Towards Post-Quantum Security for Signal's X3DH Handshake

Author(s):
Brendel, Jacqueline; Fischlin, Marc; Günther, Felix; Janson, Christian; Stebila, Douglas

Publication date:
2021-07

Permanent link:
https://doi.org/10.3929/ethz-b-000441452

Originally published in:
Lecture Notes in Computer Science 12804, https://doi.org/10.1007/978-3-030-81652-0_16

Towards Post-Quantum Security for Signal's X3DH Handshake

Jacqueline Brendel[1(✉)], Marc Fischlin[2], Felix Günther[3], Christian Janson[2], and Douglas Stebila[4]

[1] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
[2] Cryptoplexity, Technische Universität Darmstadt, Darmstadt, Germany
[3] Department of Computer Science, ETH Zürich, Zürich, Switzerland
[4] University of Waterloo, Waterloo, Canada
mail@jbrendel-info.de marc.fischlin@cryptoplexity.de
mail@felixguenther.info christian.janson@cryptoplexity.de
dstebila@uwaterloo.ca

Abstract. Modern key exchange protocols are usually based on the Diffie–Hellman (DH) primitive. The beauty of this primitive, among other things, is its potential reusage of key shares: DH shares can be either used a single time or in multiple runs. Since DH-based protocols are insecure against quantum adversaries, alternative solutions have to be found when moving to the post-quantum setting. However, most post-quantum candidates, including schemes based on lattices and even supersingular isogeny DH, are not known to be secure under key reuse. In particular, this means that they cannot be necessarily deployed as an immediate DH substitute in protocols.

In this paper, we introduce the notion of a *split key encapsulation mechanism* (split KEM) to translate the desired key-reusability of a DH-based protocol to a KEM-based flow. We provide the relevant security notions of split KEMs and show how the formalism lends itself to lifting Signal's X3DH handshake to the post-quantum KEM setting without additional message flows.

Although the proposed framework conceptually solves the raised issues, instantiating it securely from post-quantum assumptions proved to be non-trivial. We give passively secure instantiations from (R)LWE, yet overcoming the above-mentioned insecurities under key reuse in the presence of active adversaries remains an open problem. Approaching one-sided key reuse, we provide a split KEM instantiation that allows such reuse based on the KEM introduced by Kiltz (PKC 2007), which may serve as a post-quantum blueprint if the underlying hardness assumption (gap hashed Diffie–Hellman) holds for the commutative group action of CSIDH (Asiacrypt 2018).

The intention of this paper hence is to raise awareness of the challenges arising when moving to KEM-based key exchange protocols with key-reusability, and to propose split KEMs as a specific target for instantiation in future research.

Keywords: post-quantum, key encapsulation mechanisms, key exchange, Signal protocol, X3DH

1 Introduction

The core Diffie–Hellman protocol [21]—Alice sends g^x, Bob sends g^y, and both compute g^{xy} as shared secret—is a beautiful and versatile cryptographic primitive, plays a central role in modern key exchange protocols, and appears in many variants. For example, a key share (x, g^x) can be *ephemeral* (meaning used only once) or *static* (meaning reused multiple times). Furthermore, the same share can be used in role-symmetric ways, i.e., as both initiator and responder of key exchange sessions and the same message flow can give rise to different authenticated key exchange (AKE) protocols (e.g., HMQV [50], CMQV [75], NAXOS [54]). The security of DH-based constructions can in turn be based on many cryptographic assumptions over the group, ranging from simple passive assumptions like computational (CDH) or decisional (DDH) Diffie–Hellman, to interactive assumptions like GapDH [63], oracle DH (ODH) [1], or PRF-ODH [43,11].

Indeed, modern real-world cryptographic protocols employ the Diffie–Hellman key exchange protocol in ways that often rely on many aspects of this versatility. Table 1 shows key exchange patterns from various Internet protocols that employ a "signed ephemeral Diffie–Hellman" approach. In TLS 1.2 [20], the server sends the initial ephemeral public key, and the client responds, while in TLS 1.3 [69], the roles are reversed to reduce the number of round trips: the client sends the initial ephemeral public key, and the server responds. In both cases the security proofs [43,30] rely on interactive DH assumptions (variants of PRF-ODH) because of how the session key is used in the protocol prior to the session being fully authenticated.

In implicitly authenticated key exchange, static key pairs are used to derive shared secrets that can only be computed by the intended parties; having a peer who successfully computes the shared secret implicitly authenticates that peer, in contrast to the explicit authentication provided by checking a signature computed by one's peer. Implicitly authenticated key exchange protocols have long been of academic interest (e.g., [50,17,77]), and have recently started to be used in real-world protocols, such as the original handshake design of Google's QUIC protocol [68,57] or the Signal protocol [72,14], as well as OPTLS [52] which is the conceptual foundation of the TLS 1.3 handshake (cf. Table 2). In these designs, (semi-)static DH keys enable low-latency, zero round-trip time connections. The Signal protocol even focuses on *asynchronous* communication, enabling parties to initiate a connection despite their peer being offline.

Moving to post-quantum solutions. Unfortunately, DH-based protocols are not secure against quantum adversaries, so key exchange protocols need to transition to post-quantum designs. The NIST Post Quantum Cryptography Standardization process [61] is currently in the second round for identifying quantum-resistant primitives. Furthermore, experimental deployment of post-quantum algorithms in key exchange protocols has already taken place, e.g., by Google, Cloudflare, and the Open Quantum Safe project [55,56,53,73,18].

While key exchange protocols are a crucial building block for many applications, the NIST standardization process did not explicitly ask for key ex-

Protocol	Core message flow	Session key	Security
SSHv2 [78] (signed eph. DH)	$\xrightarrow{\texttt{hello}}$ $\xleftarrow{\texttt{hello}}$ $\xrightarrow{epk_A}$ $\xleftarrow{epk_B, lpk_B, \texttt{sig}}$	$\mathsf{DH}(epk_A, epk_B)$	DDH [6]
TLS 1.2 [20] (signed eph. DH)	$\xrightarrow{\texttt{hello}}$ $\xleftarrow{epk_B, \texttt{cert}[lpk_B], \texttt{sig}}$ $\xrightarrow{epk_A}$	$\mathsf{DH}(epk_A, epk_B)$	snPRF-ODH [43]
TLS 1.3 [69] (signed eph. DH)	$\xrightarrow{\texttt{hello}, epk_A}$ $\xleftarrow{epk_B, \texttt{cert}[lpk_B], \texttt{sig}}$	$\mathsf{DH}(epk_A, epk_B)$	snPRF-ODH [30]

Table 1: Signed DH key exchange patterns of selected Internet protocols. With epk_X and lpk_X we denote the ephemeral resp. long-term key of a party, `hello` is the protocol initiator's message, `sig` a signature under the long-term key, and $\texttt{cert}[lpk_X]$ the long-term key and certificate of party X.

change, but for the conceptually simpler notions of key encapsulation mechanisms (KEMs) [71,16]. A KEM allows encapsulation of a symmetric key under a public key within a ciphertext, such that the symmetric key can be decapsulated only when knowing the corresponding secret key. Security-wise, the ciphertext hides the encapsulated symmetric key indistinguishably from a random string. The proposals to the NIST standardization process mostly follow the approach to first provide a (weakly secure) public-key encryption scheme and then use well-known transforms such as the Fujisaki–Okamoto transform [36,37,41] to achieve a strongly-secure KEM with respect to active adversaries.

In principle, KEMs can directly be used to build and analyze key exchange protocols, and allow to capture (implicitly authenticated) Diffie–Hellman flows (e.g., in the static Diffie–Hellman handshake of TLS 1.2 [51]). The naive approach hence would be to simply replace every DH combination in a key exchange protocol with a KEM. However, whereas DH shares can be freely reused by both parties, particularly allowing static-static combinations (as used, e.g., in Signal [72]), the KEM concept restricts reuse to one side, namely the decapsulator. This in turn limits the possible message flows and contributions of ephemeral randomness that standard KEMs can capture, hindering a direct translation of DH-type protocols to KEM-based designs, and leading to the question:

Can we capture post-quantum KEM designs in a way that enables flexible key reuse and support efficient message flows similar to the widely-used Diffie–Hellman-based designs?

Protocol	Core message flow	Session key	Security
TLS 1.2 [20] (implicit-auth. static DH + explicit-auth. MAC)	$\texttt{hello} \rightarrow$ $\leftarrow \texttt{cert}[lpk_B], \texttt{mac}$ $epk_A, \texttt{mac} \rightarrow$	$\mathsf{DH}(epk_A, lpk_B)$	mnPRF-ODH [51]
OPTLS [52] (implicit-auth. static/eph. DH + explicit-auth. MAC)	$\texttt{hello}, epk_A \rightarrow$ $\leftarrow epk_B, \texttt{cert}[lpk_B], \texttt{mac}$	$\mathsf{DH}(epk_A, epk_B) \Vert$ $\mathsf{DH}(epk_A, lpk_B)$	GapDH, DDH [52] (ROM)
Signal [72] (X3DH: triple DH handshake + opt. eph./eph.)	$\texttt{hello} \dashrightarrow$ $\dashleftarrow lpk_B, sspk_B, epk_B^{\dagger}$ $lpk_A, epk_A \rightarrow$	$\mathsf{DH}(lpk_A, sspk_B) \Vert$ $\mathsf{DH}(epk_A, lpk_B) \Vert$ $\mathsf{DH}(epk_A, sspk_B) \Vert$ $\mathsf{DH}(epk_A, epk_B)^{\dagger}$	mmPRF-ODH, smPRF-ODH, smPRF-ODH, snPRF-ODH† [14]
QUIC [68,57] (original handshake)	$\texttt{hello}, epk_A \rightarrow$ $\leftarrow sspk_B$	$\mathsf{DH}(epk_A, lpk_B) \Vert$ $\mathsf{DH}(epk_A, sspk_B)$	GapDH [34] (ROM)

Table 2: Implicitly authenticated DH key exchange patterns of selected Internet protocols.
With epk_X and lpk_X we denote the ephemeral resp. long-term key of a party (the latter might be known to a peer in advance), `hello` is the protocol initiator's message, `mac` a message authentication code under a derived key, and $\texttt{cert}[lpk_X]$ the long-term key and certificate of party X. Dashed arrows in the Signal key exchange indicate obtaining the "prekey bundle" from the Signal server, blue values marked with † are optional.

1.1 Our Contributions

In this paper, we work towards a structure for achieving the key-reusability of Diffie–Hellman-based key exchanges in the KEM setting by introducing the notion of **split KEMs** in Section 4.1. In split KEMs, encapsulation of a shared secret takes as input not only the public key of the decapsulator but also a (potentially static) secret key of the encapsulator. Similarly, decapsulation of a ciphertext takes as input not only the decapsulator's secret key, but also the encapsulator public key corresponding to the secret key used in encapsulation.

