2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P) | 978-1-6654-6512-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/EuroSP57164.2023.00059

Asynchronous Remote Key Generation for Post-Quantum Cryptosystems from Lattices

Nick Frymann
Surrey Centre for Cyber Security
University of Surrey
Guildford, United Kingdom
n.frymann@surrey.ac.uk

Daniel Gardham
Surrey Centre for Cyber Security
University of Surrey
Guildford, United Kingdom
daniel.gardham@surrey.ac.uk

Mark Manulis
Research Institute CODE
Universität der Bundeswehr München
Munich, Germany
mark@manulis.eu

***Abstract*—Asynchronous Remote Key Generation (ARKG), introduced by Frymann et al. at CCS 2020, allows for the generation of unlinkable public keys by third parties, for which corresponding private keys may be later learned only by the key pair's legitimate owner. These key pairs can then be used in common public-key cryptosystems, including signatures, PKE, KEMs, and schemes supporting delegation, such as proxy signatures. The only known instance of ARKG generates discrete-log-based keys.**

In this paper, we introduce new ARKG constructions for lattice-based cryptosystems. The key pairs generated using our ARKG scheme can be applied to lattice-based signatures and KEMs, which have recently been selected for standardisation in the NIST PQ process, or as alternative candidates.

In particular, we address challenges associated with the noisiness of lattice hardness assumptions, which requires a new generalised definition of ARKG correctness, whilst preserving the security and privacy properties of the former instantiation. Our ARKG construction uses key encapsulation techniques by Brendel et al. (SAC 2020) coined *Split KEMs*. As an additional contribution, we also show that Kyber (Bos et al., EuroS&P 2018) can be used to construct a Split KEM. The security of our protocol is based on standard LWE assumptions. We also discuss its use with selected candidates from the NIST process and provide an implementation and benchmarks.

1. Introduction

1.1. ARKG and its applications

Asynchronous Remote Key Generation (ARKG) [1] is a key generation protocol that allows for creation of derived public keys, that are associated with original public key. A corresponding derived secret key can be computed later in time, or asynchronously, using only the derived public key, the original secret key and some auxiliary information. All derived public keys remain unlinkable with respect to the original public key, and derivation of a derived secret cannot happen without knowledge of the original secret key. Moreover, only users for whom public keys were derived may compute corresponding private keys when provided with auxiliary data, which may be public. Key pairs computed with ARKG are designed to have statistically-similar distributions to randomly generated key pairs, and thus can be securely used in public key cryptosystems.

ARKG was proposed for an application in WebAuthn [2], forming part of FIDO2, which is a W3C[1] standard for challenge-response authentication on the web using digital signatures. Its envisioned use is to enable recovery and delegation of WebAuthn accounts. In WebAuthn, authenticators, such as Yubico's YubiKey or Windows Hello, manage and store private keys on behalf of users, which means account recovery becomes problematic after loss or damage to an authenticator may render a user's online accounts inaccessible. ARKG produces unlinkable public keys that satisfy WebAuthn's strong unlinkability requirement, which means users can use a single authenticator when registering for different accounts, yet remain unlinkable across services. Furthermore, the remote feature of this protocol means that the 'backup' authenticator does not need to be present at account creation. More recently, ARKG was used in a proxy signature scheme that supports unlinkability across delegations, in a protocol called Proxy Signatures with Unlinkable Warrants [3]. It offers a delegator the ability to delegate signing rights to the proxy signer using a delegation-by-warrant approach—in which a public key for the proxy is signed by the delegator. These proxy signatures were used to enable delegation for WebAuthn accounts. The use of ARKG as a building block results in warrants that are unlinkable with respect to the proxy signer, and the asynchronicity allows the delegation process to be non-interactive.

1.2. Post-quantum ARKG and main challenges

The only known[2] construction for ARKG is for discrete-log key pairs, i.e. of the form $(\mathsf{sk}, \mathsf{pk}) = (x, g^x)$ for some group generator g, which are known to be susceptible to quantum attackers [5]. This threat has motivated huge efforts to create protocols that are resistant to quantum attacks. As such, NIST[3] have recently selected primitives to standardise post-quantum cryptography based on a range of differing hard problems including isogenies, multivariate polyonmials and code-based problems. Many of the candidates submitted to the process are based on

1. https://www.w3.org/
2. We note new ARKG constructions for pairing-based cryptosystems introduced concurrently by Frymann et al. at ACNS 2023 [4].
3. https://csrc.nist.gov/Projects/post-quantum-cryptography

DOI 10.1109/EuroSP57164.2023.00059

lattices, such as the signatures in [6]–[8] and KEMs and PKEs in [9]–[11]. Moreover, 3 of the 4 candidates selected for standardisation fall into this category, which is in part due to their relatively more mature security analysis and wide versatility. The focus of our work is, therefore, in the design of suitable ARKG constructions for lattice-based cryptosystems, including those from the current NIST PQ standardisation process.

However, lattices introduce new problems that are not inherent in group-based cryptography. Namely, one of the major security assumptions, Learning with Errors (LWE) [12] critically relies on introducing noise for its security. This noise can affect correctness properties of protocols, and requires careful parameter selection to balance correctness with security. This adds additional complexity to our work as we aim to derive keys compatible with other cryptosystems for which the structure and parameter values of the keys may differ. For example, KEMs and PKE typically rely on LWE based problems where as signatures tend to use the assumed hardness of the Short Integer Solution problem, or SIS. In this case, the distribution of the secret key is not so important (provided there is enough entropy) but the size now plays a critical role. Balancing these two requirements is a new challenge for ARKG in the lattice-based setting.

1.3. Contributions

In this work, we give a new generic construction of ARKG allowing for the use of the primitive with lattice-based public key cryptosystems. We aim to capture a wide range of lattice protocols despite differing requirements on their keys. Lattice schemes based on Short Integer Solution (SIS) [13] often use bounded short vectors for their keys, whereas KEMs/PKEs based on Learning with Errors (LWE) [12] typically require keys with a specific and proper distribution (typically discretised Gaussian). Our ARKG is the first to support LWE and SIS based-schemes, encompassing a wide range of potential applications. We build this general lattice ARKG (LARKG) using Split KEM (sKEM) [14] functionality to achieve the required asynchronicity.

Whilst maintaining ARKG's original security and privacy properties, we overcome the inherent challenges with lattice assumptions' noisiness. The need to select cryptographically hard parameters for composed protocols means consideration must be given to the correctness versus security trade-off–this can be challenging for signature schemes using SIS because, as the norm of the key increases, security is weakened. Similarly, PKE typically requires well distributed secret and error vectors. To address this, we introduce a new definition of correctness, that we call τ-correctness that captures the probability that some derived keys may fail or be too large. This parameter is related to the *repetition rate* of the rejection sampling techniques inherent in our generic construction. These parameters can be varied to trade-off correctness against efficiency.

Additionally, we provide two instantiations of our LARKG based on Frodo [9] and Kyber [10]. Since Kyber has previously not known to have sKEM functionality, we also state how to construct this and give proof it has security properties of an sKEM. We give a discussion on potential applications in lattice-based KEMs, signatures, and encryption schemes. In particular, we discuss integration with Dilithium [6] and Kyber, which have recently been selected for standardisation by the NIST PQC process. We provide benchmarking of both of our instantiations using a publicly available implementation in Python. This performance analysis shows that our implmentations achieve practical performance when generating derived key pairs, compared to the standard key generation algorithms.

1.4. Organisation

We start in Section 2 by recalling the ARKG primitive, with a new correctness definition to make it compatible with lattices. In Section 3, we define the necessary preliminaries, building blocks and security assumptions before we present our general lattice-based ARKG construction and security analysis. Section 4 presents two instantiations of LARKG and discusses its usage in lattice-based KEMs, PKE, and signatures. In this section we also show how to construct an sKEM based on Kyber, and give proof it meets standard security properties of an sKEM. We conclude this section with implementation details and benchmarks of our protocol. Our conclusion and a future work discussion is presented in Section 5.

2. Asynchronous Remote Key Generation

In this section we recall Asynchronous Remote Key Generation based on the model introduced by Frymann et al [1]. We give the syntax and security properties, and modify the definition of correctness to account for some failure probability.

2.1. ARKG Model

For the key pair $(\mathsf{sk}, \mathsf{pk})$ and credential cred, ARKG allows arbitrary public keys pk' to be derived from an original pk , with corresponding sk' being calculated at a later time—requiring private key sk .

Definition 1 (ARKG [1])**.** *The asynchronous remote key generation scheme* $\mathsf{ARKG} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DerivePK}, \mathsf{DeriveSK}, \mathsf{Check})$ *consists of the following five algorithms:*

- $\mathsf{Setup}(\lambda)$ *generates and outputs public parameters* pp *of the scheme for the security parameter* $\lambda \in \mathbb{N}$*. We assume* pp *is taken as implicit input to all algorithms.*
- $\mathsf{KeyGen}()$ *computes and returns a key pair* $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{DerivePK}(\mathsf{pk}, \mathsf{aux})$ *probabilistically returns a new public key* pk' *together with the link* cred *between* pk *and* pk'*, for the inputs* pp*,* pk *and auxiliary data* aux*. The input* aux *is always required but may be empty.*
- $\mathsf{DeriveSK}(\mathsf{sk}, \mathsf{cred})$*, computes and outputs either the new private key* sk'*, corresponding to the public key* pk' *using* cred*, or* $\perp$.
- $\mathsf{Check}(\mathsf{sk}', \mathsf{pk}')$*, on input* $(\mathsf{sk}', \mathsf{pk}')$*, returns 1 if* $(\mathsf{sk}', \mathsf{pk}')$ *forms a valid private-public key pair, where* sk' *is the corresponding private key to public key* pk'*, otherwise 0.*

Correctness. We modify the correctness property from Frymann et al. [1] to allow for some probability of failure that DeriveSK does not output a valid derived secret

key after correct execution of a corresponding DerivePK. We capture this failure probability with the correctness parameter τ. Note that our definition is consistent with Frymann et al. when $\tau \to \infty$. In this case we say the scheme has perfect correctness.

