9-2020

A lattice-based key-insulated and privacy-preserving signature scheme with publicly derived public key

Wenling LIU
Shanghai Jiaotong University

Zhen LIU

Khoa NGUYEN

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Yu YU

Citation

LIU, Wenling; LIU, Zhen; NGUYEN, Khoa; YANG, Guomin; and YU, Yu. A lattice-based key-insulated and privacy-preserving signature scheme with publicly derived public key. (2020). *Computer Security: ESORICS 2020: 25th European Symposium on Research in Computer Security, Guildford, September 14-18: Proceedings*. 12309, 357-377.
Available at: https://ink.library.smu.edu.sg/sis_research/7410

A Lattice-Based Key-Insulated and Privacy-Preserving Signature Scheme with Publicly Derived Public Key

Wenling Liu[1,2], Zhen Liu[1](✉), Khoa Nguyen[3], Guomin Yang[4], and Yu Yu[1,2](✉)

[1] Shanghai Jiao Tong University, Shanghai, China
{lingeros,liuzhen,yyuu}@sjtu.edu.cn
[2] Shanghai Qi Zhi Institute, Shanghai, China
[3] Nanyang Technological University, Singapore, Singapore
khoantt@ntu.edu.sg
[4] University of Wollongong, Wollongong, Australia
gyang@uow.edu.au

Abstract. As a widely used privacy-preserving technique for cryptocurrencies, Stealth Address constitutes a key component of Ring Confidential Transaction (RingCT) protocol and it was adopted by Monero, one of the most popular privacy-centric cryptocurrencies. Recently, Liu et al. [EuroS&P 2019] pointed out a flaw in the current widely used stealth address algorithm that once a derived secret key is compromised, the damage will spread to the corresponding master secret key, and all the derived secret keys thereof. To address this issue, Liu et al. introduced Key-Insulated and Privacy-Preserving Signature Scheme with Publicly Derived Public Key (PDPKS scheme), which captures the functionality, security, and privacy requirements of stealth address in cryptocurrencies. They further proposed a paring-based PDPKS construction and thus provided a provably secure stealth address algorithm. However, while other privacy-preserving cryptographic tools for RingCT, such as ring signature, commitment, and range proof, have successfully found counterparts on lattices, the development of lattice-based stealth address scheme lags behind and hinders the development of quantum-resistant privacy-centric cryptocurrencies following the RingCT approach.

In this paper, we propose the first lattice-based PDPKS scheme and prove its security in the random oracle model. The scheme provides

Z. Liu—Supported by the National Natural Science Foundation of China (No. 61672339) and the National Cryptography Development Fund (No. MMJJ20170111).
K. Nguyen—Supported by the Gopalakrishnan - NTU Presidential Postdoctoral Fellowship 2018.
G. Yang—Supported by the Australian Research Council Discovery Project DP200100144.
Y. Yu—Supported by the The National Key Research and Development Program of China (Grant No. 2018YFA0704701), National Natural Science Foundation of China (Grant Nos. 61872236 and 61971192), the National Cryptography Development Fund (Grant No. MMJJ20170209), and the Major Program of Guangdong Basic and Applied Research (Grant No. 2019B030302008).

L. Chen et al. (Eds.): ESORICS 2020, LNCS 12309, pp. 357–377, 2020.
https://doi.org/10.1007/978-3-030-59013-0_18

(potentially) quantum security not only for the stealth address algorithm but also for the deterministic wallet. Prior to this, the existing deterministic wallet algorithms, which have been widely adopted by most Bitcoin-like cryptocurrencies due to its easy backup/recovery and trustless audits, are not quantum resistant.

Keywords: Lattice-based · Signature · Privacy preservation · Stealth address

1 Introduction

The past decade has witnessed the rapid development of cryptocurrencies since the invention of the Bitcoin [20]. Bitcoin had been regarded as an innovative payment network with high anonymity, and its emergence brings the prosperity of blockchain technology and cryptocurrency. However, as shown by [17,26], Bitcoin is not truly "anonymous" but only "pseudonymous". In Bitcoin-like cryptocurrency systems, digital signature schemes are used to authorize and authenticate transactions. Each coin is assigned a public key and a value, where the public key specifies the ownership of the coin. When a user wants to spend a coin (pk_{in}, v) and transfer the value to another owner with public key pk_{out}, he issues a transaction tx that takes in (say, consumes) the coin (pk_{in}, v) and outputs (say, generates) a new coin (pk_{out}, v), and associate the transaction with a signature σ which is valid with respect to the transaction (as the signed message) and the spent coin's public key pk_{in}. Bitcoin achieves only pseudonym since information including the sender, the receiver, and the amount of the transactions are public and accessible to all participants.

Note that privacy preservation is one of the top desired features for cryptocurrencies, since the privacy weakness in Bitcoin was identified [17,26], enhancing user's privacy in cryptocurrencies has attracted much attention from community [6,13,19,27]. Among the proposed privacy-preserving technologies, ***Stealth Address*** [28,29], provides a simple yet efficient way to enhance privacy by hiding the receiver of transactions. Roughly speaking, stealth address is a key-derivation mechanism. In a cryptocurrency system with the stealth address, each user publishes his long-term master public key MPK, and if a payer, say Alice, wants to transfer funds to a payee, say Bob, she can generate a *fresh* derived public key dpk from Bob's master public key MPK_B and use dpk to specify the receiver of the transaction, without any interaction with Bob. On the other side, to spend the coin on such a dpk, Bob can generate a corresponding derived secret key dsk use his long term master secret key MSK_B, and then generate a signature that can be verified using dpk only. In such a mechanism, the receiver's master public key (referred to as 'address' in cryptocurrency) never appears in the transactions or on the blockchain, and neither the derived public key or the signature leaks any information about the master public key. Due to its simplicity and convenience (i.e., each coin is assigned a *fresh* derived public key, and no interaction between the payer and payee) and privacy-preserving virtues, stealth address has been widely adopted by many cryptocurrencies. Particularly, stealth address is a core component of RingCT protocol for Monero [21], which is one of the most popular privacy-centric cryptocurrencies and ranks the 14th in all cryptocurrencies in terms of market capitalization [9].

Recently, Liu et al. [16] have pointed out the current widely used stealth address algorithms [28,29] suffers a security flaw in designs. In particular, once a derived secret key was compromised, the damage would spread to the corresponding master secret key, and all the derived secret keys thereof. To address this problem, Liu et al. [16] introduced and formalized the concept of Signature Scheme with Publicly Derived Public key (PDPKS), capturing the functionality, security, and privacy requirements that steal address should satisfy when applied in cryptocurrencies in practice. Liu et al. also proposed a pairing-based construction, with provable security and privacy based on the discrete logarithm assumption. It is worth mentioning that, as shown by Liu et al. [16], a PDPKS scheme does not only implies a secure stealth address algorithm, but also implies a secure deterministic wallet [31] algorithm, supporting the promising applications such as easy backup and recovery, trustless audits, treasurers allocating funds to departments.

On the other side, due to the advance of quantum computing technologies, quantum-resistant cryptography, especial lattice-based cryptography has been attracting much attention and making significant progress. Cryptocurrency is also developing towards post-quantum cryptocurrencies. However, to the best of our knowledge, as so far, quantum-resistant stealth address algorithm satisfying the functionality, security, and privacy requirements captured by Liu et al.'s work [16] has not been proposed yet. This lags behind other privacy-preserving cryptographic primitives for cryptocurrencies. In particular, in the RingCT approach for building privacy-centric cryptocurrencies, linkable ring signature, stealth address, and commitment with range proof are used to hide the transaction's sender, receiver, and amount, respectively. While lattice-based linkable ring signature schemes [15,30] and lattice-based commitment with range proof schemes [10,32] have been proposed, the lack of lattice-based stealth address schemes is hindering the development of quantum-resistant privacy-centric cryptocurrencies following the RingCT approach.

1.1 Our Results

In this paper, we propose a lattice-based PDPKS construction, and prove the security and privacy in the random oracle model, based on the hardness of the Learning With Errors (LWE) problem [25]. As our construction satisfies the definitions and models on functionality, security, and privacy by Liu et al. [16], our lattice-based PDPKS construction provides potential quantum-resistance for both the stealth address algorithm and the deterministic wallet scheme.

As for many LWE-based cryptographic constructions with advanced features, the public key and signature sizes of our construction are still too large for practical use. We do not want to oversell our results, but take this as a steppingstone towards the goal of practical and quantum-resistant stealth address, as this is the first concrete instantiation of PDPKS scheme that has the potential to be resistant against quantum computers.

To enable the signature scheme with publicly derived public key, where the compromising of a derived secret key will not impact other secret keys, we resort to the techniques of lattice basis delegation [1,2,8]. We noticed that the delegation algorithm by Agrawa et al. [1] has the property that the delegated lattice conceals the original one. Note that when PDPKS is applied in cryptocurrency, to achieve that no attackers can learn the master public key from the derived public key and corresponding signatures, we have to make sure that only the payee who owns the corresponding master secret key can know the secret information that was used by the payer to create a derived public key. To achieve this, we resort to the key-private public key encryption introduced by Bellare et al. [5]. Due to the adversary's adaptively querying of derived public keys in the privacy game, an adaptive key-indistinguishable PKE scheme is needed in our scheme. However, to the best of our knowledge, no explicit construction of quantum-resistant key-indistinguishable PKE has been proposed prior to us. To construct such PKE scheme, We start the passive key-indistinguishable Regev's LWE-based PKE scheme and prove the Fujisaki-Okamoto transformation [11] transforms a passively key-indistinguishable PKE scheme to an adaptively key-indistinguishable one.

1.2 Related Work

Liu et al. [15] proposed a lattice-based linkable ring signature scheme with stealth address, but the security model does not consider the case that derived secret keys are generated and compromised, while all the signatures are generated using the master secret key. The setting increases the risk of the master key being compromised and cannot support the applications of deterministic wallet, such as treasurers allocating funds to departments.

2 Preliminary

In this section, we review the definition of PDPKS by Liu et al. [16] and some lattice-based background as well as the definition of PKE with key-privacy [5].

2.1 Definition of Publicly Derived Public Key Scheme

Syntax. A PDPKS scheme consists of the following polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{PP}$. On input the security parameter 1^λ, the algorithm outputs the public parameter PP.
- $\mathsf{MasterKeyGen}(\mathsf{PP}) \to (mpk, msk)$. On input the public parameter PP, the algorithm outputs a master public-secret key pair (mpk, msk).
- $\mathsf{DpkDerive}(\mathsf{PP}, mpk) \to dpk$. On input the public parameter PP and a master public key mpk, the algorithm outputs a derived public key dpk. We say such a dpk is linked to mpk.

- $\mathsf{DpkCheck}(\mathsf{PP}, mpk, msk, dpk) \to 1/0$. On input the public parameter PP, a master key pair (mpk, msk), and a derived public key dpk, the algorithm outputs a bit $b \in \{0,1\}$, with $b = 1$ meaning that dpk is linked to mpk and $b = 0$ otherwise.
- $\mathsf{DskDerive}(\mathsf{PP}, mpk, msk, dpk) \to dsk$. On input the public parameter PP, a master key pair (mpk, msk), and a derived public key dpk that is linked to mpk, the algorithm outputs a derived secret key dsk corresponding to dpk.
- $\mathsf{Sign}(\mathsf{PP}, dpk, \mu, dsk) \to s$. On input the public parameter PP, a derived public key dpk, a message μ, and a derived secret key dsk corresponding to dpk, the algorithm outputs a signature s.
- $\mathsf{Verify}(\mathsf{PP}, dpk, \mu, s) \to 1/0$. On input the public parameter PP, a derived public key dpk, a message μ, and a signature s, the algorithm outputs a bit $b \in \{0,1\}$, with $b = 1$ meaning valid and $b = 0$ meaning invalid.