In Section 4.2, we illustrate how split KEMs enable the smooth transfer of popular DH-based key exchanges such as **Signal's** X3DH to the (post-quantum) KEM setting. Especially in the case of the Signal protocol, the complex and asynchronous initial key agreement (X3DH, short for "Extended Triple Diffie–Hellman") has been abstracted away as an idealized primitive in works on secure messaging with and without post-quantum security considerations (cf., e.g., the work by Alwen, Coretti, and Dodis [2] which is amenable to post-quantum security).

In Section 4.3, we transfer the commonly-used security notion for KEMs of indistinguishability of encapsulated keys from random to the split KEM setting. For this we introduce the notion of $\mathsf{lr\text{-}IND\text{-}CCA}$ **security** that is parametrized by $\mathsf{l} \in \{\mathsf{n}, \mathsf{s}, \mathsf{m}\}$ and $\mathsf{r} \in \{\mathsf{n}, \mathsf{m}\}$ which will indicate how often the adversary is allowed to query the "left" decapsulation oracle or the "right" encapsulation oracle. (Here, n means no query is allowed, s means a single query is allowed, and m means (polynomially) many queries are allowed.) This **novel fine-grained security notion** allows not only to distinguish between passive and active attacks, but also captures key reuse on either the decapsulator's or the encapsulator's side or on both sides.

As for **realizing the split KEM** notion, we show in Section 4.4 that plain (R)LWE-based KEMs do non-trivially match the split KEM structure and maintain security in this formalization. However, known key reuse attacks against (R)LWE-based KEMs prohibit such an instantiation from being secure against active adversaries. We furthermore give an instantiation of an actively-secure split KEM achieving $\mathsf{mn\text{-}IND\text{-}CCA}$ security based on the KEM introduced by Kiltz [48]. While not being a post-quantum secure split KEM per se, the design and hardness assumptions may be replicable in the CSIDH setting (cf. the following discussion in Section 1.2). Unfortunately, we have so far been unable to successfully develop an instantiation in the strongest $\mathsf{mm\text{-}IND\text{-}CCA}$ setting from a post-quantum assumption. We identify this as an important challenge for future work.

1.2 Related Work

Related work for our split KEM notion includes approaches towards post-quantum security of concrete protocols, notions for key reuse and the possibility of key reuse in various post-quantum settings, as well as foundational extensions to the definitional framework of KEMs.

Post-quantum protocols. Post-quantum secure protocol variants based on KEMs have been proposed for TLS 1.3 [70] and WireGuard [42]. These protocols, unlike Signal, allow (multiple) round trips and therefore do not experience the same problem we discuss in this paper. For Signal, Alwen, Coretti, and Dodis [2] give a first variant of Signal's double-ratchet that is amenable to post-quantum secure KEMs, however exclude the crucial initial key agreement. Duits [33] explores transitioning Signal to the post-quantum setting; the suggested replacement of DH with Supersingular Isogeny Diffie–Hellman (SIDH) [44,19] however is not secure under the required key reuse, as we discuss next.

Key reuse with LWE *and SIDH.* There are a number of attacks on lattice-based key exchange schemes when keys are reused [35,22,26,59,24,67,5,25,40,62]. There exist proposals to enable secure key reuse in (R)LWE-based schemes [39,23], however, these proposals only seem to at most guard against specific attacks at a time, while still being vulnerable to other attacks. All LWE-based KEMs in Rounds 2 and 3 of the NIST process rely on the Fujisaki–Okamoto transform

to achieve IND-CCA security which provides safe key reuse for one party, but comes at the cost of requiring the other party to fully disclose the secret key behind their encapsulation.

Similarly, for key exchange based on SIDH [44] there are attacks when keys are reused [38]; the SIKE NIST submission uses the FO transformation to provide security under key reuse. Azarderakhsh, Jao, and Leonardi [3] proposed k-SIDH for static-static key exchange, where each party has k static keys, and the final shared secret is computed from all k^2 combinations. Security of k-SIDH relies on an additional unproven assumption that the key exchange method be "irreducible", but the best attacks currently known are exponential in k [28]. For k-SIDH at the 128-bit security, each party would need to send $k \approx 100$ public keys and compute $k^2 \approx 10,000$ SIDH computations, making it extremely expensive. A more efficient variant of k-SIDH by Jao and Urbanik [74] was found by Basso et al. [4] to have poorer scaling than the original. There have been several additional attempts which are either inconclusive [7] or insecure [47,28,29].

Static-static key agreement via CSIDH. Castryck et al. [12] introduce a scheme based on supersingular isogenies named CSIDH that supports key reuse without the need for additional transforms. Unlike the previously mentioned supersingular isogeny-based schemes building on [44], CSIDH is considering the hardness of finding isogenies between isogenous supersingular elliptic curves over a *prime* field $\mathbb{F}_p$. While this yields a commutative group action, enabling truly DH-like non-interactive key agreement, the concrete parameter selection for CSIDH has been called into doubt [65,8] and the decisional Diffie–Hellman problem has been challenged for settings related to CSIDH [13] (although not affecting CSIDH itself). De Kock [49] and Kawashima et al. [46] recently considered a translation of the gap Diffie–Hellman assumption [63] to the CSIDH setting as the underlying assumption to construct interactive, tightly post-quantum secure key exchange protocols. So far, the intractability of this and other *interactive* hardness assumptions needed for full-fledged key exchange (cf. Tables 1 and 2) however is unknown for CSIDH.

PRF-ODH. The PRF-ODH assumption is a variant of the oracle-DH assumption [1] which enables to argue pseudorandomness of PRF outputs when keyed with related, reused DH values. It is a natural assumption in DH-based key exchanges and was introduced by Jager et al. in their analysis of TLS 1.2 [43]. Since then it has been used in the analyses of many real-world key exchange protocols, including TLS 1.3 [30,31] and Signal [14,15]. Brendel et al. [11] conducted a systematic study, including the presentation of a unified definition of the various flavors of the PRF-ODH assumption that had been employed in the previous literature. We adopt their unified approach in our definition of lr-IND-CCA security for split KEMs.

Post-quantum KEMs and KEM variants. Strongly-secure (IND-CCA) KEMs can be obtained generically through transforming post-quantum secure public-key

encryption [36,41]. These transforms, however, do not allow to reuse the encapsulator's secret randomness. Xue et al. [76] introduce the notion of *double-key* KEMs. Here, encapsulation and decapsulation take two public, resp. secret, keys as input belonging to the *same* party, while split KEM encapsulation and decapsulation take one public and one secret input from *each* party, enabling static-static key reuse. The notion of *merged* KEMs [32] aims to optimize bandwidth for public ratchets in the Signal protocol; our work and the notion of split KEMs instead is concerned with the initial key agreement with contributions from both parties. In their notion of *multi-recipient* KEMs, Katsumata et al. [45] also decompose the encapsulation process, though without allowing a secret input of the sender to enter encapsulation as in our split KEM notion.

2 Preliminaries

We begin by briefly introducing the notation we require throughout this paper. Since our main concept builds upon the notion of KEMs, we subsequently review their basic syntax and security notions. We note that we define security with respect to *quantum* polynomial time (QPT) algorithms instead of probabilistic polynomial time (PPT) as this work is motivated by the existence of adversaries with access to local quantum computing power. We note that all problems that can be solved by PPT algorithms can also be solved by QPT algorithms, whereas the reverse is not true (e.g., computing discrete logarithms in special groups or factorization of large composite numbers).

2.1 Notation

For an algorithm A we write $y \leftarrow \mathsf{A}(\cdot)$, resp. $y \xleftarrow{\$} \mathsf{A}(\cdot)$, for deterministically, resp. probabilistically, running A on given inputs and assigning the output to y. We say that an algorithm A is *efficient* if it runs in QPT in the security parameter denoted by λ. By $\mathcal{A}^{\mathcal{O}}$ we express that the adversary denoted by $\mathcal{A}$ is given access to oracle $\mathcal{O}$. Finally, we use $\perp$ as a special symbol to denote rejection or an error, and we assume that $\perp \notin \{0,1\}^*$.

2.2 Key Encapsulation Mechanisms

Definition 1. *A* key encapsulation mechanism KEM *with associated public key space $\mathcal{PK}$, secret key space $\mathcal{SK}$, ciphertext space $\mathcal{C}$, and key space $\mathcal{K}$ is a tuple of algorithms* $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *defined as follows.*

- **Key generation** KGen**:** *Takes as input the security parameter λ and outputs a public-secret key pair in $\mathcal{PK} \times \mathcal{SK}$, i.e., $(pk, sk) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$.*
- **Encapsulation** Encaps**:** *Takes as input a public key pk and outputs a ciphertext $c \in \mathcal{C}$ and the encapsulated key $K \in \mathcal{K}$, i.e., $(c, K) \xleftarrow{\$} \mathsf{Encaps}(pk)$.*
- **Decapsulation** Decaps**:** *Takes as input a ciphertext c and secret key sk and outputs $K' \in \mathcal{K} \cup \{\perp\}$, where $\perp$ indicates an error, i.e., $K' \leftarrow \mathsf{Decaps}(sk, c)$.*

$\underline{\mathcal{G}^{\mathsf{indcpa}}_{\mathsf{KEM},\mathcal{A}}(\lambda)}$:
1 $(pk, sk) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
2 $(c^\star, K_0^\star) \xleftarrow{\$} \mathsf{Encaps}(pk)$
3 $K_1^\star \xleftarrow{\$} \mathcal{K}$
4 $b \xleftarrow{\$} \{0,1\}$
5 $b' \xleftarrow{\$} \mathcal{A}(pk, c^\star, K_b^\star)$
6 **return** $[\![b' = b]\!]$

$\underline{\mathcal{G}^{\mathsf{indcca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)}$:
1 $(pk, sk) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
2 $(c^\star, K_0^\star) \xleftarrow{\$} \mathsf{Encaps}(pk)$
3 $K_1^\star \xleftarrow{\$} \mathcal{K}$
4 $b \xleftarrow{\$} \{0,1\}$
5 $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Decaps}}}(pk, c^\star, K_b^\star)$
6 **return** $[\![b' = b]\!]$

$\underline{\mathcal{O}_{\mathsf{Decaps}}(c)}$:
7 **if** $c = c^\star$
8 **return** $\perp$
9 **else**
10 **return** $\mathsf{Decaps}(sk, c)$

Fig. 1: IND-CPA and IND-CCA security games for $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ with key space $\mathcal{K}$.

We say that a KEM $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* ϵ-correct *if*

$$\Pr_{(pk,sk)\xleftarrow{\$}\mathsf{KGen}(1^\lambda),(c,K)\xleftarrow{\$}\mathsf{Encaps}(pk)}\left[K' \neq K \;\middle|\; K' \leftarrow \mathsf{Decaps}(sk, c)\right] \leq \epsilon.$$

We call KEM (perfectly) correct *if* $\epsilon = 0$.

