Towards Generalized Diffie-Hellman-esque Key Agreement via Generic Split KEM Construction

Brian Goncalves[a] and Atefeh Mashatan[b]
Cybersecurity Research Lab, Toronto Metropolitan University, Victoria Street, Toronto, Canada

Keywords: Public-Key Cryptography, Provable Security, Key Agreement, Signal Protocol, Key Encapsulation.

Abstract: The Diffie-Hellman (DH) problem is a cornerstone of countless key agreement schemes. One of these schemes is the popular instant messaging protocol, Signal. The Signal protocol relies on a subprotocol based on the DH-problem in order to create a secure session key. Unfortunately, as the threat of robust quantum computers continues to loom over traditionally hard problems such as the DH problem, quantum-resistant replacements for these schemes must be created. One candidate for a drop-in DH-style replacement is a special type of key encapsulation mechanism (KEM) called a split KEM, which maintains the same message flow of DH key agreement schemes. In this work, we present an efficient combiner to construct a split from a public key encryption scheme, a signature algorithm, and a special type of pseudorandom function (PRF), called a constrained PRF. Constrained PRFs can produce PRF keys with limited domains, and by selecting the domain to be a single point, the master secret key can be reused. We then use the remaining schemes to transport the constrained key and point and ensure the authenticity of the source of the ciphertext. We then prove that our construction reaches the split KEM formulation of traditional IND-CCA-security with a tight reduction.

1 INTRODUCTION

The Diffie-Hellman (DH) problem (Diffie and Hellman, 2006) has been a foundational tool that has been widely used in key agreement schemes (Krawczyk, 2005; Lauter and Mityagin, 2006; Boyd et al., 2008; Okamoto, 2007; Dierks and Rescorla, 2008; Rescorla, 2018). Tragically, despite the versatility and elegance of the DH problem, the underlying hardness is known to be weak against the period-finding algorithm for quantum computers developed by Peter Shor (Shor, 1994). As such, countless protocols that rely on DH-related assumptions require quantum-resistant replacements.

Among such protocols that must be replaced is the asynchronous messaging protocol Signal (sig,), which is used notably in the eponymous messaging app, as well as WhatsApp (Inc., 2016). The Signal protocol relies on its Extended Triple Diffie-Hellman (X3DH) key agreement step (Marlinspike and Perrin, 2016). The X3DH protocol allows Alice to upload her long-term as well as semistatic (and optional ephemeral) DH public keys to a server so that Bob may download them and asynchronously compute a session key before sending Alice the necessary ephemeral DH public key to compute the same session key herself once she is back online.

In (Brendel et al., 2020), Brendel *et al.* attempted to find a suitable post-quantum replacement for the X3DH protocol. As part of their efforts, they analyzed a theoretical key encapsulation mechanism (KEM) based replacement for the X3DH protocol, where Alice and Bob exchange KEM ciphertexts instead of DH keys. However, this construction was unable to accurately replicate the X3DH protocol for two reasons.

First, their KEM-based construction could not generate keying material that was analogous to the keying material generated from Alice's long-term DH key and Bob's semistatic DH key that is present in the standard X3DH protocol. In the KEM-based construction, the various keys produced from encapsulating can be thought of as being produced from Bob's public keys (long-term, semi-static, and ephemeral) and Alice's ephemeral key (in the form of the random coins used during encapsulation). Thus, it cannot fully replicate the X3DH key agreement protocol. Secondly, and more problematically, in order to be able to properly replicate the keys made in the protocol, a new message flow from Bob to Alice must be

[a] https://orcid.org/0000-0002-8850-2173
[b] https://orcid.org/0000-0001-9448-9123

Goncalves, B. and Mashatan, A.
Towards Generalized Diffie-Hellman-esque Key Agreement via Generic Split KEM Construction.
DOI: 10.5220/0012454400003648

In *Proceedings of the 10th International Conference on Information Systems Security and Privacy (ICISSP 2024)*, pages 594-608
ISBN: 978-989-758-683-5; ISSN: 2184-4356

introduced, which undermines the asynchronicity of the Signal protocol.

To address these issues, Brendel, Fiedler, Günther, Janson, and Stebila (Brendel et al., 2020) presented a new type of KEM that mirrors the message flows of DH key agreement, which they call a *split KEM*, and detailed how it can be used for DH-based key exchange, such as X3DH. The idea of a split KEM is that encapsulation (resp. decapsulation) is computed with the encapsulator's private key (resp. public key) and the decapsulator's public key (resp. private key). Furthermore, Brendel *et al.* adapted the standard KEM definition of security and defined the security notions of lr-IND-CCA, where l and r denote how many queries an adversary can make to an encapsulation and decapsulation oracle, respectively. This use of oracles for encapsulation is due to the requirement of a private key during encapsulation.

Brendel *et al.* concluded their work by presenting a pair of split KEMs. The first construction presented was built from the Ring Learning With Errors (RLWE) problem and was proven to obtain the lowest level of security defined in their work. Their second construction for a split KEM was built from the *classically* hard Gap Hashed Diffie-Hellman (Gap DH) problem introduced by Kiltz as a basis for a traditional KEM (Kiltz, 2007). Unlike their first construction, Brendel *et al.* were able to prove that their second split KEM obtains fully adaptive, IND-CCA-security. Despite being unable to construct an adaptively secure split KEM from problems that are thought to be hard for quantum computers to efficiently solve, Brendel *et al.* speculated that such a construction may come from the (quantum hard) commutative supersingular Isogeny Diffie-Hellman (CSIDH) setting (Castryck et al., 2018), but note that the intractability of commutative SIDH and similar interactive problems is unknown.

1.1 Our Contributions

In this paper, we take an alternative approach to the construction of split KEMs, that of black-box combiners. As such, we present a construction for a split KEM built from a public key encryption (PKE) scheme, a digital signature algorithm (DS), a special class of pseudorandom functions (PRF), called a *constrained* PRF (cPRF), and a key derivation function (KDF). We then prove the IND-CCA-security of our construction with a tight reduction. Moreover, we assume that the adversary in the main security proof is quantum-computing capable, so that our construction can be instantiated with quantum-resistant primitives in order to translate Diffie-Hellman-esque key agreement to the post-quantum setting by way of split KEMs. Additionally, a benefit of our approach to constructing split KEMs is that it will remain secure should any specific choice for the inputs be discovered to be insecure.

Constrained PRFs[1] were introduced independently by Boneh and Waters (Boneh and Waters, 2013), Kiayias, Papadopoulos, Triandopoulos, and Zacharias (Kiayias et al., 2013), and Boyle, Goldwasser, and Ivan (Boyle et al., 2014), cPRFs are PRFs that allow for a special "constrained" key to be computed from the master secret key that allows (constrained) PRF evaluations on some restricted subset of the domain. Since their introduction, numerous works have been published (Hofheinz et al., 2019; Brakerski and Vaikuntanathan, 2015; Attrapadung et al., 2018; Boneh et al., 2017) showing how to construct cPRFS, including post-quantum assumptions such as the Learning With Errors (LWE) problem (Peikert and Shiehian, 2018; Canetti and Chen, 2017). In our construction, we make use of this additional functionality so that the encapsulator is able to compute constrained keys over singleton sets of uniformly random points in the domain of the cPRF and give both the constrained key and the point to the decapsulator in order to produce the final key.

In more detail, our construction relies on the encapsulator being able to use a constrained PRF in order to generate KDF keying material from their own secret key and send it to the decapsulator via public-key encryption. The final key is then computed with the KDF evaluating the ciphertext and its contents. In doing this, we solve the issue of the KEM-based replacement in computed,(Brendel et al., 2020), Alice's long-term secret keys contributing to producing keying material when paired with any of Bob's keys. Thus, the full X3DH protocol can now be recreated using our split KEM construction.

1.2 Related Work

As previously discussed, Brendel *et al.* constructed two split KEMs directly from two well-known hard problems, RLWE and Gap DH (Brendel et al., 2020). However, neither construction was able to fully realize both the goals of full IND-CCA-security and quantum-resistance. The RLWE construction was limited to the weakest security level defined for split KEMs. Meanwhile, the Gap DH construction failed to address the issue of quantum vulnerability of the traditional DH scheme that necessitates its eventual

[1] Kiayias *et al.* named these PRFs "Delegated PRFs", while Boyle, Goldwasser and Ivan used the term "Functional PRFs"

abandonment.

In a subsequent work, Brendel, Fiedler, Günther, Janson, and Stebila present a construction for a *deniable* asynchronous key exchange (DAKE), called "Signal in a Post-Quantum Regime" or SPQR, that matches the characteristics of the Signal protocol including the correct combinations of static, semistatic, and ephemeral keys (Brendel et al., 2021). Their key exchange protocol is built from a designated verifier signature (DVS) scheme, twisted PRF, and a KEM used multiple times to generate the required keying material. The use of the DVS is to ensure that only Bob can verify the signature that Alice generates on the transcript. While our construction shares similarities with this protocol, we note that one limitation in directly replacing X3DH with SPQR is that, in practice, Signal has the semistatic keys signed under the user's long-term key, which is not possible with the use of a DVS. In order to address this, the authors note that additional long-term signing may be needed. Additionally, DVS schemes are more expensive than traditional signature schemes such as the one used in our construction. Finally, as mentioned in (Hashimoto et al., 2022), there is an attack that breaks the deniability of the base DAKE used to build SPQR, despite Brendel *et al.* proving a *different* version of deniability.

Hashimoto, Katsumata, Kwiatkowski, and Prest (Hashimoto et al., 2022) approached the problem of finding a X3DH replacement by constructing a new 1-round authenticated key exchange (AKE) protocol where the first message is "receiver oblivious", that is it can be generated independently by Alice. They call 1-round AKEs with this property *Signal Conforming* AKEs (SC-AKEs). They were able to successfully construct a secure Signal Conforming AKE built from a digital signature, a PRF, and a pair of standard KEMs (one IND-CCA and the other IND-CPA) whose public keys and ciphertexts contain sufficient entropy. However, their construction does not immediately address the first issue of Bindel *et al.*'s KEM-based construction, as there is no immediate analog to the key material corresponding to the semistatic key of Bob and the long-term key of Alice. As such, we believe that split KEMs act as a more direct replacement for the DH key agreement in X3DH than the SC-AKE they present.

Dobson and Galbraith proposed a SIDH-based key exchange replacement for X3DH, called SI-X3DH (Dobson and Galbraith, 2022). The core idea of their construction was that while standard SIDH key exchange are not secure against adaptive attacks, by using a Proof of Knowledge to verify the honesty of long-term SIDH public keys that adaptive security can be obtained. However, since then, there has been an efficient key recovery attack, so this construction is no longer secure to use as a potential replacement for DH key agreement (Castryck and Decru, 2023).

1.3 Overview

We organize this work as follows. In Section 2, we present the notation used throughout the remainder of the paper, along with the necessary definitions for the assumptions of our main result. In Section 3, we present our split KEM combiner and proof of security. We then conclude this paper in Section 4, where we discuss future directions and possibilities of constructing further split KEMs.

