compute tk using $(\mathsf{hk}_1, \mathsf{hk}_2)$, and tk is statistically close to uniform (because $pw \neq 0$, and tk has a term $\mathrm{H}_{\mathsf{hk}_2}(u_1', pw)$ that is statistically close to uniform by the smoothness of the ASPH). Thus, the masking part $\Delta = tk \oplus \mathsf{ECC}(sk)$ output by Π_B^j statistically hides $sk \in \{0, 1\}^\kappa$ from the adversary $\mathcal{A}$. Since sk is essentially uniformly chosen from $\{0, 1\}^\kappa$, we have that the modification in game G_9 can only introduce a negligible statistical difference, which means that $|\,\mathbf{Adv}_{\mathcal{A},9}(\kappa) - \mathbf{Adv}_{\mathcal{A},8}(\kappa)| \leq \mathrm{negl}(\kappa)$. □

Game G_{10} This game is similar to game G_9 except that each $\mathsf{Send}_1(B, j, \mathsf{msg}'_1 = (A', \mathsf{hp}'_1, c'_1))$ query is handled as follows:

- If msg'_1 was output by a previous $\mathsf{Send}_0(A', *, B)$ query, the simulator $\mathcal{C}$ performs as in game G_9 except that the ciphertext c_2 is replaced with an encryption of $0 \notin \mathcal{D}$;
- Otherwise, $\mathcal{C}$ performs exactly as in game G_9.

Lemma 10. *If $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a splittable CCA-secure scheme, then $|\,\mathbf{Adv}_{\mathcal{A},10}(\kappa) - \mathbf{Adv}_{\mathcal{A},9}(\kappa)| \leq \mathrm{negl}(\kappa)$.*

Proof. As before, it is enough to consider that $\mathcal{A}$ only makes a single $\mathsf{Send}_1(B, j, \mathsf{msg}'_1 = (A', \mathsf{hp}'_1, c'_1))$ query with msg'_1 output by some $\Pi^i_{A'}$. We now show that any PPT adversary $\mathcal{A}$ that distinguishes the two games with non-negligible advantage can be directly transformed into an algorithm $\mathcal{B}$ that breaks the modified CCA-security game of the underlying $\mathcal{PKE}$ scheme with the same advantage.

Formally, given a challenge public key pk, the algorithm $\mathcal{B}$ sets pk as the CRS of the protocol, and interacts with $\mathcal{A}$ as in game G_9. When $\mathcal{B}$ has to answer a $\mathsf{Send}_1(B, j, \mathsf{msg}'_1 = (A', \mathsf{hp}'_1, c'_1))$ query for some $c'_1 = (u'_1, v'_1)$, it first randomly chooses a hash key $\mathsf{hk}_2 \leftarrow_r K$ for the ASPH, a random session key $sk \leftarrow_r \{0,1\}^\kappa$, and computes $\mathsf{hp}_2 = \mathsf{Proj}(\mathsf{hk}_2)$. Then, $\mathcal{B}$ submits two plaintexts $(pw, 0)$ to its own challenger. After obtaining u^*_2, $\mathcal{B}$ computes $tk = \mathrm{H}_{\mathsf{hk}_1}(u^*_2, pw) \oplus \mathrm{H}_{\mathsf{hk}_2}(u'_1, pw)$, $\Delta = tk \oplus \mathsf{ECC}(sk)$, and submits $\mathsf{label}_2 := A' \| B \| \mathsf{hp}'_1 \| c'_1 \| \mathsf{hp}_2 \| \Delta$ to its own modified CCA-security challenger to obtain the challenge ciphertext $c^*_2 = (u^*_2, v^*_2)$. Finally, $\mathcal{B}$ sends $(\mathsf{hp}_2, c^*_2, \Delta)$ to the adversary $\mathcal{A}$. When $\mathcal{B}$ has to decrypt some valid label-ciphertext pair $(\mathsf{label}'_2, c'_2) \neq (\mathsf{label}_2, c^*_2)$, it submits $(\mathsf{label}'_2, c'_2)$ to its own challenger for decryption. At some time, the adversary $\mathcal{A}$ outputs a bit $b \in \{0,1\}$, $\mathcal{B}$ outputs b as its own guess.

Note that if c^*_2 is an encryption of pw, then $\mathcal{B}$ perfectly simulates the attack environment of game G_9 for adversary $\mathcal{A}$, else it simulates the attack environment of G_{10} for $\mathcal{A}$. Thus, if $\mathcal{A}$ can distinguish game G_9 and G_{10} with non-negligible advantage, then algorithm $\mathcal{B}$ can break the modified CCA-security of the PKE scheme $\mathcal{PKE}$ with the same advantage, which completes the proof. $\square$

Lemma 11. *If the adversary $\mathcal{A}$ only makes at most $Q(\kappa)$ times on-line attacks, then we have that $\mathbf{Adv}_{\mathcal{A},10}(\kappa) \leq Q(\kappa)/|\mathcal{D}| + \mathrm{negl}(\kappa)$.*

Proof. Let $\mathcal{E}$ be the event that $\mathcal{A}$ submits a ciphertext that decrypts to the real password pw. If $\mathcal{E}$ does not happen, we have that the advantage of $\mathcal{A}$ is negligible in κ (because all the session keys are uniformly chosen at random). Now, we estimate the probability that $\mathcal{E}$ happens. Since in game G_{10}, all the ciphertexts output by oracle queries are encryptions of $0 \notin \mathcal{D}$, the adversary cannot obtain useful information of the real password pw via the oracle queries. Thus, for any adversary $\mathcal{A}$ that makes at most $Q(\kappa)$ times on-line attacks, the probability that $\mathcal{E}$ happens is at most $Q(\kappa)/|\mathcal{D}|$, i.e., $\Pr[E] \leq Q(\kappa)/|\mathcal{D}|$. By a simple calculation, we have $\mathbf{Adv}_{\mathcal{A},10}(\kappa) \leq Q(\kappa)/|\mathcal{D}| + \mathrm{negl}(\kappa)$. $\square$

In all, we have that $\mathbf{Adv}_{\mathcal{A},0}(\kappa) \leq Q(\kappa)/|\mathcal{D}| + \mathrm{negl}(\kappa)$ by Lemma 1∼11. This completes the proof of Theorem 1. $\square$

4 Lattices

In this section, we first give some backgrounds on lattices. Then, we show a useful technical lemma, which was crucial for our construction in Section 5.

4.1 Backgrounds on Lattices

An m-dimensional full-rank lattice $\mathbf{\Lambda} \subset \mathbb{R}^m$ is the set of all integral combinations of m linearly independent vectors $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_m) \in \mathbb{R}^{m\times m}$, i.e., $\mathbf{\Lambda} = \mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^m x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$. The dual lattice of $\mathbf{\Lambda}$, denote $\mathbf{\Lambda}^*$ is defined to be $\mathbf{\Lambda}^* = \{\mathbf{x} \in \mathbb{R}^m : \ \forall\ \mathbf{v} \in \mathbf{\Lambda}, \langle \mathbf{x}, \mathbf{v}\rangle \in \mathbb{Z}\}$. For $\mathbf{x} \in \mathbf{\Lambda}$, define the Gaussian function $\rho_{s,\mathbf{c}}(\mathbf{x})$ over $\mathbf{\Lambda} \subseteq \mathbb{Z}^m$ centered at $\mathbf{c} \in \mathbb{R}^m$ with parameter $s > 0$ as $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x} - \mathbf{c}\|^2/s^2)$. Let $\rho_{s,\mathbf{c}}(\mathbf{\Lambda}) = \sum_{\mathbf{x}\in\mathbf{\Lambda}} \rho_{s,\mathbf{c}}(\mathbf{x})$, and define the discrete Gaussian distribution over $\mathbf{\Lambda}$ as $D_{\mathbf{\Lambda},s,\mathbf{c}}(\mathbf{y}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{y})}{\rho_{s,\mathbf{c}}(\mathbf{\Lambda})}$, where $\mathbf{y} \in \mathbf{\Lambda}$. The subscripts s and $\mathbf{c}$ are taken to be 1 and $\mathbf{0}$ (resp.) when omitted.

Lemma 12 ([50,53]). *For any positive integer $m \in \mathbb{Z}$, and large enough $s \geq \omega(\sqrt{\log m})$, we have $\Pr_{\mathbf{x}\leftarrow_r D_{\mathbb{Z}^m,s}}[\|\mathbf{x}\| > s\sqrt{m}] \leq 2^{-m+1}$.*

First introduced in [50], the smoothing parameter $\eta_\epsilon(\mathbf{\Lambda})$ for any real $\epsilon > 0$ is defined as the smallest real $s > 0$ s.t. $\rho_{1/s}(\mathbf{\Lambda}^*\backslash\{\mathbf{0}\}) \leq \epsilon$.

Lemma 13 ([50]). *For any m-dimensional lattice $\mathbf{\Lambda}$, $\eta_\epsilon(\mathbf{\Lambda}) \leq \sqrt{m}/\lambda_1(\mathbf{\Lambda}^*)$, where $\epsilon = 2^{-m}$, and $\lambda_1(\mathbf{\Lambda}^*)$ is the length of the shortest vector in lattice $\mathbf{\Lambda}^*$.*

Lemma 14 ([32]). *Let $\mathbf{\Lambda}, \mathbf{\Lambda}'$ be m-dimensional lattices, with $\mathbf{\Lambda}' \subseteq \mathbf{\Lambda}$. Then, for any $\epsilon \in (0, 1/2)$, any $s \geq \eta_\epsilon(\mathbf{\Lambda}')$, and any $\mathbf{c} \in \mathbb{R}^m$, the distribution of $(D_{\mathbf{\Lambda},s,\mathbf{c}} \bmod \mathbf{\Lambda}')$ is within distance at most 2ϵ of uniform over $(\mathbf{\Lambda} \bmod \mathbf{\Lambda}')$.*

Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, define lattices $\mathbf{\Lambda}_q^\perp(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}^m \ \ s.t.\ \mathbf{A}\mathbf{e} = 0 \mod q\}$ and $\mathbf{\Lambda}_q(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m \ s.t.\ \exists \mathbf{s} \in \mathbb{Z}^n,\ \mathbf{A}^t\mathbf{s} = \mathbf{y} \mod q\}$. We have the following facts.

Lemma 15 ([32]). *Let integers $n, m \in \mathbb{Z}$ and prime q satisfy $m \geq 2n \log q$. Then, for all but an at most $2q^{-n}$ fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we have that 1) the columns of $\mathbf{A}$ generate $\mathbb{Z}_q^n$, 2) $\lambda_1^\infty(\mathbf{\Lambda}_q(\mathbf{A})) \geq q/4$, and 3) the smoothing parameter $\eta_\epsilon(\mathbf{\Lambda}_q^\perp(\mathbf{A})) \leq \omega(\sqrt{\log m})$ for some $\epsilon = \mathrm{negl}(\kappa)$.*

Lemma 16 ([32]). *Assume the columns of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ generate $\mathbb{Z}_q^n$, and let $\epsilon \in (0, 1/2)$ and $s \geq \eta_\epsilon(\mathbf{\Lambda}_q^\perp(\mathbf{A}))$. Then for $\mathbf{e} \sim D_{\mathbb{Z}^m,s}$, the distribution of the syndrome $\mathbf{u} = \mathbf{A}\mathbf{e} \mod q$ is within statistical distance 2ϵ of uniform over $\mathbb{Z}_q^n$.*

Furthermore, fix $\mathbf{u} \in \mathbb{Z}_q^n$ and let $\mathbf{v} \in \mathbb{Z}^m$ be an arbitrary solution to $\mathbf{A}\mathbf{v} = \mathbf{u} \mod q$. Then the conditional distribution of $\mathbf{e} \sim D_{\mathbb{Z}^m,s}$ given $\mathbf{A}\mathbf{e} = \mathbf{u} \mod q$ is exactly $\mathbf{v} + D_{\mathbf{\Lambda}_q^\perp(\mathbf{A}),s,-\mathbf{v}}$.