For a cryptocurrency system with PDPKS scheme, it runs the $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$ and publish PP to all participants. Each participant can run the $(mpk, msk) \leftarrow \mathsf{MasterKeyGen}(\mathsf{PP})$ to obtain his long-term master key pair and publish mpk. When a payer, say Alice, wants to pay the payee, say Bob, Alice runs $dpk \leftarrow \mathsf{DpkDerive}(\mathsf{PP}, mpk_B)$ where mpk_B is Bob's master public key, and assigns dpk to the output coin. For Bob, when a new coin appears in the system, he runs $b \leftarrow \mathsf{DpkCheck}(\mathsf{PP}, mpk_B, msk_B, dpk)$ where msk_B is his master secret key and dpk is the coin's (derived) public key, Bob puts such a coin into his wallet only if $b = 1$. To spend such a coin, Bob runs $dsk \leftarrow \mathsf{DpkCheck}(\mathsf{PP}, mpk_B, msk_B, dpk)$ to obtain the derived secret key dsk corresponding to dpk, then he runs the sign algorithm Sign to sign a transaction spending the coin. For a transaction that consumes a coin with (derived) public key dpk, anyone can run the Verify algorithm to check whether the associated signature is valid, only using the dpk, without needing the corresponding master public key.

Correctness. The scheme must satisfy the following correctness properties: for any $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$, $(mpk, msk) \leftarrow \mathsf{MasterKeyGen}(\mathsf{PP})$, $dpk \leftarrow \mathsf{DpkDerive}(\mathsf{PP}, mpk)$, $dsk \leftarrow \mathsf{DskDerive}(\mathsf{PP}, mpk, msk, dpk)$, and any message μ, it holds that

$$\Pr\left[\mathsf{DpkCheck}(\mathsf{PP}, mpk, msk, dpk) = 1\right] = 1 - \mathrm{negl}(n), \text{ and}$$
$$\Pr\left[\mathsf{Verify}(\mathsf{PP}, dpk, \mu, \mathsf{Sign}(\mathsf{PP}, dpk, \mu, dsk)) = 1\right] = 1 - \mathrm{negl}(n).$$

Security. We define the existentially unforgeable (EUF) security of PDPKS scheme below:

Definition 1. *We say a PDPKS scheme is existentially unforgeably (EUF) secure, if all probabilistic polynomial time (PPT) adversaries $\mathcal{A}$ win the following game* $\mathtt{Game}_{\mathsf{euf}}$ *with negligible probabilities.*

– **Setup.** $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(mpk, msk) \leftarrow \mathsf{MasterKeyGen}(\mathsf{PP})$ *are run.* PP *and* mpk *are given to* $\mathcal{A}$*. An empty set* $L_{dpk} = \emptyset$ *is initialized.*[1]
– **Probing Phase.** $\mathcal{A}$ *can adaptively query the following oracles:*
  • *Derived Public Key Check Oracle* $\mathsf{ODpkCheck}(\cdot)$*:*
    On input a derived public key dpk*, this oracle returns* $c \leftarrow \mathsf{DpkCheck}(\mathsf{PP}, mpk, msk, dpk)$ *to* $\mathcal{A}$*. If* $c = 1$*, set* $L_{dpk} = L_{dpk} \cup \{dpk\}$*.*
  • *Derived Secret Key Corruption Oracle* $\mathsf{ODskCorrupt}(\cdot)$*:*
    On input a derived public key $dpk \in L_{dpk}$*, this oracle returns* $dsk \leftarrow \mathsf{DskDerive}(\mathsf{PP}, mpk, msk, dpk)$ *to* $\mathcal{A}$*.*
  • *Signing Oracle* $\mathsf{OSign}(\cdot, \cdot)$*: On input a derived public key* $dpk \in L_{dpk}$ *and a message* μ*, this oracle returns* $\sigma \leftarrow \mathsf{Sign}(\mathsf{PP}, dpk, \mu, dsk)$ *to* $\mathcal{A}$*, where* $dsk \leftarrow \mathsf{DskDerive}(\mathsf{PP}, mpk, msk, dpk)$*.*
– **Output Phase.** $\mathcal{A}$ *outputs a derived public key* $dpk^* \in L_{dpk}$*, a message* μ^**, and a signature* σ^**.*

$\mathcal{A}$ *succeeds if* $\mathsf{Verify}(\mathsf{PP}, dpk^*, \mu^*, s^*) = 1$ *under the* **restrictions** *that (1)* $\mathsf{ODskCorrupt}(dpk^*)$ *is never queried, and (2)* $\mathsf{OSign}(dpk^*, \mu^*)$ *is never queried.*

Privacy. The definition captures the fact that derived public keys and corresponding signatures do not leak the corresponding master public key.

Definition 2. *A PDPKS scheme is master public key unlinkable (MPK-UNL), if for all PPT adversaries* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the following game* $\mathtt{Game}_{\mathsf{mpkunl}}$*, denoted by* $Adv_{\mathcal{A}}^{mpkunl}$*, is negligible.*

– **Setup.** $\mathsf{PP} \leftarrow \mathsf{Setup}(\lambda)$ *is run and* PP *is given to* $\mathcal{A}$*.*
  $(mpk_0, msk_0) \leftarrow \mathsf{MasterKeyGen}(\mathsf{PP})$ *and* $(mpk_1, msk_1) \leftarrow \mathsf{MasterKeyGen}(\mathsf{PP})$ *are run, and* mpk_0, mpk_1 *are given to* $\mathcal{A}$*. Two empty sets* $L_{dpk,0} = L_{dpk,1} = \emptyset$ *are initialized.*[2]
– **Challenge Phase.** *A random bit* $b \leftarrow \{0, 1\}$ *is chosen.*
  $dpk^* \leftarrow \mathsf{DpkDerive}(\mathsf{PP}, mpk_b)$ *is given to* $\mathcal{A}$*.*
– **Probing Phase**. $\mathcal{A}$ *can adaptively query the following oracles:*
  • *Derived Public Key Check Oracle* $\mathsf{ODpkCheck}(\cdot, \cdot)$*:*
    On input a derived public key $dpk \neq dpk^*$ *and an index* $i \in \{0, 1\}$*, this oracle returns* $c \leftarrow \mathsf{DpkCheck}(\mathsf{PP}, mpk_i, msk_i, dpk)$ *to* $\mathcal{A}$*. If* $c = 1$*, set* $L_{dpk,i} = L_{dpk,i} \cup \{dpk\}$*.*
  • *Derived Secret Key Corruption Oracle* $\mathsf{ODskCorrupt}(\cdot)$*:*
    On input a derived public key $dpk \in L_{dpk,0} \cup L_{dpk,1}$*, this oracle returns* $dsk \leftarrow \mathsf{DskDerive}(\mathsf{PP}, mpk_i, msk_i, dpk)$ *to* $\mathcal{A}$*, with* $i = 0$ *if* $dpk \in L_{dpk,0}$*, and* $i = 1$ *if* $dpk \in L_{dpk,1}$*.*

[1] This set serves only to describing the game easier. It stores the derived public keys that have been checked and accepted as being linked to the target master public key, where are all known to the adversary.

[2] The two sets are defined only for describing the game easier. $L_{dpk,i} (i = 0, 1)$ stores the derived public keys that have been checked and accepted as being linked to the target master public key mpk_i. The two sets are known to the adversary.

- *Signing Oracle* $\mathsf{OSign}(\cdot,\cdot)$*: On input a derived public key* $dpk \in L_{dpk,0} \cup L_{dpk,1} \cup \{dpk^*\}$ *and a message* μ*, this oracle returns* $\sigma \leftarrow \mathsf{Sign}(\mathsf{PP}, dpk, \mu, dsk)$ *to* $\mathcal{A}$*, where* $dsk \leftarrow \mathsf{DskDerive}(\mathsf{PP}, mpk_i, msk_i, dpk)$*, with* $i = 0$ *if* $dpk \in L_{dpk,0}$*,* $i = 1$ *if* $dpk \in L_{dpk,1}$*, and* $i = b$ *if* $dpk = dpk^*$*.*

– **Guess**. $\mathcal{A}$ *outputs a bit* $b' \in \{0,1\}$ *as its guess to* b*.*

2.2 Lattice Backgrounds

Notations. We denote matrices by bold capitals, e.g., $\mathbf{A}$, and column vectors by bold small letters e.g., $\mathbf{x}$. For a matrix $\mathbf{A}$, denote its transpose by $\mathbf{A}^T$. For two matrices, $\mathbf{A}$ and $\mathbf{B}$, we denote their concatenation by $[\mathbf{A}|\mathbf{B}]$. We denote the inner product of two vectors $\mathbf{a}, \mathbf{b}$ by $\langle \mathbf{a}, \mathbf{b}\rangle = \mathbf{a}^T\mathbf{b}$. For an ordered vector set $\mathbf{T} = \{\mathbf{t}_1, \cdots, \mathbf{t}_m\}$, we denote its Gram-Schmidt orthogonalization by $\tilde{\mathbf{T}} = \{\tilde{\mathbf{t}_1}, \cdots, \tilde{\mathbf{t}}_m\}$; we also denote the matrix $[\mathbf{t}_1|\cdots|\mathbf{t}_m]$ by $\mathbf{T}$. We denote the identity matrix of order m by $\mathbf{I}_m$ and omits m without ambiguity. We denote the ℓ_2 norm of a vector $\mathbf{x}$ by $\|\mathbf{x}\|$ and define $\|\mathbf{T}\| := \max \|\mathbf{t}_i\|$. For a matrix $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$, define the parameter $s_{\mathbf{R}} = \sup_{\mathbf{x}\in\mathbb{R}^m\setminus\{\mathbf{0}\}} \frac{\|\mathbf{Rx}\|}{\|\mathbf{x}\|}$.

For a randomized algorithm or a distribution $\mathcal{A}$, we denote its once execution (or sampling) output x by $x \leftarrow \mathcal{A}$. Let S be a finite set, we abuse the notion to denote the uniform distribution over S by S. We say a function $\epsilon : \mathbb{R}_+ \to \mathbb{R}_+$ is negligible if for any polynomial p, it holds that $\epsilon(n) < 1/p(n)$ for sufficient large n. We denote an arbitrary negligible function by $\mathrm{negl}(n)$. We say a function $g : \mathbb{R}_+ \to [0,1]$ is overwhelming if $g(n) = 1 - \mathrm{negl}(n)$. For $x \in \mathbb{R}$, we define $\lfloor x \rceil = \lfloor x + \frac{1}{2} \rfloor$. For an integer $m \in \mathbb{Z}_+$, we denote $\{1, 2, \cdots, m\}$ by $[m]$.

We denote m-dimensional lattice generated by a basis $\mathbf{T}$ by $\mathcal{L}(\mathbf{T})$. Denote integer lattice $\{\mathbf{z} \in \mathbb{Z}^m : \mathbf{Az} = \mathbf{0} \mod q\}$ by $\Lambda_q^{\perp}(\mathbf{A})$ and omits q without ambiguity. Denote the discrete Gaussian distribution on lattice Λ, with Gaussian parameter s and center $\mathbf{c}$ by $D_{\Lambda,s,\mathbf{c}}$.