2 PRELIMINARIES

In this section, we cover the preliminaries used in this work. We begin with the notation used, followed by an introduction to PKEs, DSs, KDFs, cPRFS, and split KEMs, including their definitions and relevant security notions.

2.1 Notation

By $y \leftarrow \mathcal{A}(x)$ we denote an algorithm $\mathcal{A}$ (either classical or quantum), which runs on input (classical) x and output (classical) y. When $\mathcal{A}$ has access to an oracle, $\mathcal{B}$, we write this as $\mathcal{A}(x)^{\mathcal{B}(\cdot)}$. If $\mathcal{A}$ is an algorithm that uses some randomness in its execution on input x and we wish to specify what the randomness is, say r, we denote it as $\mathcal{A}(x;r)$. We refer to specific subroutines within $\mathcal{A}(\cdot)$ as $\mathcal{A}.\mathsf{Subroutine}$. We consider all adversaries as algorithms (either classical or quantum) that are probabilistic polynomial-time (PPT) on their input length. We adopt the convention that PPT will be used in the case where the adversary is a classical algorithm, and QPT for when they are a quantum PPT algorithm.

We write $x \leftarrow\!\!\$\, S$ to denote that x was outputted by S probabilistically, where if S is some algorithm, then x was selected according to some internal distribution, and if S is some space, such as $\{0,1\}^l$, then we implicitly mean that x is sampled uniformly at random.

We say that a function g mapping nonnegative integers to nonnegative reals is called *negligible*, if for all positive numbers c, there exists an integer $\lambda_0(c) \geq 0$ such that for all $\lambda > \lambda_0(c)$ we have $g(\lambda) < \frac{1}{\lambda^c}$.

2.2 Public-Key Encryption

We continue this section by reviewing two of the most basic public-key cryptosystems, PKEs and DSs. Beginning with PKEs, we present formal definitions of the cryptosystem and the necessary security definitions.

Definition 2.1 (**Public-Key Encryption**)**.** We say a triple of algorithms $\mathcal{K} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ form a Public-Key Encryption (PKE), if:

- $\mathsf{KeyGen}(1^n)$: The key generation algorithm is a probabilistic algorithm which, on input 1^n ($n \in \mathbb{N}$), outputs a related pair (ek, dk), of public encryption and secret decryption keys.
- $\mathsf{Enc}(ek, m)$: The encryption algorithm is a probabilistic algorithm that takes as input a public key ek, and a plaintext message m and produces a ciphertext c.
- $\mathsf{Dec}(dk, c)$: The decryption algorithm is a deterministic algorithm that takes as input a secret key dk, and ciphertext c, and returns a plaintext message m, or a special designated rejection symbol $\perp$.

Definition 2.2 (**Correctness of PKEs**)**.** We say that a PKE, Π, is ε-correct, if $\forall n \in \mathbb{N}$:

$$\Pr\left[\mathsf{Dec}(dk, c) \neq m : \begin{smallmatrix}(ek,dk)\leftarrow\mathsf{KeyGen}(1^n),\\ c\leftarrow\mathsf{Enc}(ek,m)\end{smallmatrix}\right] \leq \varepsilon. \quad (1)$$

We say that a PKE, Π, is perfectly correct if $\varepsilon = 0$.

We now formally define what we mean by IND-CCA-security for PKEs.

Definition 2.3 (IND-CCA **Security for PKEs**)**.** We say that a PKE, Π, is IND-CCA if, for all adversaries $\mathcal{A}$, we have that:

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\Pi,\mathcal{A}}(n) = \left|\Pr\left[\mathsf{Expt}^{\mathsf{IND\text{-}CCA}}_{\Pi,\mathcal{A}}(n) - \frac{1}{2}\right]\right| \quad (2)$$

is a negligible function in $n \in \mathbb{N}$, where $\mathsf{Expt}^{\mathsf{IND\text{-}CCA}}_{\Pi,\mathcal{A}}(n)$ is defined in Figure 1.

$\mathsf{Expt}^{\mathsf{IND\text{-}CCA}}_{\Pi,\mathcal{A}}(n)$:

1. $(ek, dk) \leftarrow\!\!\$\ \Pi.\mathsf{KeyGen}(1^n)$
2. $m_0, m_1, \mathsf{st} \leftarrow \mathcal{A}^{\Pi.\mathsf{Dec}(dk,\cdot)}(ek)$
3. $b \leftarrow\!\!\$\ \{0,1\}$
4. $c^* \leftarrow \Pi.\mathsf{Enc}(ek, m_b)$
5. $b' \leftarrow \mathcal{A}^{O_{\mathcal{K}.\mathsf{Dec}(dk,\cdot\neq c^*)}}(ek, \mathsf{st}, c^*)$
6. return $[b = b']$

Figure 1: The IND-CCA security experiments for PKEs.

An additional property that we need PKEs to have later in this work is that they are *well-spread*, which we define below.

Definition 2.4 (γ-spread PKE (Fujisaki and Okamoto, 2013))**.** Let Π=($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$) be a PKE scheme. We say Π is γ-spread if for every ek generated by $\mathsf{KeyGen}(1^n)$ and every message m in the message space of Enc it holds that

$$\max_{c\in\{0,1\}^*} \Pr[c = \mathsf{Enc}(ek, m; r) : r \leftarrow\!\!\$\ \mathsf{Coin}] \leq 2^{-\gamma}, \quad (3)$$

where Coin denotes the space in which the randomness of Enc is sampled.

We say that Π is well-spread if $\gamma = \omega(\log(n)$.

2.3 Digital Signature Algorithms

Next, we recall the formal definitions of digital signature algorithms and what it means for a signature to have strong existential unforgeability under chosen-message attacks.

Definition 2.5 (**Digital Signature Algorithms**)**.** We say a triple of algorithms Σ=(KeyGen, Sign, Vfy) form a digital signature algorithm (DS) scheme, if:

- $\mathsf{KeyGen}(1^n)$: The key generation algorithm is a probabilistic algorithm which on input 1^n ($n \in \mathbb{N}$) outputs a related pair, (vk, sk), of public verification and secret signing keys;
- $\mathsf{Sign}(sk, m)$: The signing algorithm is a probabilistic algorithm that takes two inputs, a secret signing key sk and a plaintext message m, from a designated message space $\mathcal{M}_\Sigma$, and outputs a signature σ;
- $\mathsf{Vfy}(vk, m, \sigma)$: The verification algorithm is a deterministic algorithm that takes as input a public verification key vk, a plaintext message m and a signature σ, and returns either a 0 if rejected, or a 1 if accepted.

Definition 2.6 (**Correctness of DSs**)**.** We say that a digital signature scheme, Σ, is ε-correct, if for all $m \in \mathcal{M}_\Sigma$, we have:

$$\Pr\left[\mathsf{Vfy}(vk, m, \sigma) \neq 1 : \begin{smallmatrix}(vk,sk)\leftarrow\mathsf{KeyGen}(1^n),\\ \sigma\leftarrow\mathsf{Sign}(sk,m)\end{smallmatrix}\right] \leq \varepsilon. \quad (4)$$

We say that a DS, Σ, is perfectly correct if $\varepsilon = 0$.

When working with signature algorithms, the most common notion of security is that of existential unforgeability under chosen-message attacks, denoted as EUF-CMA. The EUF-CMA-security property means that an adversary cannot produce a valid signature on a message not previously signed. However, EUF-CMA-security does not protect against *new*

signatures on *old* messages, which in the case of our split KEM may result in adversaries replaying the same PKE ciphertext with different signatures and win the security game with probability 1. Thus, this property is insufficient for our split KEM combiner.

To prevent this type of attack, we require our DS to have the property of strong existential unforgeability under chosen message attacks, denoted as SUF-CMA-security. Informally, this property means that it should be infeasible for an adversary to produce a valid message-signature *pair* that they have not previously seen, including the case of repeat messages, which is what our split KEM construction requires. Although SUF-CMA-security is a stronger property than is typically considered for digital signatures, there exist several works that provide methods on how to build them from weaker signature schemes (Huang et al., 2007; Steinfeld et al., 2006; Huang et al., 2008; Wang and Tanaka, 2015). We formally define SUF-CMA-security below.

Definition 2.7 (SUF-CMA **Security for DSs).** We say that a DS, Σ, is SUF-CMA-secure if, for all adversaries $\mathcal{A}$, we have that:

$$\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{A}}(n) = \Pr\left[\mathsf{Expt}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{A}}(n) \to 1\right], \quad (5)$$

is a negligible function in n, where $\mathrm{Expt}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{A}}(n)$ is defined in Figure 2.

$\mathrm{Expt}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{A}}(n)$:

1. $(vk, sk) \leftarrow\!\$\, \Sigma.\mathsf{KeyGen}(1^n), \mathcal{L} = \emptyset$
2. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}^*(sk,\cdot)}}(vk)$
3. Return $[(m^*, \sigma^*) \notin \mathcal{L}] \wedge [\mathsf{Vfy}(vk, m^*, \sigma^*) = 1]$
4. Else 0

$O_{\mathsf{Sign}^*(sk,m)}$:

1. $\sigma \leftarrow \mathsf{Sign}^*(sk, m), \mathcal{L} \cup (m, \sigma)$
2. return (m, σ)

Figure 2: The SUF-CMA security experiments for DSs.

2.4 Key Derivation Function

In this subsection, we review key derivation functions (KDF). Intuitively, KDFs allow users to compute a cryptographic key after input is provided and are used in countless protocols and schemes as the final step in key agreement. The primary security requirement of KDFs is that their output be indistinguishable from a uniformly random string of equal length. We formalize the definition of KDFs and their security below.

Definition 2.8. KDF (Chuah et al., 2013) A key derivation function is defined as a function $\mathsf{KDF} : \Xi \times \mathsf{Salt} \times \mathsf{Context} \times \mathbb{N} \to \{0,1\}^l$, where

- ξ is a source keying material, which is chosen from the space of all possible source keying materials Ξ. With the outputted ξ is some form (such as its distribution) of the auxiliary data a, which is publicly known.
- n is a positive integer that indicates the number of bits produced by KDF.
- s is a salt, which is a public random string chosen from the salt space Salt. This input is optional.
- t is a context string chosen from a context space Context.
- $\mathcal{K}$ is the derived l-bit cryptographic key.

Definition 2.9 (KDF Security (Chuah et al., 2013))**.** Let n be the security parameter. Let $\Xi \times \mathsf{Salt} \times \mathsf{Context} \times \mathbb{N} \to \{0,1\}^l$ be a key-derivation function with input source keying material $\xi \leftarrow \Xi$, $l \in \mathbb{N}$ the output length, $s \in \mathsf{Salt}$, and $t \in \mathsf{Context}$. We say that KDF is KDF-IND-secure if, for all adversaries $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{KDF\text{-}IND}}_{\mathsf{KDF},\mathcal{A}}(n) = \left|\Pr\left[\mathsf{Expt}^{\mathsf{KDF\text{-}IND}}_{\mathsf{KDF},\mathcal{A}}(n)\right] - \frac{1}{2}\right| \quad (6)$$

is negligible as a function of n, where $\mathrm{Expt}^{\mathsf{KDF\text{-}IND}}_{\mathsf{KDF}}(\mathcal{A})$ is defined below in Figure 3.