There exist efficient algorithms [5,8,49] to generate almost uniform matrix $\mathbf{A}$ together with a trapdoor (or a short basis of $\mathbf{\Lambda}_q^\perp(\mathbf{A})$).

Proposition 1 **([49]).** *Given any integers $n \geq 1$, $q > 2$, sufficiently large $m = O(n \log q)$, and $k = \lceil \log_2 q \rceil$, there is an efficient algorithm* $\mathsf{TrapGen}(1^n, 1^m, q)$ *that outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor $\mathbf{R} \in \mathbb{Z}_q^{(m-nk)\times nk}$ such that $s_1(\mathbf{R}) \leq \sqrt{m} \cdot \omega(\sqrt{\log n})$, and $\mathbf{A}$ is $\mathrm{negl}(n)$-close to uniform.*

Moreover, given any $\mathbf{y} = \mathbf{A}^t\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$ satisfying $\|\mathbf{e}\| \leq \frac{q}{2\sqrt{5(s_1(\mathbf{R})^2+1)}}$, there exists an efficient algorithm $\mathsf{Solve}(\mathbf{A}, \mathbf{R}, \mathbf{y})$ *that outputs the vector $\mathbf{s} \in \mathbb{Z}_q^n$.*

Let $\mathsf{dist}(\mathbf{z}, \mathbf{\Lambda}_q(\mathbf{A}))$ be the distance of the vector $\mathbf{z}$ from the lattice $\mathbf{\Lambda}_q(\mathbf{A})$. For any $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, define $Y_{\mathbf{A}} = \left\{\tilde{\mathbf{y}} \in \mathbb{Z}_q^m : \forall a \in \mathbb{Z}_q \backslash \{0\}, \mathsf{dist}(a\tilde{\mathbf{y}}, \mathbf{\Lambda}_q(\mathbf{A})) \geq \sqrt{q}/4\right\}$.

Lemma 17 **([32,40]).** *Let integers n, m and prime q satisfy $m \geq 2n \log q$. Let $\gamma \geq \sqrt{q} \cdot \omega(\sqrt{\log n})$. Then, for all but a negligible fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and for any $\mathbf{z} \in Y_A$, the distribution of $(\mathbf{A}\mathbf{e}, \mathbf{z}^t\mathbf{e})$ is statistically close to uniform over $\mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{e} \sim D_{\mathbb{Z}^m,\gamma}$.*

Lemma 18 **([40]).** *Let κ be the security parameter. Let intergers n_1, n_2, m and prime q satisfy $m \geq (n_1 + n_2 + 1)\log q$ and $n_1 = 2(n_2 + 1) + \omega(\log \kappa)$. Then, for all but a negligible fraction of $\mathbf{B} \in \mathbb{Z}_q^{m\times n_1}$, the probability that there exist numbers $a, a' \in \mathbb{Z}_q \backslash \{0\}$, vectors $\mathbf{w} \neq \mathbf{w}' \in \mathbb{Z}_q^{n_2}$, and a vector $\mathbf{c} \in \mathbb{Z}_q^m$, s.t.*

$$\mathsf{dist}(a\mathbf{y}, \mathbf{\Lambda}_q(\mathbf{B}^t)) \leq \sqrt{q}/4 \textit{ and } \mathsf{dist}(a'\mathbf{y}', \mathbf{\Lambda}_q(\mathbf{B}^t)) \leq \sqrt{q}/4$$

is negligible in κ over the uniformly random choice of $\mathbf{U} \leftarrow_r \mathbb{Z}_q^{m\times(n_2+1)}$, where $\mathbf{y} = \mathbf{c} - \mathbf{U}\begin{pmatrix}1\\ \mathbf{w}\end{pmatrix}$ and $\mathbf{y}' = \mathbf{c} - \mathbf{U}\begin{pmatrix}1\\ \mathbf{w}'\end{pmatrix}$.

Learning with Errors. For any positive integers $n, q \in \mathbb{Z}$, real $\alpha > 0$ and vector $\mathbf{s} \in \mathbb{Z}_q^n$, define the distribution $A_{\mathbf{s},\alpha} = \{(\mathbf{a}, \mathbf{a}^t\mathbf{s} + e \mod q) : \mathbf{a} \leftarrow_r \mathbb{Z}_q^n, e \leftarrow_r D_{\mathbb{Z},\alpha q}\}$. For any m independent samples $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m)$ from $A_{\mathbf{s},\alpha}$, we denote it in matrix form $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, where $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ and $\mathbf{b} = (b_1, \ldots, b_m)^t$. We say that the $\mathrm{LWE}_{n,q,\alpha}$ problem is hard if, for uniformly random $\mathbf{s} \leftarrow_r \mathbb{Z}_q^n$ and given polynomially many samples, no PPT algorithm can recover $\mathbf{s}$ with non-negligible probability. The decisional LWE problem is asked to distinguish polynomially many samples from uniform. For certain parameters, the decisional LWE problem is polynomially equivalent to its search version, which is in turn known to be at least as hard as quantumly approximating SIVP on n-dimensional lattices to within polynomial factors in the worst case [55].

4.2 An Adaptive Smoothing Lemma for q-ary Lattices

Based on a good use of Lemma 17 from [32], the authors [40] constructed the first lattice-based ASPH with adaptive projection function [31,41] (i.e., the projection key is generated after given the input ciphertext). However, Lemma 17 is not enough to obtain a non-adaptive ASPH for constructing two-round PAKEs (where the ciphertext is chosen after seeing the projection key). Specifically, it provides no guarantee for the distribution of $\mathbf{z}^t\mathbf{e}$ when the choice of $\mathbf{z} \in Y_{\mathbf{A}}$ is dependent on $\mathbf{Ae} \in \mathbb{Z}_q^n$. In particular, it is possible that for each $\mathbf{z} \in Y_{\mathbf{A}}$, there is a negligible fraction of bad values $Bad_{\mathbf{z}} \subset \mathbb{Z}_q^n$ such that for all $\mathbf{Ae} \in Bad_{\mathbf{z}}$ the distribution of $\mathbf{z}^t\mathbf{e}$ is far from uniform (and thus given a fixed $\mathbf{u} = \mathbf{Ae} \in \mathbb{Z}_q^n$, the adversary may choose $\mathbf{z} \in Y_{\mathbf{A}}$ such that $\mathbf{u} \in Bad_{\mathbf{z}}$). Instead, we show a stronger result in Lemma 19, which is very crucial for our construction in Section 5.

Lemma 19. *Let positive integers $n, m \in \mathbb{Z}$ and prime q satisfy $m \geq 2n \log q$. Let $\gamma \geq 4\sqrt{mq}$. Then, for all but a negligible fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and for any (even unbounded) function $h : \mathbb{Z}_q^n \to Y_{\mathbf{A}}$, the distribution of $(\mathbf{Ae}, \mathbf{z}^t\mathbf{e})$ is statistically close to uniform over $\mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{e} \sim D_{\mathbb{Z}^m,\gamma}$ and $\mathbf{z} = h(\mathbf{Ae})$.*

Proof. By Lemma 15, for all but a negligible fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, the columns of $\mathbf{A}$ generate $\mathbb{Z}_q^n$ and the length $\lambda_1(\mathbf{\Lambda}_q(\mathbf{A}))$ (in the l_2 norm) of the shortest vector in $\mathbf{\Lambda}_q(\mathbf{A})$ is at least $q/4$ (since $\lambda_1(\mathbf{\Lambda}_q(\mathbf{A})) \geq \lambda_1^\infty(\mathbf{\Lambda}_q(\mathbf{A})) \geq q/4$). Moreover, the smoothing parameter $\eta_\epsilon(\mathbf{\Lambda}_q^\perp(\mathbf{A})) \leq \omega(\sqrt{\log m})$ for some negligible ϵ. In the following, we always assume that $\mathbf{A}$ satisfies the above properties. Since $\gamma \geq 4\sqrt{mq} > \eta_\epsilon(\mathbf{\Lambda}_q^\perp(\mathbf{A}))$, by Lemma 16 the distribution of $\mathbf{Ae} \mod q$ is within statistical distance 2ϵ of uniform over $\mathbb{Z}_q^n$, where $\mathbf{e} \sim D_{\mathbb{Z}^m,\gamma}$. Furthermore, fix $\mathbf{u} \in \mathbb{Z}_q^n$ and let $\mathbf{v}$ be an arbitrary solution to $\mathbf{Av} = \mathbf{u} \mod q$, the conditional distribution of $\mathbf{e} \sim D_{\mathbb{Z}^m,\gamma}$ given $\mathbf{Ae} = \mathbf{u} \mod q$ is exactly $\mathbf{v} + D_{\mathbf{\Lambda}_q^\perp(\mathbf{A}),\gamma,-\mathbf{v}}$. Thus, it is enough to show that for arbitrary $\mathbf{v} \in \mathbb{Z}^m$ and $\mathbf{z} = h(\mathbf{Av}) \in Y_{\mathbf{A}}$, the distribution $\mathbf{z}^t\mathbf{e}$ is statistically close to uniform over $\mathbb{Z}_q$, where $\mathbf{e} \sim D_{\mathbf{\Lambda}_q^\perp(\mathbf{A}),\gamma,-\mathbf{v}}$.

Now, fix $\mathbf{v} \in \mathbb{Z}^m$ and $\mathbf{z} = h(\mathbf{Av}) \in Y_{\mathbf{A}}$, let $\mathbf{A}' = \begin{pmatrix} \mathbf{A} \\ \mathbf{z}^t \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m}$. By the definition $Y_{\mathbf{A}} = \{\tilde{\mathbf{y}} \in \mathbb{Z}_q^m : \forall a \in \mathbb{Z}_q \backslash \{0\}, \mathsf{dist}(a\tilde{\mathbf{y}}, \mathbf{\Lambda}_q(\mathbf{A})) \geq \sqrt{q}/4\}$, we have that the rows of $\mathbf{A}'$ are linearly independent over $\mathbb{Z}_q$. In other words, the columns of $\mathbf{A}'$ generate $\mathbb{Z}_q^{n+1}$. Let $\mathbf{x}$ be the shortest vector of $\mathbf{\Lambda}_q(\mathbf{A}')$. Note that the lattice $\mathbf{\Lambda}_q(\mathbf{A}')$ is obtained by adjoining the vector $\mathbf{z}$ to $\mathbf{\Lambda}_q(\mathbf{A})$. Without loss of generality we assume $\mathbf{x} = \mathbf{y} + a\mathbf{z}$ for some $\mathbf{y} \in \mathbf{\Lambda}_q(\mathbf{A})$ and $a \in \mathbb{Z}_q$. Then, if $a = 0$, we have $\|\mathbf{x}\| \geq q/4$ by the fact that $\lambda_1(\mathbf{\Lambda}_q(\mathbf{A})) \geq q/4$. Otherwise, for any $a \in \mathbb{Z}_q \backslash \{0\}$, we have $\|\mathbf{x}\| \geq \mathsf{dist}(a\mathbf{z}, \mathbf{\Lambda}_q(\mathbf{A})) \geq \sqrt{q}/4$. In all, we have that $\lambda_1(\mathbf{\Lambda}_q(\mathbf{A}')) = \|\mathbf{x}\| \geq \sqrt{q}/4$. By Lemma 13 and the duality $\mathbf{\Lambda}_q(\mathbf{A}') = q \cdot (\mathbf{\Lambda}_q^\perp(\mathbf{A}'))^*$, we have $\eta_\epsilon(\mathbf{\Lambda}_q^\perp(\mathbf{A}')) \leq 4\sqrt{mq} \leq \gamma$ for $\epsilon = 2^{-m}$.[10]