An ARKG scheme is τ-correct if, $\forall\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, the probability $\Pr\big[(\mathsf{Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}') = 1)\big] \geqslant 1 - 2^{-\tau}$ given

$$\begin{aligned}(\mathsf{sk}, \mathsf{pk}) &\leftarrow \mathsf{KeyGen}(\mathsf{pp});\\ (\mathsf{pk}', \mathsf{cred}) &\leftarrow \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}, \cdot);\\ \mathsf{sk}' &\leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred}).\end{aligned}$$

2.1.1. Security Properties. We recall the security properties for ARKG. The first, PK-unlinkability, is an anonymity property whereas unforgeability is provided by SK-security. This property comes in 4 flavours to capture a wide range of adversarial capabilities. Furthermore, we introduce generalised security definitions to accommodate the possibility that DeriveSK returns an error $\perp$ instead of a derived secret key. This can be seen in the challenge and private key oracles. *Adversaries and Oracles*

An adversary $\mathcal{A}$, used in our security experiments, is modelled as a probabilistic polynomial time (PPT) algorithm. The adversary $\mathcal{A}$ may make a polynomial number of queries to the following oracles:

- Derived public key oracle $\mathcal{O}_{\mathsf{pk}'}(\mathsf{pk}, \cdot)$: $\mathcal{O}_{\mathsf{pk}'}$ is parameterised with public key pk. This oracle returns the result of calling $\mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}, \mathsf{aux})$ on input aux. It records the resulting $(\mathsf{pk}', \mathsf{cred})$ in PKList: $\mathsf{PKList} \leftarrow \mathsf{PKList} \cup (\mathsf{pk}', \mathsf{cred})$. PKList is initialised as $\mathsf{PKList} \leftarrow \emptyset$.
- Challenge oracle $\mathcal{O}^b_{\mathsf{pk}'}(b, \mathsf{sk}_0, \mathsf{pk}_0)$: $\mathcal{O}^b_{\mathsf{pk}'}$ is parameterised with a bit b and fixed key pair $(\mathsf{sk}_0, \mathsf{pk}_0)$, and takes no inputs. If $b = 0$, the key-pair $(\mathsf{sk}', \mathsf{pk}')$ derived using the initial pk_0 using DerivePK and DeriveSK. If DeriveSK returns $\perp$ then a DerivePK is invoked again to create a new derived key, and this is repeated until success. If $b = 1$, then a freshly-generated key pair sampled from a distribution $\mathcal{D}$.
- Private key oracle $\mathcal{O}_{\mathsf{sk}'}(\mathsf{sk}, \cdot)$: on input cred, where $(\cdot, \mathsf{cred}) \in \mathsf{PKList}$, $\mathcal{O}_{\mathsf{sk}'}$ outputs the result of $\mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})$ and updates $\mathsf{SKList} \leftarrow \mathsf{SKList} \cup \mathsf{cred}$ if the output is not $\perp$. SKList is initialised as $\mathsf{SKList} \leftarrow \emptyset$. If $(\cdot, \mathsf{cred}) \notin \mathsf{PKList}$, the oracle aborts, otherwise it returns sk' without giving access to sk .

Remark 1. *We note that our security model does capture the possibility of timing attacks that might be present due to the variable run time of the challenge oracle. However, in the envisioned use case, such a timing attack would imply adversarial access to back up authenticators – which is beyond the security model of ARKG.*

SK-security. This security property ensures that an adversary $\mathcal{A}$ cannot derive a valid key pair $(\mathsf{sk}^\star, \mathsf{pk}^\star)$ along with corresponding $\mathsf{cred}^\star$, for a fixed challenge key pk . We recall four variants of private-key security defined in Frymann et al., modelled using the experiment $\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{LARKG},\mathcal{A}}(\lambda)$ in Figure 1 with $\mathsf{ks} \in \{\mathsf{mwKS}, \mathsf{hwKS}, \mathsf{msKS}, \mathsf{hsKS}\}$. Adversary $\mathcal{A}$ is always given access to $\mathcal{O}_{\mathsf{pk}'}$ and must find a $(\mathsf{sk}^\star, \mathsf{pk}^\star, \mathsf{cred}^\star)$ triple for a provided pk . We have altered the original definition to include a check that the derived secret key sk' is not $\perp$, this is to account for the fact that there is some probability that DeriveSK fails. The malicious (m) and honest (h) variants result from the omission or presence of the PKList check on line 8, respectively, which ensures that the triple is for an honestly-generated pk (modelled using $\mathcal{O}_{\mathsf{pk}'}$) if present. The weak (w) and strong (s) variants depend on whether $\mathcal{A}$ has access to the private key derivation oracle $\mathcal{O}_{\mathsf{sk}'}$. If $\mathcal{A}$ has access to $\mathcal{O}_{\mathsf{sk}'}$, trivially querying it with $\mathsf{cred}^\star$ is prevented through the SKList check on line 7. It was shown by Frymann et al. that strong implies weak security, and malicious implies honest security. We defer to the original work for the reduction.

Definition 2 (SK-security)**.** *An ARKG scheme provides private-key security, or* ks*-security with* $\mathsf{ks} \in \{\mathit{mwKS}, \mathit{hwKS}, \mathit{msKS}, \mathit{hsKS}\}$*, if the following advantage is negligible in* λ*:*

$$\mathsf{Adv}^{\mathsf{ks}}_{\mathsf{LARKG},\mathcal{A}}(\lambda) \coloneqq \Pr\Big[\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{LARKG},\mathcal{A}}(\lambda) = 1\Big].$$

PK-unlinkability. This property ensures that derived key pairs cannot be distinguished from a sample of a distribution $\mathcal{D}$, which also implies an adversary cannot link a derived public key to a long-term public key or second derived public key. This property is formally defined in $\mathsf{Exp}^{\mathsf{pku}}_{\mathsf{LARKG},\mathcal{A}}(\lambda)$. Intuitively, the game chooses a bit $b \in \{0, 1\}$ and generates a key pair $(\mathsf{sk}_0, \mathsf{pk}_0)$. $\mathcal{A}$ is given access to oracle $\mathcal{O}^b_{\mathsf{pk}'}$, public parameters pp, and pk_0. When called, $\mathcal{O}^b_{\mathsf{pk}'}$ returns a derived key pair $(\mathsf{sk}', \mathsf{pk}')$, which is derived from pk_0 if $b = 0$, otherwise, for $b = 1$, it samples and returns key pair $(\mathsf{sk}', \mathsf{pk}')$ according to a distribution $\mathcal{D}$. It is able to corrupt any derived key, for which corresponding sk' is output. The adversary $\mathcal{A}$ wins the game if it is able to determine whether $\mathcal{O}^b_{\mathsf{pk}'}$ is instantiated with $b = 0$ or $b = 1$.

Definition 3 (PK-unlinkability)**.** *An ARKG scheme provides PK-unlinkability if the following advantage is negligible in* λ*:*

$$\mathsf{Adv}^{\mathsf{pku}}_{\mathsf{LARKG},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\Big[\mathsf{Exp}^{\mathsf{pku}}_{\mathsf{LARKG},\mathcal{A}}(\lambda) = 1\Big] - \frac{1}{2}\right|.$$

Remark 2. *As previously mentioned, Frymann et al. [1] give a construction for discrete-log key pairs. They achieve weak variants of the scheme (i.e.* mwKS *and* hwKS*) under the discrete log assumption. For stronger flavours,* msKS *and* hsKS*, security is reduced to the snPRF-ODH assumption by Brendel et al. [15], which they introduce to study the security of TLS1.3.*

3. General Lattice-based ARKG Construction from Split KEMS

In this section we define sKEMs and the other necessary building blocks for our general ARKG construction. We present our construction in Figure 5 and prove its security.

3.1. Assumptions and Building Blocks

We now introduce the technical background, hardness assumptions and the building blocks used in this work.

$\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{LARKG},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$
2 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
3 : $(\mathsf{sk}^\star, \mathsf{pk}^\star, \mathsf{cred}^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{pk}'}, \boxed{\mathcal{O}_{\mathsf{sk}'}}}(\mathsf{pp}, \mathsf{pk})$
4 : $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred}^\star)$
5 : **return** $\mathsf{Check}(\mathsf{sk}^\star, \mathsf{pk}^\star) \stackrel{?}{=} 1$
6 : $\quad \wedge\, \mathsf{Check}(\mathsf{sk}', \mathsf{pk}^\star) \stackrel{?}{=} 1$
7 : $\quad \wedge\, \mathsf{sk}' \neq \bot$
8 : $\quad \boxed{\wedge\, \mathsf{cred}^\star \notin \mathsf{SKList}}$
9 : $\quad \boxed{\wedge\, (\mathsf{pk}^\star, \mathsf{cred}^\star) \in \mathsf{PKList}}$

(a) SK-security experiment.

$\mathsf{Exp}^{\mathsf{pku}}_{\mathsf{LARKG},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
3 : $b \leftarrow\$ \{0, 1\}$
4 : $b' \leftarrow \mathcal{A}^{\mathcal{O}^b_{\mathsf{pk}'}}(\mathsf{pp}, \mathsf{pk}_0)$
5 : **return** $b \stackrel{?}{=} b'$

(b) PK-unlinkability experiment.

Figure 1: Security experiments for LARKG. The boxes denote the four variants of the $\mathsf{ks} \in \{\mathsf{mwKS}, \mathsf{hwKS}, \mathsf{msKS}, \mathsf{hsKS}\}$ experiment. Presence of the dashed boxes gives the strong variants of ks(msKS and hsKS), the presence of the dotted box gives the honest variants (hwKS and hsKS), and the exclusion of all boxes gives mwKS.

On notation, we use $\chi^{n\times m}_{\alpha,\gamma}$ to denote an $n \times m$ matrix that is element-wise sampled from a distribution χ with parameters (α, γ). In this work, we utilise Gaussian distributions $\chi_{\alpha,\gamma}$ where α is the standard deviation and γ is the centre. If we omit γ then it is understood to be 0. We use $\chi(X)$ to denote the probability of X occuring under the distribution χ. We reserve the notation $\mathcal{U}[a, b]$ to indicate uniform sampling from the closed interval $[a, b]$. Matrices are denoted in bold capital letters, e.g. $\mathbf{C}$, and vectors are notated as bold lowercase letters such as $\mathbf{x}$. In this work we consider a polynomial ring $\mathcal{R}_q := \mathbb{Z}_q[X]/(X^n + 1)$ where $n = 2^{\hat{n}-1}$ and $\hat{n}$ is the $2^{\hat{n}}$-th cyclotomic polynomial. We use $x \leftarrow\$ X$ to denote that x is uniformly sampled from X, whereas Gaussian sampling is denoted $x \leftarrow_G X$.

Definition 4. *Decisional Learning with Errors (LWE). For LWE parameters n, q, χ and a matrix $\mathbf{A}$ sampled uniformly from $\mathbb{Z}_q^{n\times m}$, sample $\mathbf{e} \leftarrow \chi$ and a vector $\mathbf{s} \in \mathbb{Z}_q^n$. In the decisional variant of LWE, an adversary must decide, for a tuple $(\mathbf{A}, \mathbf{b})$, whether $\mathbf{b} = \mathbf{As} + \mathbf{e} \mod q$ or $\mathbf{b}$ is a uniform sample from $\mathbb{Z}_q^m$. This can be extended to the Matrix-LWE by taking $\mathbf{S}, \mathbf{E} \leftarrow_G \chi^{m\times k}$.*

We do not distinguish between these two variants since hardness of LWE implies hardness of Matrix-LWE.