$\mathrm{Expt}^{\mathsf{KDF\text{-}IND}}_{\mathsf{KDF},\mathcal{A}}(n)$:

1. $(\xi, a) \leftarrow\!\$\, \Pi.\mathsf{KeyGen}(1^n), L \leftarrow \emptyset, s \leftarrow\!\$\, \mathsf{Salt}$
2. $t^*, l^*, \mathsf{st} \leftarrow \mathcal{A}^{O_{\mathsf{KDF}(\xi,s,\cdot,\cdot)}}(a, s)$
3. $k_0 \leftarrow \mathsf{KDF}(\xi, s, t^*, l^*), k_1 \leftarrow\!\$\, \{0,1\}^{l^*}$
4. $b \leftarrow\!\$\, \{0,1\}$
5. $b' \leftarrow \mathcal{A}^{O_{\mathsf{KDF}(\xi,s,\cdot,\cdot)}}(a, s, k_b, \mathsf{st})$
6. return $[b = b'] \wedge [(t^*, l^*) \notin L]$

$\mathsf{KDF}(\xi, s, t, l)$:

1. $L \cup (t, l)$
2. return $\mathsf{KDF}(\xi, s, t, l)$

Figure 3: The IND-CCA security experiments for PKEs.

2.5 Constrained PRFs

We next review the formal definitions associated with cPRFs. Constrained PRFs are briefly a type of PRF that allows third parties to use special limited keys to evaluate the PRF as if they had the master secret key. These limited keys are produced by the holder of the master secret key through a constraining algorithm. To do this, a circuit representation of the set, along the master secret key, are used as input for the constraining algorithm. We will use the convention that a circuit f is satisfied by, or authorizes, the input x if $f(x) = 1$.

Definition 2.10. A *constrained PRF* $\mathcal{P}$ is a tuple of algorithms (Setup, Eval, Con, cEval) over domain X, range Y, and some circuit class C defined as follows (Boneh and Waters, 2013): having the following interfaces (where the domain may depend on the security parameter)

- $\mathsf{Setup}(1^n)$: given the security parameter n outputs a master secret key msk.
- $\mathsf{Eval}(msk, x)$: given the master secret key msk and an input $x \in X$, it outputs some $y \in Y$.
- $\mathsf{Con}(msk, f)$: given the master secret key and a circuit $f \in C$ outputs a constrained key τ_f.
- $\mathsf{cEval}(\tau_f, x)$, given a constrained key τ_f and an input $x \in X$, outputs some $y \in Y$.

A crucial concept for cPRFS is that of *correctness*. As previously described, the aim of cPRFs is to allow a third party to perform PRF evaluations as if they had the master secret key and to allow them and the keyholder to output the same value. This property is captured in the idea of correctness, which we formally define next.

Definition 2.11. A constrained PRF, $\mathcal{P}$ is correct if $\forall$ $n \in \mathbb{N}$, $msk \leftarrow \mathsf{Setup}(1^n)$, for every circuit $f \in C$ and input $x \in X$ for which $f(x) = 1$, the following holds with overwhelming probability,

$$\mathcal{P}.\mathsf{cEval}(\mathcal{P}.\mathsf{Con}(msk, f), x) = \mathcal{P}.\mathsf{Eval}(msk, x).$$

Now, we describe the main security concept for cPRFs. As the name implies, a cPRF should produce outputs that, like a standard PRF, are indistinguishable from random. However, because cPRFs have the additional functionality of producing constrained keys, these keys should not leak or provide any additional information to an adversary attempting to distinguish the outputs from random. In more detail, we want it to be infeasible for an adversary to distinguish a PRF evaluation at a point of their choice x^*, from a randomly chosen sample in the range, while also being allowed to obtain constrained keys for circuits of their choice. However, to prevent trivial wins, we do not allow the adversary to select an x^* that satisfies any of the circuits queried, nor allow queries to the Con oracle which can be satisfied by x^*.

Definition 2.12. A constrained PRF, $\mathcal{P}$, with domain X is said to be *pseudorandom* if for all adversary $\mathcal{A}$, and family of circuits C, we have that

$$\mathsf{Adv}^{\mathrm{rndm}}_{\mathcal{P},\mathcal{A}}(n) := \left| \Pr\left[\mathrm{Expt}^{\mathrm{rndm}}_{\mathcal{P},\mathcal{A}}(n) = 1\right] - \frac{1}{2} \right|, \quad (7)$$

is a negligible function in n, where $\mathrm{Expt}^{\mathrm{rndm}}_{\mathcal{P},\mathcal{A}}(n)$ is defined in Figure 4 (Boneh and Waters, 2013).

$\underline{\mathrm{Expt}^{\mathrm{rndm}}_{\mathcal{P},\mathcal{A}}(n)}$:

1. $msk \leftarrow\!\!\$\ \mathcal{P}.\mathsf{Setup}(1^n), L_x = \emptyset, L_f = \emptyset$
2. $x^*, \mathsf{st} \leftarrow \mathcal{A}^{O_{\mathcal{P}.\mathsf{Con}}(msk,\cdot), O_{\mathcal{P}.\mathsf{Eval}}(msk,\cdot)}(1^n)$
3. $y_0 \leftarrow \mathcal{P}.\mathsf{Eval}(msk, x^*)$
4. $y_1 \leftarrow\!\!\$\ Y$
5. $b \leftarrow\!\!\$\ \{0,1\}$
6. $b' \leftarrow \mathcal{A}^{O_{\mathcal{P}.\mathsf{Con}(msk,\cdot)}, O_{\mathcal{P}.\mathsf{Eval}(msk,\cdot)}}(y_b, \mathsf{st})$
7. return $[b = b'] \wedge [x^* \notin L_x] \wedge [\forall f \in L_f, f(x^*) = 0]$

$\underline{O_{\mathcal{P}.\mathsf{Con}}(msk, f)}$:

1. $L_f \cup \{f\}$
2. return $\mathcal{P}.\mathsf{Con}(msk, f)$

$\underline{O_{\mathcal{P}.\mathsf{Eval}}(msk, x)}$:

1. $L_x \cup \{x\}$
2. return $\mathcal{P}.\mathsf{Eval}(msk, x)$

Figure 4: The pseudorandomness experiment for constrained PRF, $\mathcal{P}$.

2.6 Split KEMs

Finally, we conclude this section by providing the definitions of split KEMs presented by Brendel *et al.* (Brendel et al., 2020).

We note here that Brendel *et al.* presented two formulations of split KEMs, *symmetric* and *asymmetric* split KEMs. In the symmetric setting, the key generation process is identical for both the encapsulator and the decapsulator. Traditional DH-based schemes are an archetypical example of such symmetric KEMs. Another example of such a symmetric split KEM is the LWE-based split KEM defined by Brendel *et al.* in (Brendel et al., 2020). Intuitively, then, asymmetric split KEMs have different key generation processes for the different roles. Once again, Brendel *et al.* presented an example of such a split KEM based on the work of Kiltz (Kiltz, 2007). A natural question that may be asked is whether it is possible to convert one type of split KEM into the other, to which the answer is yes. Brendel *et al.* provided a description of how this can be done. For the conversion of asymmetric split KEMs to symmetric split KEMs, key generation is defined to simply run *both* the encapsulator and decapsulator key generation algorithms and use the appropriate key depending on their role in a session. To see the reverse, the symmetric key generation algorithm is changed so that it is done with fixed roles for the encapsulators and decapsulators.

For the remainder of this work, when we say split KEM, we will refer to asymmetric split KEMs, which we define next.

Definition 2.13. A *split KEM* $\mathcal{K}_S$ consists of four algorithms DKeyGen, EKeyGen, sEncap, and sDecap,

where EKeyGen and sEncap are executed by the encapsulator, and DKeyGen and sDecap by the decapsulator (Brendel et al., 2020).

- **split KEM key generation:** for decapsulator and encapsulator, respectively: $(D,d) \leftarrow \mathsf{DKeyGen}(1^n)$ and $(E,e) \leftarrow \mathsf{EKeyGen}(1^n)$ are probabilistic algorithms that output a pair of keys, consisting of a public key (denoted with capital letters) and a secret key (denoted by lowercase letters, respectively).
- **split KEM encapsulation:** $(c,k) \leftarrow \mathsf{sEncap}(e,D)$ is a probabilistic algorithm executed by the encapsulator. It takes as input e, the secret key of the encapsulator, and D, the public key of the decapsulator. The algorithm sEncap then outputs the shared secret k along with its encapsulation c.
- **split KEM decapsulation:** $K/\perp \leftarrow \mathsf{sDecap}(d,E,c)$ is a deterministic algorithm executed by the decapsulator. On input a ciphertext c, the secret key of the decapsulator d, and the public key of the encapsulator E, it outputs the decapsulation k of c or $\perp$, if the operation fails.

Next, we state the security definition for split KEMs, lr-IND-CCA-security. Naturally, this definition is analogous to that of IND-CCA-security of traditional KEMs, where the adversary aims to distinguish whether a uniformly random key was given or not. The key difference is that in the IND-CCA-experiment for traditional KEMs, the adversary is given the public key so that they may produce ciphertexts themselves, as well as an oracle programmed with the secret key to decrypt ciphertexts. Split KEMs, however, require secret keys for *both* encapsulation and decapsulation. Consequently, in the lr-IND-CCA-security experiment, the adversary is given both public keys and two oracles, one for encapsulation and one for decapsulation.

Definition 2.14. Let $\mathcal{K}_\mathcal{S} = (\mathsf{EKeyGen}, \mathsf{DKeyGen}, \mathsf{sEncap}, \mathsf{sDecap})$ be a split KEM with key space K. Let $l \in \{n,s,m\}$ (n means no queries, s means a single, m means polynomially many) and $r \in \{n,m\}$. We say that $\mathcal{K}_\mathcal{S}$ provides *lr-indistinguishability under chosen-ciphertext attacks*, or, for short, $\mathcal{K}_\mathcal{S}$ is lr-IND-CCA-secure, if for all adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{lr\text{-IND-CCA}}_{\mathcal{K}_\mathcal{S},\mathcal{A}}(n)$ in winning the experiment $\mathrm{Expt}^{lr\text{-IND-CCA}}_{\mathcal{K}_\mathcal{S},\mathcal{A}}(n)$ as depicted in Figure 5 defined as

$$\mathsf{Adv}^{lr\text{-IND-CCA}}_{\mathcal{K}_\mathcal{S},\mathcal{A}}(n) := \left| \Pr\left[\mathrm{Expt}^{lr\text{-IND-CCA}}_{\mathcal{K}_\mathcal{S},\mathcal{A}}(n) = 1 \right] - \frac{1}{2} \right| \tag{8}$$

is negligible in n.