Since the columns of $\mathbf{A}' \in \mathbb{Z}_q^{(n+1)\times m}$ generate $\mathbb{Z}_q^{n+1}$, we have the set of syndromes $\{u = \mathbf{z}^t\mathbf{e} : \mathbf{e} \in \mathbf{\Lambda}_q^\perp(\mathbf{A})\} = \mathbb{Z}_q$. By the fact $\mathbf{\Lambda}_q^\perp(\mathbf{A}') = \mathbf{\Lambda}_q^\perp(\mathbf{A}) \cap \mathbf{\Lambda}_q^\perp(\mathbf{z}^t)$, the quotient group $(\mathbf{\Lambda}_q^\perp(\mathbf{A})/\mathbf{\Lambda}_q^\perp(\mathbf{A}'))$ is isomorphic to the set of syndromes $\mathbb{Z}_q$ via the mapping $\mathbf{e} + \mathbf{\Lambda}_q^\perp(\mathbf{A}') \mapsto \mathbf{z}^t\mathbf{e} \mod q$. This means that computing $\mathbf{z}^t\mathbf{e} \mod q$ for some $\mathbf{e} \in \mathbf{\Lambda}_q^\perp(\mathbf{A})$ is equivalent to reducing $\mathbf{e}$ modulo the lattice $\mathbf{\Lambda}_q^\perp(\mathbf{A}')$. By Lemma 14, for any $\epsilon = \text{negl}(n)$, any $\gamma \geq \eta_\epsilon(\mathbf{\Lambda}_q^\perp(\mathbf{A}'))$ and any $\mathbf{v} \in \mathbb{Z}^m$, the distribution of $D_{\mathbf{\Lambda}_q^\perp(\mathbf{A}),\gamma,-\mathbf{v}} \mod \mathbf{\Lambda}_q^\perp(\mathbf{A}')$ is within statistical distance at most 2ϵ of uniform over $(\mathbf{\Lambda}_q^\perp(\mathbf{A})/\mathbf{\Lambda}_q^\perp(\mathbf{A}'))$. Thus, the distribution $\mathbf{z}^t\mathbf{e}$ is statistically close to uniform over $\mathbb{Z}_q$, where $\mathbf{e} \sim D_{\mathbf{\Lambda}_q^\perp(\mathbf{A}),\gamma,-\mathbf{v}}$. This completes the proof. $\square$

[10] It is possible to set a smaller γ by a more careful analysis with $\epsilon = \text{negl}(n)$.

5 Lattice-based Splittable PKE with Associated ASPH

In order to construct a splittable PKE with associated ASPH from lattices, our basic idea is to incorporate the specific algebraic properties of lattices into the Naor-Yung paradigm [51,57], which is a generic construction of CCA-secure PKE scheme from any CPA-secure PKE scheme and simulation-sound non-interactive zero knowledge (NIZK) proof [57], and was used to achieve the first one-round PAKEs from DDH and decisional linear assumptions [41].

Looking ahead, we will use a CPA-secure PKE scheme from lattices and a simulation-sound NIZK proof for specific statements, so that we can freely apply Lemma 18 and Lemma 19 to construct a non-adaptive approximate SPH and achieve the stronger smoothness property. Formally, we need a simulation-sound NIZK proof for the following relation:

$$R_{pke} := \left\{ \begin{array}{l} ((\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta), (\mathbf{s}_0, \mathbf{s}_1, \mathbf{w})) : \\ \qquad\qquad \left\| \mathbf{c}_0 - \mathbf{A}_0^t \begin{pmatrix} \mathbf{s}_0 \\ 1 \\ \mathbf{w} \end{pmatrix} \right\| \le \beta \wedge \left\| \mathbf{c}_1 - \mathbf{A}_1^t \begin{pmatrix} \mathbf{s}_1 \\ 1 \\ \mathbf{w} \end{pmatrix} \right\| \le \beta \end{array} \right\}$$

where $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}, \mathbf{c}_0, \mathbf{c}_1 \in \mathbb{Z}_q^m, \beta \in \mathbb{R}, \mathbf{s}_0, \mathbf{s}_1 \in \mathbb{Z}_q^{n_1}, \mathbf{w} \in \mathbb{Z}_q^{n_2}$ for some integers $n = n_1 + n_2 + 1, m, q \in \mathbb{Z}$. Note that under the existence of (enhanced) trapdoor permutations, there exist NIZK proofs with efficient prover for any NP relation [28,11,33]. Moreover, Sahai [57] showed that one can transform any general NIZK proof into a simulation-sound one. Thus, there exists a simulation-sound NIZK proof with efficient prover for the relation R_{pke}. In Section 5.3, we will also show how to directly construct an efficient one from lattices.

For our purpose, we require that the NIZK proof supports labels [1], which can be obtained from a normal NIZK proof by a standard way (e.g., appending the label to the statement [41,27]). Let $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ be a labeled NIZK proof for relation R_{pke}. The algorithm $\mathsf{CRSGen}(1^\kappa)$ takes a security parameter κ as input, outputs a common reference string crs, i.e., $crs \leftarrow \mathsf{CRSGen}(1^\kappa)$. The algorithm Prove takes a pair $(x, wit) = ((\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta), (\mathbf{s}_0, \mathbf{s}_1, \mathbf{w})) \in R_{pke}$ and a $\mathsf{label} \in \{0,1\}^*$ as inputs, outputs a proof π, i.e., $\pi \leftarrow \mathsf{Prove}(crs, x, wit, \mathsf{label})$. The algorithm Verify takes as inputs x, a proof π and a $\mathsf{label} \in \{0,1\}^*$, outputs a bit $b \in \{0,1\}$ indicating whether π is valid or not, i.e., $b \leftarrow \mathsf{Verify}(crs, x, \pi, \mathsf{label})$. For completeness, we require that for any $(x, wit) \in R_{pke}$ and any $\mathsf{label} \in \{0,1\}^*$, $\mathsf{Verify}(crs, x, \mathsf{Prove}(crs, x, wit, \mathsf{label}), \mathsf{label}) = 1$. We defer more information of simulation-sound NIZK to Appendix A.

5.1 A Splittable PKE from Lattices

Let $n_1, n_2 \in \mathbb{Z}$ and prime q be polynomials in the security parameter κ. Let $n = n_1 + n_2 + 1, m = O(n \log q) \in \mathbb{Z}$, and $\alpha, \beta \in \mathbb{R}$ be the system parameters. Let $\mathcal{P} = \{-\alpha q + 1, \ldots, \alpha q - 1\}^{n_2}$ be the plaintext space. Let $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ be a simulation-sound NIZK proof for R_{pke}. Our PKE scheme $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows.

$\mathsf{KeyGen}(1^\kappa)$**:** Given the security parameter κ, compute $(\mathbf{A}_0, \mathbf{R}_0) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, $(\mathbf{A}_1, \mathbf{R}_1) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ and $crs \leftarrow \mathsf{CRSGen}(1^\kappa)$. Return the public and secret key pair $(\mathsf{pk}, \mathsf{sk}) = ((\mathbf{A}_0, \mathbf{A}_1, crs), \mathbf{R}_0)$.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{label}, \mathbf{w} \in \mathcal{P})$**:** Given $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs), \mathsf{label} \in \{0,1\}^*$ and plaintext $\mathbf{w}$, randomly choose $\mathbf{s}_0, \mathbf{s}_1 \leftarrow_r \mathbb{Z}_q^{n_1}$, $\mathbf{e}_0, \mathbf{e}_1 \leftarrow_r D_{\mathbb{Z}^m, \alpha q}$. Finally, return the ciphertext $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$, where

$$\mathbf{c}_0 = \mathbf{A}_0^t \begin{pmatrix} \mathbf{s}_0 \\ 1 \\ \mathbf{w} \end{pmatrix} + \mathbf{e}_0, \quad \mathbf{c}_1 = \mathbf{A}_1^t \begin{pmatrix} \mathbf{s}_1 \\ 1 \\ \mathbf{w} \end{pmatrix} + \mathbf{e}_1,$$

and $\pi \leftarrow \mathsf{Prove}(crs, (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta), (\mathbf{s}_0, \mathbf{s}_1, \mathbf{w}), \mathsf{label})$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{label}, C)$**:** Given $\mathsf{sk} = \mathbf{R}_0$, $\mathsf{label} \in \{0,1\}^*$ and ciphertext $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$, if $\mathsf{Verify}(crs, (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta), \pi, \mathsf{label}) = 0$, return $\perp$. Otherwise, compute

$$\mathbf{t} = \begin{pmatrix} \mathbf{s}_0 \\ 1 \\ \mathbf{w} \end{pmatrix} \leftarrow \mathsf{Solve}(\mathbf{A}_0, \mathbf{R}_0, \mathbf{c}_0),$$

and return $\mathbf{w} \in \mathbb{Z}_q^{n_2}$ (note that a valid π ensures that $\mathbf{t}$ has the right form).

Correctness. By Lemma 12, we have that $\|\mathbf{e}_0\|, \|\mathbf{e}_1\| \leq \alpha q \sqrt{m}$ hold with overwhelming probability. Thus, it is enough to set $\beta \geq \alpha q \sqrt{m}$ for the NIZK proof to work. By Proposition 1, we have that $s_1(\mathbf{R}_0) \leq \sqrt{m} \cdot \omega(\sqrt{\log n})$, and the Solve algorithm can recover $\mathbf{t}$ from any $\mathbf{y} = \mathbf{A}_0^t \mathbf{t} + \mathbf{e}_0$ as long as $\|\mathbf{e}_0\| \cdot \sqrt{m} \cdot \omega(\sqrt{\log n}) \leq q$. Thus, we can set the parameters appropriately to satisfy the correctness. Besides, for the hardness of the LWE assumption, we need $\alpha q \geq 2\sqrt{n_1}$. In order to obtain an ϵ-approximate SPH function, we require $\beta \leq \sqrt{q}/4$, $\sqrt{mq}/4 \cdot \omega(\sqrt{\log n}) \leq q$ and $\alpha \gamma m < \epsilon/8$, where $\gamma \geq 4\sqrt{mq}$ is the parameter for ASPH in Section 5.2. In all, fix $\epsilon \in (0, 1/2)$, we can set the parameters $m, \alpha, \beta, q, \gamma$ as follows (where $c \geq 0$ is a real such that q is a prime) for both correctness and security:

$$\begin{array}{ll} m = O(n \log n), & \beta > 16m\sqrt{mn}/\epsilon \\ q = 16\beta^2 + c & \alpha = 2\sqrt{n}/q \\ \gamma = 4\sqrt{mq} & \end{array} \tag{1}$$

In practice, given a target length of session keys, one can first choose an appropriate ECC scheme, and then set other parameters to satisfy Equation (1). For example, the Reed-Muller code with $\ell = 1024$ can be used to encode a 176-bit session key with $\epsilon = 1/32$, and thus is far enough to establish a 128-bit session key. In the setting of $\mathcal{P} = \{-\alpha q + 1, \ldots, \alpha q - 1\}^7$ (i.e., $n_2 = 7$), one can set $n_1 \approx 2^{11}, m \approx 2^{19}, \alpha \approx 2^{-83.5}, \beta \approx 2^{43}$ and $q \approx 2^{90}$, which provides about 105-bit security by the lwe-estimator [6]. We note that there are many tradeoffs between the parameters, and it is possible to give a more tight parameter for any targeted security level. One can also reduce the parameters by using a careful proof of Lemma 19 with smaller γ.