KEM Security. The security of key encapsulation mechanisms can be formulated in terms of indistinguishability as well as in terms of one-wayness. We recap *indistinguishability* (of encapsulated keys), defined under either passive (*chosen-plaintext*, IND-CPA) or active (*chosen-ciphertext*, IND-CCA) attacks. The notion of one-wayness captures the (non-)recoverability of the encapsulated key from the ciphertext. In this paper we focus on the notion of indistinguishability but note that all security notions may be transferred to the one-wayness setting.

Definition 2. *Let* KEM *be a KEM with key space* $\mathcal{K}$*. We say that* KEM *is* IND-CPA-secure*, resp.* IND-CCA-*secure, if for every QPT adversary* $\mathcal{A}$ *the advantage function for winning the game* $\mathcal{G}^{\mathsf{indatk}}_{\mathsf{KEM},\mathcal{A}}$ *(with* $\mathsf{atk} = \mathsf{cpa}$*, resp.* $\mathsf{atk} = \mathsf{cca}$*) from Figure 1, defined as*

$$\mathsf{Adv}^{\mathsf{indatk}}_{\mathsf{KEM},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathcal{G}^{\mathsf{indatk}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|,$$

is negligible in the security parameter λ.

3 Instantiating Signal's X3DH Key Exchange with KEMs

In this section, we illustrate the challenges arising when translating DH-based key exchange protocols to the KEM setting following the example of Signal's X3DH initial key exchange design [72] for secure messaging. We will see that simply replacing DH operations with KEM encapsulations results in additional message flows and altered ephemeral/static share combinations for deriving keying material of the involved parties.

While purely ephemeral DH-based key exchanges generally map well to KEMs, many protocol designs further include DH-based combinations of static or semi-static keys with (semi-)static or ephemeral keys, most importantly for implicit

authentication (as in Signal [72] or the Noise framework [66]). However, KEMs allow only for restricted key reuse (namely only on the decapsulator's side) and are hence limited in their support of static key share combinations.

3.1 X3DH: The Initial Key Agreement in Signal

X3DH [60] is part of the Signal secure messaging protocol [72] and establishes the initial keys. We limit the following discussion to this initial key exchange. For further information on the remaining cryptographic building blocks of the Signal protocol, especially on the ratcheting stages following X3DH, we refer the interested reader to, e.g., the analyses of Cohn-Gordon et al. [14] and Alwen et al. [2].

In Figure 2 we give an illustration of X3DH, where Alice wishes to establish a shared key with Bob. The session setup in Signal involves three parties, namely the communicating parties Alice and Bob, plus a central server S. This is due to the fact that Signal aims to provide secure messaging in an *asynchronous* setting, i.e., chats can be initiated and encrypted messages can be exchanged even if not *all* communication partners are online. For this, all users need to register their long-lived identity key and further cryptographic key material with the central server S. In more detail, every user U provides the server S with the public keys of the following key pairs:

- a long-lived static identity key pair (lpk_U, lsk_U),
- a medium-lived semi-static (signed) prekey pair $(sspk_U, sssk_U)$, and
- n ephemeral prekey pairs $(eppk_U^1, epsk_U^1), \ldots, (eppk_U^n, epsk_U^n)$.

When Alice wants to initiate a chat with Bob, she simply requests the necessary information and cryptographic key material of Bob (the so-called "prekey bundle") from the central server S. From this she then derives an initial shared secret that secures her first message(s) to Bob. Once Bob comes online again and receives the first message from Alice (via the server), he requests Alice's cryptographic key material from the server S to be able to derive the same initial key to decrypt Alice's message.

More formally, Alice initiates a session with Bob by first pinging the server S and requesting Bob's public key material: the static identity key lpk_B, the semi-static prekey $sspk_B$, as well as (optionally, if available) a single ephemeral (one-time) prekey $eppk_B$. Alice then generates an ephemeral key pair (epk_A, esk_A) of her own and derives the master secret ms as

$$\mathsf{ms} \leftarrow sspk_B^{lsk_A} || lpk_B^{esk_A} || sspk_B^{esk_A} || eppk_B^{esk_A},$$

where the last DH value $eppk_B^{esk_A}$ is only present if Alice has received one of Bob's ephemeral prekeys $eppk_B$ from the server. More on this below in Remark 1.

Alice then derives the initial key K from the master secret via a pseudo-random function F keyed with ms and can then use this key to encrypt her first message to Bob. Finally, Alice sends her ephemeral public key epk_A to Bob (alongside identifiers for, e.g., Bob's semi-static and ephemeral prekeys that she

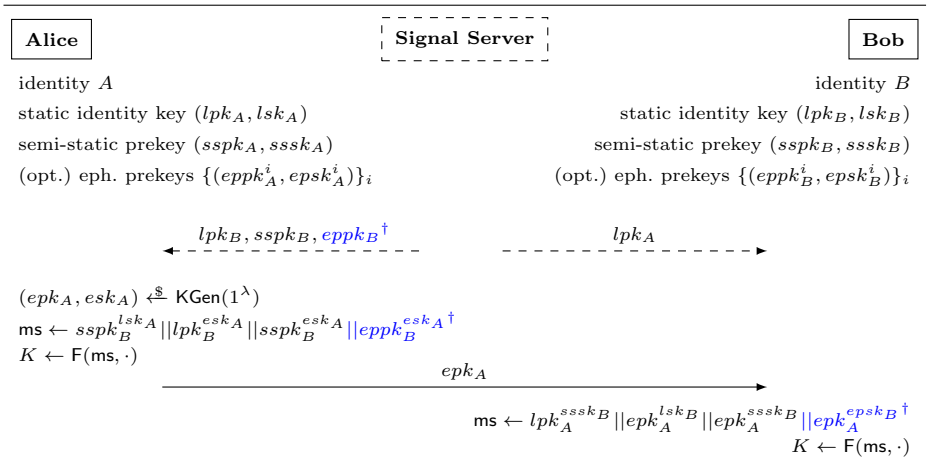

Fig. 2: Signal's X3DH key exchange. Interaction with the Signal server is dashed, the optional ephemeral prekey (combination) is depicted in blue marked with †.

received from the server). Once Bob comes online he will receive this message and can then request Alice's static identity key lpk_A from the server. Analogously to Alice he can then compute the master secret ms and thus the final initial key K that decrypts Alice's first encrypted message to him.

Remark 1 (Exhaustion of ephemeral prekeys). Note that each of the n stored ephemeral prekeys is only handed out once by the server, i.e., in case Charlie wishes to also initiate a session with Bob, he will receive an ephemeral prekey of Bob that is different from the one Alice received. However, if many users initiate a session with Bob while he is offline, it may be the case that the stored ephemeral prekeys on the server are exhausted. Hence, the initial shared secret is only derived from the static identity key lpk_B and the semi-static prekey $sspk_B$.

3.2 A KEM-based X3DH Variant

Considering preparations for a post-quantum secure messaging design, one may ask if any candidate of NIST's post-quantum cryptography process can be used smoothly in the above setting. Unfortunately this is not the case. As mentioned before, replacing the Diffie–Hellman operations in Signal's X3DH protocol with KEMs causes difficulties, as we illustrate in Figure 3 and discuss in the following.

As before, when Alice initiates a session with Bob, she requests and receives Bob's static identity key lpk_B, his semi-static prekey $sspk_B$, as well as a single ephemeral prekey $eppk_B$ (if available) from the server. Alice then separately encapsulates key material under each of these keys and sends the resulting ciphertexts to Bob, establishing three shared keys K_1, K_2, and K_3 (if available).

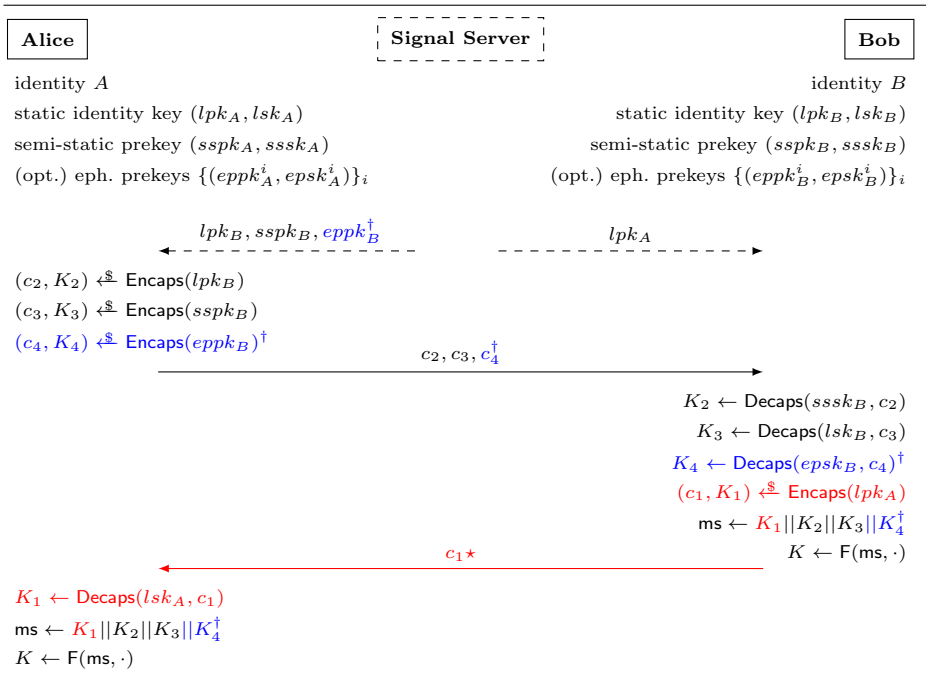

Fig. 3: Signal's X3DH key exchange with KEMs replacing the DH operations. Interaction with the Signal server is dashed, the optional ephemeral prekey (combination) is depicted in blue marked with †. The last flow (in red marked with ⋆), necessary for the key share combination involving Alice's long-term key, breaks the asynchronicity of X3DH.

Yet, in order to fully transfer X3DH to the KEM setting, these three keys are not enough: they constitute, in order, the KEM analogues of the DH secrets $\mathsf{DH}(lpk_B, epk_A)$, $\mathsf{DH}(sspk_B, epk_A)$, and $\mathsf{DH}(eppk_B, epk_A)$, where Alice's ephemeral contribution via epk_A is replaced by (differing) randomness inputs on Alice's side to the encapsulation algorithm. What is missing to complete the master secret computation—and thus key derivation—in the same fashion as in X3DH is the analogue of the DH combination of Alice's static identity key and Bob's semi-static key, i.e., $\mathsf{DH}(sspk_B, lpk_A)$.