$\underline{\mathrm{Expt}^{lr-\mathsf{IND\text{-}CCA}}_{\mathcal{K}_\mathcal{S},\mathcal{A}}(n)}$:

1. $n_l, n_r \leftarrow 0$
2. $(D,d) \leftarrow \mathsf{DKeyGen}(1^n)$
3. $(E,e) \leftarrow \mathsf{EKeyGen}(1^n)$
4. $c^*, k_0^* \leftarrow \mathsf{sEncap}(e,D)$
5. $k_1^* \leftarrow\!\!\$\, K$
6. $b \leftarrow\!\!\$\, \{0,1\}$
7. $b' \leftarrow \mathcal{A}^{O_E(\cdot), O_D(\cdot,\cdot)}(E,D,c^*,k_b^*)$
8. return $[b = b']$

$\underline{O_D(E',c)}$:

1. if $n_l \geq l$, return $\perp$
2. else $n_l += 1$
3. if $(E',c) = (E,c^*)$, return $\perp$
4. $k \leftarrow \mathsf{sDecap}(d,E',c)$

$\underline{O_E(D')}$:

1. if $n_r \geq r$, return $\perp$
2. else $n_r += 1$
3. $(c,k) \leftarrow \mathsf{sEncap}(e,D')$
4. if $(D',c) = (D,c^*)$, return $\perp$
5. else (c,k)

Figure 5: The $lr-\mathsf{IND\text{-}CCA}$ security experiments for split KEMs.

Notably, nn-IND-CCA and mm-IND-CCA-security pair with the traditional notions of IND-CPA and IND-CCA-security.

Finally, we include the definition of correctness for (asymmetric) split KEMs for completeness.

Definition 2.15 (Correctness of Asymmetric Split KEMs). We say that an asymmetric split KEM, $\mathcal{K}_\mathcal{S}$, is ε-correct, if:

$$\Pr[\mathsf{sDecap}(d,E,c) \neq k : \begin{smallmatrix}(E,e)\leftarrow\mathsf{EKeyGen},\\ (D,d)\leftarrow\mathsf{DKeyGen},\\ (c,k)\leftarrow\mathsf{sEncap}(e,D)\end{smallmatrix}] \leq \varepsilon. \tag{9}$$

We say a split KEM, $\mathcal{K}_\mathcal{S}$, is perfectly correct, if $\varepsilon = 0$.

2.6.1 The Trivial Split KEM

One natural question that arises when considering split KEMs is whether standard KEMs or public-key encryption (PKE) algorithms can be used to construct a split KEM without additional algorithms. As noted in the introduction, Brendel *et al.* attempted to replace the DH values with KEM ciphertexts (Brendel et al., 2020). This attempt can be formalized as the split KEM in Figure 6, where K is a KEM.

Once again, the issue with this construction becomes apparent in the context of the X3DH protocol. As the encapsulator's keys are simply defined to be 0, it cannot produce meaningful keying material that replicates the original protocol. In particular, the keying material computed from the long-term DH of Alice and the semistatic DH key of Bob is missing.

$S.\mathsf{EKeyGen}(1^n)$:
1. $E = 0, e = 0$

$S.\mathsf{DKeyGen}(1^n)$:
1. $K.\mathsf{KeyGen}(1^n) \to (ek, dk)$
2. $D = ek, d = (ek, dk)$

$S.\mathsf{sEncap}(e, D)$:
1. If $e \neq 0$ return $\perp$
2. Else $K.\mathsf{Encap}(ek) \to c, k$
3. Return (c, k)

$S.\mathsf{sDecap}(d, E, c)$:
1. If $E \neq 0$, $\perp = k'$
2. Else, $K.\mathsf{Decap}(dk, c) = k'$
3. Return k'

Figure 6: The trivial split KEM, S.

In this construction, the randomness the encapsulator uses acts in place of an ephemeral DH key, let alone the long-term key of Alice.

The second issue comes as a result of the first. In order to correct this lack of keying material, a (standard) KEM ciphertext from Bob to Alice is needed. However, this causes the asynchronicity property of the X3DH protocol to fail, as both Alice and Bob need to be online *simultaneously* in order to compute the session key. As a result, while a trivial split KEM can be defined, it does not address the issues that Brendel *et al.* found with (standard) KEM only replacements for X3DH.

3 SPLIT KEM COMBINER

In this section, we present the main result of this paper, a *mm*-IND-CCA split KEM, and a high-level comparison with past works. We first introduce our new construction and then prove *mm*-IND-CCA security.

3.1 Construction

We now present our split KEM, denoted by $\mathfrak{S}$, in Fig. 7. We let Π denote a IND-CCA PKE, Σ a SUF-CMA digital signature scheme, and $\mathcal{P}$ be a cPRF. We note here that we use the notation id_x to represent the circuit that is 1 when evaluated on x and 0 everywhere else.

3.2 Constrained PRF for Singleton Sets

Before we proceed to the security proof for our construction, we note that a constrained PRF, $\mathcal{P}$, suitable for our construction can be obtained from a PRF. We provide a brief description here and the full details in the Appendix. Given a PRF, P, with master secret key msk we define $\mathcal{P}.\mathsf{Eval}$ on input x as $P.\mathsf{Eval}(P.\mathsf{Eval}(msk, x), x)$, and the constrained key for the point x is defined as $\tau_x = P.\mathsf{Eval}(msk, x)$. Clearly,

$\mathfrak{K}.\mathsf{EKeyGen}(1^n)$:
1. $\mathcal{P}.\mathsf{Setup}(1^n) \to msk$
2. $\Sigma.\mathsf{KeyGen}(1^n) \to vk, sk$
3. $E = vk, e = (vk, sk, msk)$

$\mathfrak{K}.\mathsf{DKeyGen}(1^n)$:
1. $\Pi.\mathsf{KeyGen}(1^n) \to (ek, dk)$
2. $D = ek, d = (ek, dk)$

$\mathfrak{K}.\mathsf{sEncap}(e, D)$:
1. $x \leftarrow\!\!\$\, X$
2. $\mathcal{P}.\mathsf{Con}(msk, \mathsf{id}_x) \to \tau_x$
3. $\Pi.\mathsf{Enc}(ek, vk \| x \| \tau_x) \to c$
4. $\Sigma.\mathsf{Sign}(sk, vk \| c) \to \sigma$
5. $\mathcal{P}.\mathsf{Eval}(msk, x) \to \kappa$
6. $\mathsf{KDF}(\kappa, \emptyset, x \| c \| \sigma, n) \to k$
7. return $((c, \sigma), k)$

$\mathfrak{K}.\mathsf{sDecap}(d, E = vk, (c, \sigma))$:
1. $\Sigma.\mathsf{Vfy}(vk, vk \| c, \sigma) \to b$
   - If $b = 0$, return $\perp$
   - If $b = 1$ then $\Pi.\mathsf{Dec}(dk, c) \to vk' \| x' \| \tau'$
   - If $vk \neq vk'$, return $\perp$
   - Else $\mathcal{P}.\mathsf{cEval}(\tau', x') \to \kappa'$
   - Return $\mathsf{KDF}(\kappa', \emptyset, x' \| c \| \sigma, n)$

Figure 7: Our $mm-$IND-CCA-secure split KEM, $\mathfrak{S}$.

given τ_x and x, anyone can derive $\mathcal{P}.\mathsf{Eval}(msk, x)$.

Thus, it is possible to avoid the complicated tools necessary to construct robust constrained PRFs such as NC^1 circuits (Canetti and Chen, 2017) or indistinguishability obfuscation (Davidson et al., 2020) when implementing our split KEM $\mathfrak{K}$.

3.3 Security Proof

Theorem 3.1. *Suppose that $\mathcal{P}$ is a constrained PRF (cPRF), that Π is a well-spread,* IND-CCA*-secure PKE, that* KDF *is a* KDF-IND*-secure KDF, and that Σ is an* SUF-CMA*-secure DS against efficient QPT adversaries, then the split KEM described in Fig. 7 is a mm-*IND-CCA*-secure split KEM against efficient QPT adversaries. More precisely, for any efficient QPT adversary $\mathcal{A}$, against the split-KEM $\mathfrak{K}$, that makes q_{sEncap} sEncap queries, there exist efficient QPT adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$, such that*

$$\mathsf{Adv}^{mm\text{-IND-CCA}}_{\mathfrak{K},\mathcal{A}}(n) \leq \frac{q_{\mathsf{sEncap}}}{2^\gamma} + \mathsf{Adv}^{rndm}_{\mathcal{P},\mathcal{B}_1}(n) + \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\Pi,\mathcal{B}_2}(n) + \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{B}_3}(n) + \mathsf{Adv}^{\mathsf{KDF\text{-}IND}}_{\mathsf{KDF},\mathcal{B}_4}(n) \tag{10}$$

Proof. We will prove our main result by performing a series of game hops. The aim is to decouple the ciphertext and key $\mathcal{A}$ receives from each other, and thus leave the adversary with no better option than to flip its own coin to guess b.

Game 0. We begin by considering the original security experiment in Fig. 8 as G_0. Thus, we have

$$\mathsf{Adv}^{G_0}_{\mathfrak{K},\mathcal{A}}(n) = \left| \Pr\left[\mathrm{Expt}^{mm\text{-IND-CCA}}_{\mathfrak{K},\mathcal{A}}(n) = 1 \right] - \frac{1}{2} \right|. \tag{11}$$

$\underline{\text{Expt}^{mn\text{-IND-CCA}}_{\mathfrak{K},\mathcal{A}}(n)}$:

1. $n_l, n_r \leftarrow 0$
2. $\Pi.\mathsf{KeyGen}(1^n) \rightarrow (ek, dk)$
3. $D^* = ek, d^* = dk$
4. $\mathcal{P}.\mathsf{Setup}(1^n) \rightarrow msk$
5. $\Sigma.\mathsf{KeyGen}(1^n) \rightarrow sk, vk$
6. $E^* = vk, e^* = (vk, sk, msk)$
7. $x^* \leftarrow\!\$\, X$
8. $\mathcal{P}.\mathsf{Con}(msk, \mathsf{id}_{x^*}) \rightarrow \tau_{x^*}$
9. $\Pi.\mathsf{Enc}(ek, E^* \| x^* \| \tau_{x^*}) \rightarrow c^*$
10. $\Sigma.\mathsf{Sign}(sk, E^* \| c) \rightarrow \sigma^*$
11. $\mathcal{P}.\mathsf{Eval}(msk, x^*) \rightarrow \kappa^*$
12. $\mathsf{KDF}(\kappa^*, \emptyset, x^* \| c^* \| \sigma^*, n) \rightarrow k_0^*$
13. $k_1^* \leftarrow\!\$\, Y$
14. $b \leftarrow\!\$\, \{0,1\}$
15. $b' \leftarrow \mathcal{A}^{O_E(\cdot), O_D(\cdot,\cdot)}(E^*, D^*, (c^*, \sigma^*), k_b^*)$
16. return $b = b'$

$\underline{O_D(E', (c, \sigma))}$:

1. if $n_l \geq l$ return $\perp$
2. else $n_l += 1$
3. if $(E', (c, \sigma)) = (E^*, (c^*, \sigma^*))$ return $\perp$
4. $k \leftarrow \mathsf{sDecap}(d^*, E', (c, \sigma))$

$\underline{O_E(D')}$:

1. if $n_r \geq r$ return $\perp$
2. else $n_r += 1$
3. $((c, \sigma), k) \leftarrow \mathsf{sEncap}(e^*, D')$
4. if $(D', (c, \sigma)) = (D^*, (c^*, \sigma^*))$ return $\perp$
5. else $((c, \sigma), k)$

Figure 8: Game 0.