Security. For any $C = (\mathbf{c}_0, \mathbf{c}_1, \pi) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{label}, \mathbf{w})$, let r be the corresponding random coins which includes $(\mathbf{s}_0, \mathbf{s}_1, \mathbf{e}_0, \mathbf{e}_1)$ for generating $(\mathbf{c}_0, \mathbf{c}_1)$, and the randomness used for generating π. We define functions (f, g) as follows:

- The function f takes $(\mathsf{pk}, \mathbf{w}, r)$ as inputs, computes $(\mathbf{c}_0, \mathbf{c}_1)$ with random coins r, and returns $(\mathbf{c}_0, \mathbf{c}_1)$, i.e., $(\mathbf{c}_0, \mathbf{c}_1) = f(\mathsf{pk}, \mathbf{w}, r)$;
- The function g takes $(\mathsf{pk}, \mathsf{label}, \mathbf{w}, r)$ as inputs, computes the Prove algorithm with random coins r and returns the result π, i.e., $\pi = g(\mathsf{pk}, \mathsf{label}, \mathbf{w}, r)$.

We fix the two functions (f, g) in the rest of Section 5, and have the following theorem for security.

Theorem 2. *Let $n = n_1 + n_2 + 1, m \in \mathbb{Z}, \alpha, \beta, \gamma \in \mathbb{R}$ and prime q be as in Equation (1). If $\mathrm{LWE}_{n_1,q,\alpha}$ is hard, $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ is a simulation-sound NIZK proof, then the scheme $\mathcal{PKE}$ is a splittable CCA-secure PKE scheme.*

Since $\mathcal{PKE}$ is essentially an instantiation of the Naor-Yung paradigm [51,57] using a special LWE-based CPA scheme (similar to the ones in [40,49]), and a SS-NIZK for a special relation R_{pke}, this theorem can be shown by adapting the proof techniques in [51,57]. We deter the proof to Appendix B.

5.2 An Associated Approximate SPH

Fix a public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$ of the PKE scheme $\mathcal{PKE}$. Given any string $\mathsf{label} \in \{0,1\}^*$ and $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$, we say that (label, C) is a valid label-ciphertext pair with respect to pk if $\mathsf{Verify}(crs, (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta), \pi, \mathsf{label}) = 1$. Let sets X, L and $\bar{L}$ be defined as in Section 3.2. Define the associated ASPH function $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk} \in K}, S, \mathsf{Proj} : K \to S)$ for $\mathcal{PKE}$ as follows.

- The hash key is an ℓ-tuple of vectors $\mathsf{hk} = (\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$, where $\mathbf{x}_i \sim D_{\mathbb{Z}^m, \gamma}$. Write $\mathbf{A}_0^t = (\mathbf{B} \| \mathbf{U}) \in \mathbb{Z}_q^{m \times n}$ such that $\mathbf{B} \in \mathbb{Z}_q^{m \times n_1}$ and $\mathbf{U} \in \mathbb{Z}_q^{m \times (n_2+1)}$. Define the projection key $\mathsf{hp} = \mathsf{Proj}(\mathsf{hk}) = (\mathbf{u}_1, \ldots, \mathbf{u}_\ell)$, where $\mathbf{u}_i = \mathbf{B}^t \mathbf{x}_i$.
- $\mathrm{H}_{\mathsf{hk}}(x) = \mathrm{H}_{\mathsf{hk}}((\mathbf{c}_0, \mathbf{c}_1), \mathbf{w})$: Given $\mathsf{hk} = (\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$ and $x = (\mathsf{label}, C, \mathbf{w}) \in X$ for some $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$, compute $z_i = \mathbf{x}_i^t \left(\mathbf{c}_0 - \mathbf{U} \begin{pmatrix} 1 \\ \mathbf{w} \end{pmatrix} \right)$ for $i \in \{1, \ldots, \ell\}$. Then, treat each z_i as a number in $\{-(q-1)/2, \ldots, (q-1)/2\}$. If $z_i = 0$, then set $b_i \leftarrow_r \{0,1\}$. Else, set
$$b_i = \begin{cases} 0 \text{ if } z_i < 0 \\ 1 \text{ if } z_i > 0 \end{cases}.$$
Finally, return $\mathrm{H}_{\mathsf{hk}}((\mathbf{c}_0, \mathbf{c}_1), \mathbf{w}) = (b_1, \ldots, b_\ell)$.
- $\mathsf{Hash}(\mathsf{hp}, x, \mathbf{s}_0) = \mathsf{Hash}(\mathsf{hp}, ((\mathbf{c}_0, \mathbf{c}_1), \mathbf{w}), \mathbf{s}_0)$: Given $\mathsf{hp} = (\mathbf{u}_1, \ldots, \mathbf{u}_\ell), x = (\mathsf{label}, (\mathbf{c}_0, \mathbf{c}_1, \pi), \mathbf{w}) \in L$ and $\mathbf{s}_0 \in \mathbb{Z}_q^{n_1}$ such that $\mathbf{c}_0 = \mathbf{B}\mathbf{s}_0 + \mathbf{U} \begin{pmatrix} 1 \\ \mathbf{w} \end{pmatrix} + \mathbf{e}_0$ for some $\mathbf{e}_0 \leftarrow_r D_{\mathbb{Z}^m, \alpha q}$, compute $z_i' = \mathbf{u}_i^t \mathbf{s}_0$. Then, treat each z_i' as a number in $\{-(q-1)/2, \ldots, (q-1)/2\}$. If $z_i' = 0$, then set $b_i' \leftarrow_r \{0,1\}$. Else, set
$$b_i' = \begin{cases} 0 \text{ if } z_i < 0 \\ 1 \text{ if } z_i' > 0 \end{cases}.$$

Finally, return $\mathsf{Hash}(\mathsf{hp}, ((\mathbf{c}_0, \mathbf{c}_1), \mathbf{w}), \mathbf{s}_0) = (b'_1, \ldots, b'_\ell)$.

Theorem 3. *Let $\epsilon \in (0, 1/2)$, and let $n, m, q, \alpha, \beta, \gamma$ be as in Theorem 2. Let ℓ be polynomial in the security parameter κ. Then, $(K, \ell, \{\mathrm{H}_{\mathsf{hk}} : X \to \{0,1\}^\ell\}_{\mathsf{hk} \in K}, S, \mathsf{Proj} : K \times C_{\mathsf{pk}} \to S)$ is an ϵ-approximate SPH as in Definition 5.*

Proof. Clearly, there are efficient algorithms for (1) sampling a hash key $\mathsf{hk} \leftarrow_r K$, (2) computing $\mathrm{H}_{\mathsf{hk}}((\mathbf{c}_0, \mathbf{c}_1), \mathbf{w})$ for all $\mathsf{hk} \in K$ and all $x = (\mathsf{label}, C, \mathbf{w}) \in X$ with $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$, and (3) computing $\mathsf{hp} = \mathsf{Proj}(\mathsf{hk})$ for all $\mathsf{hk} \in K$. In addition, for any $x = (\mathsf{label}, C, \mathbf{w}) \in L$, the values $\mathsf{Hash}(\mathsf{hp}, ((\mathbf{c}_0, \mathbf{c}_1), \mathbf{w}), \mathbf{s}_0)$ can be efficiently computed, where $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$ and $\mathbf{c}_0$ is generated using the randomness $\mathbf{s}_0$. In the following, we show that the above construction also satisfies the approximate correctness and the smoothness given in Definition 5.

First, let $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$ be a ciphertext such that $\mathbf{c}_0 = \mathbf{B}\mathbf{s}_0 + \mathbf{U}\begin{pmatrix}1\\ \mathbf{w}\end{pmatrix} + \mathbf{e}_0$ for some $\mathbf{s}_0 \leftarrow_r \mathbb{Z}_q^{n_1}$ and $\mathbf{e}_0 \leftarrow_r D_{\mathbb{Z}^m, \alpha q}$. For any $i \in \{1, \ldots, \ell\}$, we have that $z_i = \mathbf{x}_i^t\left(\mathbf{c}_0 - \mathbf{U}\begin{pmatrix}1\\ \mathbf{w}\end{pmatrix}\right) = \mathbf{x}_i^t(\mathbf{B}\mathbf{s}_0 + \mathbf{e}_0) = \mathbf{u}_i^t\mathbf{s}_0 + \mathbf{x}_i^t\mathbf{e}_0$. This means that $|z_i - z'_i| \leq |\mathbf{x}_i^t\mathbf{e}_0| \leq \gamma\sqrt{m} \cdot \alpha q\sqrt{m} < \epsilon/2 \cdot q/4$ with overwhelming probability. Using the fact that $\mathbf{B} \in \mathbb{Z}_q^{m \times n_1}$ is statistically close to uniform, we have that $\mathbf{u}_i = \mathbf{B}^t\mathbf{x}_i$ is statistically close to uniform over $\mathbb{Z}_q^{n_1}$ for all $i \in \{1, \ldots, \ell\}$ by Lemma 16. Moreover, for any non-zero $\mathbf{s}_0 \in \mathbb{Z}_q^n$ (note that the probability that $\mathbf{s}_0 = 0$ is at most q^{-n_1}, which is negligible in κ), we have that $z'_i = \mathbf{u}_i^t\mathbf{s}_0$ is uniformly random. By a simple calculation, we have the probability that $b_i \neq b'_i$ is at most $\frac{\epsilon}{2}$. By a Chernoff bound, the Hamming distance between $\mathrm{H}_{\mathsf{hk}}((\mathbf{c}_0, \mathbf{c}_1), \mathbf{w}) = (b_1, \ldots, b_\ell)$ and $\mathsf{Hash}(\mathsf{hp}, ((\mathbf{c}_0, \mathbf{c}_1), \mathbf{w}), \mathbf{s}_0) = (b'_1, \ldots, b'_\ell)$ is at most $\epsilon\ell$ with overwhelming probability. This shows the approximate correctness.

Second, for any $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$ and $((\mathsf{label}, C), \mathbf{w}) \in X \backslash \bar{L}$, let $\mathbf{w}'$ be the decryption result of (label, C) using the secret key sk corresponding to pk (note that the validity of π ensures that the existence of $\mathbf{w}' \neq \perp$). By assumption, we know that $\mathbf{w}' \neq \mathbf{w}$. Let $\mathbf{y} = \mathbf{c}_0 - \mathbf{U}\begin{pmatrix}1\\ \mathbf{w}\end{pmatrix} \in \mathbb{Z}_q^m$ and $\mathbf{y}' = \mathbf{c}_0 - \mathbf{U}\begin{pmatrix}1\\ \mathbf{w}'\end{pmatrix}$, we have $\mathsf{dist}(\mathbf{y}', \mathbf{\Lambda}_q(\mathbf{B}^t)) \leq \beta \leq \frac{\sqrt{q}}{4}$ by the soundness of the NIZK proof π. Note that the matrix $\mathbf{U}$ is statistically close to uniform by Proposition 1. Hence, with overwhelming probability we always have that

$$\mathbf{y} = \mathbf{c}_0 - \mathbf{U}\begin{pmatrix}1\\ \mathbf{w}\end{pmatrix} \in Y = \left\{\tilde{\mathbf{y}} \in \mathbb{Z}_q^m : \forall a \in \mathbb{Z}_q \backslash \{0\}, \mathsf{dist}(a\tilde{\mathbf{y}}, \mathbf{\Lambda}_q(\mathbf{B}^t)) \geq \sqrt{q}/4\right\}$$

for any $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$ and $((\mathsf{label}, C), \mathbf{w}) \in X \backslash \bar{L}$ by Lemma 18. In addition, if $z_i = \mathbf{x}_i^t\mathbf{y}$ is uniformly random over $\mathbb{Z}_q$, then by the definition the i-th bit b_i of $\mathrm{H}_{\mathsf{hk}}((\mathbf{c}_0, \mathbf{c}_1), \mathbf{w})$ is uniformly random over $\{0,1\}$. Thus, for smoothness, it suffices to show that for any (even unbounded) function $h : \mathbb{Z}_q^{n_1 \times \ell} \to Y, \mathsf{hk} = (\mathbf{x}_1, \ldots, \mathbf{x}_\ell) \leftarrow_r (D_{\mathbb{Z}^m, \gamma})^\ell, \mathsf{hp} = (\mathbf{B}^t\mathbf{x}_1, \ldots, \mathbf{B}^t\mathbf{x}_\ell) = \mathsf{Proj}(\mathsf{hk}), \mathbf{y} = h(\mathsf{hp}), \mathbf{z} = (\mathbf{x}_1^t\mathbf{y}, \ldots, \mathbf{x}_\ell^t\mathbf{y})$ and $\mathbf{z}' \leftarrow_r \mathbb{Z}_q^\ell$, the statistical distance between $(\mathsf{hp}, \mathbf{z})$ and $(\mathsf{hp}, \mathbf{z}')$