KEMs, however, do not provide for a *non-ephemeral* contribution of the encapsulating party to the Encaps algorithm. In the KEM-based X3DH variant, Bob can thus at most encapsulate under Alice's static identity key lpk_A, which introduces an additional message flow (depicted in red in Figure 3). This however

eradicates a key feature of instant messaging: asynchronicity, i.e., the ability to send encrypted messages even if the receiving party is offline.[5]

4 Split Key Encapsulation Mechanisms

To tackle the above mentioned issues of KEMs in a DH-based protocol, we introduce a new primitive called *split key encapsulation mechanism*, or split KEM, for short. Split KEMs enable a more fine-grained notion of key encapsulation mechanisms, where the encapsulation procedure is divided up into key generation and a subsequent shared-key computation step. As it turns out, the passively-secure ($\mathsf{IND\text{-}CPA}$) versions of many proposals for KEMs submitted to the NIST Post-Quantum Cryptography Standardization process [61], especially those based on lattices, seem to naturally fit into the split KEM format: their encapsulation procedure can be split into a key generation and a shared-key computation part.

4.1 Definition of Split KEMs

Intuitively, a split KEM is a KEM in which *both* parties can contribute to the encapsulation, with either one-time or (semi-)static keys. The key generation on the encapsulator's side (that does implicitly take place in many KEMs) is decoupled from the encapsulation algorithm, thus allowing key reuse similar to the DH setting. Figure 4 shows the communication flow when using a split KEM to establish a shared secret.

Notation. Let enc denote the encapsulating party (in the following referred to as the *encapsulator*) and similarly, dec denotes the decapsulating party (or *decapsulator*). Let $\mathcal{PK}_{\mathsf{enc}}$ and $\mathcal{SK}_{\mathsf{enc}}$ be the public and secret key space of the encapsulator, and $\mathcal{PK}_{\mathsf{dec}}$ and $\mathcal{SK}_{\mathsf{dec}}$ analogously for the decapsulator (if irrelevant, or clear from the context, we will in the following omit the explicit mention of these key spaces). Let $\mathcal{C}$ be the ciphertext space and $\mathcal{K}$ the key space.

Definition 3. *A* split KEM sKEM *consists of four algorithms* $\mathsf{KGen}_{\mathsf{dec}}, \mathsf{KGen}_{\mathsf{enc}}, \mathsf{sEncaps}$, *and* $\mathsf{sDecaps}$, *where* $\mathsf{KGen}_{\mathsf{enc}}$ *and* $\mathsf{sEncaps}$ *are executed by the encapsulator, and* $\mathsf{KGen}_{\mathsf{dec}}$ *and* $\mathsf{sDecaps}$ *by the decapsulator.*

- ***split KEM key generation*** *for decapsulator and encapsulator, respectively:* $(D, d) \xleftarrow{\$} \mathsf{KGen}_{\mathsf{dec}}(1^\lambda)$ *and* $(E, e) \xleftarrow{\$} \mathsf{KGen}_{\mathsf{enc}}(1^\lambda)$ *are probabilistic algorithms that output a key pair, consisting of a public key (denoted by capital letters) and a secret key (denoted by lowercase letters) in* $\mathcal{PK}_{\mathsf{dec}} \times \mathcal{SK}_{\mathsf{dec}}$ *and* $\mathcal{PK}_{\mathsf{enc}} \times \mathcal{SK}_{\mathsf{enc}}$*, respectively.*

[5] Note that it is in general not possible for Bob to precompute and store ciphertext(s) on the server alongside his public keys to avoid the additional message flow since Bob may not know in advance which user wishes to establish a secure chat with him.

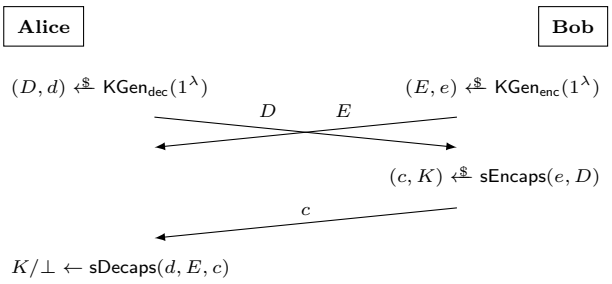

Fig. 4: Communication flow of a split KEM $\mathsf{sKEM} = (\mathsf{KGen}_{\mathsf{dec}}, \mathsf{KGen}_{\mathsf{enc}}, \mathsf{sEncaps}, \mathsf{sDecaps})$, where Alice is the decapsulator and Bob the encapsulator.

- ***split KEM encapsulation****:* $(c, K) \xleftarrow{\$} \mathsf{sEncaps}(e, D)$ *is a probabilistic algorithm executed by the encapsulator* enc*. It takes as input* $e \in \mathcal{SK}_{\mathsf{enc}}$*, the secret key of the encapsulator, and* $D \in \mathcal{PK}_{\mathsf{dec}}$*, the public key of the decapsulator. Algorithm* sEncaps *then outputs the shared secret* $K \in \mathcal{K}$ *along with its* encapsulation $c \in \mathcal{C}$*. It is common to simply refer to the encapsulation* c *of* K *as* ciphertext*.*
- ***split KEM decapsulation****:* $K/\perp \leftarrow \mathsf{sDecaps}(d, E, c)$ *is a deterministic algorithm executed by the decapsulator* dec*. 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* $\perp$*, if the operation fails.*

We say that a split key encapsulation $\mathsf{sKEM} = (\mathsf{KGen}_{\mathsf{dec}}, \mathsf{KGen}_{\mathsf{enc}}, \mathsf{sEncaps}, \mathsf{sDecaps})$ *is* ϵ-correct *if*

$$\Pr_{(D,d)\xleftarrow{\$}\mathsf{KGen}_{\mathsf{dec}},(E,e)\xleftarrow{\$}\mathsf{KGen}_{\mathsf{enc}},(c,K)\xleftarrow{\$}\mathsf{sEncaps}(e,D)}[K' \neq K \mid K' \leftarrow \mathsf{sDecaps}(d,E,c)] \leq \epsilon.$$

We call sKEM (perfectly) correct *if* $\epsilon = 0$*.*

Symmetric Split KEMs. In some supersingular-isogeny-based KEMs the specification of the key generation algorithm depends on the role of the generating party (cf., e.g., the NIST Round 2 candidate SIKE [19]). In these schemes, Alice and Bob generate public points in different subgroups of the curve during key generation, i.e., $\mathsf{KGen}_{\mathsf{dec}} \neq \mathsf{KGen}_{\mathsf{enc}}$. However, there are also many natural examples (e.g., DH- or LWE-based KEMs), where the key generation algorithms for the encapsulator and the decapsulator do not differ. This allows generated key pairs to be used as input for both the encapsulation and decapsulation algorithms, i.e, across roles. In order to capture these special types of split KEMs, we introduce the notion of a *symmetric* split KEM.

Definition 4 (Symmetric Split KEM). *We call a split KEM* $\mathsf{sKEM} = (\mathsf{KGen}_{\mathsf{dec}}, \mathsf{KGen}_{\mathsf{enc}}, \mathsf{sEncaps}, \mathsf{sDecaps})$ symmetric *if* $\mathsf{KGen}_{\mathsf{dec}} = \mathsf{KGen}_{\mathsf{enc}}$ *and the same key pair of a party is reused in both roles. In particular, this means that*

$\mathcal{PK}_{\mathsf{dec}} = \mathcal{PK}_{\mathsf{enc}}$ *and* $\mathcal{SK}_{\mathsf{dec}} = \mathcal{SK}_{\mathsf{enc}}$*. For sake of simplicity, in this case we will often simply refer to the key generation algorithm as* KGen *instead of* $\mathsf{KGen}_{\mathsf{dec}}$ *and* $\mathsf{KGen}_{\mathsf{enc}}$*, respectively.*
We say that $\mathsf{sKEM} = (\mathsf{KGen}, \mathsf{sEncaps}, \mathsf{sDecaps})$ *is* ϵ-correct *if both*

$$\Pr_{(D,d),(E,e)\xleftarrow{\$}\mathsf{KGen}(1^\lambda),(c,K)\xleftarrow{\$}\mathsf{sEncaps}(e,D)}\left[K' \neq K \;\middle|\; K' \leftarrow \mathsf{sDecaps}(d,E,c)\right] \leq \epsilon$$

and

$$\Pr_{(D,d),(E,e)\xleftarrow{\$}\mathsf{KGen}(1^\lambda),(c,K)\xleftarrow{\$}\mathsf{sEncaps}(d,E)}\left[K' \neq K \;\middle|\; K' \leftarrow \mathsf{sDecaps}(e,D,c)\right] \leq \epsilon.$$

Again, as before, a symmetric split KEM is called (perfectly) correct *if* $\epsilon = 0$.

We stress that it is not necessary to move to the symmetric split KEM setting if the key generation algorithms are the same for the encapsulator and decapsulator, but a resulting key pair is only ever reused for a fixed role. However, the symmetric split KEM setting is predestined for protocols like Signal's X3DH with KEMs. There, the long-term identity keys are used in both roles, either as the initiating party (the encapsulator) or the responder (the decapsulator).

4.2 X3DH with Split KEMs

We briefly show that using the split KEM formalism could solve the aforementioned problems when switching from the DH to the KEM setting. Figure 5 illustrates the flow between Alice and Bob using only split KEMs. On the one hand, the formalization of split KEMs may now allow both parties to reuse key pairs and have them both contribute to the encapsulation operation(s). Furthermore, regarding the issue of having to encapsulate without knowing the corresponding public key, the split KEM formalism gets rid of the additional message flow from Bob to Alice, thereby effectively regaining the asynchronicity of the secure messaging application.