Definition 5. *Decisional Module Learning with Errors (MLWE) [16]. For parameters m, k, χ the MLWE problem is stated for a matrix $\mathbf{A}$ sampled uniformly from $\mathcal{R}_q^{k\times k}$ and $\mathbf{e}, \mathbf{s} \leftarrow \chi_\eta^k$. The problem requires an adversary to decide, for a tuple $(\mathbf{A}, \mathbf{b})$, whether $\mathbf{b} = \mathbf{As} + \mathbf{e}$ or $\mathbf{b}$ is a uniform sample from $\mathcal{R}_q^k$. MLWE becomes Ring-LWE (RLWE) when the dimension of the ring is set to 1, i.e. when the parameter $k = 1$.*

Definition 6. *Short Integer Solution (SIS) [13]. The SIS problem $\mathsf{SIS}_{n,m,q,\beta}$ requires an adversary, given a uniformly sampled matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, to find a non-zero vector $\mathbf{z} \in \mathbb{Z}_q^m$ such that $\|\mathbf{z}\| \leqslant \beta$ and $\mathbf{Az} = \mathbf{0} \mod q$.*

Definition 7. *Module Short Integer Solution (MSIS) [16]. The MSIS problem $\mathsf{MSIS}_{n,m,q,\beta}$ requires an adversary who, given a uniformly sampled matrix $\mathbf{A} \in \mathcal{R}_q^{k\times m}$ to find a non-zero vector $\mathbf{z} \in \mathcal{R}_q^k$ such that $\|\mathbf{z}\| \leqslant \beta$ and $\mathbf{Az} = \mathbf{0} \in \mathcal{R}_q^k$. MSIS becomes Ring-SIS (RSIS) when the parameter k is set to 1.*

Rejection Sampling. This is a technique that ensures we can control the shape of distributions, first used in lattice cryptography by Lyubashevsky [17]. Let f and g be probability distributions and $M \in \mathbb{R}$ a scalar constant, such that for all x we have $f(x) \leqslant Mg(x)$. Then, one can sample y from g and output it with probability $f(y)/Mg(y)$, and the resulting distribution of output values is precisely f. The *repetition rate* M is the expected amount of time needed to output a sample.

Split Key Encapsulation Mechanism. The fundamental building block for our ARKG protocol is a *Split Key Encapsulation Mechanism*, or sKEM. It is related to standard notion of a KEM, where the encapsulation algorithm can be divided into two phases for key generation and shared-key computation. This primitive was introduced by Brendel et al. [14] and used to propose quantum secure key exchange for the Signal protocol [18]. We recall their model and security definitions.

Definition 8 (sKEM [14])**.** *An* sKEM *consists of four algorithms* KeyGenEnc,KeyGenDec, sEncaps *and* sDecaps, *where* KeyGenEnc *and* sEncaps *are executed by the encapsulator, and* KeyGenDec *and* sDecaps *by the decapsulator.*

- KeyGenEnc *is the probabilistic key generation algorithm for the encapsulator* $(E, e) \leftarrow \mathsf{KeyGenEnc}(1^\lambda)$.
- KeyGenDec *is the probabilistic key generation algorithm for the decapsulator* $(D, d) \leftarrow \mathsf{KeyGenDec}(1^\lambda)$.
- $\mathsf{sEncaps}(e; D)$ *is a probabilistic algorithm executed by the encapsulator. It takes as input $e \in \mathcal{K}_{enc}$, the secret key of the encapsulator, and $D \in \mathcal{K}_{dec}$, the public key of the decapsulator. Algorithm* sEncaps *then outputs the shared secret $K \in \mathcal{K}$ along with its encapsulation $c \in C$, sometimes referred to as the ciphertext.*
- $\mathsf{sDecaps}(d; E; c)$ *is a deterministic algorithm executed by the decapsulator. On input a ciphertext c, the decapsulator's secret key d, and encapsulator's public key E, it outputs either the decapsulation K of c or $\bot$, if it fails.*

Security Properties. sKEMs can offer a range of indistinguishability properties, that are denoted as $\mathsf{l}\text{r-IND-CCA}$ where the parameters l, r are in the set $\{\mathsf{n}, \mathsf{s}, \mathsf{m}\}$. Here, l indicates whether the adversary is allowed to make no ($\mathsf{l} = \mathsf{n}$), a single ($\mathsf{l} = \mathsf{s}$), or polynomially many ($\mathsf{l} = \mathsf{m}$) queries to the decapsulation oracle $\mathcal{O}_{\mathsf{sDecaps}}$. Analogously, r indicates the number of queries the adversary is allowed to make to the encapsulation oracle $\mathcal{O}_{\mathsf{sEncaps}}$. The case that $\mathsf{r} = \mathsf{s}$ is excluded since the adversary cannot make

the encapsulator encapsulate only once more under the secret key e used for challenge generation. The key pair of the encapsulator is used either solely for the challenge generation or may be queried up to a bound which is $poly(\lambda)$. The 'many' case models key re-use attacks and has historically been a challenge for post-quantum cryptography. As noted by Brendel et al. [14], nn-IND-CCA security corresponds to the notion of IND-CPA security for KEMs, i.e. the adversary is only able to learn the public keys and challenge ciphertext and key, but remains passive. It is also restricted from querying the decapsulation oracle.

Definition 9. *Let* sKEM = *(*KeyGenEnc*;* KeyGenDec*;* sEncaps*;* sDecaps*) be a split KEM with key space* $\mathcal{K}$*. Let* $l, r \in \{n, s, m\}$*. We say* sKEM *provides* lr*-indistinguishability under chosen-ciphertext attacks, or for short,* sKEM *is* lr-IND-CCA *secure, if for every PPT adversary* $\mathcal{A}$ *the following advantage is negligble in* λ*.*

$$\mathsf{Adv}^{\mathsf{lr\text{-}IND\text{-}CCA}}_{\mathsf{sKEM},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\mathsf{Exp}^{\mathsf{lr\text{-}IND\text{-}CCA}}_{\mathsf{sKEM},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$$

In Figure 3, we recall a generic lattice-based construction from Brendel et al. [14], presented in Figure 3. Many lattice KEM candidates fit this generic 2-phase structure of generating ephemeral keys and then combing and reconciling a shared key [9], [19]–[21].

Key Derivation Function (KDF) [22] A key generation function $\mathsf{KDF}(k, l)$ takes source key k and label l and returns a new key k'. It is secure if the advantage $\mathsf{Adv}^{\mathsf{KDF}}_{\mathcal{A}}(\lambda)$ is negligible in λ for a PPT adversary $\mathcal{A}$ to distinguish outputs of the KDF from random bitstrings of the same length. In this work we consider uniform outputs of KDF_1 with small Gaussian outputs (paramterised by χ) and KDF_2 with uniform outputs.

Public Key Encryption (PKE) A PKE scheme consists of three algorithms KeyGen, Enc and Dec. The former outputs a key-pair based on a security parameters λ. The encryption algorithm takes as input a key k and plaintext m and returns a ciphertext c. Decrypt then outputs the plaintext m on input of a ciphertext and corresponding secret key. The game is formally defined in Figure 4.

It is IND-CPA secure if the advantage $\mathsf{Adv}^{\mathsf{PKE}}_{\mathcal{A}}(\lambda)$ is negligible in λ for a PPT adversary $\mathcal{A}$ to decide which of two chosen messages a ciphertext contains. Precisely, PKE is IND-CPA secure if the following advantage is negligible in λ.

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathsf{PKE}}(\lambda) \coloneqq \left|\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\lambda) = 1\right] - \frac{1}{2}\right|.$$

3.2. Our LARKG scheme based on Split KEMs

We now present our ARKG construction based on lattices. In our construction, key pairs $(\mathsf{pk}, \mathsf{sk})$ are given by the vectors of matrices $(\mathbf{A}, \mathbf{B} = \mathbf{AS}+\mathbf{E}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times k}$ which form an LWE sample. The derived key pair $(\mathsf{pk}', \mathsf{sk}')$, i.e. the outputs of DerivePK and DeriveSK respectively, are given by $\mathbf{P}$ and $\mathbf{S}''$ also form an LWE sample. For this construction, we focus on plain LWE and the simple case where the secret and error matrices are taken from the same distribution χ. This is for simplicity in presention, but we emphasise that our techniques readily extend to Ring-LWE and Module-LWE settings. Furthermore, the restricted distribution is not a requirement of our scheme and can be chosen independently according to the application.

The core building block of our construction is the sKEM primitive. The main benefit of using this over standard KEMs is that the flavours of security that allow for oracle queries capture the notion of key reuse. As our security theorems will show, security in the presence of key reuse allows us to achieve all, including the strongest, security flavours of SK security for ARKG.

Intuitively, we use a split KEM to create a shared secret K with recovery information $\mathbf{c}$. The format of sKEMs allows us to assume the structure of the KEM mechanism, in particular, that the encapsulation and decapsulation algorithms both take in public and secret keys. This enables an asynchronous approach to decapsulating a shared key, which forms the basis of our DerivePK and DeriveSK algorithms. The KDFs are used in one case to create a shared secret with appropriate distribution, i.e. a small Gaussian distribution χ. The other allows DeriveSK to abort since an incorrect long-term key or auxiliary information was use, and thus Check would fail.

Our protocol makes use of rejection sampling to ensure that output of DeriveSK follows a desired distribution. We note that flooding techniques, e.g. [23], could also be implemented here but at cost of increased parameter size. Instead, rejection sampling allows us to keep the parameters fixed and instead vary the correctness probability with repetition rate. That is, the protocol will need to be executed an expected M times to obtain a derived secret key.

We give a high-level description of the algorithms and present the full algorithms in Figure 5.

Setup This algorithm takes as input a security parameter λ and outputs public parameters pp that contains LWE parameters n, q, χ and $\mathbf{A} \leftarrow\$ \mathbb{Z}_q^{n\times m}$.

KeyGen On input of public parameters, this algorithm outputs a key-pair as $(\mathsf{sk}, \mathsf{pk})$ such that $\mathsf{sk} = \mathbf{S}$ is a small gaussian and $\mathsf{pk} = \mathbf{B} = \mathbf{A} \cdot \mathbf{S} + \mathbf{E}$. That is, $(\mathbf{A}, \mathbf{B})$ form an LWE sample.

DerivePK This algorithm generates a derived public key pk' from input of a public key $\mathsf{pk} = \mathbf{B}$. Intuitively, it samples an ephemeral key pair as $(\mathsf{sk}', \mathsf{pk}') = (\mathbf{S}', \mathbf{P} = \mathbf{A} \cdot \mathbf{S}' + \mathbf{E}')$. It computes a shared key K by using a split KEM with inputs $(\mathsf{sk}', \mathsf{pk}) = (\mathbf{S}', \mathbf{B})$, and uses this to derive a credential key $\mathbf{K}$. The derived public key is computed as $\mathsf{pk}' = \mathbf{B} + \mathbf{A} \cdot \mathbf{K} + \mathbf{E}'$. This step also outputs ciphertext $\mathbf{c}$. A second key derivation function KDF_2 is used to generate μ over the inputs $\mathbf{P}, \mathsf{aux}$ using KDF_2. It builds the recovery information as $\mathsf{cred} = (\mathbf{D}, \mathbf{c}, \mu)$.

DeriveSK This algorithm takes as input sk and cred. It recomputes the key K by executing the decapsulation algorithm on its sk and with $\mathbf{P}$. A candidate $\mu^* \leftarrow \mathsf{KDF}_2(K)$ is derived from K, and it is checked that $\mu^* \stackrel{?}{=} \mu$. If this holds, then it proceeds to recover its derived secret key as $\mathbf{K} \leftarrow \mathsf{KDF}_1(K)$ and setting $\mathsf{sk}' = \mathbf{S} + \mathbf{K}$. Finally, rejection sampling is performed, so that the key is only output with probability $\chi_{b,\mathbf{S}}(\mathbf{S}'')/(M \cdot \chi_a(\mathbf{S}))$, for some M.