is negligible in κ. Since $\gamma \geq 4\sqrt{mq}$ and $\mathbf{B} \in \mathbb{Z}_q^{m\times n_1}$ is statistically close to uniform, by Lemma 19 we have that for any function $h' : \mathbb{Z}_q^{n_1} \to Y$, the distribution of $(\mathbf{B}^t\mathbf{x}, \mathbf{x}^t\mathbf{y}')$ is statistically close to uniform over $\mathbb{Z}_q^{n_1} \times \mathbb{Z}_q$, where $\mathbf{x} \sim D_{\mathbb{Z}^m,\gamma}$ and $\mathbf{y}' = h'(\mathbf{B}^t\mathbf{x})$. Using the facts that Lemma 19 holds for arbitrary choice of $h' : \mathbb{Z}_q^{n_1} \to Y$ and that each $\mathbf{x}_i$ is independently chosen from $D_{\mathbb{Z}^m,\gamma}$, we have that $(\mathsf{hp}, \mathbf{z}) = ((\mathbf{B}^t\mathbf{x}_1, \ldots, \mathbf{B}^t\mathbf{x}_\ell), (\mathbf{x}_1^t\mathbf{y}, \ldots, \mathbf{x}_\ell^t\mathbf{y}))$ is statistically close to uniform by a standard hybrid argument. This completes the proof Theorem 3. $\square$

5.3 Achieving Simulation-Sound NIZK for R_{pke} on Lattices

In this section, we will show how to construct an simulation-sound NIZK for R_{pke} from lattices in the random oracle model. Formally, let $n = n_1 + n_2 + 1, m, q \in \mathbb{Z}$ be defined as in Section 5.1. We begin by defining a variant relation R'_{pke} of R_{pke} (in the l_∞ form):

$$R'_{pke} := \left\{ \begin{array}{l} ((\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \zeta), (\mathbf{s}_0, \mathbf{s}_1, \mathbf{w})) : \|\mathbf{w}\|_\infty \leq \zeta \wedge \\ \qquad \left\| \mathbf{c}_0 - \mathbf{A}_0^t \begin{pmatrix} \mathbf{s}_0 \\ 1 \\ \mathbf{w} \end{pmatrix} \right\|_\infty \leq \zeta \wedge \left\| \mathbf{c}_1 - \mathbf{A}_1^t \begin{pmatrix} \mathbf{s}_1 \\ 1 \\ \mathbf{w} \end{pmatrix} \right\|_\infty \leq \zeta \end{array} \right\},$$

where $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n\times m}, \mathbf{c}_0, \mathbf{c}_1 \in \mathbb{Z}_q^m, \zeta \in \mathbb{R}, \mathbf{s}_0, \mathbf{s}_1 \in \mathbb{Z}_q^{n_1}$ and $\mathbf{w} \in \mathbb{Z}_q^{n_2}$. Write $\mathbf{A}_0^t = (\mathbf{B}_0 \| \mathbf{U}_0) \in \mathbb{Z}_q^{m\times n_1} \times \mathbb{Z}_q^{m\times(n_2+1)}$. Note that for large enough $m = O(n_1 \log q)$, the rows of a uniformly random $\mathbf{B}_0 \in \mathbb{Z}_q^{m\times n_1}$ generate $\mathbb{Z}_q^{n_1}$ with overwhelming probability. By the duality [48], one can compute a parity check matrix $\mathbf{G}_0 \in \mathbb{Z}_q^{(m-n_1)\times m}$ such that 1) the columns of $\mathbf{G}_0$ generate $\mathbb{Z}_q^{m-n_1}$, and 2) $\mathbf{G}_0\mathbf{B}_0 = \mathbf{0}$. Now, let vector $\mathbf{e}_0 \in \mathbb{Z}^m$ satisfy

$$\mathbf{c}_0 = \mathbf{A}_0^t \begin{pmatrix} \mathbf{s}_0 \\ 1 \\ \mathbf{w} \end{pmatrix} + \mathbf{e}_0 = \mathbf{B}_0\mathbf{s}_0 + \mathbf{U}_0 \begin{pmatrix} 1 \\ \mathbf{w} \end{pmatrix} + \mathbf{e}_0. \qquad (2)$$

By multiplying Equation (2) with matrix $\mathbf{G}_0$ and rearranging the terms, we have the equation $\mathbf{D}_0\mathbf{w} + \mathbf{G}_0\mathbf{e}_0 = \mathbf{b}_0$, where $(\mathbf{a}_0 \| \mathbf{D}_0) = \mathbf{G}_0\mathbf{U}_0 \in \mathbb{Z}_q^{(m-n_1)\times(1+n_2)}$, and $\mathbf{b}_0 = \mathbf{G}_0\mathbf{c}_0 - \mathbf{a}_0 \in \mathbb{Z}_q^{m-n_1}$. Similarly, by letting $\mathbf{A}_1^t = (\mathbf{B}_1 \| \mathbf{U}_1)$ and $\mathbf{c}_1 = \mathbf{B}_1\mathbf{s}_1 + \mathbf{U}_1 \begin{pmatrix} 1 \\ \mathbf{w} \end{pmatrix} + \mathbf{e}_1$, we can compute an equation $\mathbf{D}_1\mathbf{w} + \mathbf{G}_1\mathbf{e}_1 = \mathbf{b}_1$, where $\mathbf{G}_1 \in \mathbb{Z}_q^{(m-n_1)\times m}$ is a parity check matrix for $\mathbf{B}_1$, $(\mathbf{a}_1 \| \mathbf{D}_1) = \mathbf{G}_1\mathbf{U}_1 \in \mathbb{Z}_q^{(m-n_1)\times(1+n_2)}$, and $\mathbf{b}_1 = \mathbf{G}_1\mathbf{c}_1 - \mathbf{a}_1 \in \mathbb{Z}_q^{m-n_1}$. As in [46,42], in order to show $((\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \zeta), (\mathbf{s}_0, \mathbf{s}_1, \mathbf{w})) \in \mathbf{R}'_{pke}$, it is enough to prove that there exists $(\mathbf{w}, \mathbf{e}_0, \mathbf{e}_1)$ such that $((\mathbf{D}_0, \mathbf{G}_0, \mathbf{D}_1, \mathbf{G}_1, \mathbf{b}_0, \mathbf{b}_1, \zeta), (\mathbf{w}, \mathbf{e}_0, \mathbf{e}_1)) \in \tilde{R}'_{pke}$:

$$\tilde{R}'_{pke} := \left\{ \begin{array}{l} ((\mathbf{D}_0, \mathbf{G}_0, \mathbf{D}_1, \mathbf{G}_1, \mathbf{b}_0, \mathbf{b}_1, \zeta), (\mathbf{w}, \mathbf{e}_0, \mathbf{e}_1)) : \\ \qquad \begin{pmatrix} \mathbf{D}_0 & \mathbf{G}_0 & \mathbf{0} \\ \mathbf{D}_1 & \mathbf{0} & \mathbf{G}_1 \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{e}_0 \\ \mathbf{e}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \end{pmatrix} \wedge \\ \qquad \|\mathbf{w}\|_\infty \leq \zeta \wedge \|\mathbf{e}_0\|_\infty \leq \zeta \wedge \|\mathbf{e}_1\|_\infty \leq \zeta \end{array} \right\},$$

which is essentially a special case of the ISIS relation R_{ISIS} (in the l_∞ norm):

$$R_{ISIS} := \{((\mathbf{M}, \mathbf{b}, \zeta), \mathbf{x}) : \mathbf{M}\mathbf{x} = \mathbf{b} \wedge \|\mathbf{x}\|_\infty \leq \zeta\}\,.$$

Notice that if there is a three-round public-coin honest-verifier zero-knowledge (HVZK) proof for the relation R_{ISIS}, one can obtain an NIZK proof for R_{ISIS} by applying the Fiat-Shamir transform [29] in the random oracle model [16]. Moreover, if the basic protocol additionally has the *quasi unique responses* property [30,26,14], the literature [15,26,14] shows that the resulting NIZK proof derived from the Fiat-Shamir transform meets the simulation-soundness needed for constructing CCA-secure PKE via the Naor-Yung paradigm [51,57]. Fortunately, we do have an efficient three-round public-coin HVZK proof with quasi unique responses in [44],[11] which is extended from the Stern protocol [58] and has the same structure as the latter. Specifically, the protocol [44] has three messages (a, e, z), where a consists of several commitments sent by the prover, e is the challenge sent by the verifier, and the third message z (i.e., the response) consists of the openings to the commitments specified by the challenge e.

Note that the quasi unique responses property [30,26] essentially requires that it is computationally infeasible for an adversary to output (a, e, z) and (a, e, z') such that both (a, e, z) and (a, e, z') are valid. Thus, if, as is usually the case, the parameters of the commitment scheme are priorly fixed for all users, the protocol in [44] naturally has the quasi unique responses property by the binding property of the commitment scheme. In other words, the NIZK proof for R_{ISIS} [43,45] (and thus for $\tilde{R}'_{pke}$) obtained by applying the Fiat-Shamir transform to the protocol in [44] suffices for our PKE scheme (where labels can be incorporated into the input of the hash function used for the transformation).

Finally, we clarify that the protocol [44] is designed for R_{ISIS} in the l_∞ norm, while the l_2 norm is used in Section 5.1. This problem can be easily fixed by setting $\zeta = \alpha q \cdot \omega(\sqrt{\log n})$ in the NIZK proof, and setting the parameter β in Equation (1) such that $\beta \geq 2\zeta\sqrt{n}$ holds, since 1) for $\mathbf{e}_0, \mathbf{e}_1 \leftarrow_r D_{\mathbb{Z}^m,\alpha q}$, both $\Pr[\|\mathbf{e}_0\|_\infty \geq \zeta], \Pr[\|\mathbf{e}_1\|_\infty \geq \zeta]$ are negligible in n by [32, Lemma. 4.2]; and 2) $\mathcal{P} = \{-\alpha q + 1, \ldots, \alpha q - 1\}^{n_2}$ in our PKE scheme $\mathcal{PKE}$. By [44], the resulting NIZK can be achieved with total communication cost $\log_2 \beta \cdot \widetilde{O}(m \log q)$.

Acknowledgments. We would like to thank the anonymous reviewers for their helpful suggestions. Jiang Zhang is supported by the National Key Research and Development Program of China (Grant No. 2017YFB0802005), the National Grand Fundamental Research Program of China (Grant No. 2013CB338003), the National Natural Science Foundation of China (Grant Nos. 61602046, 61602045), and the Young Elite Scientists Sponsorship Program by CAST (2016QNRC001). Yu Yu is supported by the National Natural Science Foundation of China (Grant Nos. 61472249, 61572192, 61572149), the National Cryptography Development

[11] More precisely, the authors [44] showed a zero-knowledge proof of knowledge for R_{ISIS}, which has a constant soundness error about 2/3. By repeating the basic protocol $t = \omega(\log n)$ times in parallel, we can obtain a desired public-coin HVZK proof for R_{ISIS} with negligible soundness error [43,45].

Fund MMJJ20170209, and International Science & Technology Cooperation & Exchange Projects of Shaanxi Province (2016KW-038).