4.3 Security of Split KEMs

When translating the security definitions from the KEM setting (cf. Section 2.2) to split KEMs, we need to address that encapsulation now contains a secret-key input e and the potential reuse of keys. We arrive at fine-grained security notions that we term lr-IND-CCA (cf. Figure 6), which are parametrized by $\mathsf{l} \in \{\mathsf{n}, \mathsf{s}, \mathsf{m}\}$ and $\mathsf{r} \in \{\mathsf{n}, \mathsf{m}\}$. Here, $\mathsf{l} \in \{\mathsf{n}, \mathsf{s}, \mathsf{m}\}$ 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, $\mathsf{r} \in \{\mathsf{n}, \mathsf{m}\}$ 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 polynomially many times. More formally:

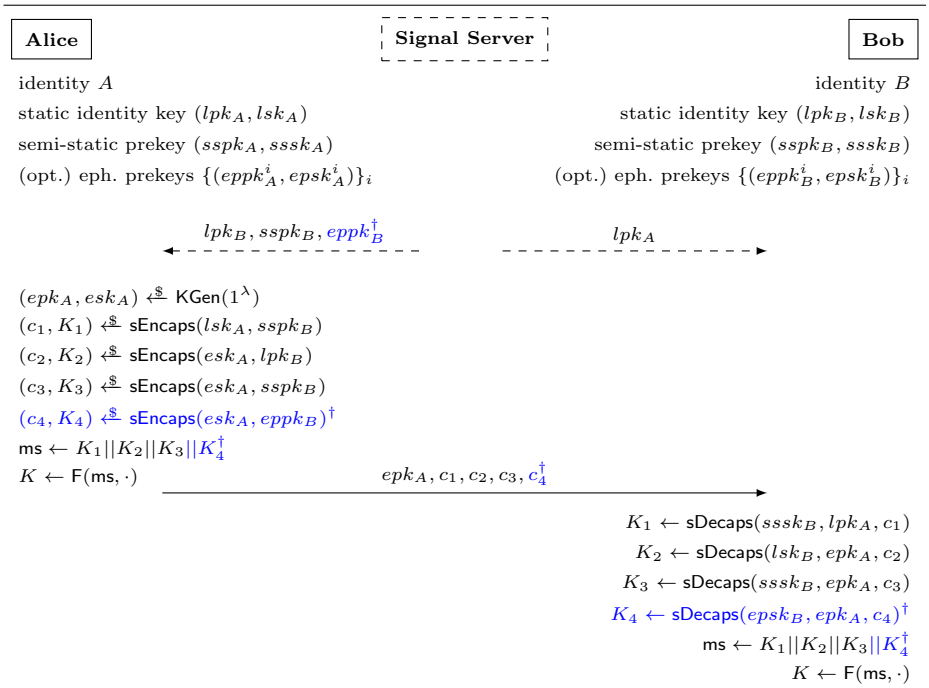

Fig. 5: Split KEM flow for the KEM-based version of Signal's X3DH handshake. Interaction with the Signal server is dashed, the optional ephemeral prekey (combination) is depicted in blue marked with †.

Definition 5. *Let* $\mathsf{sKEM} = (\mathsf{KGen}_{\mathsf{dec}}, \mathsf{KGen}_{\mathsf{enc}}, \mathsf{sEncaps}, \mathsf{sDecaps})$ *be a split KEM with key space* $\mathcal{K}$. *Let* $\mathsf{l} \in \{\mathsf{n}, \mathsf{s}, \mathsf{m}\}$ *and* $\mathsf{r} \in \{\mathsf{n}, \mathsf{m}\}$. *We say* sKEM *provides* lr-indistinguishability under chosen-ciphertext attacks, *or for short,* sKEM *is* lr-IND-CCA-*secure, if for every QPT adversary* $\mathcal{A}$ *the advantage* $\mathsf{Adv}^{\mathsf{lr\text{-}indcca}}_{\mathsf{sKEM},\mathcal{A}}(\lambda)$ *in winning the game* $\mathcal{G}^{\mathsf{lr\text{-}indcca}}_{\mathsf{sKEM},\mathcal{A}}(\lambda)$ *as depicted in Figure 6 defined as*

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

is negligible in the security parameter λ.

At this point, a couple of remarks are in order to motivate the definition:

Remark 2. One may wonder, why, in contrast to the regular KEM setting, the adversary $\mathcal{A}$ is given access to an encapsulating oracle $\mathcal{O}_{\mathsf{sEncaps}}$. This oracle must be provided to the adversary since in the split KEM setting, the encapsulation algorithm sEncaps not only takes as input the public key D of the decapsulating party, but also its own secret key e. As elaborated in the next remark, in some settings $\mathcal{A}$ must however be able to learn encapsulations $\mathsf{sEncaps}(e, D')$ for public

$\underline{\mathcal{G}^{\mathsf{lr\text{-}indcca}}_{\mathsf{sKEM},\mathcal{A}}(\lambda)}$:
1 $n_l, n_r \leftarrow 0$
2 $(D, d) \xleftarrow{\$} \mathsf{KGen}_{\mathsf{dec}}(1^\lambda)$
3 $(E, e) \xleftarrow{\$} \mathsf{KGen}_{\mathsf{enc}}(1^\lambda)$
4 $(c^\star, K_0^\star) \xleftarrow{\$} \mathsf{sEncaps}(e, D)$
5 $K_1^\star \xleftarrow{\$} \mathcal{K}$
6 $b \xleftarrow{\$} \{0,1\}$
7 $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{sDecaps}}, \mathcal{O}_{\mathsf{sEncaps}}}(D, E, c^\star, K_b^\star)$
8 **return** $[\![b' = b]\!]$

$\underline{\mathcal{O}_{\mathsf{sDecaps}}(E', c)}$:
9 **if** $n_l \geq \mathsf{l}$
10 **return** $\perp$
11 $n_l = n_l + 1$
12 **if** $(E', c) = (E, c^\star)$
13 **return** $\perp$
14 $K \leftarrow \mathsf{sDecaps}(d, E', c)$
15 **return** K

$\underline{\mathcal{O}_{\mathsf{sEncaps}}(D')}$:
16 **if** $n_r \geq \mathsf{r}$
17 **return** $\perp$
18 $n_r = n_r + 1$
19 $(c, K) \xleftarrow{\$} \mathsf{sEncaps}(e, D')$
20 **if** $(D', c) = (D, c^\star)$
21 **return** $\perp$
22 **else**
23 **return** (c, K)

$\underline{\mathcal{G}^{\mathsf{mm\text{-}sym\text{-}indcca}}_{\mathsf{sKEM},\mathcal{A}}(\lambda)}$:
1 $(D, d) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
2 $(E, e) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
3 $(c^\star, K_0^\star) \xleftarrow{\$} \mathsf{sEncaps}(e, D)$
4 $K_1^\star \xleftarrow{\$} \mathcal{K}$
5 $b \xleftarrow{\$} \{0,1\}$
6 $b' \xleftarrow{\$} \mathcal{A}^{\left[\mathcal{O}^{sk}_{\mathsf{sDecaps}}, \mathcal{O}^{sk}_{\mathsf{sEncaps}}\right]_{sk \in \{d,e\}}}(D, E, c^\star, K_b^\star)$
7 **return** $[\![b' = b]\!]$

$\underline{\mathcal{O}^{sk}_{\mathsf{sDecaps}}(pk, c)}$:
8 **if** $(sk = d \wedge (pk, c) = (E, c^\star))$
$\vee\ (sk = e \wedge (pk, c) = (D, c^\star))$
9 **return** $\perp$
10 **else**
11 **return** $\mathsf{sDecaps}(sk, pk, c)$

$\underline{\mathcal{O}^{sk}_{\mathsf{sEncaps}}(pk)}$:
12 $(c, K) \xleftarrow{\$} \mathsf{sEncaps}(sk, pk)$
13 **if** $(pk, c) = (D, c^\star)$
14 **return** $\perp$
15 **else**
16 **return** (c, K)

Fig. 6: Top: lr-IND-CCA security of split KEMs, where $\mathsf{l}, \mathsf{r} \in \{\mathsf{n}, \mathsf{s}, \mathsf{m}\}$. Note that we only consider $\mathsf{lr} \in \{\mathsf{nn}, \mathsf{sn}, \mathsf{mn}, \mathsf{sm}, \mathsf{mm}\}$ to be relevant in our setting (see Remark 3). Bottom: Definition of mm-sym-IND-CCA security of symmetric split KEMs, where the key pairs (D, d) and (E, e) are reused across roles. In numerical evaluations for l and r we naturally define $\mathsf{n} = 0, \mathsf{s} = 1$, and $\mathsf{m} = \infty$.

keys D' of its own choosing. As this operation cannot be executed by $\mathcal{A}$ itself due to the secret-key input e, an encapsulation oracle will be provided in these cases.

Remark 3. We next discuss the six flavors $\mathsf{lr} \in \{\mathsf{nn}, \mathsf{sn}, \mathsf{sm}, \mathsf{mn}, \mathsf{nm}, \mathsf{mm}\}$. In the following, let (D, d) and (E, e) be the respective decapsulating and encapsulating key pairs used for generating the challenge ciphertext $c^\star$ and real encapsulated key $K_0^\star$ in the lr-IND-CCA game.

- $\mathsf{lr} = \mathsf{nn}$: The notion of nn-IND-CCA security corresponds to the IND-CPA case in the classical setting, where the adversary is only able to learn the public keys and challenge ciphertext and key, but remains passive. Furthermore, $\mathcal{A}$ is passive and may thus not learn decapsulations of ciphertexts $c' \neq c^\star$.
- $\mathsf{lr} = \mathsf{sn}$: The notion of sn-IND-CCA captures that an active adversary may alter the challenge ciphertext $c^\star$ on the transit from the encapsulator to the decapsulator in a setting where no keys are reused. Thus, $\mathcal{A}$ is allowed to learn a single decapsulation $\mathsf{sDecaps}(d, E', c)$ for $(E', c) \neq (E, c^\star)$.
- $\mathsf{lr} = \mathsf{sm}$: The notion of sm-IND-CCA is similar to the notion of sn-IND-CCA, but here the encapsulator's key pair (E, e) is reused across multiple encapsulations. Therefore, in addition to the single query to $\mathcal{O}_{\mathsf{sDecaps}}$, $\mathcal{A}$ may now also learn multiple encapsulations $\mathsf{sEncaps}(e, D')$.
- $\mathsf{lr} = \mathsf{mn}$: The notion of mn-IND-CCA security corresponds to a reuse of the key material (D, d) on the decapsulator's side in the presence of an active

adversary. Hence, the adversary is able to learn multiple decapsulations $\mathsf{sDecaps}(d, E', c)$ for $(E', c) \neq (E, c^\star)$.
- $\mathsf{lr} = \mathsf{nm}$: The notion of nm-IND-CCA security corresponds to a reuse of the key material (E, e) on the encapsulator's side in the presence of an active adversary. Hence, the adversary is able to learn multiple encapsulations $\mathsf{sEncaps}(e, D')$ for $(D', c) \neq (D, c^\star)$. Having $\mathsf{l} = \mathsf{n}$ encodes that the adversary however cannot obtain the decapsulation of any different ciphertext $c' \neq c^\star$ under D (e.g., due to $c^\star$ being authentically transmitted to the decapsulator).
- $\mathsf{lr} = \mathsf{mm}$: Finally, this notion mm-IND-CCA corresponds to reuse of keys (D, d) and (E, e) on both the decapsulator's and encapsulator's side (in fixed roles). This entails that the adversary can learn multiple related decapsulations and encapsulations.

We note the similarity to the lrPRF-ODH assumption formalization introduced in [11]. This assumption captures security of DH-based key exchanges in the presence of active adversaries in different reuse scenarios.