Game 1. The first Game, G_1, is identical to the original experiment, except that we abort the game if the challenge ciphertext c^* is outputted by the encapsulation oracle. By assumption Π is well-spread and so the probability of this abort condition occurring is bounded by $\frac{q_{\mathsf{sEncap}}}{2^\gamma}$. Thus, we have that

$$\mathsf{Adv}^{Go}_{\mathfrak{K},\mathcal{A}}(n) \leq \mathsf{Adv}^{G_1}_{\mathfrak{K},\mathcal{A}}(n) + \frac{q_{\mathsf{sEncap}}}{2^\gamma}. \quad (12)$$

Game 2. In the next Game, G_2 Fig. 9, we will modify the decapsulation oracle so that it will return $\perp$ on any query involving c^*. We claim that this change cannot meaningfully be noticed by the adversary unless they are able to undermine the SUF-CMA-security of Σ.

Claim 1.

$$\mathsf{Adv}^{G_1}_{\mathfrak{K},\mathcal{A}}(n) \leq \mathsf{Adv}^{G_2}_{\mathfrak{K},\mathcal{A}}(n) + \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{B}_1}(n) \quad (13)$$

Claim 1. To prove our claim, we will consider the structure of queries containing c^* and how they are handled in G_1.

1. $(E^*, (c^*, \sigma^*))$. This query is prohibited in G_1; therefore, in both games, $\perp$ is returned.
2. $(E \neq E^*, (c^*, \sigma^*))$. In this query, the verification key submitted is different from the challenge verification key encrypted within c^*. In G_1, such a query would result in the signature verification algorithm returning 0, thus returning $\perp$.

$\underline{\text{Expt}^{mn\text{-IND-CCA}}_{\mathfrak{K},\mathcal{A}}(n)}$:

1. $n_l, n_r \leftarrow 0$
2. $\Pi.\mathsf{KeyGen}(1^n) \rightarrow (ek, dk)$
3. $D^* = ek, d^* = dk$
4. $\mathcal{P}.\mathsf{Setup}(1^n) \rightarrow msk$
5. $\Sigma.\mathsf{KeyGen}(1^n) \rightarrow sk, vk$
6. $E^* = vk, e^* = (vk, sk, msk)$
7. $x^* \leftarrow\!\$\, X$
8. $\mathcal{P}.\mathsf{Con}(msk, \mathsf{id}_{x^*}) \rightarrow \tau_{x^*}$
9. $\Pi.\mathsf{Enc}(ek, E^* \| x^* \| \tau_{x^*}) \rightarrow c^*$
10. $\Sigma.\mathsf{Sign}(sk, E^* \| c) \rightarrow \sigma^*$
11. $\mathcal{P}.\mathsf{Eval}(msk, x^*) \rightarrow \kappa^*$
12. $\mathsf{KDF}(\kappa^*, \emptyset, x^* \| c^* \| \sigma^*, n) \rightarrow k_0^*$
13. $k_1^* \leftarrow\!\$\, Y$
14. $b \leftarrow\!\$\, \{0,1\}$
15. $b' \leftarrow \mathcal{A}^{O_E(\cdot), O_D(\cdot,\cdot)}(E^*, D^*, (c^*, \sigma^*), k_b^*)$
16. return $b = b'$

$\underline{O_D(E', c)}$:

1. if $n_l \geq l$ return $\perp$
2. else $n_l += 1$
3. $\boxed{\text{if } c = c^*, \text{return } \perp}$
4. $k \leftarrow \mathsf{sDecap}(d^*, E', (c, \sigma))$

$\underline{O_E(D')}$:

1. if $n_r \geq r$ return $\perp$
2. else $n_r += 1$
3. $((c, \sigma), k) \leftarrow \mathsf{sEncap}(e^*, D')$
4. $\boxed{\text{if } c = c^*, \text{return } \perp}$
5. else $((c, \sigma), k)$

Figure 9: Game 2.

3. $(E \neq E^*, (c^*, \sigma \neq \sigma^*))$. In this query, both the verification key and the signature differ from the challenge values. Regardless, if the signature verifies in G_0, the oracle will then check if $E = E^*$, which by assumption they are not, and thus the oracle will return $\perp$.
4. $(E^*, (c^*, \sigma \neq \sigma^*))$. In this query, only the signature is different from the challenge. In G_1, this query does not return $\perp$ if the signature can be correctly verified with E^*, while in G_2 it will always result in rejection.

Next, we show that the probability that a query described in 4 is bounded above by the probability that an SUF-CMA adversary, $\mathcal{B}_1$, is able to win the SUF-CMA experiment using $\mathcal{A}$.

To see this, suppose that $\mathcal{B}_1$ is an SUF-CMA-attacker of Σ, as described in Fig. 2. After being given the verification key, $E^* = vk^*$, by its challenger, as well as the signing oracle, $\mathcal{B}_1$ then runs $\Pi.\mathsf{KeyGen}(1^n)$ and $\mathcal{P}.\mathsf{Setup}(1^n)$ to obtain $(D^* = ek, dk)$ and msk, respectively.

Next, $\mathcal{B}_1$ then randomly selects x^* from the domain of $\mathcal{P}$, computes the constrained key τ_{x^*}, before encrypting vk^*, x^*, τ_{x^*} with its encryption key ek, to create c^*. It then computes the real key k_0^* as in Figure 7, selects k_1^* at random, and flips a bit b before sending $E^*, D^*, (c^*, \sigma^*), k_b^*$ to $\mathcal{A}$, where σ^* is created by submitting $E^* \| c^*$ to its oracle.

Equipped with keys and the signing oracle, $\mathcal{B}_1$ can

simply impersonate the $\mathcal{A}$' oracles. By maintaining its own list of signatures created by its oracle, if given a decryption query as described in 4, which is verified by the challenge key E^* and contains a signature it did not record, $\mathcal{B}_1$ then submits this query as its answer to the SUF-CMA-experiment and wins.

By the assumption that Σ is SUF-CMA secure, this can happen only with negligible probability, and thus our claim holds.

□

Game 3. In the next Game G_3, Fig. 10, we will modify the sEncap algorithm to replace the constrained key τ_{x^*} with a string of 1s of equal length during encryption. We claim that this replacement cannot non-negligibly change the probability of $\mathcal{A}$ winning. Before we prove this claim formally, we provide an informal explanation. By assumption, we have that Π is IND-CCA-secure. If this change results in $\mathcal{A}$ being more likely to win the experiment, then an attacker against Π's IND-CCA-security can simulate $\mathcal{A}$ and estimate its probability of winning to decide which message was encrypted. Now we provide the formal details.

$\underline{\text{Expt}^{mn\text{-IND-CCA}}_{\mathfrak{K},\mathcal{A}}(n)}$:

1. $n_l, n_r \leftarrow 0$
2. $\Pi.\mathsf{KeyGen}(1^n) \rightarrow (ek, dk)$
3. $D^* = ek, d^* = dk$
4. $\mathcal{P}.\mathsf{Setup}(1^n) \rightarrow msk$
5. $\Sigma.\mathsf{KeyGen}(1^n) \rightarrow sk, vk$
6. $E^* = vk, e^* = (vk, sk, msk)$
7. $x^* \leftarrow\!\!\$\, X$
8. $\mathcal{P}.\mathsf{Con}(msk, \mathsf{id}_{x^*}) \rightarrow \tau_{x^*}$
9. $\boxed{\Pi.\mathsf{Enc}(ek, E^* \| x^* \| 1^{|\tau_{x^*}|}) \rightarrow c^*}$
10. $\Sigma.\mathsf{Sign}(sk, E^* \| c) \rightarrow \sigma^*$
11. $\mathcal{P}.\mathsf{Eval}(msk, x^*) \rightarrow \kappa^*$
12. $\mathsf{KDF}(\kappa^*, \emptyset, x^* \| c^* \| \sigma^*, n) \rightarrow k_0^*$
13. $k_1^* \leftarrow\!\!\$\, Y$
14. $b \leftarrow\!\!\$\, \{0,1\}$
15. $b' \leftarrow \mathcal{A}^{O_E(\cdot), O_D(\cdot,\cdot)}(E^*, D^*, (c^*, \sigma^*), k_b^*)$
16. return $b = b'$

$\underline{O_D(E', c, \sigma)}$:

1. if $n_l \geq l$ return $\perp$
2. else $n_l += 1$
3. $\boxed{\text{if } c = c^*, \text{return } \perp}$
4. $k \leftarrow \mathsf{sDecap}(d^*, E', c)$

$\underline{O_E(D')}$:

1. if $n_r \geq r$ return $\perp$
2. else $n_r += 1$
3. $((c, \sigma), k) \leftarrow \mathsf{sEncap}(e^*, D')$
4. $\boxed{\text{if } c = c^*, \text{return } \perp}$
5. else $((c, \sigma), k)$

Figure 10: Game 3.

Claim 2.

$$\mathsf{Adv}^{G_2}_{\mathfrak{K},\mathcal{A}}(n) \leq \mathsf{Adv}^{G_3}_{\mathfrak{K},\mathcal{A}}(n) + \mathsf{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{B}_2}(n). \quad (14)$$

Claim 2. Suppose that $\mathcal{B}_2$ is an IND-CCA-attacker of Π, as described in Fig. 1. After being given the public key $D^* = ek$ by the challenger, $\mathcal{B}_2$ first runs $\Sigma.\mathsf{KeyGen}(1^n)$ and $\mathcal{P}.\mathsf{Setup}(1^n)$ to obtain $(sk, vk = E^*)$ and, msk, respectively. Next, $\mathcal{B}_2$ chooses x^* uniformly at random from the domain of $\mathcal{P}$, and computes $\mathcal{P}.\mathsf{Con}(msk, \mathsf{id}_{x^*}) \rightarrow \tau_{x^*}$. Afterwards, $\mathcal{B}_2$ submits $E^* \| x^* \| \tau_{x^*}$ and $E^* \| x^* \| 1^{|\tau_{x^*}|}$ as its two challenge messages to the challenger, and receives c^* back. Next, $\mathcal{B}_2$ signs $E^* \| c^*$ to create the signature σ^*, computes the real key k_0^* as in Figure 7, selects k_1^* uniformly at random. Finally, $\mathcal{B}_2$ flips a bit $b \leftarrow \{0,1\}$ and forwards $(E^*, D^*, (c^*, \sigma^*), k_b^*)$ to $\mathcal{A}$.