For Gaussian Distribution we have the following Lemma 1 and Lemma 2, where Lemma 2 is obtained by combining the smoothing lemma [18] and "the new bound of smoothing parameter" in [12].

Lemma 1 ([18]). *Let* $q \geq 2$ *and* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$*. Let* $\mathbf{T}$ *be a basis of* $\Lambda^{\perp}(\mathbf{A})$*,* $s \geq \|\tilde{\mathbf{T}}\| \cdot \omega(\sqrt{\log m})$*. Then for any* $\mathbf{c} \in \mathbb{Z}_q^m$*,*

$$\Pr_{\mathbf{x}\leftarrow D_{\Lambda^{\perp}(\mathbf{A}),s,\mathbf{c}}}[\|\mathbf{x} - \mathbf{c}\| > s\sqrt{m}] = \mathrm{negl}(n).$$

Lemma 2 ([12,18]). *For any* m*-dimensional lattice* Λ*, define*

$$\tilde{bl}(\Lambda) = \min_{\mathbf{T}:\Lambda=\mathcal{L}(\mathbf{T})} \|\tilde{\mathbf{T}}\|.$$

Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *be a matrix whose columns generate* $\mathbb{Z}_q^n$*,* $s \geq \tilde{bl}(\Lambda^{\perp}(\mathbf{A})) \cdot \omega(\sqrt{\log m})$ *be a real number. Then for a* $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,s}$*, the distribution of* $\mathbf{u} = \mathbf{Ax} \mod q$ *is statistically close to the uniform distribution over* $\mathbb{Z}_q^n$*.*

Assumptions. The security and privacy of our PDPKS construction will be based on the following Short Integer Solution (**SIS**) assumption and **LWE** assumption.

Definition 3 (SIS Assumption) ([3,12,18,22]). *Let q, β, m be functions of n. Define $\mathsf{SIS}_{n,q,\beta,m}$ problem as: Given a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, find a non-zero integer vector $\mathbf{z} \in \mathbb{Z}^m$ s.t. $\mathbf{Az} = \mathbf{0} \mod q$ and $\|\mathbf{z}\| \leq \beta$.*

For $m, \beta = \text{poly}(n)$, $q \geq \beta \cdot \tilde{O}(\sqrt{n})$, no (quantum) algorithm can solve $\mathsf{SIS}_{n,q,\beta,m}$ problem in polynomial time.

Definition 4 (LWE Assumption) ([22,25]). *Let m, q be functions of n, $q > 2$, χ be a distribution on $\mathbb{Z}_q$ called the error distribution, defines the LWE distribution $A_{\mathbf{s},\chi}$ as: Choose a vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and an error $e \leftarrow \chi$, output $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e)$. Defines the Search-$\mathsf{LWE}_{n,q,\chi,m}$ problem as: fix an $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, given at most m samples from $A_{\mathbf{s},\chi}$, work out $\mathbf{s}$. Define the Decision-$\mathsf{LWE}_{n,q,\chi,m}$ problem as: For a uniformly chosen $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, given the oracle to be (1) $A_{\mathbf{s},\chi}$ or (2) the uniform distribution over $\mathbb{Z}_q^{n+1}$, decide which is the case with at most m oracle calls.*

For parameters $m = \text{poly}(n)$, $q \leq 2^{\text{poly}(n)}$, $r = 2\sqrt{n}$ and χ be the (discrete) Gaussian distribution with Gaussian parameter r, no (quantum) algorithm can solve the (Search/Decision)-$\mathsf{LWE}_{n,q,\chi,m}$ problem in polynomial time.

Lemma 3 ([24]). *With such parameters in SIS assumption and LWE assumption, SIS assumption implies LWE assumption for $\beta \leq q/r$.*

Algorithms on Lattices. Our construction will use the following $\mathsf{SamplePre}$ and $\mathsf{TrapGen}$ algorithms.

Lemma 4 ($\mathsf{SamplePre}$ **Algorithm** [12]). *There exists a PPT algorithm $\mathsf{SamplePre}$ that, on input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a basis $\mathbf{T} \in \mathbb{Z}_q^{m\times m}$ of $\Lambda = \Lambda^{\perp}(\mathbf{A})$, a Gaussian parameter $s \geq \|\tilde{\mathbf{T}}\| \cdot \omega(\sqrt{\log m})$ and a vector $\mathbf{u} \in \mathbb{Z}_q^n$, outputs a vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{u} \mod q$ and the distribution of $\mathbf{x}$ is statistically close to the distribution of $\mathbf{x}' \leftarrow D_{\mathbb{Z}^m,s}$ conditioned on $\mathbf{Ax}' = \mathbf{u} \mod q$. The short basis $\mathbf{T}$ is called the trapdoor of $\mathbf{A}$.*

Lemma 5 ($\mathsf{TrapGen}$ **Algorithm**, [4,7]). *For fixed constant $\delta > 0$, there is a PPT algorithm $\mathsf{TrapGen}$ that, on input n (in unary), an odd prime $q = \text{poly}(n)$, and $m \geq (5 + 3\delta)n \log q$, outputs a statistically $(m \cdot q^{-\delta_0 n/2})$-close to uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T}$ of $\Lambda^{\perp}(\mathbf{A})$ such that with overwhelming probability $\|\tilde{\mathbf{T}}\| \leq O(\sqrt{n \log q})$.*

Note that we can set $\delta = 1/3$, then we have $m \geq 6n \log q$ and output matrix $\mathbf{A}$ by the above $\mathsf{TrapGen}$ algorithm distributes statistically $(6n \cdot q^{-n/6} \log q)$-close to the uniform distribution over $\mathbb{Z}_q^{n\times m}$. In addition, the following Lemma 6 shows that the output matrix $\mathbf{A}$ by the above $\mathsf{TrapGen}$ algorithm generates $\mathbb{Z}_q^n$ with overwhelming probability.

Lemma 6 ([12]). *Let $m \geq 2n \log q$, then for all but q^{-n} fractions of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, the columns of $\mathbf{A}$ generate $\mathbb{Z}_q^n$.*

Trapdoor Delegation Algorithms. On the basis/trapdoor delegation, we have the following Lemma 7, 8, and 9.

Lemma 7 ([1,7]). *There exists a PPT algorithm* $\mathsf{DeleRight}'$ *that, on input a matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, *a matrix* $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ *whose columns generate* $\mathbb{Z}_q^n$, *a trapdoor* $\mathbf{T_B}$ *of* $\mathbf{B}$ *s.t.* $\|\tilde{\mathbf{T}}_\mathbf{B}\| \le L$, *and a matrix* $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$, *outputs a trapdoor* $\mathbf{T}'_\mathbf{F}$ *for* $\mathbf{F} = [\mathbf{A}|\mathbf{AR}+\mathbf{B}]$ *s.t.* $\|\tilde{\mathbf{T}}'_\mathbf{F}\| \le L \cdot (s_\mathbf{R}+1)$.

Lemma 8 ([8]). *There exists a PPT algorithm* $\mathsf{DeleLeft}'$ *that, on input a matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *whose columns generate* $\mathbb{Z}_q^n$, *a matrix* $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$, *and a trapdoor* $\mathbf{T_A}$, *outputs a trapdoor* $\mathbf{T}'_\mathbf{F}$ *for* $\mathbf{F} = [\mathbf{A}|\mathbf{C}]$ *s.t.* $\|\tilde{\mathbf{T}}'_\mathbf{F}\| = \|\tilde{\mathbf{T}}_\mathbf{A}\|$.

Lemma 9 ([8]). *There exists a PPT algorithm* $\mathsf{RandBasis}$ *that, on input a basis* $\mathbf{T}'$ *of an* m*-dimensional integer lattice* Λ *s.t.* $\|\tilde{\mathbf{T}}'\| < L$, *a real number* $s \ge \|\tilde{\mathbf{T}}'\| \cdot \omega(\sqrt{\log m})$, *outputs a basis* $\mathbf{T}$ *of* Λ *s.t.* $\|\tilde{\mathbf{T}}\| \le s\sqrt{m}$. *Moreover, for any two basis* $\mathbf{T}_1, \mathbf{T}_2$ *of* Λ, *let* $s \ge \max\{\|\tilde{\mathbf{T}}_1\|, \|\tilde{\mathbf{T}}_2\|\} \cdot \omega(\sqrt{\log m})$, *then the outputs of* $\mathsf{RandBasis}(\mathbf{T}_1, s)$ *and* $\mathsf{RandBasis}(\mathbf{T}_2, s)$ *are statistically close.*

With the parameter $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{R} \leftarrow \{-1,1\}^{m\times m}$ as chosen in [1], applying the above Lemma 7, 8, and 9, we have the following $\mathsf{DeleRight}$ and $\mathsf{DeleLeft}$ algorithms as shown by Theorem 1 and Theorem 2 respectively. Our PDPKS construction will be based on the $\mathsf{DeleRight}$ algorithm, while the proofs will be based on the $\mathsf{DeleLeft}$ algorithm.

Theorem 1 ($\mathsf{DeleRight}$ **Algorithm**). *Let* $q > 2$ *be a prime, fix some* $s_R = O(\sqrt{m})$. *There exists a PPT algorithm* $\mathsf{DeleRight}$ *that, on input a matrix* $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, *a matrix* $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ *whose columns generate* $\mathbb{Z}_q^n$, *a trapdoor* $\mathbf{T_B}$ *of* $\mathbf{B}$ *s.t.* $\|\tilde{\mathbf{T}}_\mathbf{B}\| \le L$, *and a matrix* $\mathbf{R} \leftarrow \{-1,1\}^{m\times m}$, *a real number* $s \ge L \cdot s_R \cdot \omega(\sqrt{\log m})$, *outputs a trapdoor* $\mathbf{T_F}$ *for* $\mathbf{F} = [\mathbf{A}|\mathbf{AR}+\mathbf{B}]$ *such that* $\mathbf{F}$ *distributes statistical close to the uniform distribution over* $\mathbb{Z}_q^{n\times 2m}$ *and* $\|\tilde{\mathbf{T}}_\mathbf{F}\| \le s \cdot \sqrt{2m}$.

Moreover, for any two trapdoors $\mathbf{T}_1, \mathbf{T}_2$ *of* $\mathbf{B}$ *s.t.* $\|\tilde{\mathbf{T}}_1\| \le L$ *and* $\|\tilde{\mathbf{T}}_2\| \le L$, *the distribution of* $\mathsf{DeleRight}(\mathbf{A}, \mathbf{B}, \mathbf{T}_1, \mathbf{R}, s)$ *and* $\mathsf{DeleRight}(\mathbf{A}, \mathbf{B}, \mathbf{T}_2, \mathbf{R}, s)$ *are statistically close.*

Theorem 2 ($\mathsf{DeleLeft}$ **Algorithm**). *Let* $q > 2$ *be a prime. There exists a PPT algorithm* $\mathsf{DeleLeft}$ *that, on input a matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *s.t. columns of* $\mathbf{A}$ *generate* $\mathbb{Z}_q^n$, *a trapdoor* $\mathbf{T_A}$ *of* $\mathbf{A}$ *s.t.* $\|\tilde{\mathbf{T}}_\mathbf{A}\| \le L$, *a matrix* $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$ *and a real number* $s \ge L \cdot \omega(\sqrt{\log m})$, *outputs a trapdoor* $\mathbf{T_F}$ *for* $\mathbf{F} = [\mathbf{A}|\mathbf{C}]$ *s.t.* $\|\tilde{\mathbf{T}}_\mathbf{F}\| \le L \cdot \sqrt{2m} \cdot \omega(\sqrt{\log m})$.