References

1. Abdalla, M., Benhamouda, F., MacKenzie, P.: Security of the J-PAKE password-authenticated key exchange protocol. In: IEEE S&P 2015. pp. 571–587 (May 2015)
2. Abdalla, M., Benhamouda, F., Pointcheval, D.: Disjunctions for hash proof systems: New constructions and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, LNCS, vol. 9057, pp. 69–100. Springer, Heidelberg (2015)
3. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth projective hashing for conditionally extractable commitments. In: Halevi, S. (ed.) CRYPTO 2009, LNCS, vol. 5677, pp. 671–689. Springer, Heidelberg (2009)
4. Abe, M., Cui, Y., Imai, H., Kiltz, E.: Efficient hybrid encryption from ID-based encryption. Designs, Codes and Cryptography 54(3), 205–240 (2010)
5. Ajtai, M.: Generating hard instances of the short basis problem. In: Wiedermann, J., van Emde Boas, P., Nielsen, M. (eds.) Automata, Languages and Programming, LNCS, vol. 1644, pp. 706–706. Springer, Heidelberg (1999)
6. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. In Journal of Mathematical Cryptology 9, 169–203 (oct 2015)
7. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange—a new hope. In: USENIX Security 2016. pp. 327–343. USENIX Association (2016)
8. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In STACS pp. 75–86 (2009)
9. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) CRYPTO 2005, LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005)
10. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000, LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
11. Bellare, M., Yung, M.: Certifying cryptographic tools: The case of trapdoor permutations. In: Brickell, E.F. (ed.) CRYPTO '92, LNCS, vol. 740, pp. 442–460. Springer, Heidelberg (1992)
12. Bellovin, S.M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy. pp. 72–84 (May 1992)
13. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, LNCS, vol. 8042, pp. 449–475. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
14. Bernhard, D., Fischlin, M., Warinschi, B.: Adaptive proofs of knowledge in the random oracle model. In: Katz, J. (ed.) PKC 2015, LNCS, vol. 9020, pp. 629–649. Springer, Heidelberg (2015)
15. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012, LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (2012)
16. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: STOC '88. pp. 103–112. ACM (1988)

17. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000, LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
18. Bresson, E., Chevassut, O., Pointcheval, D.: Security proofs for an efficient password-based key exchange. In: CCS 2003. pp. 241–250. ACM (2003)
19. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS '01. pp. 136–145. IEEE (2001)
20. Canetti, R., Goyal, V., Jain, A.: Concurrent secure computation with optimal query complexity. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, LNCS, vol. 9216, pp. 43–62. Springer, Heidelberg (2015)
21. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004, LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
22. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005, LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
23. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002, LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
24. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001, LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
25. Diffie, W., Hellman, M.: New directions in cryptography. Information Theory, IEEE Transactions on 22(6), 644 – 654 (nov 1976)
26. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012, LNCS, vol. 7668, pp. 60–79. Springer, Heidelberg (2012)
27. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous non-malleable codes. In: Lindell, Y. (ed.) TCC 2014, LNCS, vol. 8349, pp. 465–488. Springer, Heidelberg (2014)
28. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string. In: FOCS '90. pp. 308–317. IEEE (1990)
29. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A. (ed.) CRYPTO '86, LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
30. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005, LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)
31. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003, LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)
32. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC 2008. pp. 197–206. ACM (2008)
33. Goldreich, O.: Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. In: Goldreich, O. (ed.) Studies in Complexity and Cryptography, LNCS, vol. 6550, pp. 406–421. Springer, Heidelberg (2011)
34. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) CRYPTO 2001, LNCS, vol. 2139, pp. 408–432. Springer, Heidelberg (2001)

35. Gong, L., Lomas, M.A., Needham, R.M., Saltzer, J.H.: Protecting poorly chosen secrets from guessing attacks. IEEE Journal on Selected Areas in Communications 11(5), 648–656 (Jun 1993)
36. Goyal, V., Jain, A., Ostrovsky, R.: Password-authenticated session-key generation on the internet in the plain model. In: Rabin, T. (ed.) CRYPTO 2010, LNCS, vol. 6223, pp. 277–294. Springer, Heidelberg (2010)
37. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: CCS 2010. pp. 516–525. ACM (2010)
38. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. ACM Trans. Inf. Syst. Secur. 2(3), 230–268 (Aug 1999)
39. Katz, J., Ostrovsky, R., Yung, M.: Efficient and secure authenticated key exchange using weak passwords. J. ACM 57(1), 3:1–3:39 (Nov 2009)
40. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Matsui, M. (ed.) ASIACRYPT 2009, LNCS, vol. 5912, pp. 636–652. Springer, Heidelberg (2009)
41. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011, LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (2011)
42. Laguillaumie, F., Langlois, A., Libert, B., Stehlé, D.: Lattice-based group signatures with logarithmic signature size. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, LNCS, vol. 8270, pp. 41–61. Springer, Heidelberg (2013)
43. Libert, B., Mouhartem, F., Nguyen, K.: A lattice-based group signature scheme with message-dependent opening. In: Manulis, M., Sadeghi, A.R., Schneider, S. (eds.) ACNS 2016. pp. 137–155. Springer International Publishing (2016)
44. Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013, LNCS, vol. 7778, pp. 107–124. Springer, Heidelberg (2013)
45. Ling, S., Nguyen, K., Wang, H.: Group signatures from lattices: Simpler, tighter, shorter, ring-based. In: Katz, J. (ed.) PKC 2015, LNCS, vol. 9020, pp. 427–449. Springer, Heidelberg (2015)
46. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012, LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012)
47. MacKenzie, P., Patel, S., Swaminathan, R.: Password-authenticated key exchange based on RSA. In: Okamoto, T. (ed.) ASIACRYPT 2000, LNCS, vol. 1976, pp. 599–613. Springer, Heidelberg (2000)
48. Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) CRYPTO 2011, LNCS, vol. 6841, pp. 465–484. Springer, Heidelberg (2011)
49. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012, LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012)
50. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. 37, 267–302 (April 2007)
51. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC '90. pp. 427–437. ACM (1990)
52. Peikert, C.: Lattice cryptography for the internet. In: Mosca, M. (ed.) Post-Quantum Cryptography, LNCS, vol. 8772, pp. 197–219. Springer International Publishing (2014)

53. Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006, LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006)
54. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008, LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
55. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005. pp. 84–93. ACM (2005)
56. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. In: Reingold, O. (ed.) Theory of Cryptography, LNCS, vol. 5444, pp. 419–436. Springer, Heidelberg (2009)
57. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS '99. pp. 543–553. IEEE Computer Society (1999)
58. Stern, J.: A new paradigm for public key identification. IEEE Transactions on Information Theory 42(6), 1757–1768 (Nov 1996)
59. Zhang, J., Chen, Y., Zhang, Z.: Programmable hash functions from lattices: Short signatures and ibes with small key sizes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, LNCS, vol. 9816, pp. 303–332. Springer, Heidelberg (2016)
60. Zhang, J., Zhang, Z., Ding, J., Snook, M., Dagdelen, Ö.: Authenticated key exchange from ideal lattices. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, LNCS, vol. 9057, pp. 719–751. Springer, Heidelberg (2015)
61. Zhang, R.: Tweaking TBE/IBE to PKE transforms with chameleon hash functions. In: Katz, J., Yung, M. (eds.) Applied Cryptography and Network Security, LNCS, vol. 4521, pp. 323–339. Springer, Heidelberg (2007)

A Simulation-Sound Non-Interactive Zero-Knowledge

In this section, we recall the definitions of (labeled) NIZK proof, and simulation-sound NIZK from [16,57,33,41].

Definition 6 (NIZK). *An efficient (labeled) NIZK proof for an NP language L with witness relation R is a tuple of probability polynomial time (PPT) algorithms ($\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify}, \mathcal{S}_1, \mathcal{S}_2$) such that the following holds:*

- ***Completeness:*** *For all n, all $x \in L \cap \{0,1\}^n$, all wit such that $(x, wit) \in R$, all $\mathsf{label} \in \{0,1\}^*$ and all strings $crs \leftarrow \mathsf{CRSGen}(1^n)$, it holds that* $\mathsf{Verify}(crs, x, \mathsf{Prove}(crs, x, wit, \mathsf{label}), \mathsf{label}) = 1$.
- ***Soundness:*** *For all (even unbounded) adversaries $\mathcal{A}$, the following is negligible in n:*

$$\Pr[crs \leftarrow \mathsf{CRSGen}(1^n); (x, \mathsf{label}, \pi) \leftarrow \mathcal{A}(1^n, crs) : \mathsf{Verify}(crs, x, \pi, \mathsf{label}) = 1 \wedge x \notin L]$$

- ***Adaptive Zero Knowledge:*** *For all PPT adversaries $\mathcal{A}$, the following is negligible in n:*

$$\left| \Pr[crs \leftarrow \mathsf{CRSGen}(1^n) : \mathcal{A}^{\mathsf{Prove}(crs,\cdot,\cdot,\cdot)}(1^n, crs) = 1] - \Pr[(crs, \tau) \leftarrow \mathcal{S}_1(1^n) : \mathcal{A}^{\mathcal{S}'(crs,\tau,\cdot,\cdot,\cdot)}(1^n, crs) = 1] \right|,$$

where $\mathcal{S}'(crs, \tau, \cdot, \cdot, \cdot)$ *is defined as:*

$$\mathcal{S}'(crs, \tau, x, wit, \mathsf{label}) = \begin{cases} \mathcal{S}_2(crs, \tau, x, \mathsf{label}), & \text{if } (x, wit) \in R \wedge x \in \{0,1\}^n; \\ \perp & , \text{otherwise.} \end{cases}$$

Definition 7 (Simulation-soundness). *A labeled NIZK proof (CRSGen, Prove, Verify,* $\mathcal{S}_1, \mathcal{S}_2$*) for a NP language* L *with witness relation* R *is simulation-sound if for all PPT adversaries* $\mathcal{A}$*, it holds that* $\Pr[\mathbf{Exp}_{\mathcal{A}}(n) = 1]$ *is negligible in* n*, where* $\Pr[\mathbf{Exp}_{\mathcal{A}}(n)$ *denotes the following experiment:*

$(crs, \tau) \leftarrow \mathcal{S}_1(1^n)$
$(x, \mathsf{label}, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_2(crs,\tau,\cdot,\cdot)}(1^n, crs)$
Let Q *be the set of query-responses for* $\mathcal{S}_2(crs, \tau, \cdot, \cdot)$*, above*
Return 1 iff $((x, \mathsf{label}), \pi) \notin Q$ *and* $x \notin L$ *and* $\mathsf{Verify}(crs, x, \pi, \mathsf{label}) = 1$.

Moreover, the NIZK proof is one-time simulation-sound if the adversary $\mathcal{A}$ *is only allowed to submit a single query to* $\mathcal{S}_2(crs, \tau, \cdot, \cdot)$ *in the above experiment.*

Note that the notion of one-time simulation-soundness is enough for our construction of CCA-secure PKE. Besides, assuming the existence of doubly enhanced trapdoor permutations, every NP language has a simulation-sound NIZK proof [24].

B The Proof of Theorem 2

For convenience, we first restate Theorem 2 in the following.

Theorem 2. *Let* $n = n_1 + n_2 + 1, m \in \mathbb{Z}, \alpha, \beta, \gamma \in \mathbb{R}$ *and prime* q *be as in Equation (1). If* $\mathrm{LWE}_{n_1,q,\alpha}$ *is hard, (CRSGen, Prove, Verify) is a simulation-sound NIZK proof, then the scheme* $\mathcal{PKE}$ *is a splittable CCA-secure PKE scheme.*

Proof. We will prove Theorem 2 via a sequence of games from G_0 to G_{12}, where G_0 is a game with $b^* = 0$, while G_{12} is a game with $b^* = 1$. The security is established by showing that the adversary's advantages in game G_0 and G_{12} are negligibly close (and thus the adversary cannot distinguish b^* with non-negligible advantage). Let $\mathbf{Adv}_{\mathcal{A},i}(\kappa)$ be the adversary $\mathcal{A}$'s advantage in game G_i.