Check Takes as input a candidate key pair $(\mathsf{sk}, \mathsf{pk}) = (\mathbf{S}, \mathbf{P})$ and returns 1 if $\|\mathbf{P} - \mathbf{A} \cdot \mathbf{S}\| \leqslant \beta$, where the

$\mathsf{Exp}^{lr\text{-}\mathsf{IND-CCA}}_{\mathsf{sKEM},\mathcal{A}}(\lambda)$

$n_l, n_r \leftarrow 0$
$(D, d) \leftarrow \mathsf{KeyGenDec}(\mathsf{pp})$
$(E, e) \leftarrow \mathsf{KeyGenEnc}(\mathsf{pp})$
$(c^*, K_0^*) \leftarrow \mathsf{sEncaps}(e, D)$
$K_1^* \leftarrow\$ \mathcal{K}$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sDecaps}}, \mathcal{O}_{\mathsf{sEncaps}}}(D, E, c^*, K_b^*)$
return $b' \stackrel{?}{=} b$

$\mathcal{O}_{\mathsf{sDecaps}}(E', c)$

if $n_l \geqslant l$
  return $\perp$
$n_l = n_l + 1$
if $(E', c) \stackrel{?}{=} (E, c^*)$
  return $\perp$
$L \leftarrow \mathsf{sDecaps}(d, E', c)$
return K

$\mathcal{O}_{\mathsf{sEncaps}}(D')$

if $n_r \geqslant r$
  return $\perp$
$n_r = n_r + 1$
$(c, K) \leftarrow \mathsf{sEncaps}(e, D')$
if $(D', c) \stackrel{?}{=} (D, c^*)$
  return $\perp$
else
  return (c, K)

Figure 2: Security experiments and oracles for sKEMs. The values of n_l and n_r are set to as either 0, 1 or k (which is $poly(\lambda)$) according to the $\{\mathsf{n}, \mathsf{s}, \mathsf{m}\}$-IND-CPA security, respectively.

$\mathsf{KeyGenEnc}(\mathsf{pp})$

$\mathbf{S}' \leftarrow\$ \chi^{m \times k}, \mathbf{E}' \leftarrow\$ \chi^{n \times k}$
$\mathbf{B}' \leftarrow \mathbf{A}\mathbf{S}' + \mathbf{E}'$
return $(\mathsf{sk}', \mathsf{pk}') := (\mathbf{S}', \mathbf{B}')$

$\mathsf{KeyGenDec}(\mathsf{pp})$

$\mathbf{S} \leftarrow\$ \chi^{m \times k}, \mathbf{E} \leftarrow\$ \chi^{n \times k}$
$\mathbf{B} \leftarrow \mathbf{S}\mathbf{A} + \mathbf{E}$
return $(\mathsf{sk}, \mathsf{pk}) := (\mathbf{S}, \mathbf{B})$

$\mathsf{sEncaps}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}')$

$\mathbf{E}'' \leftarrow \chi^{n \times k}$
$\mathbf{V} \leftarrow \mathbf{B}\mathbf{S}' + \mathbf{E}''$
$c \leftarrow \mathsf{HelpRec}(\mathbf{V})$
$K \leftarrow \mathsf{Rec}(\mathbf{V}, c)$
return (K, c)

$\mathsf{sDecaps}(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}, c)$

$\mathbf{V}' \leftarrow \mathbf{B}'\mathbf{S}$
$K \leftarrow \mathsf{Rec}(\mathbf{V}', c)$
return K

Figure 3: Lattice-based sKEM with generalised Rec and HelpRec algorithms by Brendel et al [14].

$\mathsf{Exp}^{\mathsf{IND-CPA}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$
$m_0, m_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
$\mathsf{c} \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b)$
$b' \leftarrow \mathcal{A}_2(c)$
return $b' \stackrel{?}{=} b$

Figure 4: IND-CPA security experiment for PKE.

distribution χ_a is bounded by β.

3.3. τ-Correctness

We observe that τ-correctness follows from correctness of the split KEM and that $\mathbf{P}$ is well formed:

$$\begin{aligned}\mathbf{P} &= \mathbf{B} + \mathbf{E} + \mathbf{A}\mathbf{K} + \mathbf{E}' = \mathbf{A}\mathbf{S} + \mathbf{A}\mathbf{K} + \mathbf{E} + \mathbf{E}' \\ &= \mathbf{A}(\mathbf{S} + \mathbf{K}) + \mathbf{E}'' = \mathbf{A}\mathbf{S}'' + \mathbf{E}''\end{aligned}$$

We note that our use of rejection sampling in DeriveSK ensures that the sum $\mathbf{S} + \mathbf{K}$ is statistically close to discrete Gaussian distribution χ_a, i.e. the distribution which $\mathcal{S}$ follows. If this is chosen to be the distribution for a compatible protocol, then the correctness is perfect and $t = \infty$. However, this may require a high repetition rate M. We can trade τ off against M to be most suitable for an application, the exact relationship for which is described by an appropriate choice of parameters and distributions. If we instantiate the error vectors to be sampled from Gaussian distributions as χ_a, χ_b, then the resultant distribution of the derived key $\mathbf{S}''$ is $\chi_{b,\mathbf{S}}$, i.e. a Gaussian distribution with variance b centred at $\mathbf{S}$. If χ_a is bounded by a value β_1, and $\sigma_b = \alpha\beta_1$ for some positive value α, then following [17], we can compute the repetition rate M as follows:

$$M = e^{\frac{12}{\alpha} + \frac{1}{2\alpha^2}}$$

$\mathsf{KeyGen}(\mathsf{pp})$

return $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{sKEM.KeyGenDec}(\mathsf{pp})$

$\mathsf{DerivePK}(\mathbf{B})$

$(\mathbf{S}', \mathbf{B}') \leftarrow \mathsf{sKEM.KeyGenEnc}(\mathsf{pp})$
$(K, \mathbf{c}) \leftarrow \mathsf{sKEM.sEncaps}(\mathbf{B}, \mathbf{S}')$
$\mathbf{K} \leftarrow \mathsf{KDF}_1(K)$
$\mu \leftarrow \mathsf{KDF}_2(K)$
$\mathbf{E}' \leftarrow\$ \chi_b^{n \times k}$
$\mathbf{P} \leftarrow \mathbf{B} + \mathbf{A}\mathbf{K} + \mathbf{E}' \mod q$
return $(\mathbf{P}, \mathbf{B}', \mathbf{c}, \mu)$

$\mathsf{Check}(\mathbf{P}, \mathbf{S}'')$

return $\|\mathbf{P} - \mathbf{A}\mathbf{S}''\| \leqslant \beta'$

$\mathsf{DeriveSK}(\mathbf{B}', \mathbf{S}, \mu, c)$

$\mathbf{S}'' \leftarrow \perp$
$K \leftarrow \mathsf{sKEM.sDecaps}(\mathbf{B}', \mathbf{S}, c)$
$\mathbf{K} \leftarrow \mathsf{KDF}_1(K)$
$\mu^* \leftarrow \mathsf{KDF}_2(K)$
if $\mu^* \stackrel{?}{=} \mu$ **then**
  $\mathbf{S}'' \leftarrow \mathbf{K} + \mathbf{S}$
$u \leftarrow \mathcal{U}[0,1]$
if $u < \left(\frac{M \cdot \chi_{b,\mathbf{S}}(\mathbf{S}'')}{\chi_a(\mathbf{S}'')}\right)$, **return** $\mathbf{S}''$
return $\perp$

Figure 5: General Lattice-based ARKG from sKEM

If we consider the scenario where the error $\mathbf{E}''$ and secret $\mathbf{S}''$ are derived from the distinct distributions, then a repetition rate would need to be computed for each component, denoted $M_{\mathbf{S}}$ and $M_{\mathbf{E}}$. Then M, the realised repetition rate of the protocol, would be set to $\max\{M_{\mathbf{S}}, M_{\mathbf{E}}\}$.

Since the distribution of the derived secret key $\mathbf{S}''$ is χ_a, then for an appropriate choice of parameters, it forms a valid LWE instance. For composable protocols that use Gaussian distributed keys, all that is required is to select a and b such that χ_a and χ_b are cryptographically hard distributions. If these conditions are met, then we can use the composability theorem in Frymann et al. [1] to generate keys for protocols that include many of the NIST PQC lattice KEM and encryption schemes such as Frodo [9] and Kyber [10], but also some signatures such as qTESLA [7].

For schemes based on the SIS assumption [13], where the bound β' on the norm plays an important factor in the security of the scheme it may be necessary or desirable to set $\beta' < 2\beta$. However, the wider distribution of the derived key $\mathbf{S}''$ may be too large and thus Check could fail. Nonetheless, the trade off in lower repetition rate versus a larger value of β' may be desirable if the composed protocol has parameter values chosen to accommodate this. The exact values in these cases are dependent on protocol specific parameters, see Section 4.4 for an example using the lattice signature Dilithium [6].

3.4. Security Analysis

Theorem 1. LARKG *satisfies PK-unlinkability if* sKEM *is* nn-IND-CCA *secure and* $\mathsf{LWE}_{n,q,\chi_a}$ *is hard.*

Proof. Game_0: Defined to be the PK-unlinkability experiment $\mathsf{Exp}^{\mathsf{pku}}_{\mathsf{LARKG}}(\lambda)$.