Moreover, for $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ *and* $\mathbf{R} \leftarrow \{-1,1\}^{m\times m}$, *let* $\mathbf{C} = \mathbf{AR}+\mathbf{B}$ *for some* $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, s_R *be such parameter in Theorem 1, if columns of* $\mathbf{B}$ *generate* $\mathbb{Z}_q^n$ *and* $\mathbf{T_B}$ *be a trapdoor of* $\mathbf{B}$ *s.t.* $\|\tilde{\mathbf{T}}_\mathbf{B}\| \le L$, *then the outputs of* $\mathsf{DeleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{C}, L \cdot s_R \cdot \omega(\sqrt{\log m}))$ *and* $\mathsf{DeleRight}(\mathbf{A}, \mathbf{B}, \mathbf{T_B}, \mathbf{R}, L \cdot s_R \cdot \omega(\sqrt{\log m}))$ *are statistically close.*

2.3 Key-Privacy in Public Key Encryption

Our construction is based on public key encryption (PKE) with **key-privacy**, which was introduced by Bellare et al. [5]. In particular, key-privacy requires that an adversary in possession of a ciphertext is not able to tell which specific public key, out of a set of known public keys.

Syntax. A public-key encryption scheme is a tuple of four PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{GP}$. On input a security parameter 1^λ, the algorithm outputs the common global public parameters GP, that all users in the system will share, including the security parameter, the message space $\mathcal{M}$, the ciphertext space $\mathcal{C}$, etc.
- $\mathsf{KeyGen}(\mathsf{GP}) \to (pk, sk)$. On input GP, the algorithm outputs a public-secret key pair (pk, sk).
- $\mathsf{Enc}(\mathsf{GP}, pk, \mu) \to c \in \mathcal{C}$: On input GP, a public key pk and a plaintext $\mu \in \mathcal{M}$, the algorithm outputs a ciphertext $c \in \mathcal{C}$.
- $\mathsf{Dec}(\mathsf{GP}, c, pk, sk) \to \mu' / \perp$. On input GP, a secret key sk and a ciphertext $c \in \mathcal{C}$, the algorithm outputs the a plaintext $\mu' \in \mathcal{M}$ or $\perp$.

Correctness and Security. The correctness, CPA-Security, and CCA2-security are identical to that of conventional PKE, and we omit the details here.

Key-Privacy. The key-privacy is captured by the following "indistinguishable of keys under adaptive chosen-ciphertext attack" (IK-CCA) property [5]:

Definition 5 (IK-CCA). *A PKE scheme is IK-CCA secure if for any PPT adversary $\mathcal{A}$, the advantage in the following IK-CCA game $\mathsf{Game}_{\mathsf{ikcca}}$, denoted by $Adv_{\mathcal{A}}^{ikcca}$, is negligible.*

1. **Setup.** $\mathsf{GP} \leftarrow \mathsf{Setup}(1^\lambda)$ *is computed and given to* $\mathcal{A}$. $(pk_0, sk_0) \leftarrow \mathsf{KeyGen}(\mathsf{GP})$ *and* $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(\mathsf{GP})$ *are run, and* pk_1 *and* pk_2 *are sent to adversary* $\mathcal{A}$.
2. **Probing Phase 1.** $\mathcal{A}$ *can adaptively query the decryption oracle* $\mathsf{ODec}(\cdot,\cdot)$*: On input a ciphertext* $c \in \mathcal{C}$ *and an index* $i \in \{0,1\}$*, this oracle returns* $\mu \leftarrow \mathsf{Dec}(\mathsf{GP}, c, pk_i, sk_i)$ *to* $\mathcal{A}$.
3. **Challenge Phase.** $\mathcal{A}$ *chooses a challenge plaintext* $\mu^* \in \mathcal{M}$*. A uniform coin* $b \leftarrow \{0,1\}$ *is tossed.* $c^* \leftarrow \mathsf{Enc}(\mathsf{GP}, pk_b, \mu^*)$ *is given to* $\mathcal{A}$.
4. **Probing Phase 2.** $\mathcal{A}$ *can adaptively query the decryption oracle* $\mathsf{ODec}(\cdot,\cdot)$*, but cannot make query on* $(c^*, 0)$ *or* $(c^*, 1)$.
5. **Output Phase**. $\mathcal{A}$ *outputs a bit* $b' \in \{0,1\}$ *as its guess to* b.

3 Our Lattice-Based PDPKS Construction

• $\mathsf{Setup}(1^\lambda) \to \mathsf{PP}$. The algorithm takes λ (in unary) as input. Let n be a polynomial of λ, $q > 2$ be a prime, and $m \geq 6n \log q$. Fix some $\omega_1 = \omega(\sqrt{\log m})$

and $\omega_2 = \omega(\sqrt{\log 2m})$, let $s_R = O(\sqrt{m})$, $\sigma_B = L \cdot s_R \cdot \omega_1$ and $\sigma_F = \sigma_B \cdot \sqrt{2m} \cdot \omega_2$ for some $L \geq O(\sqrt{n \log q})$.

Let $\Pi_{\mathsf{pke}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a lattice-based CCA2 secure and IK-CCA secure PKE scheme. The algorithm runs $\mathsf{GP} \leftarrow \Pi_{\mathsf{pke}}.\mathsf{Setup}(1^\lambda)$. Let $\mathcal{M}_{\mathsf{pke}}$ and $\mathcal{C}_{\mathsf{pke}}$ be the message space and ciphertext space in GP respectively. Let k be polynomials of λ, and let $G_1 : \mathcal{M}_{\mathsf{pke}} \times \mathcal{C}_{\mathsf{pke}} \rightarrow \mathbb{Z}_q^{n \times m}$, $G_2 : \mathcal{M}_{\mathsf{pke}} \times \mathcal{C}_{\mathsf{pke}} \rightarrow \{-1, 1\}^{m \times m}$, and $H : \{0,1\}^* \times \{0,1\}^k \rightarrow \mathbb{Z}_q^n$ be functions that will be modeled as random oracles in the proofs. The algorithm produces as output the public parameter $\mathsf{PP} = (1^\lambda, n, m, q, s_R, \sigma_B, \sigma_F, k, \mathsf{GP}, (G_1, G_2, H), \Pi_{\mathsf{pke}})$.

- $\mathsf{MasterKeyGen}(\mathsf{PP}) \rightarrow (mpk, msk)$. On input PP, the algorithm runs $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n)$ to generate a random $\mathbf{B}$ and its trapdoor $\mathbf{T_B}$. It runs $(epk, esk) \leftarrow \Pi_{\mathsf{pke}}.\mathsf{KeyGen}(\mathsf{GP})$ to generate a PKE public-secret key pair. It outputs the master public key and master secret key as $mpk = (\mathbf{B}, epk), msk = (\mathbf{T_B}, esk)$.
- $\mathsf{DpkDerive}(\mathsf{PP}, mpk) \rightarrow dpk$. On input PP, a master public key $mpk = (\mathbf{B}, epk)$, the algorithm samples $t \leftarrow \mathcal{M}_{\mathsf{pke}}$ and computes $\tau \leftarrow \Pi_{\mathsf{pke}}.\mathsf{Enc}(\mathsf{GP}, epk, t)$, $\mathbf{A} = G_1(t, \tau)$ and $\mathbf{R} = G_2(t, \tau)$. It then sets $\mathbf{F} = [\mathbf{A}|\mathbf{AR}+\mathbf{B}]$ and outputs the derived public key $dpk = (\mathbf{F}, \tau)$.
- $\mathsf{DpkCheck}(\mathsf{PP}, mpk, msk, dpk)$: On input PP, a master key pair ($mpk = (\mathbf{B}, epk)$, $msk = (\mathbf{T_B}, esk)$), and a derived public key $dpk = (\mathbf{F} = [\mathbf{A}|\mathbf{C}], \tau)$, the algorithm computes $t \leftarrow \Pi_{\mathsf{pke}}.\mathsf{Dec}(\mathsf{GP}, \tau, epk, esk)$ and $\mathbf{A}' = G_1(t, \tau)$, $\mathbf{R}' = G_2(t, \tau)$. It outputs 0 if $\mathbf{A}' \neq \mathbf{A}$ or $\mathbf{C} \neq \mathbf{A}'\mathbf{R}' + \mathbf{B}$. Otherwise, it outputs 1.
- $\mathsf{DskDerive}(\mathsf{PP}, mpk, msk, dpk) \rightarrow dsk$. On input PP, a master key pair ($mpk = (\mathbf{B}, epk), msk = (\mathbf{T_B}, esk)$), and a derived public key $dpk = (\mathbf{F} = [\mathbf{A}|\mathbf{C}], \tau)$, the algorithm computes $t \leftarrow \Pi_{\mathsf{pke}}.\mathsf{Dec}(\mathsf{GP}, \tau, epk, esk)$, $\mathbf{A}' = G_1(t, \tau)$ and $\mathbf{R}' = G_2(t, \tau)$. It outputs $\perp$ if $\mathbf{A}' \neq \mathbf{A}$ or $\mathbf{C} \neq \mathbf{A}'\mathbf{R}' + \mathbf{B}$. Otherwise (i.e., $\mathbf{A}' = \mathbf{A}$ and $\mathbf{C} = \mathbf{AR}' + \mathbf{B}$), it runs $\mathbf{T_F} \leftarrow \mathsf{DeleRight}(\mathbf{A}, \mathbf{B}, \mathbf{T_B}, \mathbf{R}', \sigma_B)$ to sample a trapdoor $\mathbf{T_F}$ for $\mathbf{F}$, then outputs the derived secret key $dsk = \mathbf{T_F}$ for dpk.
- $\mathsf{Sign}(\mathsf{PP}, dpk, \mu, dsk) \rightarrow s$. On input PP, a derived public key $dpk = (\mathbf{F}, \tau)$, a message $\mu \in \{0,1\}^*$, and the derived secret key $dsk = \mathbf{T_F}$ corresponding to dpk, the algorithm samples a random string $r \leftarrow \{0,1\}^k$ and computes $\mathbf{u} = H(\mu, r)$. It runs $\mathbf{z} \leftarrow \mathsf{SamplePre}(\mathbf{F}, \mathbf{T_F}, \mathbf{u}, \sigma_F)$, and outputs $s = (\mathbf{z}, r)$ as a signature for μ.
- $\mathsf{Verify}(\mathsf{PP}, dpk, \mu, s) \rightarrow 1/0$. On input PP, a derived public key $dpk = (\mathbf{F}, \tau)$, a message μ, and a signature $s = (\mathbf{z}, r)$, the algorithm outputs 1 (accepts) if $\|\mathbf{z}\| \leq \sqrt{2m} \cdot \sigma_F$ and $\mathbf{Fz} = H(\mu, r) \mod q$, otherwise, it outputs 0 (rejects).

3.1 Correctness

Correctness of $\mathsf{DpkCheck}()$. Due to the correctness of Π_{pke}, for a derived public key $dpk = (\mathbf{F} = [\mathbf{A}|\mathbf{C}], \tau)$, the t under τ can be recovered correctly with overwhelming probability. This implies that the recovered $\mathbf{A}' = G_1(t, \tau)$ and $\mathbf{R}' = G_2(t, \tau)$ will pass the checks $\mathbf{A}' = \mathbf{A}$ and $\mathbf{C} = \mathbf{AR}' + \mathbf{B}$.