Game G_0 This game is the real game considered in Definition 4. Formally, the challenger $\mathcal{C}$ first computes $(\mathbf{A}_0, \mathbf{R}_0) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, $(\mathbf{A}_1, \mathbf{R}_1) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ and $crs \leftarrow \mathsf{CRSGen}(1^\kappa)$. Then, it keeps $\mathsf{sk} = \mathbf{R}_0$ private, and gives the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$ to the adversary $\mathcal{A}$.

Upon receiving a decryption query (label, C) for some $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$, the challenger $\mathcal{C}$ checks if $\mathsf{Verify}(crs, (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta), \pi, \mathsf{label}) = 1$. If not, $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ computes

$$\mathbf{t} = \begin{pmatrix} \mathbf{s} \\ 1 \\ \mathbf{w} \end{pmatrix} \leftarrow \mathsf{Solve}(\mathbf{A}_0, \mathbf{R}_0, \mathbf{c}_0),$$

and returns $\mathbf{w} \in \mathbb{Z}_q^{n_2}$ to $\mathcal{A}$.

At some time, $\mathcal{A}$ outputs two equal-length plaintexts $\mathbf{w}_0, \mathbf{w}_1 \in \mathcal{P}$. Then, $\mathcal{C}$ chooses $\mathbf{s}_0^*, \mathbf{s}_1^* \leftarrow_r \mathbb{Z}_q^{n_1}$, $\mathbf{e}_0^*, \mathbf{e}_1^* \leftarrow_r D_{\mathbb{Z}^m, \alpha q}$, and returns $(\mathbf{c}_0^*, \mathbf{c}_1^*)$ to $\mathcal{A}$, where

$$\mathbf{c}_0^* = \mathbf{A}_0^t \begin{pmatrix} \mathbf{s}_0^* \\ 1 \\ \mathbf{w}_0 \end{pmatrix} + \mathbf{e}_0^*, \quad \mathbf{c}_1^* = \mathbf{A}_1^t \begin{pmatrix} \mathbf{s}_1^* \\ 1 \\ \mathbf{w}_0 \end{pmatrix} + \mathbf{e}_1^*.$$

Upon receiving a string $\mathsf{label}^* \in \{0,1\}^*$ from $\mathcal{A}$, the challenger $\mathcal{C}$ computes $\pi^* \leftarrow \mathsf{Prove}(crs, (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta), (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{w}_0), \mathsf{label}^*)$, and returns the challenge ciphertext $C^* = (\mathbf{c}_0^*, \mathbf{c}_1^*, \pi^*)$ to $\mathcal{A}$.

Game G_1 This game is similar to game G_0 except that the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$ is generated by uniformly choosing $\mathbf{A}_1 \leftarrow_r \mathbb{Z}_q^{n \times m}$ at random.

Lemma 20. *Let $n = n_1 + n_2 + 1, m, q, \alpha, \beta, \gamma$ be as in Equation (1). Then, $|\,\mathbf{Adv}_{\mathcal{A},1}(\kappa) - \mathbf{Adv}_{\mathcal{A},0}(\kappa)| \le \mathrm{negl}(\kappa)$.*

Proof. Since in game G_0 the matrix $\mathbf{A}_1$ output by the trapdoor generation algorithm is statistically close to uniform by Proposition 1, and the only difference between game G_0 and G_1 is the generation of $\mathbf{A}_1$, we have that G_1 is statistically indistinguishable from G_0. Thus, $|\,\mathbf{Adv}_{\mathcal{A},1}(\kappa) - \mathbf{Adv}_{\mathcal{A},0}(\kappa)| \le \mathrm{negl}(\kappa)$. □

Game G_2 This game is similar to game G_1 except that the NIZK simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ are used to generate the crs and the proof π^* in the challenge ciphertext c^*, respectively.

Lemma 21. *If $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify}, \mathcal{S}_1, \mathcal{S}_2)$ is a NIZK proof, then we have $|\,\mathbf{Adv}_{\mathcal{A},2}(\kappa) - \mathbf{Adv}_{\mathcal{A},1}(\kappa)| \le \mathrm{negl}(\kappa)$.*

Proof. We complete the proof by showing that G_2 is indistinguishable from G_1 for any PPT adversary $\mathcal{A}$. Now, assume that the adversary $\mathcal{A}$ can distinguish G_2 from G_1 with any non-negligible probability, we construct an algorithm $\mathcal{B}$ that breaks the adaptive zero-knowledge of the NIZK proof.

Formally, given a common reference string crs as input, $\mathcal{B}$ first generates $\mathbf{A}_0, \mathbf{A}_1$ and $\mathbf{R}_0$ as in game G_1. Then, it gives the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$ to $\mathcal{A}$, and uses $\mathsf{sk} = \mathbf{R}_0$ to honestly answer the decryption query from $\mathcal{A}$. At some time, $\mathcal{A}$ outputs two equal-length plaintexts $\mathbf{w}_0, \mathbf{w}_1 \in \mathcal{P}$. Then, $\mathcal{B}$ chooses $\mathbf{s}_0^*, \mathbf{s}_1^* \leftarrow_r \mathbb{Z}_q^{n_1}$, $\mathbf{e}_0^*, \mathbf{e}_1^* \leftarrow_r D_{\mathbb{Z}^m, \alpha q}$, and returns $(\mathbf{c}_0^*, \mathbf{c}_1^*)$ to $\mathcal{A}$, where

$$\mathbf{c}_0^* = \mathbf{A}_0^t \begin{pmatrix} \mathbf{s}_0^* \\ 1 \\ \mathbf{w}_0 \end{pmatrix} + \mathbf{e}_0^*, \quad \mathbf{c}_1^* = \mathbf{A}_1^t \begin{pmatrix} \mathbf{s}_1^* \\ 1 \\ \mathbf{w}_0 \end{pmatrix} + \mathbf{e}_1^*.$$

Upon receiving a $\mathsf{label}^* \in \{0,1\}^*$ from $\mathcal{A}$, the algorithm $\mathcal{B}$ submits $x^* = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta)$, $wit^* = (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{w}_0)$ and label^* to its own NIZK proof oracle to obtain a proof π^*. Finally, $\mathcal{B}$ returns the challenge ciphertext $C^* = (\mathbf{c}_0^*, \mathbf{c}_1^*, \pi^*)$ to $\mathcal{A}$. Whenever $\mathcal{A}$ outputs a guess $b \in \{0,1\}$, $\mathcal{B}$ outputs b and terminates.

Clearly, if $\mathcal{A}$ can distinguish game G_2 from G_1 with non-negligible advantage, $\mathcal{B}$ can break the zero-knowledge of the NIZK with the same advantage. □

Game G_3 This game is similar to game G_2 except that $\mathbf{c}_1^* = \mathbf{d}_1 + \mathbf{U}_1 \begin{pmatrix} 1 \\ \mathbf{w}_0 \end{pmatrix}$ is generated by uniformly choosing $\mathbf{d}_1 \leftarrow_r \mathbb{Z}_q^m$ at random, where $\mathbf{A}_1^t = (\mathbf{B}_1 \| \mathbf{U}_1)$.

Lemma 22. *Let* $n = n_1 + n_2 + 1, m, q, \alpha, \beta, \gamma$ *be as in Equation (1). If* $\mathrm{LWE}_{n_1,q,\alpha}$ *is hard, then* $|\,\mathbf{Adv}_{\mathcal{A},3}(\kappa) - \mathbf{Adv}_{\mathcal{A},2}(\kappa)| \leq \mathrm{negl}(\kappa)$.

Proof. We complete the proof by showing that under the LWE assumption, G_3 is indistinguishable from G_2 for any PPT adversary $\mathcal{A}$. Assume that the adversary $\mathcal{A}$ can distinguish G_3 from G_2 with any non-negligible advantage, we construct an algorithm $\mathcal{B}$ that breaks the LWE assumption with the same advantage.

Formally, given a LWE challenge tuple $(\mathbf{B}_1^t, \mathbf{d}_1) \in \mathbb{Z}_q^{n_1 \times m} \times \mathbb{Z}_q^m$, $\mathcal{B}$ first randomly chooses $\mathbf{U}_1 \leftarrow_r \mathbb{Z}_q^{m \times (n_2+1)}$, generates $(\mathbf{A}_0, \mathbf{R}_0) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ and $(crs, \tau) \leftarrow \mathcal{S}_1(1^\kappa)$ as in game G_2. Then, it sets $\mathbf{A}_1^t = (\mathbf{B}_1 \| \mathbf{U}_1) \in \mathbb{Z}_q^{m \times n}$, gives the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$ to $\mathcal{A}$, and uses $\mathsf{sk} = \mathbf{R}_0$ to honestly answer the decryption query from $\mathcal{A}$. At some time, $\mathcal{A}$ outputs two equal-length plaintexts $\mathbf{w}_0, \mathbf{w}_1 \in \mathcal{P}$. Then, $\mathcal{B}$ chooses $\mathbf{s}_0^* \leftarrow_r \mathbb{Z}_q^{n_1}, \mathbf{e}_0^* \leftarrow_r D_{\mathbb{Z}^m, \alpha q}$, and returns $(\mathbf{c}_0^*, \mathbf{c}_1^*)$ to $\mathcal{A}$, where

$$\mathbf{c}_0^* = \mathbf{A}_0^t \begin{pmatrix} \mathbf{s}_0^* \\ 1 \\ \mathbf{w}_0 \end{pmatrix} + \mathbf{e}_0^*, \quad \mathbf{c}_1^* = \mathbf{d}_1 + \mathbf{U}_1^t \begin{pmatrix} 1 \\ \mathbf{w}_0 \end{pmatrix}.$$

Upon receiving a $\mathsf{label}^* \in \{0,1\}^*$ from $\mathcal{A}$, the algorithm $\mathcal{B}$ computes $\pi^* \leftarrow \mathcal{S}_2(crs, \tau, x^*, \mathsf{label}^*)$, where $x^* = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta)$. Finally, $\mathcal{B}$ returns the challenge ciphertext $C^* = (\mathbf{c}_0^*, \mathbf{c}_1^*, \pi^*)$ to $\mathcal{A}$. Whenever $\mathcal{A}$ outputs a guess $b \in \{0,1\}$, $\mathcal{B}$ outputs b and terminates.

Clearly, if $(\mathbf{B}_1^t, \mathbf{d}_1) \in \mathbb{Z}_q^{n_1 \times m} \times \mathbb{Z}_q^m$ is a valid LWE tuple, then $\mathcal{B}$ simulates the attack environment of game G_2 for $\mathcal{A}$, else it simulates the attack environment of game G_3 for $\mathcal{A}$. Thus, if $\mathcal{A}$ can distinguish game G_3 from G_2 with non-negligible advantage, $\mathcal{B}$ can break the LWE assumption with the same advantage. □

Game G_4 This game is similar to game G_3 except that $\mathbf{c}_1^* = \mathbf{d}_1 + \mathbf{U}_1 \begin{pmatrix} 1 \\ \mathbf{w}_1 \end{pmatrix}$ is generated by using $\mathbf{d}_1 \leftarrow_r \mathbb{Z}_q^m$ and $\mathbf{w}_1$, where $\mathbf{A}_1^t = (\mathbf{B}_1 \| \mathbf{U}_1)$.

Lemma 23. $|\,\mathbf{Adv}_{\mathcal{A},4}(\kappa) - \mathbf{Adv}_{\mathcal{A},3}(\kappa)| \leq \mathrm{negl}(\kappa)$.