Security of Symmetric Split KEMs. In the following, we also provide the indistinguishability-based security notion for symmetric split KEMs. Since symmetric split KEMs inherently model key reuse across roles, we only consider the notion of mm-sym-IND-CCA to be relevant in practical settings and therefore do not define lr-sym-IND-CCA in its full generality.[6]

Definition 6. *Let* $\mathsf{sKEM} = (\mathsf{KGen}, \mathsf{sEncaps}, \mathsf{sDecaps})$ *be a symmetric split KEM with key space* $\mathcal{K}$*. We say* sKEM *provides symmetric* mm-*indistinguishability under* chosen-ciphertext attacks*, in short* sKEM *is* mm-sym-IND-CCA*-secure, if for every QPT adversary* $\mathcal{A}$ *the advantage* $\mathsf{Adv}^{\mathsf{mm\text{-}sym\text{-}indcca}}_{\mathsf{sKEM},\mathcal{A}}$ *in winning the game* $\mathcal{G}^{\mathsf{mm\text{-}sym\text{-}indcca}}_{\mathsf{sKEM},\mathcal{A}}(\lambda)$ *as depicted in Figure 6 defined as*

$$\mathsf{Adv}^{\mathsf{mm\text{-}sym\text{-}indcca}}_{\mathsf{sKEM},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathcal{G}^{\mathsf{mm\text{-}sym\text{-}indcca}}_{\mathsf{sKEM},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|$$

is negligible in the security parameter λ*.*

Remark 4. One may wonder whether it is possible to turn every secure symmetric split KEM into a split KEM, and vice versa. It is easy to see that every mm-sym-IND-CCA secure symmetric split KEM is also mm-IND-CCA secure in the sense of non-symmetric split KEMs. The key generation algorithm is simply executed in fixed decapsulator and encapsulator roles and the reduction is straightforward by directly embedding the mm-sym-IND-CCA challenge into the mm-IND-CCA game and relaying the oracle queries and answers to the respective oracles. For the other direction, let $\mathsf{sKEM} = (\mathsf{KGen}_{\mathsf{dec}}, \mathsf{KGen}_{\mathsf{enc}}, \mathsf{sEncaps},$

[6] Note that the symmetric split KEM setting implies key reuse, obsoleting $\mathsf{lr} = \mathsf{nn}$. We further consider the notions $\mathsf{lr} \in \{\mathsf{sn}, \mathsf{mn}, \mathsf{sm}, \mathsf{nm}\}$ to be artificial as these notions encode that only some parties reuse keys across roles while other do not.

sDecaps) be an mm-IND-CCA-secure split KEM. Then $\mathsf{sKEM}' = (\mathsf{KGen}', \mathsf{sEncaps}', \mathsf{sDecaps}')$ as defined in Figure 7 is a mm-sym-IND-CCA-secure symmetric split KEM. Again, this can be shown by a simple reduction, where the mm-IND-CCA reduction embeds its challenge $(D, E, c^\star, k^\star)$ in the following manner:

- The mm-sym-IND-CCA adversary $\mathcal{A}$ expects as input $(pk, pk', c^\star, K^\star)$, where pk and pk' are outputs of KGen' and thus are of the form $pk = (pk_{\mathsf{dec}}, pk_{\mathsf{enc}})$ and $pk' = (pk'_{\mathsf{dec}}, pk'_{\mathsf{enc}})$.
- $\mathcal{B}$ now sets $pk_{\mathsf{dec}} \leftarrow D$ and $pk'_{\mathsf{enc}} \leftarrow E$. It generates the remaining key components itself via the respective key generation algorithm.
- Oracle queries concerning d and e of $\mathcal{A}$ can simply be relayed to $\mathcal{B}$'s respective oracles. The other oracles, $\mathcal{B}$ can answer itself due to the knowledge of the secret keys.
- At some point, $\mathcal{A}$ will output its guess b' and $\mathcal{B}$ can output the same guess.

$\underline{\mathsf{KGen}'(1^\lambda)}$:
1 $(D, d) \xleftarrow{\$} \mathsf{KGen}_{\mathsf{dec}}(1^\lambda)$
2 $(E, e) \xleftarrow{\$} \mathsf{KGen}_{\mathsf{enc}}(1^\lambda)$
3 $pk \leftarrow (D, E)$
4 $sk \leftarrow (d, e)$
5 **return** (pk, sk)

$\underline{\mathsf{sEncaps}'(sk = (d, e), pk' = (D', E'))}$:
6 $(c, K) \xleftarrow{\$} \mathsf{sEncaps}(e, D')$
7 **return** (c, K)

$\underline{\mathsf{sDecaps}'(sk' = (d', e'), pk = (D, E), c)}$:
8 $K \leftarrow \mathsf{sDecaps}(d', E, c)$
9 **return** K

Fig. 7: Transform of mm-IND-CCA-secure split KEM $\mathsf{sKEM} = (\mathsf{KGen}_{\mathsf{dec}}, \mathsf{KGen}_{\mathsf{enc}}, \mathsf{sEncaps}, \mathsf{sDecaps})$ to mm-sym-IND-CCA-secure symmetric split KEM $\mathsf{sKEM}' = (\mathsf{KGen}', \mathsf{sEncaps}', \mathsf{sDecaps}')$.

4.4 Instantiations of Split KEMs

We now turn to giving instantiations of secure split KEMs. Let us start by showing that plain lattice-based KEMs secure under the *(Ring-)Learning with Errors* assumption (R)LWE naturally fit the split KEM flow and maintain their security against passive adversaries in this setting.

nn-IND-CCA Security from (R)LWE. Figure 8 illustrates a generic (R)LWE-based key exchange viewed as a split KEM. Encapsulation on Bob's side is split into the generation of Bob's key pair as well as the final encapsulation of the shared key via the computation of an approximate shared secret and so-called *reconciliation information*, which constitutes the ciphertext. As mentioned before, these constructions are not secure against active attacks and/or key reuse in the "standard" KEM setting, and thus they naturally shouldn't achieve security above nn-IND-CCA in their split KEM formalization. We reduce the nn-IND-CCA to what is commonly referred to as the DDH-*like problem* (DDHℓ), which we state for LWE in the following. Note that the hardness of decision (R)LWE implies hardness of the DDH-like problem (cf., e.g., [58,10]).

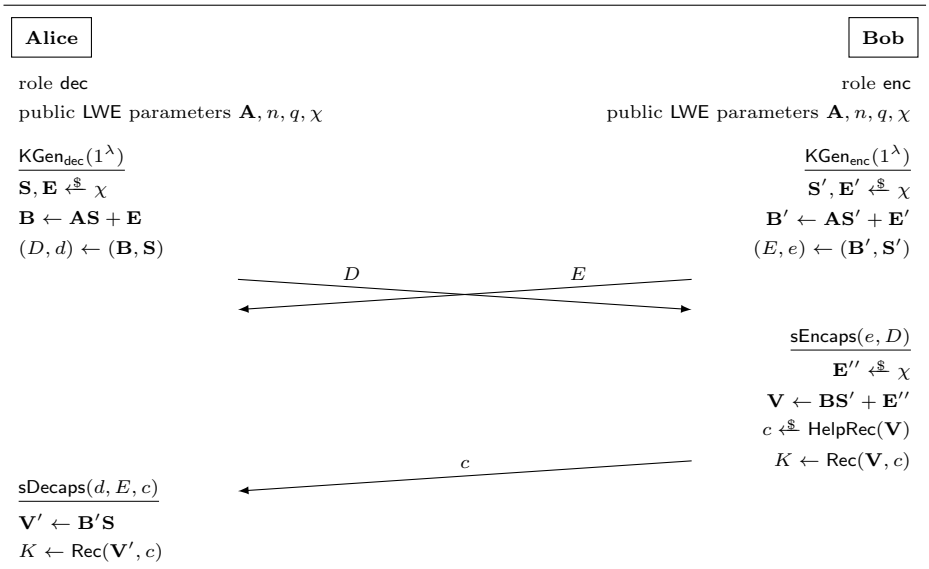

Fig. 8: Instantiation of split KEM flow with plain LWE as, e.g., in [58,27,64,9] with LWE parameters n, q, χ and fixed, public $\mathbf{A}$. The functions HelpRec and Rec aid computation of the shared secret K from the approximate shared secrets $\mathbf{V}$, $\mathbf{V}'$ and vary among different (R)LWE-based schemes.

Definition 7 (DDHℓ Problem). *Let $\mathbf{A}, n, q, \chi$ be* LWE *parameters. Given reconciliation information c, the* decision Diffie–Hellman-like *problem* (DDHℓ) *for $\mathbf{A}, q, n, \chi$ is to distinguish $(\mathbf{B}, \mathbf{B}', c, K)$ from $(\mathbf{B}, \mathbf{B}', c, K')$, where*

- $\mathbf{S}, \mathbf{S}', \mathbf{E}, \mathbf{E}', \mathbf{E}'' \xleftarrow{\$} \chi$,
- $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$, *and* $\mathbf{B}' \leftarrow \mathbf{AS}' + \mathbf{E}'$,
- $\mathbf{V} \leftarrow \mathbf{BS}' + \mathbf{E}''$, *and* $c \xleftarrow{\$} \mathsf{HelpRec}(\mathbf{V})$,
- $K \leftarrow \mathsf{Rec}(\mathbf{V}, c)$ *and* $K' \xleftarrow{\$} \mathcal{K}$ *is a random element in the key space.*

For an algorithm $\mathcal{A}$ we define the distinguishing advantage to be

$$\mathsf{Adv}^{\mathsf{DDH}\ell}_{(\mathbf{A},n,q,\chi),\mathcal{A}}(\lambda) := \big|\Pr[\mathcal{A}(\mathbf{B}, \mathbf{B}', c, K) = 1] - \Pr[\mathcal{A}(\mathbf{B}, \mathbf{B}', c, K') = 1]\big|,$$

and say DDHℓ *is hard if $\mathsf{Adv}^{\mathsf{DDH}\ell}_{(\mathbf{A},n,q,\chi),\mathcal{A}}$ is negligible in the security parameter λ.*

Theorem 1. *Let* sKEM $= (\mathsf{KGen}_{\mathsf{dec}}, \mathsf{KGen}_{\mathsf{enc}}, \mathsf{sEncaps}, \mathsf{sDecaps})$ *be a split KEM with key space $\mathcal{K}$ as in Figure 8 for secure* LWE *parameters $\mathbf{A}, n, q, \chi$. Then for any QPT adversary $\mathcal{A}$* sKEM *is* nn-IND-CCA *secure, assuming the hardness of the* DDHℓ *problem for $\mathbf{A}, n, q, \chi$.*

Proof. By straightforward reduction. We show that if there exists an efficient adversary $\mathcal{A}$ against the nn-IND-CCA security of sKEM, then this immediately

$\underline{\mathsf{KGen}_{\mathsf{dec}}(1^\lambda)}$:
1 $x, y \xleftarrow{\$} \mathbb{Z}_p^\star$
2 $X \leftarrow g^x$
3 $Y \leftarrow g^y$
4 $D \leftarrow (X, Y)$
5 $d \leftarrow (x, y)$
6 **return** (D, d)

$\underline{\mathsf{KGen}_{\mathsf{enc}}(1^\lambda)}$:
7 $e \xleftarrow{\$} \mathbb{Z}_p^\star$
8 $E \leftarrow g^e$
9 **return** (E, e)