Game_1 is defined as Game_0 with the exception that line 6 of DerivePK is replaced with "$\mathbf{S}', \mathbf{E} \leftarrow_G \chi_a, \mathbf{B}' \leftarrow \mathbf{AS}' + \mathbf{E}, \mathbf{P} \leftarrow \mathbf{B}' + \mathbf{AK} + \mathbf{E}''$". It follows that $\mathbf{B}$ and $\mathbf{B}'$ are statistically indistinguishable from decisional LWE, since rejection sampling ensures that the derived secret key $\mathbf{S}''$ is distributed according to χ_a. Hence the probability that an adversary wins is preserved.

$$|\Pr[\mathsf{Game}_0 = 1] - \Pr[\mathsf{Game}_1 = 1]| \leqslant \mathsf{Adv}^{\mathsf{LWE}_{n,q,\chi_a}}_{\mathcal{B}_1}(\lambda)$$

Game_2 is defined as Game_1 with the exception that line 4 of the oracle is replaced with "$\mathcal{K}^*, \mathbf{c}^* \leftarrow \mathsf{sEncaps}(\mathbf{B}, \mathbf{S}'), \mathbf{V} \leftarrow \mathbf{BS}' + \mathbf{E}'', K_0^* \leftarrow \$\mathcal{K}$. That is, we replace the shared-key K_0^* with a uniformly random sample from the key space $\mathcal{K}$. We argue that the nn-IND-CCA property of sKEM implies that the difference in success probability for the adversary is negligible between games Game_1 and Game_0. To see this, we construct an adversary $\mathcal{B}$ against the nn-IND-CCA property of sKEM using Game_2 as a distinguisher. Precisely, $\mathcal{A}$ plays the role of challenger against $\mathcal{B}$, and initiates the experiment as described in Figure 2, where it sets the public parameters $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{B}'$ to be those from its nn-IND-CCA experiment. We note that $\mathcal{A}$ does not have access to any oracles in aide it, other than the challenge oracle. Instead of executing the oracle as described, it uses the values it has obtained from its own game. Since all values are known to $\mathcal{B}$, it perfectly simulates the game for $\mathcal{A}$, in particular, if nn-IND-CCA has been instantiated with challenge bit $b = 0$ then it coincides with Game_1 and if $b = 1$ then it coincides with Game_2. Therefore, if $\mathcal{A}$ can distinguish between Game_2 and Game_1, $\mathcal{B}$ can construct a correct response to the nn-IND-CCA experiment, that succeeds if $\mathcal{A}$ succeeds.

$$|\Pr[\mathsf{Game}_2 = 1] - \Pr[\mathsf{Game}_1 = 1]| \leqslant \mathsf{Adv}^{\mathsf{nn\text{-}IND\text{-}CCA}}_{\mathcal{B}_2,\mathsf{sKEM}}(\lambda).$$

Now, the output of the challenge oracle is independent of the challenge bit b, and thus the probability that $\mathcal{A}$ outputs a correct responses is $\frac{1}{2}$.

$$\Pr[\mathsf{Game}_2 = 1] = \frac{1}{2} \implies \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{B}_2,\mathsf{sKEM}}(\lambda) = 0.$$

The claimed results of the theorem then follow from Game_0 to Game_2, and the observation that this argument requires KDF_2 to be secure. We conclude the advantage of an adversary against $\mathsf{Exp}^{\mathsf{pku}}_{\mathsf{LARKG}}(\lambda)$ is bound by the advantage of an adversary against nn-IND-CCA of sKEM.

$$\mathsf{Adv}^{\mathsf{pku}}_{\mathsf{LARKG}}(\lambda) \leqslant \mathsf{Adv}^{\mathsf{KDF}_2}_{\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{nn\text{-}IND\text{-}CCA}}_{\mathcal{B}_2,\mathsf{sKEM}}(\lambda) + \mathsf{Adv}^{\mathsf{LWE}_{n,q,\chi_a}}_{\mathcal{B}_1}(\lambda)$$

Hence we have shown that LARKG has PK-unlinkability if LWE is hard and sKEM is nn-IND-CCA secure. □

Theorem 2. LARKG *satisfies honest strong SK-secrecy if* $\mathsf{LWE}_{n,q,\chi_a}$ *is hard.*

Proof. We directly embed an LWE challenge into the protocol as follows. Line 2 of the key generation algorithm in the experiment is replaced with "$(\mathsf{pk}, \mathsf{sk}) \leftarrow (\mathbf{B}^*, \perp)$ where $(\mathbf{A}^*, \mathbf{B}^*)$ is an LWE challenge. Since $\mathcal{A}$ has access to no oracles, $\mathcal{A}$ can simulate the experiment to $\mathcal{B}$. After some time, $\mathcal{B}$ outputs a forgery $\mathbf{S}''$. $\mathcal{A}$ the computes $\mathbf{S}^* = \mathbf{S}'' - \mathbf{K}$, which it can do so since $\mathbf{K}$ was computed by $\mathcal{A}$ during the DerivePK oracle call that created $\mathbf{P}$. If $\|\mathbf{P} - \mathbf{AS}^*\|_\infty = \|\mathbf{E}^*\|_\infty \leqslant 2\beta$ then $\mathcal{A}$ knows that the challenge $\mathbf{B}^*$ is an LWE sample, and thus wins the decisional $\mathsf{LWE}_{n,q,\chi_a}$ game, else it was a uniformly sampled matrix. In either case, if $\mathcal{A}$ wins its game against HSKS, then $\mathcal{B}$ wins its LWE game. This we have:

$$\mathsf{Adv}^{\mathrm{hsks}}_{\mathsf{LARKG}}(\lambda) \leqslant \mathsf{Adv}^{\mathsf{LWE}_{n,q,\chi_a}}_{\mathcal{B}}(\lambda).$$

We conclude the advantage of an adversary against SK-security is bound by the advantage of an adversary against LWE, which proves the theorem. □

Theorem 3. LARKG *satisfies malicious strong SK-secrecy if* sKEM *is* sn-IND-CCA *secure and* $\mathsf{LWE}_{n,q,\chi_a}$ *is hard.*

Proof. Let Game_0 be defined as the MSKS experiment in Figure 1. Then, Game_1 is defined as Game_0 with the exception that line 7 of DerivePK is replaced with "$(\hat{\mathbf{S}}, \hat{\mathbf{B}}) \leftarrow \mathsf{LARKG.KeyGen}(), \mathbf{P} \leftarrow \hat{\mathbf{B}} + \mathbf{AK} + \mathbf{E}' \bmod q$. This change is undetectable by the adversary since $\mathbf{AK} + \mathbf{E}'$ is uniformly distributed by LWE. Thus $\mathbf{P}$ is uniformly distributed in both Game_0 and in Game_1 and hence the winning probability of $\mathcal{A}$ is unchanged.

$$|\Pr[\mathsf{Game}_0 = 1] - \Pr[\mathsf{Game}_1 = 1]| \leqslant \mathsf{Adv}^{\mathsf{LWE}_{n,q,\chi_a}}_{\mathcal{B}_1}(\lambda)$$

We now argue that $\mathcal{A}$ has negligible advantage in Game_1 if the sKEM is sn-IND-CCA secure. To see this, let $\mathcal{B}$ set up the challenge as described in Game_1. It obtains challenge

$(\mathbf{B}, \mathbf{B}^*, c, K)$ and embeds $\mathbf{B}^*$ into the LARKG keys in line 2 of the experiment, i.e. replace with $(\perp, \mathbf{B}^*)$. We note that $\mathcal{B}$ is able to answer all oracle queries made by $\mathcal{A}$ since it uses the public challenge key $\mathbf{B}^*$ to compute the sKEM output $\mathbf{K} \leftarrow \mathsf{sKEM.KeyGenEnc}(\mathbf{B}^*, \mathbf{E})$ (where $\mathbf{E}$ is the ephemeral key created by $\mathcal{B}$ in the DerivePK oracle). Eventually, the adversary outputs a forgery of the form $(\mathbf{S}^*, \mathbf{P}^*, (\mathbf{c}^*, \mathbf{B}^*))$. It extracts the reconciliation data $\mathbf{c}^*$ from aux^* and uses this in its single oracle query. It receives back the corresponding shared key K which it used to compute $\mathbf{K}$. $\mathcal{A}$ then constructs the response as $\mathbf{S}^* \leftarrow \mathbf{S}'' - \mathbf{K}$. It is trivial for $\mathcal{B}$ to then compute $\mathbf{K} \leftarrow \mathsf{KDF}_2(K)$. If $\mathbf{K} \stackrel{?}{=} \mathbf{K}^*$ then set $b = 0$, otherwise set $b = 1$. The simulation only fails if $\mathcal{A}$ queries $\mathbf{B}^*$ which happens with low probability q_o/p^n. Thus the advantage of $\mathcal{A}$ against Game_1 is bound by $\mathcal{B}$ against sn-IND-CCA of sKEM.

$$\mathsf{Adv}^{\mathrm{msks}}_{\mathsf{LARKG}}(\lambda) \leqslant \mathsf{Adv}^{\mathsf{sn\text{-}IND\text{-}CCA}}_{\mathcal{B}_2,\mathsf{sKEM}}(\lambda) + \mathsf{Adv}^{\mathsf{LWE}_{n,q,\chi_a}}_{\mathcal{B}_1}(\lambda).$$

We conclude that LARKG has malicious key secrecy if sKEM is sn-IND-CCA secure and $\mathsf{LWE}_{n,q,\chi_a}$ is hard. □

4. LARKG Instantiations, Usage and Performance

In this section we instantiate our generic scheme with the sKEMs based on passively-secure versions of Frodo and Kyber. That is, we do not perform the FO transforms and thus are limited to IND-CPA security. Before we can use our general results for (IND-CPA-secure) Kyber, we first show that Kyber can be used to construct an nn-IND-CCA secure sKEM. We discuss composability with other cryptographic primitives and give benchmark timings for our implementation.

The constructions for sKEM by Brendel et al. [14] are limited to providing nn-IND-CCA security, which is also the security property we achieve for our Kyber-based approach. This is inherent in both designs since they are based on KEMs that do not have adaptive security. Indeed, as specified in their NIST submissions, both Frodo and Kyber achieve IND-CCA security via use of variants of the FO-transform [24], [25]. However, in the construction by Brendel et al. and in this work, we use the IND-CPA versions. This limits our LARKG constructions to honest weak and honest strong key security only. In the discrete-log setting, a similar restriction is observed where a stronger sn-PRF-ODH assumption [14] is required to achieve malicious security [1]. Nonetheless, malicious security may not be required—since it models the instance an attacker can perform (potentially arbitrary) secret key derivation queries, which may be too strong for many applications. This is the case for the original ARKG-based recovery [1] and delegation mechanisms [3] for WebAuthn, where honest strong key secrecy is sufficient.

4.1. LARKG Instantiation from FrodoKEM

We instantiate our protocol with passively-secure FrodoKEM, i.e. with IND-CPA security, which was shown to be an nn-IND-CCA-secure sKEM by Brendel et al. [14]. We call our instantiation LARKG-Frodo and present it in Figure 6. Illustrated in Figure 3, lines 1 and 2 of the

$\mathsf{KeyGen}(\mathsf{pp})$

1 : $\mathbf{A} \leftarrow\!\$\ \mathbb{Z}_q^{n\times m}$
2 : $\mathbf{S}, \mathbf{E} \leftarrow\!\$\ \chi$
3 : $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
4 : **return** $(\mathsf{sk}, \mathsf{pk}) = (\mathbf{S}, (\mathbf{A}, \mathbf{B}))$

$\mathsf{Decapsulate}(\mathbf{B}', \mathbf{S}, \mu, \mathbf{C})$

1 : **return** $\mathsf{rec}(\mathbf{B}'\mathbf{S}, \mathbf{C})$

$\mathsf{Encapsulate}(\mathbf{A}, \mathbf{B})$

1 : $(\mathbf{S}', \mathbf{E}') \leftarrow\!\$\ \chi$
2 : $\mathbf{B}' \leftarrow \mathbf{AS}' + \mathbf{E}'$
3 : $\mathbf{E}'' \leftarrow\!\$\ \chi$
4 : $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
5 : $\mathbf{C} \leftarrow \langle \mathbf{V} \rangle_{2^B}$
6 : $K \leftarrow \lfloor \mathbf{V} \rceil_{2^B}$
7 : **return** $(\mathsf{ss}, C) = (K, \mathbf{C})$

Figure 6: IND-CPA FrodoKEM [9] based on LWE parameters (n, m, χ). This implies an nn-IND-CCA-secure sKEM as detailed in Figure 3.