Correctness of $\mathsf{Verify}()$. For $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n)$, recall Lemma 5 and Lemma 6, we have that $\|\tilde{\mathbf{T}}_\mathbf{B}\| \le L$, the distribution of $\mathbf{B}$ is statistically close to the uniform distribution over $\mathbb{Z}_q^{n\times m}$, and the columns of $\mathbf{B}$ generate $\mathbb{Z}_q^n$ with overwhelming probability. Thus, for the $\mathbf{B}$, $\mathbf{A}$, and $\mathbf{R}$ in the construction, the distribution of $\mathbf{F}$ is statistically close to the uniform distribution over $\mathbb{Z}_q^{n\times 2m}$. And this implies that the columns of $\mathbf{F}$ generate $\mathbb{Z}_q^n$ with overwhelming probability. For $dsk \leftarrow \mathsf{DskDerive}(\mathsf{PP}, mpk, msk, dpk)$, due to Theorem 1, we have that $dsk = \mathbf{T_F}$ is a basis of $\Lambda^\perp(\mathbf{F})$ and $\|\tilde{\mathbf{T}}_\mathbf{F}\| \le \sigma_B \cdot \sqrt{2m}$. For $s \leftarrow \mathsf{Sign}(\mathsf{PP}, dpk, \mu, dsk)$ where $s = (\mathbf{z}, r)$, due to Lemma 4, we have that the distribution of $\mathbf{z}$ is statistically close to $\mathbf{z}'$ s.t. $\mathbf{z}' \leftarrow D_{\mathbb{Z}^{2m},\sigma_F}$ conditioned on $\mathbf{F}\mathbf{z}' = \mathbf{u} = H(\mu, r) \mod q$. This implies that with overwhelming probability, $\mathbf{z}$ satisfies $\|\mathbf{z}\| \le \sigma_F \cdot \sqrt{2m}$, i.e., the Verify algorithm accepts such (μ, s) as valid (message, signature) pair with overwhelming probability.

3.2 Proof of Security

Theorem 3. *If the $\mathsf{SIS}_{n,q,2\beta,2m}$ assumption holds with $\beta = \sqrt{2m}\cdot\sigma_F$, then the PDPKS scheme is secure in the random oracle model.*

Proof. Let $\mathcal{F}$ be a forger of the PDPKS scheme that wins the game $\mathtt{Game}_{\mathsf{euf}}$ (w.r.t. Definition 1) with non-negligible probability $\epsilon(n)$. We construct an SIS solver $\mathcal{S}$ that invokes $\mathcal{F}$ as a subroutine and solves the $\mathsf{SIS}_{n,q,2\beta,2m}$ problem with non-negligible probability.

Setup. $\mathcal{S}$ is given an instance of SIS problem $\mathsf{SIS}_{n,q,2\beta,2m}$ with $\beta = \sqrt{2m}\cdot\sigma_F$, i.e., $\mathbf{F} = [\mathbf{A}|\mathbf{C}] \in \mathbb{Z}_q^{n\times 2m}$, where $\mathbf{A}, \mathbf{C} \in \mathbb{Z}_q^{n\times m}$. $\mathcal{S}$ samples $\mathbf{z} \leftarrow D_{\mathbb{Z}_q^{2m},\sigma_F}$ and computes $\mathbf{u} = \mathbf{F}\mathbf{z} \mod q$.

$\mathcal{S}$ setups $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$ as in the construction and gives PP to $\mathcal{F}$, We assume WLOG that $\mathcal{F}$ queries G_1, G_2 and H for at most Q_1, Q_2 and Q_H times respectively and set $Q_G = \max\{Q_1, Q_2\}$. $\mathcal{S}$ chooses $k \leftarrow [Q_G]$ and $\ell \leftarrow [Q_H]$. $\mathcal{S}$ initializes empty lists L_1, L_2, L_H to record the oracle query-results of G_1, G_2, and H respectively.

$\mathcal{S}$ samples $\mathbf{R} \leftarrow \{-1,1\}^{m\times m}$ and runs $(epk, esk) \leftarrow \Pi_{\mathsf{pke}}.\mathsf{KeyGen}(\mathsf{GP})$. $\mathcal{S}$ then sets $\mathbf{B} = \mathbf{C} - \mathbf{AR}$, $mpk = (\mathbf{B}, epk)$ and gives mpk to $\mathcal{F}$. Note that $\mathbf{B}$ distributes statistically close to that in the real game. $\mathcal{S}$ initializes an empty list $\hat{L}_{dpk}$ to record the derived public keys linked to mpk and corresponding information.

Probing Phase. $\mathcal{F}$ can adaptively query the following oracles:

- For the j-th distinct query to G_1 on (t_j, τ_j): If $j \neq k$, $\mathcal{S}$ runs $(\mathbf{A}_j, \mathbf{T}_{\mathbf{A}_j}) \leftarrow \mathsf{TrapGen}(1^{\lambda_1})$, samples $\mathbf{R}_j \leftarrow \{-1,1\}^{m\times m}$, stores $(t_j, \tau_j, \mathbf{A}_j, \mathbf{T}_{\mathbf{A}_j})$ and $(t_j, \tau_j, \mathbf{R}_j)$ into L_1 and L_2 respectively, and replies with $\mathbf{A}_j$. If $j = k$, $\mathcal{S}$ stores $(t_j, \tau_j, \mathbf{A}, \top)$ and $(t_j, \tau_j, \mathbf{R})$ into L_1 and L_2 respectively, and replies with $\mathbf{A}$.
- For a query to G_2 on any (t, τ): If $(t, \tau, \mathbf{R}')$ exists in L_2 with some $\mathbf{R}'$, $\mathcal{S}$ replies with $\mathbf{R}'$, otherwise $\mathcal{S}$ makes a query to G_1 on (t, τ), which triggers a new $(t, \tau, \mathbf{R}')$ to be put into L_2, then replies with the corresponding $\mathbf{R}'$.

• For j-th distinct query to H on (μ_j, r_j): If $j \neq \ell$, $\mathcal{S}$ chooses $\mathbf{z}_j \leftarrow D_{\mathbb{Z}^{2m},\sigma_F}$, sets $\mathbf{u}_j = \mathbf{F}\mathbf{z}_j \bmod q$, stores $(\mu_j, r_j, \mathbf{u}_j, \mathbf{z}_j)$ into L_H, and replies with $\mathbf{u}_j$. If $j = \ell$, $\mathcal{S}$ stores $(\mu_j, r_j, \mathbf{u}, \mathbf{z})$ into L_H, and replies with $\mathbf{u}$.
• For a query to $\mathsf{ODpkCheck}(\cdot)$ on $dpk' = (\mathbf{F}' = [\mathbf{A}'|\mathbf{C}'], \tau')$: $\mathcal{S}$ runs $t' \leftarrow \Pi_{\mathsf{pke}}.\mathsf{Dec}(\mathsf{GP}, \tau', epk, esk)$, and makes query to G_1 and G_2 on (t', τ') respectively. Let $\mathbf{A}'' = G_1(t', \tau')$, $\mathbf{R}'' = G_2(t', \tau')$. If $\mathbf{A}' = \mathbf{A}''$ and $\mathbf{C}' = \mathbf{A}''\mathbf{R}'' + \mathbf{B}$, $\mathcal{S}$ replies with 1 and sets $\hat{L}_{dpk} = \hat{L}_{dpk} \cup \{(dpk, \mathbf{A}'', \mathbf{R}'')\}$, otherwise replies with 0.
• For a query to $\mathsf{ODskCorrupt}(\cdot)$ on $dpk' = (\mathbf{F}' = [\mathbf{A}'|\mathbf{C}'], \tau') \in L_{dpk}$: $\mathcal{S}$ finds $dpk' \in \hat{L}_{dpk}$, let $(\mathbf{A}', \mathbf{R}')$ be the corresponding matrices. We have that there is a tuple $(t', \tau', \mathbf{A}', \mathbf{T}'_{\mathbf{A}}) \in L_1$ and a tuple $(t', \tau', \mathbf{R}') \in L_2$, where $t' = \Pi_{\mathsf{pke}}.\mathsf{Dec}(\mathsf{GP}, \tau', epk, esk)$. If $\mathbf{A}' = \mathbf{A}$, note that $\mathbf{T}_{\mathbf{A}'} = \top$, $\mathcal{S}$ aborts the game. Otherwise, $\mathcal{S}$ computes $\mathbf{T}_{\mathbf{F}'} \leftarrow \mathsf{DeleLeft}(\mathbf{A}', \mathbf{A}'\mathbf{R}' + \mathbf{B}, \mathbf{T}_{\mathbf{A}'}, \sigma_B)$ and replies with $\mathbf{T}_{\mathbf{F}'}$.
• For a query to $\mathsf{OSign}(\cdot, \cdot)$ on $(dpk' = (\mathbf{F}', \tau'), \mu')$ such that $dpk' \in L_{dpk}$: If $\mathbf{F}' = \mathbf{F}$, $\mathcal{S}$ samples $r' \leftarrow \{0,1\}^k$ and makes a query to H on (μ', r'). With $(\mu', r', \mathbf{u}', \mathbf{z}')$ in L_H, $\mathcal{S}$ replies with $s' = (\mathbf{z}', r')$. If $\mathbf{F}' \neq \mathbf{F}$, $\mathcal{S}$ runs $t' \leftarrow \Pi_{\mathsf{pke}}.\mathsf{Dec}(\mathsf{GP}, \tau', epk, esk)$. Let $(t', \tau', \mathbf{A}', \mathbf{T}_{\mathbf{A}'}) \in L_1$ be the tuple corresponding to (t', τ'), $\mathcal{S}$ computes $\mathbf{T}_{\mathbf{F}'} \leftarrow \mathsf{DeleLeft}(\mathbf{A}', \mathbf{A}'\mathbf{R}' + \mathbf{B}, \mathbf{T}_{\mathbf{A}'}, \sigma_B)$ and sets $dsk' = \mathbf{T}_{\mathbf{F}'}$, then replies with $s' \leftarrow \mathsf{Sign}(\mathsf{PP}, dsk', dpk', \mu')$.

Output Phase. $\mathcal{F}$ outputs a forge $(dpk^*, \mu^*, s^* = (\mathbf{z}^*, r^*))$. $\mathcal{S}$ outputs $\mathbf{z}^* - \mathbf{z}$ as its solution to the SIS problem.

Analysis. Before analyzing the reduction, we prove the following claims.

Claim 1. *G_2 is perfectly simulated, and the responses of G_1, H are statistically close to such in the real game.*

Proof. Due to Lemma 5, the output of G_1 simulated by $\mathcal{S}$ is statistical close to uniform. Recall that by Lemma 2 the output of H simulated by $\mathcal{S}$ is statistically close to uniform.

Claim 2. *The replies of* $\mathsf{ODpkCheck}(\cdot)$ *simulated by $\mathcal{S}$ is statistical close to those in the real game.*

Proof. The only difference between the $\mathsf{ODpkCheck}(\cdot)$ simulated by $\mathcal{S}$ and such in the real game is that whether the simulated or the real G_1, G_2 are used. Due to Claim 1, the claim holds.