$\underline{\mathsf{sEncaps}(e, D = (X, Y))}$:
10 $t \leftarrow \mathsf{G}(g^e)$
11 $c \leftarrow (X^t Y)^e$
12 $K \leftarrow \mathsf{H}(X^e)$
13 **return** (c, K)

$\underline{\mathsf{sDecaps}(d, E, c)}$:
14 $t' \leftarrow \mathsf{G}(E)$
15 **if** $E^{xt'+y} \neq c$
16 **return** $\perp$
17 **else**
18 $K \leftarrow \mathsf{H}(E^x)$
19 **return** K

Fig. 9: Instantiation of mn-IND-CCA-secure split KEM from the KEM by Kiltz [48]. Here, $\mathsf{G}\colon \mathbb{G} \to \mathbb{Z}_p$ is a target collision resistant function defined over a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order p. Furthermore, $\mathsf{H}\colon \mathbb{G} \to \{0,1\}^\lambda$ is a random hash function such that the gap hashed Diffie–Hellman assumption GapHDH is intractable in the security parameter λ wrt. $(\mathbb{G}, g, p, \mathsf{H})$.

implies an efficient solver $\mathcal{B}$ for the$\mathsf{DDH}\ell$ problem over the same parameter set $\mathbf{A}, n, q, \chi$. $\mathcal{B}$ receives as input a tuple $(\mathbf{B}, \mathbf{B}', c, K_b^\star)$, where $\mathbf{B}, \mathbf{B}', c$ are computed as $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$, $\mathbf{B}' \leftarrow \mathbf{AS}' + \mathbf{E}'$ and $c \xleftarrow{\$} \mathsf{HelpRec}(\mathbf{V})$, where $\mathbf{V} \leftarrow \mathbf{BS}' + \mathbf{E}''$a nd $\mathbf{S}, \mathbf{S}', \mathbf{E}, \mathbf{E}', \mathbf{E}'' \xleftarrow{\$} \chi$. Depending on the internal $\mathsf{DDH}\ell$ challenge bit $b \xleftarrow{\$} \{0,1\}$, $K_b^\star$ is either $\mathsf{Rec}(\mathbf{V}, c)$ (for $b = 0$) or a random key from the key space $\mathcal{K}$ (for $b = 1$). $\mathcal{B}$ then runs $\mathcal{A}$ on input $(\mathbf{B}, \mathbf{B}', c, K_{b'}^\star)$, i.e., $D \leftarrow B, E \leftarrow B', c^\star \leftarrow c$, and $K_b^\star$. Note that in the nn-IND-CCA case, $\mathcal{A}$ has no access to $\mathcal{O}_{\mathsf{sEncaps}}$ and $\mathcal{O}_{\mathsf{sDecaps}}$, i.e., $\mathcal{B}$ must not simulate any oracle queries. At some point, $\mathcal{A}$ will then output a guess bit b', and $\mathcal{B}$ will output the same bit. If $\mathcal{A}$ is successful in the nn-IND-CCA game it has successfully distinguished whether $K_b^\star$ is the decapsulation of c or a random key, analogous to the $\mathsf{DDH}\ell$ challenge. □

mn-IND-CCA security from GapHDH. For a non-trivial mn-IND-CCA secure instantiation of a split KEM we rephrase the IND-CCA secure KEM by Kiltz [48] based on the intractability of the gap hashed-DH assumption (GapHDH) as a split KEM. Informally, the GapHDH assumption says that an adversary cannot distinguish the two distributions $(g^x, g^y, \mathsf{H}(g^{xy}))$ and (g^x, g^y, R) for random exponents x, y, and random bit string R from the range of the hash function R, even when given a DDH oracle which on input (g^a, g^b, g^c) determines whether $g^c = g^{ab}$ or not.

Recall that the split KEM notion exploits that some encapsulation algorithms of KEMs perform an implicit key generation before computing the ciphertext and the encapsulated shared key. Thus, Figure 9 is the same as the original KEM in [48], but with the key generation of the encapsulator made explicit in $\mathsf{KGen}_{\mathsf{enc}}$. Obviously, this KEM is not post-quantum secure. However, if we postulate the hardness of gap hashed-DH in the CSIDH setting, post-quantum security is achieved.[7]

[7] Recently, de Kock [49] and Kawashima et al. [46] used a translation of the conceptually related gap Diffie–Hellman (GapDH [63]) assumption to the CSIDH setting to construct interactive, post-quantum secure key exchange protocols with tight security.

Theorem 2. *Let* $(\mathbb{G}, g, p, \mathsf{H})$*, where* $\mathbb{G} = \langle g \rangle$ *is a cyclic group of prime order* p *and* $\mathsf{H}\colon \mathbb{G} \to \{0,1\}^\lambda$ *is a random hash function, chosen such that* GapHDH *is intractable in the security parameter* λ*. Furthermore, let* $\mathsf{G}\colon \mathbb{G} \to \mathbb{Z}_p$ *be a target collision resistant function. Then* $\mathsf{sKEM} = (\mathsf{KGen}_{\mathsf{dec}}, \mathsf{KGen}_{\mathsf{enc}}, \mathsf{sEncaps}, \mathsf{sDecaps})$ *as defined in Figure 9 is an* mn-IND-CCA *secure split KEM.*

Proof Sketch. The proof works analogous to the proof given in [48]. We suppose there exists an efficient adversary $\mathcal{A}$ against the mn-IND-CCA security of sKEM (cf. Figure 9) and show that this immediately implies an efficient adversary $\mathcal{B}$ against GapHDH. The reduction $\mathcal{B}$ gets as input a GapHDH challenge of the form (g^x, g^y, Z) and shall decide whether Z equals $\mathsf{H}(g^{xy})$ or a random bit string in $\{0,1\}^\lambda$. $\mathcal{B}$ samples $w \xleftarrow{\$} \mathbb{Z}_p^\star$ and computes $t^\star \leftarrow \mathsf{G}(g^y)$ for the target collision resistant function G specified by sKEM.

$\mathcal{B}$ initiates $\mathcal{A}$ on input $(D \leftarrow (g^x, Y), E \leftarrow g^y, c^\star, Z)$, where $c^\star \leftarrow g^{yw}$ and $Y \leftarrow (g^x)^{-t^\star} g^w$. $\mathcal{A}$ may then query the decapsulation oracle $\mathcal{O}_{\mathsf{sDecaps}}$ multiple times on public encapsulator keys E' and ciphertexts c of its choice. The reduction simulates $\mathcal{O}_{\mathsf{sDecaps}}$ analogously to the decapsulation oracle in the IND-CCA proof in [48]. $\mathcal{B}$ rejects pairs (E', c) that are equal to $(E, c^\star)$. Otherwise it checks consistency of the ciphertexts corresponding to Line 15 of the decapsulation process by querying its DDH oracle on $(g^{xt}Y, E', c)$, where $t \leftarrow \mathsf{G}(E')$.

- **Case 1:** If $t = t^\star$ but $E' \neq E$: $\mathcal{B}$ has found a collision in G (contradicting G's security) and aborts.
- **Case 2:** If $t \neq t^\star$, $\mathcal{B}$ can compute the decapsulation K as $\mathsf{H}\big(\big(\frac{c}{E'^w}\big)^{(t-t^\star)^{-1}}\big)$ and return K to $\mathcal{A}$.

At some point $\mathcal{A}$ will output a guess b' whether $Z = \mathsf{H}(g^{xy})$ or random and $\mathcal{B}$ will output the same bit. □

5 Conclusion

We have seen that split KEMs could be a way to capture DH-style key exchange flows with post-quantum security through enabling both parties to contribute to the KEM encapsulation. The starting point for this discussion and the need for a split-KEM–like notion stemmed from the fact that key exchange protocols based on DH must eventually be transitioned to post-quantum secure alternatives, which are given in the form of KEMs. For "simple" protocols, that only combine two ephemeral DH key pairs at a time, this should not pose too much of an issue. For specialized usages, such as 0-RTT modes based on DH or intricate patterns with many different DH combinations as in the initial key agreement of Signal, involving static keys, we have seen that standard KEMs are often inadequate.

We thus introduced the notion of split key encapsulation mechanisms. However, a major challenge remains, when it comes to showing that known KEMs fulfill the split KEM notion with reuse of keys on both sides: while, e.g., many passively-secure lattice-based KEMs are a prime example of the structure of split KEMs (since their encapsulation can be divided up into key generation and key

agreement on the encapsulator's side), we know that these are not secure when keys are reused. A promising candidate are constructions that replace the DH operations by CSIDH [12]. However, to achieve full-fledged key exchange security, interactive hardness assumptions such as the gap (hashed) DH assumption, strong DH assumption, or PRF-ODH are needed. We see it as an open problem to define these assumptions in the CSIDH setting and establish their intractability. Only answers to these questions can truly establish CSIDH as a viable building block for key exchange protocols.

Finally, it remains an open question to develop post-quantum solutions that support static-static key exchange, or that can accommodate reversed message flows; in other words, it is an open question to develop strongly-secure post-quantum constructions that have the same flexibility as Diffie–Hellman-based primitives. We believe the notion of split KEMs to be an adequate starting point to this exploration.

Acknowledgements

We thank Håkon Jacobsen for helpful discussions in the early phase of this work.

Marc Fischlin and Christian Janson have been (partially) funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297. Felix Günther has been supported in part by Research Fellowship grant GU 1859/1-1 of the German Research Foundation (DFG) and National Science Foundation (NSF) grants CNS-1526801 and CNS-1717640. Douglas Stebila has been supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada Discovery grant RGPIN-2016-05146 and a NSERC Discovery Accelerator Supplement.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (Apr 2001)
2. Alwen, J., Coretti, S., Dodis, Y.: The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 129–158. Springer, Heidelberg (May 2019)
3. Azarderakhsh, R., Jao, D., Leonardi, C.: Post-quantum static-static key agreement using multiple protocol instances. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 45–63. Springer, Heidelberg (Aug 2017)
4. Basso, A., Kutas, P., Merz, S.P., Petit, C., Weitkämper, C.: On adaptive attacks against Jao-Urbanik's isogeny-based protocol. In: Nitaj, A., Youssef, A. (eds.) Progress in Cryptology - AFRICACRYPT 2020. pp. 195–213. Springer International Publishing, Cham (2020)
5. Bauer, A., Gilbert, H., Renault, G., Rossi, M.: Assessment of the key-reuse resilience of NewHope. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 272–292. Springer, Heidelberg (Mar 2019)