Encapsulate algorithm that generates an LWE key pair $(\mathbf{S}', \mathbf{B}')$ are lifted into a separate algorithm KeyGenEnc. The encapsulate algorithm must now take in the secret key $\mathbf{S}$' to correctly compute the shared secret. FrodoKEM has some protocol specific parameters $\overline{n}, \overline{m}$ and $\overline{B}$, which are defined as $\overline{B} = (\log q) - B$ and where $\overline{n}, \overline{m}$ and B are chosen such that $\overline{n} \cdot \overline{m} \cdot B > \lambda$, the security parameter. The function $\lfloor \cdot \rceil_{2^B} : \mathbb{Z}_q^{n\times m} \to \mathbb{Z}_{2^B-1}$ is a rounding function defined as $\lfloor v \rceil_{2^B} : v \to \lfloor 2^{-\overline{B}} v \rfloor \mod 2^B$, which partitions $\mathbb{Z}_q$ into 2^B intervals of integers with matching B significant bits. Similarly, the function $\langle \cdot \rangle_{2^B} : \mathbb{Z}_q^{n\times m} \to \{0, 1\}$ is a cross-rounding function defined as $\langle v \rangle_{2^B} : v \to \lfloor 2^{-\overline{B}+1} v \rfloor \mod 2$, which divides $\mathbb{Z}_q$ into 2 partitions according to the value of the bit at position $B + 1$. We can now define the reconciliation function rec, which takes as input $\mathbf{B}'\mathbf{S}$ and recovery information $\mathbf{C}$. It element-wise computes a matrix $\mathbf{V}'$ such that each element is the closest which satisfies $\langle \mathbf{V}_{i,j} \rangle_{2^B} = \mathbf{C}_{i,j}$. It outputs the shared secret as $\lfloor \mathbf{V}_{i,j} \rceil_{2^B}$. Crucially, leaking $\langle \mathbf{V} \rangle_{2^B}$ has been shown to not reveal any information about the shared secret, we refer to Bos et al. [9] for further details. We have chosen our notation carefully so that parameters and variables in FrodoKEM (Figure 6) align those used to describe our generic LARKG construction in Figure 5. We instantiate KDF with HKDF [22] with double length output, i.e. of length 2λ, parsed as $\rho || \mu$, each a bit string of length λ. To sample a Gaussian distributed key K from ρ, we use Frodo's algorithm `SampleG`, that takes as input a uniform seed ρ, a centre α and a standard deviation η and outputs $K \sim \chi_\eta$ centred at α.

The key pairs derived by our LARKG-Frodo instance from Figure 5 are compatible with LWE and SIS-based KEM/PKE and signature schemes. We note that the construction of sKEM using FrodoKEM by Brendel at al. only meets nn-IND-CCA security, our ARKG instance is at most hsKS-secure. This property also implies LARKG-

$\mathsf{KeyGen}(\mathsf{pp})$

$\mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k\times k}$
$\mathbf{s}, \mathbf{e} \leftarrow\!\!\$\ \chi_\eta^k$
$\mathbf{b} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$
return $(\mathsf{sk}, \mathsf{pk}) = (\mathbf{s}, (\mathbf{A}, \mathbf{b}))$

$\mathsf{Enc}(\mathsf{pp}, \mathbf{b}, m)$

$\mathbf{r}, \mathbf{e} \leftarrow\!\!\$\ \chi_\eta^k$
$\mathbf{u} \leftarrow \mathbf{A}^T\mathbf{r} + \mathbf{e}$
$e' \leftarrow\!\!\$\ \chi_\eta$
$v \leftarrow \mathbf{b}^T\mathbf{r} + e' + K$
return $\mathsf{c} = (\mathbf{u}, v)$

$\mathsf{Dec}(\mathbf{s}, \mathbf{u}, v)$

$m \leftarrow [v - \mathbf{s}^T\mathbf{u}]_{q/2}$
return m

Figure 7: Kyber PKE algorithms with MLWE parameters (m, k, χ). This gives an nn-IND-CCA-secure sKEM shown in Figure 8.

Frodo is PK-unlinkable.

Remark 3. *We note a small error in the sKEM construction described in Brendel et al. [14] that prevents correctness of the protocol in the plain-LWE setting. We have presented a correction in our protocol described in Figure 3. Their analogous RLWE-based sKEM is unaffected due to commutativity in the ring.*

4.2. LARKG Instantiation from Kyber

In this section we show how to construct LARKG based on Kyber, which we call LARKG-Kyber. We cannot directly use the generic construction of sKEMs as given by Brendel et al. [14] since Kyber uses a PKE scheme instead of the reconciliation methods used by e.g. Frodo [9]. We first show that we can construct an sKEM based on Kyber, the security of which relies on the IND-CPA property of the underlying PKE scheme. Intuitively, our sKEM is constructed by extracting some computations (which can be viewed as creating an LWE key) in the encapsulation algorithm into a key generation algorithm, KeyGenEnc, for sKEM.

In particular, the encapsulation algorithm works by generating some randomness for key generation, using Kyber's PKE to encrypt the shared key. In our scheme, we separate out part of the Encryption algorithm into a key generation algorithm for sKEM, namely, KeyGenEnc. We base our construction on Kyber PKE which we have recalled in Figure 7. Note that we have abstracted away the use of $\mathtt{compress}_q, \mathtt{decompress}_q, \mathtt{NTT}$ and $\mathtt{NTT}^{-1}$ algorithms present in the specification of Kyber [10] for conciseness. Compression is mainly used to discard low-order bits (which has limited impact on correctness), which allows for smaller parameter choices. The only exception to this during the decryption algorithm where $\mathtt{compress}_q$ is used to 'round' the decryption to 0 or $\lceil \frac{q}{2} \rfloor$. We have modelled this behaviour with a rounding function, which we denote by $[\cdot]_{q/2}$. Looking ahead, our implementation will use compression and NTT techniques since this offers fair comparisons with Kyber implementations. The security of Kyber is based on MLWE, and so our construction is also presented using module lattices.

Our algorithms are given in Figure 8. In Theorem 4, we show that our construction is nn-IND-CCA secure based on on the security of Kyber's IND-CPA secure PKE. We can then use the generic construction in Figure 5 to construct an ARKG scheme based on Kyber.

The correctness of our sKEM is implied by the correctness of the underlying PKE. Intuitively, we have split the encapsulation algorithm into two algorithms for sKEM but not changed the underlying computations. Hence, we choose parameters to ensure PKE is correct, and correctness of our sKEM follows.

Theorem 4. *Our sKEM construction based on Kyber in Figure 8 is* nn-IND-CCA *secure.*

Proof. We consider the PK-Unlinkability experiment $\mathsf{Exp}^{\mathsf{nn\text{-}IND\text{-}CCA}}_{\mathsf{sKEM}}(\lambda)$ and give a direct reduction to the IND-CPA property of the Kyber's PKE scheme. To do so, we construct an adversary $\mathcal{B}$ that sets up the sKEM experiment as described with the following exceptions. It invokes the IND-CPA game to receive the matrix $\mathbf{A}$, the challenge key $\mathsf{pk}^\star$ and a challenge ciphertext $(\mathbf{u}^\star, v^\star)$ based on one of two distinct messages $K_b, b \in \{0, 1\}$. Note that here, $\mathbf{u}^\star$ takes the form of the keys created by KeyGenEnc. $\mathcal{B}$ forwards the following sKEM challenge to $\mathcal{A}$: $(\mathsf{pk}^\star, \mathbf{u}^\star, v^\star, K_1)$. Note that the adversary cannot distinguish which key queries are not genuine since $\mathbf{u}^\star$ is distributed properly in $\mathcal{R}_q^k$. Note that $\mathcal{B}$ does not need to respond to any oracle queries for nn-IND-CCA.

$\mathcal{A}$ guesses a bit b', which $\mathcal{B}$ forwards as a response to its own game. It wins its IND-CPA experiment whenever $\mathcal{A}$ wins its nn-IND-CCA game. Thus, we have:

$$\Pr\left[\mathsf{Exp}^{\mathsf{nn\text{-}IND\text{-}CCA}}_{\mathsf{sKEM}}(\lambda) = 1\right] = \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\lambda) = 1\right]$$

which implies

$$\mathsf{Adv}^{\mathsf{nn\text{-}IND\text{-}CCA}}_{\mathcal{A},\mathsf{sKEM}}(\lambda) = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{B},\mathsf{PKE}}(\lambda).$$

Hence, we have shown that sKEM nn-IND-CCA secure if Kyber PKE is IND-CPA secure. □

In this instantiation we use our result from Theorem 4. The distribution of χ is chosen to be a centered binomial distribution (CBD) instead of small discretised Gaussian, so that our LARKG protocol outputs derived keys that are also distributed according to a CBD. We emphasise this is not an incompatibility with Kyber, rather the decision to use CBD is due to the relative inefficiencies of sampling Gaussian distributions. LWE-based encryption does not depend on the exact distribution but rather the standard deviation [10], thus we do not lose security of guarantees of LWE when error and secrets are sampled from a CBD. We instantiate KDF with HKDF [22] with double length output, i.e. of length 2λ, parsed as $\rho||\mu$, each a bit string of length λ. To sample a centered binomial distributed key K from ρ, we use Kyber's algorithm CBD, that takes as input a uniform seed ρ, a centre α and a standard deviation η and outputs $K \sim \chi_\eta$ centred at α.

Setup(pp)

1 : $\mathbf{A} \leftarrow\$ \mathcal{R}_q^{k \times k}$
2 : **return** $\mathsf{pp} = \mathbf{A}$

KeyGenDec(pp)

1 : $\mathbf{s}, \mathbf{e} \leftarrow\$ \chi_\eta^k$
2 : $\mathbf{b} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$
3 : **return** $(\mathsf{sk}, \mathsf{pk}) = (\mathbf{s}, (\mathbf{A}, \mathbf{b}))$

KeyGenEnc(pp)

1 : $\mathbf{r}, \mathbf{e} \leftarrow\$ \chi_\eta^k$
2 : $\mathbf{u} \leftarrow \mathbf{A}^T\mathbf{r} + \mathbf{e}$
3 : **return** $(\mathsf{sk}, \mathsf{pk}) = (\mathbf{r}, \mathbf{u})$

Encapsulate(pp, $\mathbf{b}$)

1 : $K \leftarrow\$ \mathcal{R}_q$
2 : $e' \leftarrow\$ \chi_\eta$
3 : $v \leftarrow \mathbf{b}^T\mathbf{r} + e' + K$
4 : **return** $(\mathsf{ss}, C) = (K, v)$

Decapsulate($\mathbf{u}, \mathbf{s}, \mathbf{C}$)

1 : Parse $\mathbf{C}$ as v
2 : $K \leftarrow v - \mathbf{s}^T\mathbf{u}$
3 : **return** K

Figure 8: sKEM based on Kyber with MLWE parameters (m, k, χ).

Remark 4. *We observe that our approach could be applied to other lattice-based KEMs that use the PKE method of computing a shared secret. In particular, the team behind Frodo made changes in their submission to Round 3 of the NIST PQC standardisation process. They shifted away from reconciliation techniques in favour of a PKE based solution*[4]*, which we distinguish from Frodo by calling this variant FrodoKEM. Thus both Frodo (proof by Brendel et al. [14]) and FrodoKEM (analogous to Theorem 4) are sKEMs and therefore can be used to construct* LARKG.