Before $\mathcal{B}_2$ submits the challenge messages, they are able to perfectly simulate the oracles of $\mathcal{A}$ with the aid of the decryption oracle. Furthermore, after $\mathcal{B}_2$ submits its messages, its decryption oracle is changed so that it will reject c^*. However, by G_1 all queries involving c^* will return $\perp$, and therefore $\mathcal{B}_2$ does not need to query its oracle on c^* to be able to answer queries involving c^*. Thus, $\mathcal{B}_2$ can still perfectly simulate the oracles $\mathcal{A}$ using its secret keys and oracle.

At this point, depending on which message is chosen by the IND-CCA-challenger, $\mathcal{B}_2$'s simulation will correspond to either G_2 or G_3. If the change made between games results in a non-negligible change in $\mathcal{A}$'s probability of winning, by estimating this probability, $\mathcal{B}_2$ can distinguish which message was chosen and win the IND-CCA-experiment. Thus, we conclude that

$$\mathsf{Adv}^{G_2}_{\mathfrak{K},\mathcal{A}}(n) \leq \mathsf{Adv}^{G_3}_{\mathfrak{K},\mathcal{A}}(n) + \mathsf{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{B}_2}(n). \quad (15)$$

□

Game 4. In the next Game G_4, Figure 11, we modify the sEncap algorithm to replace $\mathcal{P}.\mathsf{Eval}(msk, x^*)$ with a uniformly random κ'. We claim that if this change results in $\mathcal{A}$ being able to win G_3 with some non-negligible probability, then they can be used by another adversary, $\mathcal{B}_3$, to win the $\text{Expt}^{\text{rndm}}_{\mathcal{P},\mathcal{B}_3}$ with the same probability.

Claim 3.

$$\mathsf{Adv}^{G_3}_{\mathfrak{K},\mathcal{A}}(n) \leq \mathsf{Adv}^{G_4}_{\mathfrak{K},\mathcal{A}}(n) + \mathsf{Adv}^{\text{rndm}}_{\mathcal{P},\mathcal{B}_3}(n) \quad (16)$$

Claim 3. To see this, consider $\mathcal{B}_3$ in the $\text{Expt}^{\text{rndm}}_{\mathcal{P},\mathcal{B}_3}(n)$ experiment described in Fig. 4.

After the experiment begins and $\mathcal{B}_3$ is given 1^n and its oracles, they run $\Pi.\mathsf{KeyGen}(1^n)$ and $\Sigma.\mathsf{KeyGen}(1^n)$ to obtain $(D^* = ek, dk)$ and $(E^* = vk, sk)$, respectively.

To create the challenge input for $\mathcal{A}$, $\mathcal{B}_3$ does the following: selects x^* uniformly at random and submits it to the experiment challenger to receive y_b, which is either $y_0 \leftarrow \mathcal{P}.\mathsf{Eval}(msk, x^*)$ or $y_1 \leftarrow\!\!\$\, Y$, depending on the bit chosen by the challenger. Next, $\mathcal{B}_3$ uses D^* to create the ciphertext c^* of the following message $E^* \| x^* \| 1^l$, where l is the length

$\underline{\text{Expt}^{mn\text{-IND-CCA}}_{\mathcal{R},\mathcal{A}}(n)}$:

1. $n_l, n_r \leftarrow 0$
2. $\Pi.\mathsf{KeyGen}(1^n) \rightarrow (ek, dk)$
3. $D^* = ek, d^* = dk$
4. $\mathcal{P}.\mathsf{Setup}(1^n) \rightarrow msk$
5. $\Sigma.\mathsf{KeyGen}(1^n) \rightarrow sk, vk$
6. $E^* = vk, e^* = (vk, sk, msk)$
7. $x^* \leftarrow\!\!\$\, X$
8. $\mathcal{P}.\mathsf{Con}(msk, \mathsf{id}_{x^*}) \rightarrow \tau_{x^*}$
9. $\boxed{\Pi.\mathsf{Enc}(ek, E^* \| x^* \| 1^{|\tau_{x^*}|}) \rightarrow c^*}$
10. $\Sigma.\mathsf{Sign}(sk, E^* \| c) \rightarrow \sigma^*$
11. $\boxed{\xi \leftarrow\!\!\$\, X}$
12. $\mathsf{KDF}(\xi, \emptyset, x^* \| c^* \| \sigma^*, n) \rightarrow k_0^*$
13. $k_1^* \leftarrow\!\!\$\, Y$
14. $b \leftarrow\!\!\$\, \{0,1\}$
15. $b' \leftarrow \mathcal{A}^{O_E(\cdot), O_D(\cdot,\cdot)}(E^*, D^*, (c^*, \sigma^*), k_b^*)$
16. return $b = b'$

$\underline{O_D(E', c, \sigma)}$:

1. if $n_l \geq l$ return $\perp$
2. else $n_l += 1$
3. $\boxed{\text{if } c = c^*, \text{return } \perp}$
4. $k \leftarrow \mathsf{sDecap}(d^*, E', c)$

$\underline{O_E(D')}$:

1. if $n_r \geq r$ return $\perp$
2. else $n_r += 1$
3. $((c, \sigma), k) \leftarrow \mathsf{sEncap}(e^*, D')$
4. $\boxed{\text{if } c = c^*, \text{return } \perp}$
5. else $((c, \sigma), k)$

Figure 11: Game 4.

of constrained key computed from x^*. Subsequently, $\mathcal{B}_3$ sk to sign $E^* \| c^*$ obtaining σ^*. Finally, $\mathcal{B}_3$ computes $\mathsf{KDF}(y_b, \emptyset, x^* \| c^* \| \sigma^*, n)$ and submits $(E^*, D^*, (c^*, \sigma^*), y_b)$ to $\mathcal{A}$.

In order to answer an encapsulation query D', $\mathcal{B}_3$ does the following: they chose x' uniformly at random and not equal to x^*. Next, it queries both the Con and Eval oracles on x' to obtain $\tau_{x'}$ and ξ', respectively. They then encrypt $E^* \| x' \| \tau_{x'}$ to produce c and then creates the signature σ for the message $E^* \| c$. Lastly, $\mathcal{B}_3$ computes the key as $k' = \mathsf{KDF}(\xi', \emptyset, x' \| c \| \sigma, n)$. Finally, $\mathcal{B}_3$ returns c, σ, k' to $\mathcal{A}$.

When $\mathcal{A}$ makes a decryption query (E', c, σ), $\mathcal{B}_3$ first checks if the signature is valid on $E' \| c$. If it is not, they reject and return $\perp$. Otherwise, $\mathcal{B}_3$ decrypts the ciphertext c using its decryption key dk to recover the message $E'' \| x' \| \tau'$, and confirms if $E' = E''$. If not, they return $\perp$, and otherwise computes $\mathcal{P}.\mathsf{cEval}(\tau', x') = \xi'$ and returns $\mathsf{KDF}(\xi', \emptyset, x' \| c \| \sigma, n)$.

We note that $\mathcal{B}_3$ handles decryption queries with c^* by simply rejecting and returning $\perp$ as in Game G_2. This ensures that they avoid having to query their own oracles on x^* and lose their experiment.

Eventually, $\mathcal{A}$ submits its guess bit b' which $\mathcal{B}_3$ forwards to its own challenger. If $\mathcal{B}_3$'s challenger's bit was 0, then $\mathcal{B}_3$ received the true evaluation of x^*, and $\mathcal{A}$ was given the real key as part of the input $(E^*, D^*, (c^*, \sigma^*), y_b)$. Thus, $\mathcal{B}_3$ was simulating G_3 to $\mathcal{A}$. Similarly, if the bit was 1 then $\mathcal{B}_3$, and consequentially $\mathcal{A}$, were given a random key, and so $\mathcal{B}_3$ was simulating G_4 to $\mathcal{A}$. Thus, if the change from G_3 to G_4 results in a significant increase in $\mathcal{A}$'s probability of winning, then $\mathcal{B}_3$ can win its experiment with the same probability. However, by assumption $\mathcal{P}$ is a pseudorandom constrained PRF, and thus, the difference must be negligible.

□

Finally, we will show that $\mathcal{A}$'s of winning in G_4 is at most negligible, and if not implies an attack on the KDF-IND-security of KDF.

Claim 4.

$$\mathsf{Adv}^{G_4}_{\mathcal{R},\mathcal{A}}(n) \leq \mathsf{Adv}^{\mathsf{KDF\text{-}IND}}_{\mathcal{P},\mathcal{B}_4}(n) \tag{17}$$

Claim 4. To see this, consider $\mathcal{B}_4$ in the $\text{Expt}^{\mathsf{KDF\text{-}IND}}_{\mathcal{P},\mathcal{B}_3}(n)$ experiment described in Fig. 3

After the KDF-IND experiment has begun, $\mathcal{B}$ is given (a, s), and access to the KDF oracle, they then run $\Sigma.\mathsf{KeyGen}(1^n)$, $\Pi.\mathsf{KeyGen}(1^n)$, and $\mathcal{P}.\mathsf{Setup}(1^n)$ to receive $(E^* = vk, sk)$, $(D^* = ek, dk)$ and msk, respectively. From there, they sample a uniformly random x^*, and compute $\mathcal{P}.\mathsf{Con}(msk, \mathsf{id}_{x^*})$ to obtain τ_{x^*} and record the length. Next, they generate the ciphertext $c^* = \Pi.\mathsf{Enc}(ek, E^* \| x^* \| 1^{|\tau_{x^*}|})$ and the signature $\sigma^* = \Sigma.\mathsf{Sign}(sk, E^* \| c)$. Finally, $\mathcal{B}_4$ submits as its challenge $(x^* \| c^* \| \sigma^*, n)$, receives k_b^* and sends $\mathcal{A}$ $(E^*, D^*, (c^*, \sigma^*), k_b^*)$.

In order to answer $\mathcal{A}$'s encapsulation queries, $\mathcal{B}_4$ does the following: selects an $x' =\neq x^*$, and then using the keys it generated computes $\tau_{x'}$, the ciphertext $c = \Pi.\mathsf{Enc}(ek, E^* \| x' \| \tau_{x'})$, and signature σ for $E^* \| c$. Lastly, $\mathcal{B}_4$ queries its oracle on $(x^* \| c \| \sigma, n)$ to get k and outputs $((c, \sigma), k)$ to $\mathcal{A}$.

Decapsulation queries are handled by $\mathcal{B}_4$ as follows. Any queries involving c^* have $\perp$ returned according to the change made in G_2. For all other queries $(E, (c, \sigma))$, $\mathcal{B}_4$ first verifies the signature σ for the message $E \| c$ using E. If this fails, $\mathcal{B}_4$ aborts and returns $\perp$. Otherwise, $\mathcal{B}_4$ continues and uses the decryption key dk it possesses to decrypt c to recover $E' \| x' \| \tau'$. Next, $\mathcal{B}_4$ confirms that $E = E'$ and aborts if not. If $E = E'$ then $\mathcal{B}_4$ computes $\mathcal{P}.\mathsf{cEval}(\tau') = \kappa$, then computes and returns $\mathsf{KDF}(\kappa, \emptyset, x' \| c \| \sigma, n)$.