Proof. This lemma follows from the fact that 1) $\mathbf{d}_1$ is uniformly chosen from $\mathbb{Z}_q^m$ in both game G_4 and G_3; and 2) $\mathbf{w}_b$ for $b \in \{0,1\}$ is perfectly hidden in $\mathbf{c}_1^* = \mathbf{d}_1 + \mathbf{U}_1 \begin{pmatrix} 1 \\ \mathbf{w}_b \end{pmatrix}$. □

Game G_5 This game is similar to game G_4 except that $\mathbf{c}_1^* = \mathbf{B}_1 \mathbf{s}_1^* + \mathbf{e}_1^* + \mathbf{U}_1 \begin{pmatrix} 1 \\ \mathbf{w}_1 \end{pmatrix}$ is generated by using $\mathbf{s}_1^* \leftarrow_r \mathbb{Z}_q^{n_1}, \mathbf{e}_1^* \leftarrow_r D_{\mathbb{Z}^m, \alpha q}$, where $\mathbf{A}_1^t = (\mathbf{B}_1 \| \mathbf{U}_1)$.

Lemma 24. *Let* $n = n_1 + n_2 + 1, m, q, \alpha, \beta, \gamma$ *be as in Equation (1). If* $\mathrm{LWE}_{n_1,q,\alpha}$ *is hard, then* $|\,\mathbf{Adv}_{\mathcal{A},5}(\kappa) - \mathbf{Adv}_{\mathcal{A},4}(\kappa)| \leq \mathrm{negl}(\kappa)$.

Proof. The proof is similar to that of Lemma 22, we omit the details. □

Game G_6 This game is similar to game G_5 except that the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$ is generated by computing $(\mathbf{A}_1, \mathbf{R}_1) \leftarrow_r \mathsf{TrapGen}(1^n, 1^m, q)$.

Lemma 25. *Let* $n = n_1 + n_2 + 1, m, q, \alpha, \beta, \gamma$ *be as in Equation (1). Then,* $|\,\mathbf{Adv}_{\mathcal{A},6}(\kappa) - \mathbf{Adv}_{\mathcal{A},5}(\kappa)| \leq \text{negl}(\kappa)$.

Proof. Since in game G_6 the matrix $\mathbf{A}_1$ output by the trapdoor generation algorithm is statistically close to uniform by Proposition 1, and the only difference between game G_6 and G_5 is the generation of $\mathbf{A}_1$, we have that G_6 is statistically indistinguishable from G_5. Thus, $|\,\mathbf{Adv}_{\mathcal{A},6}(\kappa) - \mathbf{Adv}_{\mathcal{A},5}(\kappa)| \leq \text{negl}(\kappa)$. □

Game G_7 This game is similar to game G_6 except that $\mathbf{R}_1$ is used to answer the decryption queries from the adversary.

Lemma 26. *Let* $n = n_1 + n_2 + 1, m, q, \alpha, \beta, \gamma$ *be as in Equation (1). If* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify}, \mathcal{S}_1, \mathcal{S}_2)$ *is a simulation-sound NIZK proof, then we have* $|\,\mathbf{Adv}_{\mathcal{A},7}(\kappa) - \mathbf{Adv}_{\mathcal{A},6}(\kappa)| \leq \text{negl}(\kappa)$.

Proof. For a decryption query (label, C) from the adversary with $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$, let $x = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta)$, where the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$. Note that the simulator will always return $\perp$ if $\mathsf{Verify}(crs, x, \pi, \mathsf{label}) \neq 1$. Moreover, if there exists $wit = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{w})$ such that $(x, wit) \in R_{pke}$, the decryption results in both games are identical except with negligible probability by Proposition 1. Besides, by the definition of CCA-security, the adversary is not allowed to make a decryption query with $(\mathsf{label}, C) = (\mathsf{label}^*, C^*)$ after seeing the challenge ciphertext (label^*, C^*) for some $C^* = (\mathbf{c}_0^*, \mathbf{c}_1^*, \pi^*)$. Let E be the event that the adversary $\mathcal{A}$ makes a decryption query (label, C) with $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$ such that 1) (label, C) is not output in the challenge phase (i.e., the decryption query (label, C) is made before the challenge phase or $(\mathsf{label}, C) \neq (\mathsf{label}^*, C^*)$ after the challenge phase); 2) there does not exist $wit = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{w})$ such that $(x, wit) \in R_{pke}$; and 3) $\mathsf{Verify}(crs, x, \pi, \mathsf{label}) = 1$. Obviously, if E does not happen, then game G_7 and G_6 are indistinguishable from the adversary.

Now, we show that E can only happen with negligible probability. In fact, we can construct an algorithm $\mathcal{B}$ that breaks the simulation-soundness of the NIZK by interacting with any adversary $\mathcal{A}$ that makes E happen with non-negligible probability. Formally, $\mathcal{B}$ can simulate the attack environment as in game G_6 for the adversary $\mathcal{A}$ except that it directly obtains crs from its own challenger, and generates the proof π^* using its NIZK proof oracle. Clearly, if $\mathcal{A}$ can make E happen with non-negligible probability, then $\mathcal{B}$ can make E happen with the same probability, and thus breaks the simulation-soundness of the NIZK. By our assumption on the NIZK, we have that E can only happen with negligible probability. In other words, $|\,\mathbf{Adv}_{\mathcal{A},7}(\kappa) - \mathbf{Adv}_{\mathcal{A},6}(\kappa)| \leq \text{negl}(\kappa)$. □

Game G_8 This game is similar to game G_7 except that the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$ is generated by uniformly choosing $\mathbf{A}_0 \leftarrow_r \mathbb{Z}_q^{n\times m}$ at random.

Lemma 27. *Let* $n = n_1 + n_2 + 1, m, q, \alpha, \beta, \gamma$ *be as in Equation (1). Then,* $|\,\mathbf{Adv}_{\mathcal{A},8}(\kappa) - \mathbf{Adv}_{\mathcal{A},7}(\kappa)| \le \mathrm{negl}(\kappa)$.

Proof. The proof is similar to Lemma 20, we omit the details. □

Game G_9 This game is similar to game G_8 except that $\mathbf{c}_0^* = \mathbf{B}_0\mathbf{s}_0^* + \mathbf{e}_0^* + \mathbf{U}_0\begin{pmatrix}1\\ \mathbf{w}_1\end{pmatrix}$ is generated by choosing $\mathbf{s}_0^* \leftarrow_r \mathbb{Z}_q^{n_1}, \mathbf{e}_0^* \leftarrow_r D_{\mathbb{Z}^m,\alpha q}$, where $\mathbf{A}_0^t = (\mathbf{B}_0\|\mathbf{U}_0)$.

Lemma 28. *Let* $n = n_1+n_2+1, m, q, \alpha, \beta, \gamma$ *be as in Equation (1). If* $\mathrm{LWE}_{n_1,q,\alpha}$ *is hard, then* $|\,\mathbf{Adv}_{\mathcal{A},9}(\kappa) - \mathbf{Adv}_{\mathcal{A},8}(\kappa)| \le \mathrm{negl}(\kappa)$.

Proof. This lemma can be shown via a sequence of games similar to that from game G_2 to G_5, we omit the details. □

Game G_{10} This game is similar to game G_9 except that 1) the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$ is generated by computing $crs \leftarrow \mathsf{CRSGen}(1^\kappa)$; and 2) the challenge ciphertext $C^* = (\mathbf{c}_0^*, \mathbf{c}_1^*, \pi^*)$ for $\mathsf{label}^* \in \{0,1\}^*$ is generated by using $\pi^* \leftarrow \mathsf{Prove}(crs, (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0^*, \mathbf{c}_1^*, \beta), (\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{w}_1), \mathsf{label}^*)$.

Lemma 29. *If* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify}, \mathcal{S}_1, \mathcal{S}_2)$ *is a NIZK proof, then we have* $|\,\mathbf{Adv}_{\mathcal{A},10}(\kappa) - \mathbf{Adv}_{\mathcal{A},9}(\kappa)| \le \mathrm{negl}(\kappa)$.

Proof. Note that $(\mathbf{c}_0^*, \mathbf{c}_1^*)$ is honestly generated using $(\mathbf{s}_1^*, \mathbf{s}_2^*, \mathbf{w}_1)$, the proof is similar to Lemma 21, we omit the details. □

Game G_{11} This game is similar to game G_{10} except that the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$ is generated by computing $(\mathbf{A}_0, \mathbf{R}_0) \leftarrow_r \mathsf{TrapGen}(1^n, 1^m, q)$.

Lemma 30. *Let* $n = n_1 + n_2 + 1, m, q, \alpha, \beta, \gamma$ *be as in Equation (1). Then,* $|\,\mathbf{Adv}_{\mathcal{A},11}(\kappa) - \mathbf{Adv}_{\mathcal{A},10}(\kappa)| \le \mathrm{negl}(\kappa)$.

Proof. The proof is similar to Lemma 25, we omit the details. □

Game G_{12} This game is similar to game G_{11} except that $\mathbf{R}_0$ is used to answer the decryption queries from the adversary.

Lemma 31. *Let* $n = n_1+n_2+1, m, q, \alpha, \beta, \gamma$ *be as in Equation (1). If* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify}, \mathcal{S}_1, \mathcal{S}_2)$ *is a NIZK proof, then* $|\,\mathbf{Adv}_{\mathcal{A},12}(\kappa) - \mathbf{Adv}_{\mathcal{A},11}(\kappa)| \le \mathrm{negl}(\kappa)$.

Proof. For a decryption query (label, C) from the adversary with $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$, let $x = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}_0, \mathbf{c}_1, \beta)$, where the public key $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, crs)$. Note that the simulator will always return $\perp$ if $\mathsf{Verify}(crs, x, \pi, \mathsf{label}) \neq 1$. Moreover, if there exists $wit = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{w})$ such that $(x, wit) \in R_{pke}$, the decryption results in both games are identical except with negligible probability by Proposition 1. Let E be the event that the adversary $\mathcal{A}$ makes a decryption query (label, C) with $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$ such that 1) $\mathsf{Verify}(crs, x, \pi, \mathsf{label}) = 1$; and 2) there does not exist $wit = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{w})$ such that $(x, wit) \in R_{pke}$. Obviously, if E does not happen, then game G_{12} and G_{11} are indistinguishable from the adversary.

Now, we show that E can only happen with negligible probability. In fact, we can construct an algorithm $\mathcal{B}$ that breaks the soundness of the NIZK by interacting with any adversary $\mathcal{A}$ that makes E happen with non-negligible probability. Formally, $\mathcal{B}$ can simulate the attack environment as in game G_{11} for the adversary $\mathcal{A}$ except that it directly obtains crs from its own challenger, and honestly runs the Prove algorithm to generate the proof π^* for the challenge label-ciphertext pair (label^*, C^*) for some $C^* = (\mathbf{c}_0^*, \mathbf{c}_1^*, \pi^*)$. Recall that (label^*, C^*) is actually a valid encryption of $\mathbf{w}_1$, we always have $(\mathbf{c}_0, \mathbf{c}_1) \neq (\mathbf{c}_0^*, \mathbf{c}_1^*)$ for any (label, C) that makes E happen, where $C = (\mathbf{c}_0, \mathbf{c}_1, \pi)$. Clearly, if $\mathcal{A}$ can make E happen with non-negligible probability, then $\mathcal{B}$ can make E happen with the same probability, and thus breaks the soundness of the NIZK. By our assumption on the NIZK, we have that E can only happen with negligible probability. In other words, $|\,\mathbf{Adv}_{\mathcal{A},12}(\kappa) - \mathbf{Adv}_{\mathcal{A},11}(\kappa)| \leq \mathrm{negl}(\kappa)$. $\square$

By combining the previous lemmas, we have $|\,\mathbf{Adv}_{\mathcal{A},12}(\kappa) - \mathbf{Adv}_{\mathcal{A},0}(\kappa)| \leq \mathrm{negl}(\kappa)$. Since game G_0 is a real game where the challenge ciphertext is an encryption of $\mathbf{w}_0$, and game G_{12} is a real game where the challenge ciphertext is an encryption of $\mathbf{w}_1$, we have the claims in Theorem 2. $\square$