4.3. LARKG usage in KEMs and PKE

KEMs are an important primitive used in many cryptographic protocols, including for challenge-response for authentication. Instead of requiring a user to sign a challenge string, the user must decrypt a challenge KEM issued by the relying party. The user sends back the decrypted plaintext, and the sever accepts if the plaintexts match. This is the approach proposed in KEMTLS [26], which is a post-quantum replacement for TLS1.3. The benefit over more traditional signature-based solutions is that lattice KEMs typically offer more efficient constructions. For this reason, we conjecture that KEM based challenge-response protocols (such as WebAuthn) are likely to become more wide spread as post-quantum protocols are adopted.

Our lattice ARKG has been designed to produce key pairs that are compatible with lattice-based KEMs and signature schemes. Furthermore, security of the composed protocol is maintained when used with keys generated by LARKG. This follows from the general composability theorem in [1] which states that ARKG can be securely composed with arbitrary public-key protocols provided the key distributions match, and ARKG is PK-unlinkable.

We first consider LARKG-Frodo, presented in Section 4.1, with Frodo [9] as a KEM. This composition of protocols follows form the fact that our LARKG-Frodo outputs derived keys that have a small Gaussian distribution (if χ is also a small Gaussian), which is exactly the structure of secret keys of Frodo. All that remains is to select parameters to meet cryptographic requirements. It is sufficient to instantiate the protocol with the parameters from Frodo directly. This follows because the long term and ephemeral public key forms a cryptographically hard LWE instances, and the derived key has the same dimensions with an error distribution that has standard deviation (s.d.) $\overline{\sigma} = 2\sigma$, where σ is the standard deviation from the Frodo parameters. Therefore the derived keys are secure since $\overline{\sigma} > 2\sqrt{n}$ if $\sigma > 2\sqrt{n}$, which ensures cryptographic hardness [12]. Note that this does not impact correctness since the KEM correctly decrypts the ciphertext since $\|\mathbf{E}''\| \ll 2^{\overline{B}-2}$.

We next consider the LARKG-Kyber instance in Section 4.2. To select parameters to meet cryptographic requirements, we follow the parameter levels included in the round 3 Kyber specification [5]. Intuitviely, because the security of our construction for an sKEM based on Kyber relies the underlying PKE, parameters that ensure a secure PKE instance also imply a secure sKEM. Thus we are assured the keys used in sKEM form cryptographically hard LWE instances if we instantiate with secure Kyber parameters. The resulting LARKG instance outputs keys with a wider error distribution. However this does not affect correctness provided $\left\|\mathbf{e}^T\mathbf{r} + e_2 + \mathbf{c}_v + \mathbf{c}_t^T\mathbf{r} - \mathbf{s}^T\mathbf{e}_1 - \mathbf{s}^T\mathbf{c}_u\right\|_\infty \leqslant \lceil q/4 \rfloor$, where $\mathbf{e}$ and $\mathbf{r}$ are CBD error vectors, e_2 is a small scalar, $\mathbf{c}_t, \mathbf{C}_u$ are (small Gaussian) rounding error vectors, and $\mathbf{s}$ is the derived secret key. The only term that changes from the original correctness analysis is that $\mathbf{s}$ is now distributed according to a centered binomial distribution with variance η, rather than with variance $\eta/2$ that $\mathbf{s}$ would typically have. The exact amount this changes will depend on the selected parameter choices. To give an example, if we consider Kyber Level 5 parameter sets then the probability of failure is 2^{-174}. When composed with LARKG-Kyber also instantiated with Kyber's level 5 parameter set, then we estimate using a security estimator for Kyber [27]. This gives a value of $M = 9$ (at the expense of additional derived public keys) we are able to obtain correctness values of $1 - 2^{-174}$, or equivalently $\tau = 274$, when the distribution χ_b is instantiated with standard deviation $\sigma_b = 5\sigma_a$. Further optimisation of parameter choices may improve correctness or reduce M yet retain the desired security level may be possible, but we leave identifying such parameters future work as it will be application and security level dependent.hat tr

Furthermore, we note that since many lattice KEMs can be extended to PKE algorithms by replacing the sampling of a shared key with the message to be encrypted, we conclude that our LARKG construction is also compatible

4. https://frodokem.org/files/FrodoKEM-specification-20210604.pdf

5. https://www.pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf

Table 1: Parameter trade-offs for our Lattice-ARKG based on Dilithium level 5 parameters.

β'	$\Pr[\|\mathbf{S}\|_\infty \leqslant \beta']$	τ
5	0	0
6	6×10^{-5}	0.0009
7	0.1	0.15
8	0.63	1.43
9	0.92	3.64
10	1	∞

with corresponding lattice PKE schemes. Concrete examples of this include FrodoPKE and KyberPKE. Finally, our LARKG construction is compatible with many lattice KEMs that achieve chosen ciphertext security via (variants of) the FO-transform [24], [25], again such examples include Frodo, FrodoKEM or Kyber.

4.4. LARKG usage in Signatures

Some lattice digital signatures, such as qTESLA [7] also support small Gaussian keys, and are compatible with the previous techniques described for KEMs. However, other schemes based on SIS use short vectors as secret keys (provided they meet some min-entropy [28]). We consider Dilithium [29], which was selected for standardisation in the NIST post quantum process and takes this form. The key structure consists of two short vectors $\mathbf{s}_1$ and $\mathbf{s}_2$, bounded by a value β that satisfy $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{b}$. Since our protocol outputs keys with Gaussian distribution $\chi_{2\sigma}$, we need to ensure that the parameters selected for Dilithium remain secure, despite the wider error distribution. To achieve this, one could select a tighter distribution for the long term keys. This corresponds to a potentially higher-security level, and therefore larger key sizes, to ensure the derived keys meet their desired minimum security level.

Alternatively, another approach would be to choose $\beta' < 2\beta$ for the composed signature, where there is some probability that a derived secret key has norm $> \beta'$ and therefore would be an invalid key. However, as this process is interactive, it is not possible to simply repeat the procedure as required until a 'good' key is found. Instead, it must be somehow accounted for during execution of DerivePK and before the values of $(\mathbf{S}_1, \mathbf{S}_2)$ are known. To achieve this one could provide t candidate keys, that probabilistically, at least one of t will create a valid derived key pair. We illustrate this trade off in Table 1, based on parameters for Dilithium level 5 (where $\beta = 3$). We vary the value of β' to show how this affects correctness of our protocol and corresponding τ value. Intuitively, increased failure rate allows for higher security since the norm bound is smaller. We emphasise that the value of β' must be selected based on security analysis of the composed signature.

4.5. LARKG Implementation, Parameters and Performance

In our code repository [30], we provide implementations of our LARKG-Frodo and LARKG-Kyber schemes, and we present the benchmarking of these in Table 2, as well as respective sKEM instantiations. We taken as the mean of 100 invocations on a Intel i7-8700 processor, with a clock speed of 3.20GHz. Note that the libraries we used are not optimised or side-channel secure, instead we illustrate here the relative performance of LARKG versus the underlying KEMs.

Table 2: Performance of Frodo- and Kyber-based KEM and LARKG primitives for a single call, in seconds, calculated as the mean of 10 invocations.

		Instantiation	
Primitive	Algorithm	Frodo	Kyber
KEM	KeyGen	3.412	0.011
	Encaps	3.624	0.016
	Decaps	3.546	0.024
LARKG	KeyGen	1.146	0.010
	DerivePK	2.536	0.027
	DeriveSK	1.641	0.014

For LARKG-Frodo, we instantiate the scheme with parameter set `FrodoKEM-976-AES`, i.e., $n = 976$, $m = k = 8$, $q = 2^{16}$, with the error distribution χ having $\sigma = 2.3$. This matches parameter choices for NIST Level 3 of FrodoKEM. We used a Python implementation of FrodoKEM[6]. Compared to FrodoKEM's KeyGen, the figures in Table 2 give an overall performance cost of 56% for LARKG-Frodo when generating a derived key pair, i.e., calling KeyGen, then DerivePK, and finally DeriveSK. Note that our LARKG-Frodo.KeyGen is 298% faster than FrodoKEM's KeyGen as we fix the matrix A as part of our public parameters, whereas FrodoKEM must recompute this matrix from a seed on each invocation.

For LARKG-Kyber, we instantiated Kyber with the Level 5 parameter set, `Kyber1024`, using a Python library[7]. Precisely, the parameters we use are: $n = 256, k = 4, q = 3329, n_1 = n_2 = 2, d_u = 11, d_v = 5$. Note that we also used the number theoretic transform (NTT) to perform polynomial multiplication to give a fair comparison to the base Kyber implementation. We also demonstrate the correctness of our implementation in our code repository [30] by running an instance of Kyber's IND-CPA PKE using LARKG-Kyber keys, as well as give a Kyber with split KEM functionality based on the required changes presented in Figure 8.

We present timings given in Table 2 that do not account for repeat executions required by rejection sampling, and an example whereby we do consider the probability of failure. We do so because the rejection sampling parameters are highly dependent on scheme and security level, whereas a single run demonstrates the relative performance of the underlying computations with the understanding that it would most likely require repeated executions. By setting the distribution χ_b to have standard deviation $\sigma_b = 5\sigma_a$, then we get $M = 9$ (which is similar to other works [17]), then we instead measure a full correctness value of 2^{-174} which corresponds to $\tau = 174$, still targeting 256-bit security. In particular, the wider distribution of LARKG-Kyber keys has increased the difficulty in solving MLWE, where the optimal block size of dual and primal attacks has increased from 868 to 953, which gives an estimated classical bit security of 278 versus the 253 of `Kyber1024` [27].

6. github.com/microsoft/PQCrypto-LWEKE/tree/master/python3
7. github.com/jack4818/kyber-py

As an *illustrative example*, we measure the average time for a successful run of the complete LARKG primitive (i.e., KeyGen, followed by DerivePK, then DeriveSK), when using rejection sampling and instantiating χ_a and χ_b to be the same CBD distributions with $\eta = 2$. The total run time of LARKG-Kyber is then 0.19s with an acceptance rate of 0.32—measured over 1000 runs on an i7-8700. This is an increase in execution time of 272.5%, compared to the average without rejection sampling—which, for Kyber, is 0.051s. This means that, for these particular parameters, a successful run of DeriveSK requires around three executions of LARKG. These measurements vary considerably based on parameter selection.

Looking at Table 2, we note that, similarly to LARKG-Frodo, LARKG-Kyber's DeriveSK takes 92% less time than DerivePK. Without any optimisations, we see an overall cost of 51ms for LARKG-Kyber when generating derived key pairs compared to Kyber's 11ms for standalone key generation. We note that currently existing Python libraries behind Frodo and Kyber which we implement upon, differ in that the Kyber library is optimised whereas Frodo library not. This explains why in our measurements in Table 2 Frodo appear much slower (with values being in s range) in comparison to Kyber (in ms range) than one would expect. We anticipate that an optimised underlying Python library for Frodo might yield a somewhat slower implementation (based on similar security levels) than Kyber albeit not at the same magnitude. We emphasise this does not affect our claim of practicality since we compare relative performance.