Eventually, $\mathcal{A}$ will output a guess b and $\mathcal{B}_4$ will copy this guess. Thus, when $\mathcal{A}$ wins its experiment, $\mathcal{B}_4$ will also win their experiment. Therefore, we have

$$\mathsf{Adv}^{G_4}_{\mathcal{R},\mathcal{A}}(n) \leq \mathsf{Adv}^{\mathsf{KDF\text{-}IND}}_{\mathcal{P},\mathcal{B}_4}(n). \tag{18}$$

Thus, our claim holds, and our proof is concluded.

□

□

3.4 Discussion and Limitations

When split KEMs were origially introduced by Brendel *et al.* as an option to replace DH key agreement with a quantum-resistant alternative, they constructed a split KEM directly from the RLWE problem (Brendel et al., 2020). Although their construction is elegant in design, it was only shown to be *nn*-IND-CCA secure. Thus, while our main result does require a more involved construction with several input primitives, we realize full *mm*-IND-CCA-security. Furthermore, as a generic construction, it allows for plug-and-play implementations, rather than being limited to a single hard problem that may eventually be efficiently solved.

Brendel, Fiedler, Günther, Janson, and Stebila followed up their work on split KEMs with the deniable asynchronous key exchange protocol SPQR (Brendel et al., 2021). As stated in the introduction, our construction shares similiarites with SPQR, with two key differences. First, SPQR possess a deniablility property, whereas, we do not prove such a property for our construction. The second key distinction is their use of a designated verifier signature (DVS) scheme. By using a DVS their scheme does not allow for the signing of semistatic keys, which is present in the Signal protocol, as siging requires a specific partners public key. Our construction relies on the use of a standard DS, and so the signing of semistatic keys can be performed without the use of an additional DS being deployed which would be needed for SPQR.

Compared to SC-AKE of Hashimoto, Katsumata, Kwiatkowski, and Prest, our construction requires stronger assumptions on the ingredients inputs. We require an IND-CCA-secure PKE and a SUF-CMA DS, where SC-AKE needs an IND-CCA-secure KEM and an EUF-CMA DS. However, despite having relaxed assumptions, SC-AKE does not clearly address the problem of a KEM-based replacement for Signal, as shown by Bindel *et al.* (Brendel et al., 2020). There is a lack of keying material produced from the semistatic key of Bob and the long-term key of Alice. Split KEMs were created to ensure that this key material is produced and, as such, our construction is able to replicate this part of the Signal protocol.

4 CONCLUSION AND FUTURE WORK

The Diffie-Hellman problem is an extremely widespread and efficient tool to construct key agreement schemes that must be replaced with a post-quantum scheme. One such scheme that is put at risk by quantum attacks is the X3DH subprotocol of the Signal protocol, which relies on the intractability of the Diffie-Hellman problem. Crucially, the X3DH step allows for asynchronous key generation between parties.

Although there have been several works that have presented replacements for the X3DH protocol from the point of view of AKEs, an alternative option is the use of the relatively novel cryptosystem of split KEMs. These types of KEMs present a generic way to replicate the message flows of DH-based key agreement schemes. In particular, they can be used to exactly mirror the X3DH protocol, by producing keying material that is parallel to those used in the traditional protocol, something that standard KEMs cannot do without breaking the desirable asynchronicity property of X3DH.

In this work, we present a split KEM combiner from a PKE, a signature algorithm, and a constrained PRF to create a fully adaptive $mm-$IND-CCA-secure split KEM. We leverage the ability of the constrained PRF to produce restricted domain keys from the master secret key so that a third party may evaluate the limited domain as if they held the master secret key. In our split KEM, the encapsulator computes a constrained key, whose domain is a single, randomly chosen point. This key and point pair are then sent to the decapsulator using the PKE, while the signature is placed on the ciphertext to verify that there has not been any interference by an attacker. We then prove the *lr*-IND-CCA-security of our split KEM with a tight reduction in the standard model. Notably, due to our construction constraining to singleton set domains, we are able to convert standard PRFs into suitable constrained PRFs directly.

As we were able to achieve fully adaptive $mm-$IND-CCA-security, our split KEM presents a viable solution to replace the current DH-based key agreement. Furthermore, as our construction is a generic black-box construction, it will remain an evergreen method to ensure that DH-like communication is possible with suitable cryptographic primitives in the future.

4.1 Future Work

We leave it as an open question whether a strongly secure split KEM can be directly constructed from post-quantum assumptions, such as LWE. Another question inspired directly from our result is whether there can be any relaxation of our assumptions, in particular, the need for an IND-CCA-secure PKE. Furthermore, there is still an open question as to how to construct additional split KEMs via black-box combiners.

As for future work, the most significant limitation of our work is that we have restricted ourselves to split KEMs, which do not have the same history of study and familiarity as AKEs. Thus, we plan to investigate how to transform our construction into an appropriate type of AKE, such as a Signal-Conforming AKE. Moreover, we will investigate whether there are any possible efficiency gains from translating our split KEM into an appropriate AKE. This would include things such as lowering the bandwidth of communication by removing any redundancies introduced by a generic conversion from split KEM to SC-AKE or deniable AKE. Another important direction for the future of split KEMs is to define the notion of deniability. The Signal protocol possesses the property that transcripts between Alice and Bob cannot confirm with certainty that either truly participated, as the DH shares are used for authentication as opposed to signatures. As our construction relies on the use of traditional signatures, it intuitively cannot be a deniable scheme. Thus, in contexts where deniability is vital, how to construct a split KEM with this property is an open problem.

REFERENCES

Signal protocol. Technical documentation.

Attrapadung, N., Matsuda, T., Nishimaki, R., Yamada, S., and Yamakawa, T. (2018). Constrained prfs for nc^1 in traditional groups. In Shacham, H. and Boldyreva, A., editors, *Advances in Cryptology – CRYPTO 2018*, pages 543–574, Cham. Springer International Publishing.

Boneh, D., Lewi, K., and Wu, D. J. (2017). Constraining pseudorandom functions privately. In Fehr, S., editor, *Public-Key Cryptography – PKC 2017*, pages 494–524, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boneh, D. and Waters, B. (2013). Constrained pseudorandom functions and their applications. In Sako, K. and Sarkar, P., editors, *Advances in Cryptology - ASIACRYPT 2013*, pages 280–300, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boyd, C., Cliff, Y., Gonzalez Nieto, J., and Paterson, K. G. (2008). Efficient one-round key exchange in the standard model. In Mu, Y., Susilo, W., and Seberry, J., editors, *Information Security and Privacy*, pages 69–83, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boyle, E., Goldwasser, S., and Ivan, I. (2014). Functional signatures and pseudorandom functions. In Krawczyk, H., editor, *Public-Key Cryptography – PKC 2014*, pages 501–519, Berlin, Heidelberg. Springer Berlin Heidelberg.

Brakerski, Z. and Vaikuntanathan, V. (2015). Constrained key-homomorphic prfs from standard lattice assumptions. In Dodis, Y. and Nielsen, J. B., editors, *Theory of Cryptography*, pages 1–30, Berlin, Heidelberg. Springer Berlin Heidelberg.

Brendel, J., Fiedler, R., Günther, F., Janson, C., and Stebila, D. (2021). Post-quantum asynchronous deniable key exchange and the signal handshake. In *IACR Cryptology ePrint Archive*.

Brendel, J., Fischlin, M., Günther, F., Janson, C., and Stebila, D. (2020). Towards post-quantum security for Signal's X3DH handshake. In Jr., M. J. J., Dunkelman, O., and O'Flynn, C., editors, *Proc. 27th Conference on Selected Areas in Cryptography (SAC) 2020*, LNCS. Springer. To appear. Cryptology ePrint Archive, Report 2019/1356. http://eprint.iacr.org/2019/1356.

Canetti, R. and Chen, Y. (2017). Constraint-hiding constrained prfs for nc^1 from lwe. In Coron, J.-S. and Nielsen, J. B., editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 446–476, Cham. Springer International Publishing.

Castryck, W. and Decru, T. (2023). An efficient key recovery attack on sidh. In Hazay, C. and Stam, M., editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 423–447, Cham. Springer Nature Switzerland.

Castryck, W., Lange, T., Martindale, C., Panny, L., and Renes, J. (2018). Csidh: An efficient post-quantum commutative group action. In Peyrin, T. and Galbraith, S., editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 395–427, Cham. Springer International Publishing.

Chuah, C. W., Dawson, E., and Simpson, L. (2013). Key derivation function: The sckdf scheme. In Janczewski, L. J., Wolfe, H. B., and Shenoi, S., editors, *Security and Privacy Protection in Information Processing Systems*, pages 125–138, Berlin, Heidelberg. Springer Berlin Heidelberg.

Davidson, A., Katsumata, S., Nishimaki, R., Yamada, S., and Yamakawa, T. (2020). Adaptively secure constrained pseudorandom functions in the standard model. In Micciancio, D. and Ristenpart, T., editors, *Advances in Cryptology – CRYPTO 2020*, pages 559–589, Cham. Springer International Publishing.

Dierks, T. and Rescorla, E. (2008). The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard). Updated by RFCs 5746, 5878, 6176.

Diffie, W. and Hellman, M. (2006). New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654.

Dobson, S. and Galbraith, S. D. (2022). Post-quantum signal key agreement from sidh. In Cheon, J. H. and Johansson, T., editors, *Post-Quantum Cryptography*, pages 422–450, Cham. Springer International Publishing.

Fujisaki, E. and Okamoto, T. (2013). Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101.

Hashimoto, K., Katsumata, S., Kwiatkowski, K., and Prest, T. (2022). An efficient and generic construction for signal's handshake (x3dh): Post-quantum, state leakage secure, and deniable. *Journal of Cryptology*, 35(3).

Hofheinz, D., Kamath, A., Koppula, V., and Waters, B. (2019). Adaptively secure constrained pseudorandom functions. In Goldberg, I. and Moore, T., editors, *Financial Cryptography and Data Security*, pages 357–376, Cham. Springer International Publishing.

Huang, Q., Wong, D. S., Li, J., and Zhao, Y.-M. (2008). Generic transformation from weakly to strongly unforgeable signatures. *Journal of Computer Science and Technology*, 23(2):240–252.

Huang, Q., Wong, D. S., and Zhao, Y. (2007). Generic transformation to strongly unforgeable signatures. In Katz, J. and Yung, M., editors, *Applied Cryptography and Network Security*, pages 1–17, Berlin, Heidelberg. Springer Berlin Heidelberg.

Inc., W. (2016). Whatsapp encryption overview technical white paper.

Kiayias, A., Papadopoulos, S., Triandopoulos, N., and Zacharias, T. (2013). Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 669–684, New York, NY, USA. Association for Computing Machinery.

Kiltz, E. (2007). Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In Okamoto, T. and Wang, X., editors, *Public Key Cryptography – PKC 2007*, pages 282–297, Berlin, Heidelberg. Springer Berlin Heidelberg.