Claim 3. *With negligible probability, $\mathcal{F}$ adds some $dpk' = (\mathbf{F}' = [\mathbf{A}|\mathbf{C}'], \tau')$ for $\mathbf{C}' \neq \mathbf{C}$ to L_{dpk}.*

Proof. If $dpk' = (\mathbf{F}' = [\mathbf{A}|\mathbf{C}'], \tau')$ is added to L_{dpk}, then $\tau' \neq \tau$. Since $\mathbf{F}'$ is determined by τ', but distributes uniform before making queries to G_1, G_2 on (t', τ') where $t' = \mathsf{Dec}(\mathsf{GP}, \tau', epk, esk)$, this happens with negligible probability due to the limited query times.

Claim 4. *$\mathcal{F}$ produces a forgery with regards to $\mathbf{A}$ with probability $\epsilon(n)/Q_G - \mathrm{negl}(n)$.*

Proof. To facilitate the analysis, suppose that there is an imaginary computational unbounded $\mathcal{S}'$ that behaves identical to $\mathcal{S}$ except when $\mathcal{F}$ queries $\mathsf{ODskCorrupt}$ on $(\mathbf{F}, \tau)$, where $\mathcal{F}$ is the SIS instance to solve and τ is arbitrary. Upon such a query, $\mathcal{S}'$ computes a trapdoor $\mathbf{T_A}$ of $\mathbf{A}$ s.t. $\|\tilde{\mathbf{T}}_\mathbf{A}\| \leq L$ and replies with the trapdoor delegated from $\mathbf{T_A}$. Then $\mathcal{S}'$ never aborts. In the game simulated by $\mathcal{S}'$, the view of $\mathcal{F}$ is statistically close to such in the real game. Since $\mathcal{F}$ outputs a forge in the real game with probability larger than $\epsilon(n)$, $\mathcal{F}$ outputs a forge with probability $\epsilon(n) - \text{negl}(n)$ in the game simulated by $\mathcal{S}'$. If $\mathcal{F}$ output a forge, there exists some keys in L_{dpk} that hasn't been queried to $\mathsf{ODskCorrupt}$. Due to the uniformity of $\mathbf{A}$, any such key has probability at least $1/Q_G$ regards to $\mathcal{A}$. Then with probability larger than $\epsilon(n)/Q_G - \text{negl}(n)$, $\mathcal{F}$ does not query $\mathsf{ODskCorrupt}$ with keys regard to $\mathbf{A}$. Since before $\mathcal{S}$ abort, the view of $\mathcal{F}$ in the game simulated by $\mathcal{S}$ is identical to such in the game simulated by $\mathcal{S}'$, then with probability $\epsilon(n)/Q_G - \text{negl}(n)$, $\mathcal{F}$ produces a forgery with regards to $\mathbf{A}$.

With probability larger than $\epsilon(n)/Q_G - \text{negl}(n)$, $\mathcal{F}$ produces a forgery with regards to $\mathbf{A}$, also under this case, the probability that the forgery happens on $\mathbf{u}$ is $1/Q_H$, so the total probability is $\epsilon(n)/(Q_G \cdot Q_H) - \text{negl}(n)$. If $\mathcal{F}$ produces a forgery with regards to $\mathbf{A}$, $\mathcal{S}$ will not abort. Since the view of $\mathcal{F}$ in the game simulated by $\mathcal{S}$ is statistically close to such in the real game, then if $\mathcal{S}$ does not abort, $\mathcal{F}$ outputs a valid forge $s^* = (\mathbf{z}^*, r^*)$ for some message with probability $\epsilon(n) - \text{negl}(n)$. If $\mathcal{F}$ outputs a valid forge $(\mathbf{z}^*, r)$, then $\|\mathbf{z}^*\| < \sigma_F \cdot \sqrt{2m}$ with overwhelming probability. If such $\mathbf{z}$ is forged with $\mathbf{F}$, then $\mathbf{F}\mathbf{z}^* = \mathbf{u} \mod q$. Due to min-entropy of Gaussian distribution shown in [12,23], $\mathbf{z}$ has min-entropy at least $O(m)$, $\mathbf{z} \neq \mathbf{z}^*$ with overwhelming probability. With all the above events happen, $\|\mathbf{z} - \mathbf{z}^*\| \leq 2\sigma_F \cdot \sqrt{2m}$, $\mathbf{z} - \mathbf{z}^*$ is a solution to the SIS problem.

In conclusion, $\mathcal{S}$ outputs a valid solution $\mathbf{z} - \mathbf{z}^*$ of the given SIS instance will probability large than $\epsilon(n)^2/(Q_H \cdot Q_G^2) - \text{negl}(n)$, which is non-negligible.

3.3 Proof of Privacy

Theorem 4. *If the CCA2-security and IK-CCA security of Π_{pke} holds, the PDPKS scheme is MPK-UNL privacy-preserving in random oracle model.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks the privacy of the PDPKS scheme with non-negligible probability $\epsilon(n)$, we construct a PPT adversary $\mathcal{B}$ that breaks the IK-CCA security of Π_{pke} with non-negligible probability.

Setup. $\mathcal{B}$ is given the global public parameter GP of Π_{pke}, and two public keys epk_0, epk_1. $\mathcal{B}$ simulates the MPK-UNL game for $\mathcal{A}$ as follows: $\mathcal{B}$ sets $n, m, q, s_R, \sigma_B, \sigma_F, k, G_1, G_2, H$ as in the construction, and gives $\mathsf{PP} = (1^\lambda, n, m, q, s_R, \sigma_B, \sigma_F, k, \mathsf{GP}, (G_1, G_2, H), \Pi_{\mathsf{pke}})$ to $\mathcal{A}$. $\mathcal{B}$ runs $(\mathbf{B}_0, \mathbf{T}_0) \leftarrow \mathsf{TrapGen}(1^n)$ and $(\mathbf{B}_1, \mathbf{T}_1) \leftarrow \mathsf{TrapGen}(1^n)$, and gives $mpk_0 = (\mathbf{B}_0, epk_0)$ and $mpk_1 = (\mathbf{B}_1, epk_1)$ to $\mathcal{A}$.

$\mathcal{B}$ initializes empty lists L_1, L_2, L_H to record the oracle query-results of G_1, G_2, and H respectively. $\mathcal{B}$ initializes two empty lists $\hat{L}_{dpk,i} (i = 0, 1)$ to record the derived public keys linked to mpk_i and the corresponding information. $\mathcal{B}$ samples $t^* \leftarrow \mathcal{M}_{\mathsf{pke}}$ and submits t^* to his challenge in the IK-CCA game, and obtains a challenge ciphertext $\tau^* \leftarrow \Pi_{\mathsf{pke}}.\mathsf{Enc}(\mathsf{GP}, epk_b, t^*)$. $\mathcal{B}$ simulates dpk^* by running $(\mathbf{A}^*, \mathbf{T}_{\mathbf{A}^*}) \leftarrow \mathsf{TrapGen}(1^n)$, sampling $\mathbf{C}^* \leftarrow \mathbb{Z}_q^{n\times m}$ and then setting the challenge derived public key as $dpk^* = (\mathbf{F}^* = [\mathbf{A}^*|\mathbf{C}^*], \tau^*)$ and sending dpk^* to $\mathcal{A}$.

Probing Phase. $\mathcal{B}$ then answers $\mathcal{A}$'s oracle queries as follows:

• For the j-th distinct query to G_1 on (t_j, τ_j), if $t_j = t^*$, $\mathcal{B}$ aborts the game; otherwise, $\mathcal{B}$ samples $\mathbf{A}_j \leftarrow \mathbb{Z}_q^{n\times m}$ and stores $(t_j, \tau_j, \mathbf{A}_j)$ into L_1, then replies with $\mathbf{A}_j$.

• For the j-th distinct query to G_2 on (t_j, τ_j), if $t_j = t^*$, $\mathcal{B}$ aborts the game; otherwise, $\mathcal{B}$ samples $\mathbf{R}_j \leftarrow \{-1, 1\}^{m\times m}$ and stores $(t_j, \tau_j, \mathbf{R}_j)$ into L_2, then replies with $\mathbf{R}_j$.

• For the j-th distinct query to H on (μ_j, r_j), $\mathcal{B}$ samples $\mathbf{u}_j \leftarrow \mathbb{Z}_q^n$ and stores $(\mu_j, r_j, \mathbf{u}_j)$ into L_H, then replies with $\mathbf{u}_j$.

• For a query to $\mathsf{ODpkCheck}(\cdot, \cdot)$ on (dpk, i) where $i \in \{0, 1\}$ and $dpk = (\mathbf{F}, \tau) \neq dpk^*$: If $\tau \neq \tau^*$, $\mathcal{B}$ make a query to $\mathsf{ODec}(\cdot, \cdot)$ on (τ, i) and obtain a $t \in \mathcal{M}_{pke}$. Then $\mathcal{B}$ make a query to G_1 on (t, τ) and a query to G_2 on (t, τ), and sets $\mathbf{A} = G_1(t, \tau), \mathbf{R} = G_2(t, \tau)$. If $\mathbf{F} = [\mathbf{A}|\mathbf{AR}+\mathbf{B}_i]$, $\mathcal{B}$ sets $\hat{L}_{dpk,i} = \hat{L}_{dpk,i} \cup \{(dpk, \mathbf{A}, \mathbf{R})\}$ and replies with 1, otherwise replies with 0. If $(\tau = \tau^*) \wedge (\mathbf{F} \neq \mathbf{F}^*)$, $\mathcal{B}$ returns 0.

• For a query to $\mathsf{ODskCorrupt}(\cdot)$ on $dpk = (\mathbf{F}, \tau) \in L_{dpk,i}$ s.t. $i \in \{0, 1\}$: as $dpk \in L_{dpk,i}$, $\mathcal{B}$ can find the corresponding $(\mathbf{A}, \mathbf{R})$ from $\hat{L}_{dpk,i}$ such that $\mathbf{F} = [\mathbf{A}|\mathbf{AR} + \mathbf{B}_i]$, then $\mathcal{B}$ computes $\mathbf{T}_{\mathbf{F}} \leftarrow \mathsf{DeleRight}(\mathbf{A}, \mathbf{B}_i, \mathbf{T}_i, \mathbf{R}, \sigma_B)$ and replies with $dsk = \mathbf{T}_{\mathbf{F}}$.

• For a query to $\mathsf{OSign}(\cdot, \cdot)$ on a $dpk = (\mathbf{F}, \tau) \in L_{dpk,0} \cup L_{dpk,1} \cup \{dpk^*\}$ and a message μ: If $dpk \in L_{dpk,i}$ for $i \in \{0, 1\}$, $\mathcal{B}$ can find the corresponding $(\mathbf{A}, \mathbf{R})$ from $\hat{L}_{dpk,i}$ such that $\mathbf{F} = [\mathbf{A}|\mathbf{AR} + \mathbf{B}_i]$, then $\mathcal{B}$ computes $\mathbf{T}_{\mathbf{F}} \leftarrow \mathsf{DeleRight}(\mathbf{A}, \mathbf{B}_i, \mathbf{T}_i, \mathbf{R}, \sigma_B)$, and runs $s \leftarrow \mathsf{Sign}(\mathsf{PP}, dpk, \mu, dsk = \mathbf{T}_{\mathbf{F}})$ and replies with s. If $dpk = dpk^*$, $\mathcal{B}$ computes $\mathbf{T}_{\mathbf{F}^*} \leftarrow \mathsf{DeleLeft}(\mathbf{A}^*, \mathbf{T}_{\mathbf{A}^*}, \mathbf{C}^*, \sigma_B)$, samples $r \leftarrow \{0, 1\}^k$, and computes $\mathbf{u} = H(\mu, r)$. Then $\mathcal{B}$ runs $\mathbf{z} \leftarrow \mathsf{SamplePre}$ $(\mathbf{F}^*, \mathbf{T}_{\mathbf{F}^*}, \sigma_F)$ and outputs $s = (\mathbf{z}, r)$ as the signature.