5. Conclusion

In this paper we have given the first Asynchronous Remote Key Generation scheme for use with lattice-based public key cryptosystems that use either SIS or LWE-based keys. We have given a general lattice ARKG (LARKG) using Split KEMs which we prove is PK-unlinkable under the nn-IND-CCA property of the sKEM. We showed that if LWE is hard, then our construction has hsKS security, which is extended to msKS security under the sn-IND-CCA property of the underlying sKEM. We gave two instantiations of our hsKS-secure protocol, LARKG-Frodo and LARKG-Kyber, which are based on popular KEMs. To use Kyber in LARKG-Kyber, we first had to show that it could be used to realise an sKEM and showed that it satisfies the nn-IND-CCA security property. Other KEMs based on an IND-CPA PKE could also be used to create sKEMs in a similar way.

To capture the need to restrict the norm of the derived secret key, we have defined a relaxed notion of ARKG correctness to account for inherent noise of our underlying lattice building blocks. Our constructions support lattice schemes that use both LWE and SIS style keys, and thus are compatible with several KEMs, PKE and signatures selected for standardisation by NIST, or that appear in round 3 as alternative candidate. We discussed integration details with selected primitives Frodo, Kyber and Dilithium. We highlighted the necessary the parameter selections needed to ensure that ARKG keys are compatible with the selected primitive, without compromising security or correctness.

Additionally, we have presented publicly-accessible implementations in Python of our LARKG based on both FrodoKEM and Kyber, along with timings and a discussion of our results. We note that, despite our unoptimised implementations, our performance metrics show that our protocols are practical.

Future Work. A future direction for this work could be to develop sKEMs that are secure against adaptive adversaries, in particular, to satisfy the sn-IND-CCA security. This would allow LARKG schemes with stronger key secrecy properties, i.e. msKS and mwKS, that would allow for use in wider applications with stronger adversarial models. The difficulty in existing techniques to construct sKEMs lies in the use of the FO transform to achieve IND-CCA security in the corresponding KEM, e.g. Kyber or Frodo. In particular, the decapsulation requires reconstruction of the ciphertext using the same randomness to ensure consistency. However, when translated to the sKEM setting using either the techniques by Brendel et al. [14] or Theorem 4, some of this 'randomness' forms the secret key of the encapsulator and thus is not known during decapsulation. This prevents use of the FO-like transform to achieve adaptive security in sKEMs, i.e. sn-IND-CCA security. Overcoming these challenges would be interesting future work.

We also observe that the original discrete-log construction of ARKG was previously used to enable anonymous payments in cryptocurrencies. Coined stealth addresses [31], [32], a payer can generate an ephemeral address for the receiver based on its long term key. This corresponds to the DerivePK, and DeriveSK corresponds to the receiver computing the secret key for the ephemeral address. The only known constructions are based on the group-theoretic cryptography, and we speculate that lattice ARKG could be used to give a post-quantum construction for stealth addresses.

References

[1] N. Frymann, D. Gardham, F. Kiefer, E. Lundberg, M. Manulis and D. Nilsson, 'Asynchronous remote key generation: An analysis of yubico's proposal for w3c webauthn', *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[2] D. Balfanz, A. Czeskis, J. Hodges *et al.*, 'Web authentication: An API for accessing public key credentials level 1', Tech. Rep., 2019. [Online]. Available: https://www.w3.org/TR/webauthn.

[3] N. Frymann, D. Gardham and M. Manulis, 'Unlinkable delegation of webauthn credentials', in *Computer Security – ESORICS 2022*, V. Atluri, R. Di Pietro, C. D. Jensen and W. Meng, Eds., Cham: Springer Nature Switzerland, 2022, pp. 125–144, ISBN: 978-3-031-17143-7.

[4] N. Frymann, D. Gardham, M. Manulis and H. Nartz, *Generalised asynchronous remote key generation for pairing-based cryptosystems*, Cryptology ePrint Archive, Paper 2023/456, https://eprint.iacr.org/2023/456, 2023. [Online]. Available: https://eprint.iacr.org/2023/456.

[5] P. W. Shor, 'Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer', *SIAM J. Comput.*, vol. 26, no. 5,

pp. 1484–1509, 1997, ISSN: 0097-5397. DOI: 10.1137/S0097539795293172. [Online]. Available: https://doi.org/10.1137/S0097539795293172.

[6] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler and D. Stehle, *Crystals – dilithium: Digital signatures from module lattices*, Cryptology ePrint Archive, Report 2017/633, https://eprint.iacr.org/2017/633, 2017.

[7] E. Alkim, P. S. L. M. Barreto, N. Bindel, J. Krämer, P. Longa and J. E. Ricardini, 'The lattice-based digital signature scheme qtesla', in *Applied Cryptography and Network Security: 18th International Conference, ACNS 2020, Rome, Italy, October 19–22, 2020, Proceedings, Part I*, Rome, Italy: Springer-Verlag, 2020, pp. 441–460, ISBN: 978-3-030-57807-7.

[8] P.-A. Fouque, J. Hoffstein, P. Kirchner *et al.*, *Falcon: Fast-fourier lattice-based compact signatures over ntru.* 2017.

[9] J. Bos, C. Costello, L. Ducas *et al.*, 'Frodo: Take off the ring! practical, quantum-secure key exchange from lwe', in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, Vienna, Austria: Association for Computing Machinery, 2016, pp. 1006–1018, ISBN: 9781450341394. DOI: 10.1145/2976749.2978425. [Online]. Available: https://doi.org/10.1145/2976749.2978425.

[10] J. W. Bos, L. Ducas, E. Kiltz *et al.*, 'CRYSTALS - kyber: A cca-secure module-lattice-based KEM', in *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, IEEE, 2018, pp. 353–367. DOI: 10.1109/EuroSP.2018.00032. [Online]. Available: https://doi.org/10.1109/EuroSP.2018.00032.

[11] E. Alkim, L. Ducas, T. Pöppelmann and P. Schwabe, 'Post-quantum key exchange: A new hope', in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16, Austin, TX, USA: USENIX Association, 2016, pp. 327–343, ISBN: 9781931971324.

[12] O. Regev, 'On lattices, learning with errors, random linear codes, and cryptography', in *STOC*, 2005.

[13] M. Ajtai, 'Generating hard instances of lattice problems (extended abstract)', in *STOC '96*, 1996.

[14] J. Brendel, M. Fischlin, F. Gunther, C. Janson and D. Stebila, 'Towards post-quantum security for signal's x3dh handshake', in *Proceedings of the Selected Areas in Cryptography, 27th InternationalnConference.*, ser. SAC 2020, Springer, 2020.

[15] J. Brendel, M. Fischlin, F. Günther and C. Janson, 'Prf-odh: Relations, instantiations, and impossibility results', in *Advances in Cryptology – CRYPTO 2017*, J. Katz and H. Shacham, Eds., Cham: Springer International Publishing, 2017, pp. 651–681, ISBN: 978-3-319-63697-9.

[16] Z. Brakerski, C. Gentry and V. Vaikuntanathan, '(leveled) fully homomorphic encryption without bootstrapping', in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12, Cambridge, Massachusetts: Association for Computing Machinery, 2012, pp. 309–325, ISBN: 9781450311151. DOI: 10.1145/2090236.2090262. [Online]. Available: https://doi.org/10.1145/2090236.2090262.

[17] V. Lyubashevsky, 'Lattice signatures without trapdoors', in *Advances in Cryptology – EUROCRYPT 2012*, D. Pointcheval and T. Johansson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 738–755, ISBN: 978-3-642-29011-4.

[18] J. Brendel, R. Fiedler, F. Günther, C. Janson and D. Stebila, 'Post-quantum asynchronous deniable key exchange and the signal handshake', in *Public-Key Cryptography – PKC 2022*, G. Hanaoka, J. Shikata and Y. Watanabe, Eds., Cham: Springer International Publishing, 2022, pp. 3–34, ISBN: 978-3-030-97131-1.

[19] J. Ding, X. Xie and X. Lin, *A simple provably secure key exchange scheme based on the learning with errors problem*, Cryptology ePrint Archive, Paper 2012/688, https://eprint.iacr.org/2012/688, 2012. [Online]. Available: https://eprint.iacr.org/2012/688.

[20] R. Lindner and C. Peikert, 'Better key sizes (and attacks) for lwe-based encryption', in *Topics in Cryptology – CT-RSA 2011*, A. Kiayias, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 319–339, ISBN: 978-3-642-19074-2.

[21] C. Peikert, 'Lattice cryptography for the internet', in *Post-Quantum Cryptography*, Cham: Springer International Publishing, 2014, pp. 197–219, ISBN: 978-3-319-11659-4.

[22] H. Krawczyk, 'Cryptographic extraction and key derivation: The hkdf scheme', in *Advances in Cryptology – CRYPTO 2010*, T. Rabin, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 631–648, ISBN: 978-3-642-14623-7.

[23] C. Gentry, 'A fully homomorphic encryption scheme', crypto.stanford.edu/craig, Ph.D. dissertation, Stanford University, 2009.

[24] E. Fujisaki and T. Okamoto, 'Secure integration of asymmetric and symmetric encryption schemes', in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '99, Berlin, Heidelberg: Springer-Verlag, 1999, pp. 537–554, ISBN: 3540663479.

[25] E. Fujisaki and T. Okamoto, 'Secure integration of asymmetric and symmetric encryption schemes', *J. Cryptol.*, vol. 26, no. 1, pp. 80–101, 2013. DOI: 10.1007/s00145-011-9114-1. [Online]. Available: https://doi.org/10.1007/s00145-011-9114-1.

[26] P. Schwabe, D. Stebila and T. Wiggers, 'Post-quantum tls without handshake signatures', in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1461–1480, ISBN: 9781450370899. [Online]. Available: https://doi.org/10.1145/3372297.3423350.

[27] *Github - pq-crystals/security-estimates: Security estimation scripts for kyber and dilithium.* https://github.com/pq-crystals/security-estimates, Accessed: 2022-08-01.

[28] S. Agrawal, C. Gentry, S. Halevi and A. Sahai, 'Discrete gaussian leftover hash lemma over infinite domains', in *Advances in Cryptology - ASIACRYPT 2013*, K. Sako and P. Sarkar, Eds., Berlin, Heidel-

berg: Springer Berlin Heidelberg, 2013, pp. 97–116, ISBN: 978-3-642-42033-7.

[29] L. Ducas, E. Kiltz, T. Lepoint *et al.*, 'Crystals-dilithium: A lattice-based digital signature scheme', *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 1, pp. 238–268, 2018. DOI: 10.13154/tches.v2018.i1.238-268. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/839.

[30] *Source code for FrodoKEM- and Kyber-based LARKG implementations*, https://gitlab.surrey.ac.uk/sccs/larkg.

[31] P. Todd, *Stealth addresses*, https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-January/004020.html, 2014.

[32] N. van Saberhagen, *Cryptonote v2.0*, https://cryptonote.org/whitepaper.pdf, 2013.