6. Bergsma, F., Dowling, B., Kohlar, F., Schwenk, J., Stebila, D.: Multi-ciphersuite security of the Secure Shell (SSH) protocol. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 369–381. ACM Press (Nov 2014)
7. Boneh, D., Glass, D., Krashen, D., Lauter, K., Sharif, S., Silverberg, A., Tibouchi, M., Zhandry, M.: Multiparty non-interactive key exchange and more from isogenies on elliptic curves. Journal of Mathematical Cryptology 14(1), 5 – 14 (2020), https://www.degruyter.com/view/journals/jmc/14/1/article-p5.xml
8. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 493–522. Springer, Heidelberg (May 2020)
9. Bos, J.W., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1006–1018. ACM Press (Oct 2016)
10. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: 2015 IEEE Symposium on Security and Privacy. pp. 553–570. IEEE Computer Society Press (May 2015)
11. Brendel, J., Fischlin, M., Günther, F., Janson, C.: PRF-ODH: Relations, instantiations, and impossibility results. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 651–681. Springer, Heidelberg (Aug 2017)
12. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Heidelberg (Dec 2018)
13. Castryck, W., Sotáková, J., Vercauteren, F.: Breaking the decisional Diffie-Hellman problem for class group actions using genus theory. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 92–120. Springer, Heidelberg (Aug 2020)
14. Cohn-Gordon, K., Cremers, C.J.F., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the Signal messaging protocol. In: IEEE European Symposium on Security and Privacy, EuroS&P 2017. pp. 451–466 (2017)
15. Cohn-Gordon, K., Cremers, C.J.F., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of key establishment in the Signal messaging protocol. Journal of Cryptology (2020)
16. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. 33(1), 167–226 (Jan 2004)
17. Cremers, C.J.F., Feltz, M.: Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 734–751. Springer, Heidelberg (Sep 2012)
18. Crockett, E., Paquin, C., Stebila, D.: Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. In: NIST 2nd Post-Quantum Cryptography Standardization Conference 2019 (August 2019)
19. David Jao, R.A., Campagna, M., Costello, C., Feo, L.D., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Pereira, G., Renes, J., Soukharev, V., Urbanik, D.: Supersingular isogeny key encapsulation (April 2019), https://sike.org/
20. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard) (Aug 2008), https://www.rfc-editor.org/rfc/rfc5246.txt

21. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
22. Ding, J., Alsayigh, S., RV, S., Fluhrer, S., Lin, X.: Leakage of signal function with reused keys in rlwe key exchange. In: 2017 IEEE International Conference on Communications (ICC). pp. 1–6 (2017)
23. Ding, J., Branco, P., Schmitt, K.: Key exchange and authenticated key exchange with reusable keys based on RLWE assumption. Cryptology ePrint Archive, Report 2019/665 (2019), https://eprint.iacr.org/2019/665
24. Ding, J., Cheng, C., Qin, Y.: A simple key reuse attack on LWE and ring LWE encryption schemes as key encapsulation mechanisms (KEMs). Cryptology ePrint Archive, Report 2019/271 (2019), https://eprint.iacr.org/2019/271
25. Ding, J., Deaton, J., Schmidt, K., Vishakha, Zhang, Z.: A simple and practical key reuse attack on NTRU cryptosystem. Cryptology ePrint Archive, Report 2019/1022 (2019), https://eprint.iacr.org/2019/1022
26. Ding, J., Fluhrer, S.R., RV, S.: Complete attack on RLWE key exchange with reused keys, without signal leakage. In: ACISP 2018. LNCS, vol. 10946, pp. 467–486. Springer, Heidelberg (2018)
27. Ding, J., Xie, X., Lin, X.: A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688 (2012), http://eprint.iacr.org/2012/688
28. Dobson, S., Galbraith, S.D., LeGrow, J., Ti, Y.B., Zobernig, L.: An adaptive attack on 2-SIDH. Cryptology ePrint Archive, Report 2019/890 (2019), https://eprint.iacr.org/2019/890
29. Dobson, S., Li, T., Zobernig, L.: A note on a static SIDH protocol. Cryptology ePrint Archive, Report 2019/1244 (2019), https://eprint.iacr.org/2019/1244
30. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 1197–1210. ACM Press (Oct 2015)
31. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol. Journal of Cryptology (2020)
32. Drucker, N., Gueron, S.: Continuous key agreement with reduced bandwidth. In: Dolev, S., Hendler, D., Lodha, S., Yung, M. (eds.) Cyber Security Cryptography and Machine Learning. pp. 33–46. Springer International Publishing, Cham (2019)
33. Duits, I.: The Post-Quantum Signal Protocol: Secure Chat in a Quantum World. Master's thesis, University of Twente (February 2019), http://essay.utwente.nl/77239/
34. Fischlin, M., Günther, F.: Multi-stage key exchange and the case of Google's QUIC protocol. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 1193–1204. ACM Press (Nov 2014)
35. Fluhrer, S.: Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016), http://eprint.iacr.org/2016/085
36. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999)
37. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology 26(1), 80–101 (Jan 2013)
38. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 63–91. Springer, Heidelberg (Dec 2016)

39. Gao, X., Ding, J., Li, L., Liu, J.: Practical randomized RLWE-based key exchange against signal leakage attack. IEEE Transactions on Computers 67(11), 1584–1593 (Nov 2018)
40. Greuet, A., Montoya, S., Renault, G.: Attack on LAC key exchange in misuse situation. Cryptology ePrint Archive, Report 2020/063 (2020), https://eprint.iacr.org/2020/063
41. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Heidelberg (Nov 2017)
42. Hülsing, A., Ning, K.C., Schwabe, P., Weber, F., Zimmermann, P.R.: Post-quantum wireguard. Cryptology ePrint Archive, Report 2020/379 (2020), https://eprint.iacr.org/2020/379
43. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg (Aug 2012)
44. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.Y. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011. pp. 19–34. Springer, Heidelberg (Nov / Dec 2011)
45. Katsumata, S., Kwiatkowski, K., Pintore, F., Prest, T.: Scalable ciphertext compression techniques for post-quantum KEMs and their applications. In: ASIACRYPT 2020 (2020), to appear. Available as Cryptology ePrint Archive, Report 2020/1107. https://eprint.iacr.org/2020/1107
46. Kawashima, T., Takashima, K., Aikawa, Y., Takagi, T.: An efficient authenticated key exchange from random self-reducibility on CSIDH. Cryptology ePrint Archive, Report 2020/1178 (2020), https://eprint.iacr.org/2020/1178
47. Kayacan, S.: A note on the static-static key agreement protocol from supersingular isogenies. Cryptology ePrint Archive, Report 2019/815 (2019), https://eprint.iacr.org/2019/815
48. Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (Apr 2007)
49. de Kock, B., Gjøsteen, K., Veroni, M.: Practical isogeny-based key-exchange with optimal tightness. In: SAC 2020 (2020), to appear. Available as Cryptology ePrint Archive, Report 2020/1165. https://eprint.iacr.org/2020/1165
50. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (Aug 2005)
51. Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: A systematic analysis. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 429–448. Springer, Heidelberg (Aug 2013)
52. Krawczyk, H., Wee, H.: The OPTLS protocol and TLS 1.3. In: 2016 IEEE European Symposium on Security and Privacy, EuroS&P 2016. pp. 81–96. IEEE, Saarbrücken, Germany (Mar 21–24, 2016)
53. Kwiatkowski, K., Valenta, L.: The TLS Post-Quantum Experiment. The Cloudflare Blog, https://blog.cloudflare.com/the-tls-post-quantum-experiment/ (October 2019)
54. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (Nov 2007)

55. Langley, A.: CECPQ1 results. Imperial Violet, Blog, https://www.imperialviolet.org/2016/11/28/cecpq1.html (Nov 2016)
56. Langley, A.: CECPQ2. Imperial Violet, Blog, https://www.imperialviolet.org/2018/12/12/cecpq2.html (Dec 2018)
57. Langley, A., Chang, W.T.: QUIC Crypto. https://docs.google.com/document/d/1g5nIXAIkN_Y-7XJW5K45IblHd_L2f5LTaDUDwvZ5L6g/ (Dec 2016), revision 20161206
58. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (Feb 2011)
59. Liu, C., Zheng, Z., Zou, G.: Key reuse attack on NewHope key exchange protocol. In: Lee, K. (ed.) ICISC 18. LNCS, vol. 11396, pp. 163–176. Springer, Heidelberg (Nov 2019)
60. Marlinspike, M., Perrin, T.: The X3DH key agreement protocol (Nov 2016), https://signal.org/docs/specifications/x3dh/
61. National Institute of Standards and Technology (NIST): Post-quantum cryptography. https://csrc.nist.gov/projects/post-quantum-cryptography (Aug 19, 2015)
62. Okada, S., Wang, Y., Takagi, T.: Improving key mismatch attack on NewHope with fewer queries. In: Liu, J.K., Cui, H. (eds.) ACISP 20. LNCS, vol. 12248, pp. 505–524. Springer, Heidelberg (Nov / Dec 2020)
63. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (Feb 2001)
64. Peikert, C.: Lattice cryptography for the internet. In: Mosca, M. (ed.) Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014. pp. 197–219. Springer, Heidelberg (Oct 2014)
65. Peikert, C.: He gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 463–492. Springer, Heidelberg (May 2020)
66. Perrin, T.: The Noise protocol framework, https://noiseprotocol.org/
67. Qin, Y., Cheng, C., Ding, J.: A complete and optimized key mismatch attack on NIST candidate NewHope. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019, Part II. LNCS, vol. 11736, pp. 504–520. Springer, Heidelberg (Sep 2019)
68. QUIC, a multiplexed stream transport over UDP. https://www.chromium.org/quic
69. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Proposed Standard) (Aug 2018), https://www.rfc-editor.org/rfc/rfc8446.txt
70. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum tls without handshake signatures. Cryptology ePrint Archive, Report 2020/534 (2020), https://eprint.iacr.org/2020/534
71. Shoup, V.: A Proposal for an ISO Standard for Public Key Encryption (version 2.1) (Dec 2001), https://www.shoup.net/papers/iso-2_1.pdf
72. Signal protocol: Technical documentation. https://whispersystems.org/docs/
73. Stebila, D., Mosca, M.: Post-quantum key exchange for the internet and the open quantum safe project. In: Avanzi, R., Heys, H.M. (eds.) SAC 2016. LNCS, vol. 10532, pp. 14–37. Springer, Heidelberg (Aug 2016)

74. Urbanik, D., Jao, D.: New techniques for SIDH-based NIKE. Journal of Mathematical Cryptology 14(1), 120 – 128 (2020), https://www.degruyter.com/view/journals/jmc/14/1/article-p120.xml
75. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Designs, Codes and Cryptography 46(3), 329–342 (Mar 2008)
76. Xue, H., Lu, X., Li, B., Liang, B., He, J.: Understanding and constructing AKE via double-key key encapsulation mechanism. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 158–189. Springer, Heidelberg (Dec 2018)
77. Yao, A.C.C., Zhao, Y.: OAKE: a new family of implicitly authenticated Diffie-Hellman protocols. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 1113–1128. ACM Press (Nov 2013)
78. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Transport Layer Protocol". RFC 4253 (Proposed Standard) (Jan 2005), https://www.rfc-editor.org/rfc/rfc4253.txt