Output Phase. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Analysis. We prove the following claims.

Claim 5. *If $\mathcal{A}$ does not query G_1, G_2 on t^*, the simulated $\mathbf{F}^*$ is statistical indistinguishable to the original MPK-UNL game.*

Proof. If $\mathcal{A}$ does not query G_1, G_2 on t^*, then $\mathbf{R}^* = G_2(t^*, \tau^*)$ is undefined and uniformly distributed, which means $\mathbf{F}^* = [\mathbf{A}^*|\mathbf{A}^*\mathbf{R}^* + \mathbf{B}_b^*]$ is statistically indistinguishable from $\mathbf{F}^* = [\mathbf{A}^*|\mathbf{C}^*]$ (Theorem 1).

Claim 6. *If $\mathcal{A}$ does not query G_1, G_2 on t^*, the simulation of* $\mathsf{DpkCheck}$*,* $\mathsf{DskDerive}$ *and* OSign *queries is statistically close to that in the real game.*

Proof. The only difference between the simulation and the real game is caused by the use of $\mathsf{DeleLeft}$ and $\mathsf{DeleRight}$, which produces statistically close results.

Claim 7. *$\mathcal{A}$ queries G_1, G_2 on t^* with negligible probability if Π_{PKE} is IND-CCA2 secure and $\mathcal{M}_{pke}$ has super-polynomial size.*

Proof. If $\mathcal{A}$ queries G_1 or G_2 on t^* with non-negligible probability, then we can construct $\mathcal{B}'$ to break the CCA2 security of Π_{PKE} with non-negligible probability.

$\mathcal{B}'$ is given a public key epk. $\mathcal{B}'$ then picks two random messages t_0^*, t_1^* from $\mathcal{M}_{pke}$ and obtains a challenge ciphertext $\tau^* \leftarrow \Pi_{\mathsf{pke}}.\mathsf{Enc}(\mathsf{GP}, epk, t_{b'}^*)$ where b' is chosen by the IND-CCA2 challenger. $\mathcal{B}'$ sets up the game for $\mathcal{A}$ as follows: $\mathcal{B}'$ tosses a coin b and sets $epk_b = epk$ and randomly generates (epk_{1-b}, esk_{1-b}). $\mathcal{B}'$ simulates $\mathbf{F}^*$ as $\mathcal{B}$ does and then gives epk_0, epk_1 and $dpk^* = (\mathbf{F}^*, \tau^*)$ to $\mathcal{A}$. $\mathcal{B}'$ answers all the queries as $\mathcal{B}$ does except that $\mathcal{B}'$ simulates all the queries related to epk_{1-b} honestly using esk_{1-b}. If in the game, $\mathcal{A}$ queries G_1 or G_2 on t_c^* for $c \in \{0,1\}$, then $\mathcal{B}'$ outputs c as his guess for b' in the IND-CCA2 game, otherwise, $\mathcal{B}'$ outputs a random bit. Since $\mathcal{M}_{pke}$ has super-polynomial size and $t_{1-b'}^*$ is never used in the simulation for $\mathcal{A}$, the chance that $\mathcal{A}$ outputs $t_{1-b'}^*$ in a query to G_1 or G_2 is negligible. On the other hand, by the assumption, $t_{b'}^*$ will appear in a query to G_1 or G_2 with a non-negligible probability, hence $\mathcal{B}'$ can win the IND-CCA2 game with a negligible probability.

If $\mathcal{A}$ does not query G_1 or G_2 on t^*, then $\mathcal{B}$ does not abort and the simulation is statistical close to the real game. If $\mathcal{A}$ can guess b correctly in the $\mathtt{Game}_{\mathsf{mpkunl}}$ with non-negligible advantage, $\mathcal{B}$ can break the IK-CCA security of Π_{pke} with non-negligible advantage.

3.4 Parameter Choosing

We fix the parameter $n = \lambda$. The other parameters can be instantiated in various ways. For a typical choice, we fix $k = n$ and $\epsilon > 0$ to some constant, choose $m = n^{1+\epsilon}$ and set $L = \sqrt{m}$. We fix $\omega_1 = \omega_2 = \omega(\sqrt{\log m})$ and set $\sigma_F = O(m^{3/2}) \cdot \omega(\sqrt{\log m})^2$. To ensure the security of our SIS problem, we set $\beta = \sqrt{2m} \cdot \sigma_F = O(m^2) \cdot \omega(\sqrt{\log m})^2$. According to the SIS assumption, we set $q = \tilde{O}(m^{5/2}) \cdot \omega(\sqrt{\log m})^2$.

3.5 Lattice-Based Key-Private Public Key Encryption

In this section, we construct a (quantumly) CCA2-secure and IK-CCA secure PKE based on the hardness of LWE. We states the theorem here and leave the construction of such PKE scheme and the proof to Appendix A.

Theorem 5. *Let $q > 2$ be a prime, m be some polynomial of n, χ be an efficiently sampleable distribution over $\mathbb{Z}_q$. Assume that the $\mathsf{LWE}_{n,q,\chi,m}$ problem is hard, there exists PKE scheme π that is IND-CCA2 secure and IK-CCA secure.*

4 Conclusion

Unlike other cryptographic components for RingCT (e.g., ring signature, commitment, and range proof) for which lattice-based constructions are known, we did not know any lattice based stealth address schemes, which hinders the development and deployment of quantum-resistant RingCT-based privacy-centric cryptocurrencies. In this paper, we fill this gap by proposing the first lattice-based PDPKS scheme and proving its security in the random oracle model. Our construction offers (potentially) quantum security not only for the stealth address algorithm but also for the deterministic wallet algorithm. Previously, deterministic wallet algorithms, despite their popularity in Bitcoin-like cryptocurrencies, were not quantum resistant.

A Construct Quantumly CCA2-Secure PKE Scheme with CCA2 with IK-CCA Security

The IK-CPA Privacy of PKE Schemes. The definition of IK-CPA privacy of PKE schemes follows [5].

Definition 6. *A PKE scheme is Key-Indistinguishable in Chosen-Plaintext-Attack (IK-CPA) secure if for any PPT adversary $\mathcal{A}$, the advantage in the following CCA-key-distinguish game* $\mathtt{Game}_{\mathsf{ikcpa}}$*, denoted by* $Adv_{\mathcal{A}}^{ikcca}$*, is negligible.*

1. **Setup.** $\mathsf{GP} \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk_0, sk_0) \leftarrow \mathsf{KeyGen}(\mathsf{GP}), (pk_1, sk_1) \leftarrow \mathsf{KeyGen}(\mathsf{GP})$ *are run.* GP, pk_0, pk_1 *are sent to the adversary* $\mathcal{A}$.
2. **Challenge Phase.** $\mathcal{A}$ *choose a challenger ciphertext* $\mu^* \in \mathcal{M}$*. A uniform coin* $b \leftarrow \{0,1\}$ *is tossed.* $c^* \leftarrow \mathsf{Enc}(\mathsf{GP}, pk_b, \mu^*)$ *is given to* $\mathcal{A}$.
3. **Output Phase.** $\mathcal{A}$ *outputs a bit* $b' \in \{0,1\}$ *as its guess to* b *and wins if* $b' = b$.

Let n, q, m, χ be the parameters in Theorem 5. Regev's PKE scheme [25] $\mathsf{LWEPKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a tuple of PPT algorithms.

- $\mathsf{LWEPKE.Setup}(1^\lambda)$: The algorithm computes $n = \text{poly}(n)$, $m = \text{poly}(n)$, fixes the error distribution χ according to n and outputs $\mathsf{GP} = (1^n, q, \chi, m)$.
- $\mathsf{LWEPKE.KeyGen}(\mathsf{GP})$: The algorithm samples $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, and $\mathbf{e} \leftarrow \chi^m$. It then computes $\mathbf{b} = \mathbf{A}^T\mathbf{s} + \mathbf{e}$ and outputs secret key $sk = \mathbf{s}$ and public key $pk = (\mathbf{A}^T, \mathbf{b})$.
- $\mathsf{LWEPKE.Enc}(\mathsf{GP}, pk = (\mathbf{A}^T, \mathbf{b}), \mu \in \{0,1\})$: The algorithm samples $\mathbf{r} \leftarrow \{0,1\}^m$, computes and outputs $c = (\mathbf{a}^T = \mathbf{r}^T\mathbf{A}^T, b = \mathbf{r}^T\mathbf{b} + \lfloor \frac{q}{2} \rceil)$ as ciphertext.
- $\mathsf{LWEPKE.Dec}(\mathsf{GP}, c = (\mathbf{a}^T, b), pk, sk = \mathbf{s})$: The algorithm decrypts 0 if $b - \langle \mathbf{a}, \mathbf{s} \rangle$ is closer to 0 than to $\frac{q}{2}$. Otherwise decrypts to 1.

Lemma 10. LWEPKE *is IK-CPA private.*

Proof. We define the following games for a hybrid argument:

- $\mathtt{Game}_0$: This is the original kd-cpa game of LWEPKE for $\mathcal{A}$.
- $\mathtt{Game}_1$: $\mathbf{A}_0 \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{b}_0 \leftarrow \mathbb{Z}_q^m$ are sampled. Based on $\mathtt{Game}_0$, $(\mathbf{A}_0^T, \mathbf{b}_0)$ is sent to $\mathcal{A}$ instead of pk_0. The rest of the game remains unchanged.

– $\mathtt{Game}_2$: $\mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{b}_1 \leftarrow \mathbb{Z}_q^m$ are sampled. Based on $\mathtt{Game}_1$, $(\mathbf{A}_1^T, \mathbf{b}_1)$ is sent to $\mathcal{A}$ instead of pk_1. The rest of the game remains unchanged.

Due to the LWE assumption, the views of $\mathcal{A}$ in $\mathtt{Game}_0$ and $\mathtt{Game}_1$ are computational indistinguishable, as are the views of $\mathcal{A}$ in $\mathtt{Game}_1$ and $\mathtt{Game}_2$. The rest of the proof can be done by showing that any PPT adversary $\mathcal{A}$ achieves only negligible advantage in $\mathtt{Game}_2$ by using left-over hash lemma [14].

From IK-CPA to IK-CCA. We introduce Fujisaki-Okamoto transformation that transforms a CPA-secure PKE scheme to a CCA2-secure one [11] and prove it transforms a IK-CPA private PKE scheme to a IK-CCA private PKE scheme.