Krawczyk, H. (2005). Hmqv: A high-performance secure diffie-hellman protocol. In Shoup, V., editor, *Advances in Cryptology – CRYPTO 2005*, pages 546–566, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lauter, K. and Mityagin, A. (2006). Security analysis of kea authenticated key exchange protocol. In Yung, M., Dodis, Y., Kiayias, A., and Malkin, T., editors, *Public Key Cryptography - PKC 2006*, pages 378–394, Berlin, Heidelberg. Springer Berlin Heidelberg.

Marlinspike, M. and Perrin, T. (2016). The x3dh key agreement protocol. *Open Whisper Systems*, 283:10.

Okamoto, T. (2007). Authenticated key exchange and key encapsulation in the standard model. In Kurosawa, K., editor, *Advances in Cryptology – ASIACRYPT 2007*, pages 474–484, Berlin, Heidelberg. Springer Berlin Heidelberg.

Peikert, C. and Shiehian, S. (2018). Privately constraining and programming prfs, the LWE way. In Abdalla, M. and Dahab, R., editors, *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 675–701. Springer.

Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446.

Shor, P. W. (1994). Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, pages 124–134, Washington, DC, USA. IEEE Computer Society.

Steinfeld, R., Pieprzyk, J., and Wang, H. (2006). How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In Abe, M., editor, *Topics in Cryptology – CT-RSA 2007*, pages 357–371, Berlin, Heidelberg. Springer Berlin Heidelberg.

Wang, Y. and Tanaka, K. (2015). Generic transformation to strongly existentially unforgeable signature schemes with continuous leakage resiliency. In Foo, E. and Stebila, D., editors, *Information Security and Privacy*, pages 213–229, Cham. Springer International Publishing.

APPENDIX

In this Appendix, we show how to construct a cPRF for the class of circuits of indicator functions for singleton sets, from a standard PRF. We provide both the definition of a (standard) PRF, the relevant security definition before presenting our construction, and the proof of security.

Definition 4.1. A *PRF* P is a pair of algorithms $(\mathsf{Setup}, \mathsf{Eval})$ over domain X, range Y,

- $\mathsf{Setup}(1^n)$: given the security parameter n outputs a master secret key msk.
- $\mathsf{Eval}(msk, x)$: given the master secret key msk and an input $x \in X$, outputs some $y \in Y$.

Definition 4.2. A PRF, P, with domain and range X is said to be *pseudorandom* if for all adversary $\mathcal{A}$ we have that

$$\mathsf{Adv}^{\mathrm{rndm}}_{P,\mathcal{A}}(n) := \left| \Pr\left[\mathrm{Expt}^{\mathrm{rndm}}_{P,\mathcal{A}}(n) = 1\right] - \frac{1}{2} \right|, \quad (19)$$

is a negligible function in n, where $\mathrm{Expt}^{\mathrm{rndm}}_{P,\mathcal{A}}(n)$ is defined in Figure 12(Boneh and Waters, 2013).

$\mathrm{Expt}^{\mathrm{rndm}}_{P,\mathcal{A}}(n)$:

1. $msk \leftarrow\$ \mathcal{P}.\mathsf{Setup}(1^n), L = \emptyset$
2. $x^*\mathsf{st} \leftarrow \mathcal{A}^{O_{P.\mathsf{Eval}}(msk,\cdot)}(1^n)$
3. $y_0 \leftarrow P.\mathsf{Eval}(msk, x^*), y_1 \leftarrow\$ Y$
4. $b \leftarrow\$ \{0,1\}$
5. $b' \leftarrow \mathcal{A}^{O_{P.\mathsf{Eval}}(msk,\cdot)}(y_b, \mathsf{st})$
6. return $[b = b'] \wedge [x^* \notin L]$

$O_{P.\mathsf{Eval}}(msk, x)$:

1. $L \cup \{x\}$
2. return $P.\mathsf{Eval}(msk, x)$

Figure 12: The pseudorandomness experiment for PRF, P.

We now present a construction for a constrained PRF, $\mathcal{P}$, which is suitable for our split KEM construction from the standard PRF $P = (\mathsf{Setup}, \mathsf{Eval})$ below.

We denote the constraining function for the singleton set, $\{x\}$, as f_x, where $f_x(x') = 1$ if and only if $x' = x$ and 0 otherwise, and denote the corresponding constrained key as τ_x.

$\mathcal{P}.\mathsf{Setup}(1^n)$:	$\mathcal{P}.\mathsf{Eval}(msk, x)$:
1. $P.\mathsf{Setup}(1^n) \to msk$	1. $y \leftarrow P.\mathsf{Eval}(P.\mathsf{Eval}(msk, x), x)$
2. return msk	2. return y
$\mathcal{P}.\mathsf{Con}(msk, f_x)$:	$\mathcal{P}.\mathsf{cEval}(\tau_x)$:
1. $P.\mathsf{Eval}(msk, x) \to \tau_x$	1. $P.\mathsf{Eval}(\tau_x, x) \to y'$
2. return τ_x	2. return y'

Figure 13: A constrained PRF, $\mathcal{P}$, for singleton sets from a standard PRF, P.

Theorem 4.1. *Let P be a pseudorandom PRF with domain and range X, and whose* Setup *algorithm outputs master secret keys that are statistically close to uniformly at random, then the construction, $\mathcal{P}$, in Figure 13 is a correct pseudorandom constrained PRF over the circuit class of indicator functions for singleton sets.*

Proof. We begin with the correctness of $\mathcal{P}$. We will slightly abuse the notation so that x will denote both itself and f_x the indicator function for the set $\{x\}$, for $x \in X$. By the definition of Con, $\tau_x = P.\mathsf{Eval}(msk, x)$, so we see that by construction $\mathcal{P}.\mathsf{Eval}(msk, x)$ and $\mathcal{P}.\mathsf{cEval}(\tau_x, x)$ agree on all $x' \in X$ such that $f_x(x') = 1$.

We now prove that $\mathcal{P}$ is pseudorandom via a series of game hops. We consider Game 0 as the security experiment $\mathrm{Expt}^{\mathrm{rndm}}_{\mathcal{P},\mathcal{A}}(n)$ from Figure 4. That is,

$$\mathsf{Adv}^{G_0}_{\mathcal{P},\mathcal{A}}(n) = \left| \Pr\left[\mathrm{Expt}^{\mathrm{rndm}}_{\mathcal{P},\mathcal{A}}(n) = 1\right] - \frac{1}{2} \right|. \quad (20)$$

We note that by the definition of the security experiment, the $\mathcal{P}$ adversary cannot query any potential challenge value x^* to either of its oracles without ensuring that they cannot win.

In Game 1, we modify the experiment to replace all instances of $P.\mathsf{Eval}(msk, x^*)$ with a uniformly random value $r^* \in X$. To see that this cannot change $\mathcal{A}$'s probability of winning the experiment, let $\mathcal{B}_1$ be an adversary in the pseudo-randomness experiment of P. Once $\mathcal{B}_1$ is given oracle access to, $O_{P.\mathsf{Eval}}(msk, \cdot)$ it can simulate both of the $\mathcal{A}'s$ oracles straightforwardly. Given an Eval query, x, from $\mathcal{A}$, $\mathcal{B}_1$ queries $O_{P.\mathsf{Eval}}(msk, x)$ to get y, before computing and returning $P.\mathsf{Eval}(y, x) \to z$ itself. Queries made by $\mathcal{A}$ to the Con oracle are handled with a single query to $O_{P.\mathsf{Eval}}(msk, \cdot)$. Once $\mathcal{A}$ submits a x^*, $\mathcal{B}_1$ submits this as its own challenge in its experiment. Then, y_b is given to $\mathcal{B}_1$ who then computes $P.\mathsf{Eval}(y_b, x^*) \to z_0$, selects a z_1 uniformly at random, before giving $\mathcal{A}$ one of z_0 or z_1 at random. We note here that in the case where $\mathcal{A}$ is given z_0, if $\mathcal{B}_1$ was given, y_0 then it is simulating Game 0 to $\mathcal{A}$ and Game 1 otherwise.

If the change made in Game 1 results in $\mathcal{A}$ being able to win its experiment with a non-negligible probability, then $\mathcal{B}_1$ can win in its experiment as follows: simulate $\mathcal{A}$'s probability of winning and then guessing $b' = 0$ if the probability is negligible, and 1 otherwise. Note that this method of attack works only half of the times, as $\mathcal{B}_1$ must have given $\mathcal{A}$ z_0 uniformly at random. Thus, $\mathcal{B}_1$ must run this entire process at least twice to account for this fact. By assumption that P is a pseudorandom PRF, this attack must only succeed with a negligible probability, so the difference in probabilities of $\mathcal{A}$ winning between Game 0 and Game 1 is negligible. Thus, we have

$$\mathsf{Adv}^{G_1}_{\mathcal{P},\mathcal{A}}(n) \leq \mathsf{Adv}^{G_0}_{\mathcal{P},\mathcal{A}}(n) + 2 \cdot \mathsf{Adv}^{\mathrm{rndm}}_{P,\mathcal{B}_1}(n). \quad (21)$$

In Game 2, we now replace $P.\mathsf{Eval}(r^*, x^*)$ with a uniformly random s^*. By similar arguments as in Game 1 we will show that this change cannot meaningfully impact $\mathcal{A}$'s chances of winning in the constrained PRF experiment without creating a contradiction.

Let $\mathcal{B}_2$ be an adversary in the pseudorandom experiment of P. Once $\mathcal{B}_2$ is given oracle access, it can simulate $\mathcal{A}$'s oracles as previously described in Game 1. When $\mathcal{A}$ submits its challenge x^*, $\mathcal{B}_2$ then copies this challenge. By assumption, on P, its master secret keys are close to uniformly at random so $\mathcal{B}_2$ does not need to pick an r^* itself. Next, $\mathcal{B}_2$ receives y_b and passes it to $\mathcal{A}$. In particular, if $\mathcal{B}_2$ was given the true evaluation, then it simulates Game 1 to $\mathcal{A}$ and otherwise Game 2. When $\mathcal{A}$ submits its guess bit b', $\mathcal{B}_2$ copies its answer. Since we are considering the circuit class of indicator functions for singleton sets, no query of $\mathcal{A}$ requires $\mathcal{B}_2$ to try to perform evaluations related, x^* and so it can continue to perfectly simulate the necessary oracles to $\mathcal{A}$. Should this change result in a non-negligible change to $\mathcal{A}$'s probability of success, then $\mathcal{B}_2$ is also able to win the PRF pseudorandom experiment with the same probability. However, this contradicts our assumptions on P, so we can conclude that the difference in the probabilities of $\mathcal{A}$ winning in Games 1 and 2 must be negligible. Moreover, in Game 2 $\mathcal{A}$ receives a random value independent of the bit chosen by the experiment, and consequently has 0 advantage in Game 2. Thus, we have

$$\mathsf{Adv}^{G_2}_{\mathcal{P},\mathcal{A}}(n) \leq \mathsf{Adv}^{G_1}_{\mathcal{P},\mathcal{A}}(n) + \mathsf{Adv}^{\mathrm{rndm}}_{P,\mathcal{B}_2}(n), \quad (22)$$

which concludes our proof.

□