Let λ be the security parameter and n, N, ℓ be polynomials of λ. Let $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme with message space $\{0,1\}^n$. Denote the process "under global parameter GP, encrypts plaintext μ under public key pk with randomness r" by $\mathsf{Enc}(\mathsf{GP}, pk, \mu; r)$. Assume the algorithm Enc takes at most ℓ random bits and let $G : \{0,1\}^n \to \{0,1\}^N, H : \{0,1\}^{N+n} \to \{0,1\}^\ell$, be random oracles. The $\mathsf{PKE2} = (\mathsf{Setup2}, \mathsf{KeyGen2}, \mathsf{Enc2}, \mathsf{Dec2})$ scheme is defined as:

– $\mathsf{PKE2.Setup2}(1^\lambda)$: The algorithm runs $\mathsf{GP} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and sets $\ell = \mathrm{poly}(\lambda)$, $N = \mathrm{poly}(\lambda)$. Let fix G, H and outputs $\mathsf{GP2} = (\mathsf{GP}, \ell, N, G, H)$.
– $\mathsf{KeyGen2}(\mathsf{GP2})$: The algorithm runs $(pk', sk') \leftarrow \mathsf{KeyGen}(\mathsf{GP})$. It outputs $pk = pk'$ as public key, outputs $sk = sk'$ as secret key.
– $\mathsf{Enc2}(\mathsf{GP2}, pk, \mu)$: For a public key $pk = pk'$ and a plaintext $\mu \in \{0,1\}^N$, the algorithm chooses $\sigma \leftarrow \{0,1\}^n$. It computes $w = G(\sigma) \oplus \mu$, $d \leftarrow \mathsf{PKE.Enc}(\mathsf{GP}, pk', \sigma; H(\mu, \sigma))$ and outputs $c = (d, w)$ as ciphertext.
– $\mathsf{Dec2}(\mathsf{GP2}, c, pk, sk)$: For a public key $pk = pk'$, a secret key $sk = sk'$ and a ciphertext $c = (d, w)$, the algorithm computes $\sigma \leftarrow \mathsf{Dec}(\mathsf{GP2}, sk', d)$ and $\mu = G(\sigma) \oplus w$. It outputs μ if $d = \mathsf{Enc}(\mathsf{GP}, pk', \sigma; H(\mu, \sigma))$, otherwise it outputs $\perp$.

Theorem 6. *Let* PKE *be a PKE scheme with CPA-security and IK-CPA privacy,* $\mathsf{PKE2}$ *is IK-CCA private.*

Proof. We prove by reduction. Let $\mathcal{A}$ be any PPT algorithm that breaks the IK-CCA private of $\mathsf{PKE2}$, we construct a PPT algorithm $\mathcal{B}$ with $\mathcal{A}$ as a subroutine breaks the IK-CPA privacy of PKE.

On receiving the challenge keys $\mathsf{GP}, pk_0', pk_1'$ s.t $(pk_0', sk_0') \leftarrow \mathsf{PKE.KeyGen}(\mathsf{GP})$ and $(pk_1', sk_1') \leftarrow \mathsf{PKE.KeyGen}(\mathsf{GP})$, $\mathcal{B}$ computes and sends $\mathsf{GP2}, pk_0 = pk_0', pk_1 = pk_1'$ to $\mathcal{A}$. In the $\mathsf{Game}_{\mathsf{ikcca}}$ of $\mathcal{A}$ for $\mathsf{PKE2}$, the decryption oracles of $\mathcal{A}$ are $\mathsf{ODec2}(\cdot,\cdot)$, which equal to $\mathsf{Dec2}(\mathsf{GP2}, \cdot, pk_i, sk_i)$ respectively. A query to $\mathsf{ODec2}(\cdot,\cdot)$ on $c_j = (d_j, w_j)$ defines values $\sigma_j = \mathsf{PKE.Dec}(\mathsf{GP}, k_i, d_j), \mu_j = G(\sigma_j) \oplus w_j$, where $i \in \{0,1\}$. We define the following events:
•$\mathtt{inv}_j$: For j-th query to $\mathsf{ODec2}(\cdot,\cdot)$ on $(i, c_j = (d_j, w_j))$, it replies $\perp$.
•$\mathtt{gue}_j$: Before the j-th query to $\mathsf{ODec2}(\cdot,\cdot)$ on $(i, c_j = (d_j, w_j))$, $G(\sigma_j)$ and $H(\mu_j, \sigma_j)$ are not queried, where $\sigma_j = \mathsf{Dec}(\mathsf{GP}, d_j, pk_j', sk_i')$ and $\mu_j = w_j \oplus G(\sigma_j)$.

•$\mathtt{exp}_j$: Defined to be the event $\mathtt{gue}_j \wedge \overline{\mathtt{inv}_j}$.
•$\mathtt{exp}$: Any of $\mathtt{exp}_j$ happen in the whole kd-cca game.

We prove that $\mathtt{exp}$ occurs with negligible probability. Assume that $\mathtt{exp}_j$ occurs, then c_j decrypts to μ_j. Without querying on $G(\sigma_j)$, $w_j = G(\sigma_j) \oplus \mu_j$ is uniformly random to $\mathcal{A}$. In order to achieve $\overline{\mathtt{inv}_j}$, $\mathcal{A}$ has to guess w_j right, but this happens with negligible probability. If $\mathcal{A}$ has queried $G(\sigma_j)$, then it hasn't queried $H(\mu_j, \sigma_j)$. The randomness $H(\mu_j, \sigma_j)$ in the encryption process $\mathsf{PKE.Enc}(\mathsf{GP}, pk_i, \mu_j)$ is uniformly random. Then $\mathcal{A}$ has to compute $d_j = \mathsf{Enc}(\mathsf{GP}, pk_i', \mu_j; H(\mu_j, \sigma_j))$ right with a uniformly random string $r_j \leftarrow \{0,1\}^\ell$ instead of $H(\mu_j, \sigma_j)$. This happens with negligible property, otherwise in the cpa game, one could try to encrypts the challenge plaintext to ciphertext with uniform randomness to break the CPA-security of PKE. The union bound of $\mathtt{exp}_j$ shows $\mathtt{exp}$ happens with negligible probability. Therefore, it holds that

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins} \wedge \overline{\mathtt{exp}}] + \Pr[A \text{ wins} \wedge \mathtt{exp}] \le \Pr[\mathcal{A} \text{ wins} \wedge \overline{\mathtt{exp}}] + \Pr[\mathtt{exp}] \\ &= \Pr[\mathcal{A} \text{ wins} \wedge \overline{\mathtt{exp}}] + \mathrm{negl}(n)\end{aligned}$$

Assume $\mathtt{Adv}^{ikcca}_{\mathcal{A},\mathsf{PKE2}} = |\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}|$ is non-negligible, then $|\Pr[\mathcal{A} \text{ wins} \wedge \overline{\mathtt{exp}}] - \frac{1}{2}|$ is non-negligible. $\mathcal{B}$ is committed to win its kd-cpa game when $[\mathcal{A} \text{ wins} \wedge \overline{\mathtt{exp}}]$ happens. $\mathcal{B}$ simulates the oracles G, H by uniformly sampling and recording to list L_G, L_H, similar to the strategy of G_1, G_2 in the proof of Theorem 4. When $\mathcal{B}$ receives the challenge plaintext $\mu^* \in \{0,1\}^N$ from $\mathcal{A}$, it samples a $\sigma^* \leftarrow \{0,1\}^n$ as its challenge plaintext. On receiving challenge ciphertext $d^* = \mathsf{PKE.Enc}(\mathsf{GP}, pk_b', \sigma^*; r^*)$ for some $r^* \leftarrow \{0,1\}^\ell$, $\mathcal{B}$ sends $c^* = (d^*, w^*)$ to $\mathcal{A}$ as challenge ciphertext, where $w^* = G(\sigma^*) \oplus \mu^*$. For $\mathcal{A}$'s j-th query to $\mathsf{ODec2}(\cdot,\cdot)$ on $c_j = (d_j, w_j)$, $\mathcal{B}$ scans the whole L_G, L_H. If $\mathcal{B}$ finds some $(\sigma_j, G(\sigma_j)) \in L_G$ and $(\mu_j, \sigma_j, H(\mu_j, \sigma_j)) \in L_H$ s.t. $G(\sigma_j) \oplus \mu_j = w_j$ and $d_j = \mathsf{PKE.Enc}(\mathsf{GP}, pk_i, \mu_j, H(\mu_j, \sigma_j))$, it replies with μ_j, otherwise replies $\perp$.

When $\mathcal{A}$ outputs its guess b' to b, $\mathcal{B}$ outputs b'. Conditioned on that $\overline{\mathtt{exp}}$ occurs, $\mathcal{B}$ can answer all the decryption queries, and the view of $\mathcal{A}$ in the reduction is identical to that in the real game. Therefore, we have $\Pr[\mathcal{B} \text{ wins}] \ge \Pr[\mathcal{A} \text{ wins} \wedge \overline{\mathtt{exp}}] - \mathrm{negl}(n)$. Then $\mathsf{Adv}^{ikcpa}_{\mathcal{B},\mathsf{PKE}} = |\Pr[\mathcal{B} \text{ wins}] - \frac{1}{2}| = |\Pr[\mathcal{A} \text{ wins} \wedge \overline{\mathtt{exp}}] - \frac{1}{2}| - \mathrm{negl}(n)$, which is non-negligible if $\mathtt{Adv}^{ikcca}_{\mathcal{A},\mathsf{PKE}}$ is non-negligible.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
2. Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 98–115. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_6
3. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: 28th ACM STOC, pp. 99–108 (1996)

4. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: STACS 2009, pp. 75–86. LIPIcs (2009)
5. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_33
6. Ben-Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474 (2014)
7. Boyen, X.: Lattice mixing and vanishing trapdoors: a framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_29
8. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
9. CoinMarketCap: Top 100 cryptocurrencies by market capitalization. https://coinmarketcap.com. Accessed 12 Feb 2020
10. Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: new techniques for shorter and faster constructions and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 115–146. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_5
11. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_34
12. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40th ACM STOC, pp. 197–206 (2008)
13. Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S.: TumbleBit: an untrusted bitcoin-compatible anonymous payment hub. In: NDSS 2017 (2017)
14. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions (extended abstracts). In: 21st ACM STOC, pp. 12–24 (1989)
15. Liu, Z., Nguyen, K., Yang, G., Wang, H., Wong, D.S.: A lattice-based linkable ring signature supporting stealth addresses. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019. LNCS, vol. 11735, pp. 726–746. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29959-0_35
16. Liu, Z., Yang, G., Wong, D.S., Nguyen, K., Wang, H.: Key-insulated and privacy-preserving signature scheme with publicly derived public key. In: IEEE EuroS&P 2019, pp. 215–230 (2019)
17. Meiklejohn, S., et al.: A fistful of bitcoins: characterizing payments among men with no names. Commun. ACM **59**(4), 86–93 (2016)
18. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: 45th FOCS, pp. 372–381 (2004)
19. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed E-cash from Bitcoin. In: 2013 IEEE Symposium on Security and Privacy, pp. 397–411 (2013)
20. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2009). http://www.bitcoin.org/bitcoin.pdf
21. Noether, S., Mackenzie, A.: Ring confidential transactions. Ledger **1**, 1–18 (2016)
22. Peikert, C.: A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939 (2015). http://eprint.iacr.org/2015/939
23. 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). https://doi.org/10.1007/11681878_8

24. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 89–114. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_4
25. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: 37th ACM STOC, pp. 84–93 (2005)
26. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 6–24. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_2
27. Ruffing, T., Moreno-Sanchez, P., Kate, A.: CoinShuffle: practical decentralized coin mixing for bitcoin. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 345–364. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11212-1_20
28. van Saberhagen, N.: Cryptonote v 2.0 (2013). https://cryptonote.org/whitepaper.pdf
29. Todd, P.: Stealth addresses. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-January/004020.html
30. Torres, W.A.A., et al.: Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice RingCT v1.0). In: ACISP 2018, pp. 558–576 (2018)
31. Wuille, P.: Bip32: hierarchical deterministic wallets, February 2012. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki
32. Yang, R., Au, M.H., Zhang, Z., Xu, Q., Yu, Z., Whyte, W.: Efficient lattice-based zero-knowledge arguments with standard soundness: construction and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 147–175. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_6