Practical (Post-Quantum) Key Combiners from One-Wayness and Applications to TLS

Nimrod Aviram[1], Benjamin Dowling[2], Ilan Komargodski[3],
Kenneth G. Paterson[4], Eyal Ronen[1], and Eylon Yogev[5]

[1] Tel Aviv University
[2] The University Of Sheffield
[3] Hebrew University and NTT Research
[4] ETH Zurich
[5] Bar-Ilan University

Abstract. The task of combining cryptographic keys, some of which may be maliciously formed, into one key, which is (pseudo)random is a central task in cryptographic systems. For example, it is a crucial component in the widely used TLS and Signal protocols. From an analytical standpoint, current security proofs model such *key combiners* as dual-PRFs – a function which is a PRF when keyed by either of its two inputs – guaranteeing pseudo-randomness if one of the keys is compromised or even maliciously chosen by an adversary.
However, in practice, implementations of key combiners significantly depart from the "dual-PRF" standard set by provable schemes. Specifically, existing implementations typically use heuristic approaches and rely on strong assumptions that are only known to be satisfied in ideal models (such as modelling underlying hash functions as random oracles), or which are not justified at all by known security results. We describe several cases of deployed protocols where this is the case, focussing on the use of HKDF as a dual-PRF. Unfortunately, such strong assumptions on cryptographic hash functions tend not to withstand the test of time, often leading to deployed systems that eventually become completely insecure; experience also shows that upgrading already-deployed cryptography from deprecated hash functions to newer ones is a slow process that can take many years. Finally, we consider it sub-optimal that the new hybrid key exchange protocols combining classical and post-quantum key exchanges and that are in the process of development risk more deeply embedding the improper use of key combiners.
In this work, we narrow the gap between theory and practice for key combiners. In particular, we give a construction of a dual-PRF that can be used as a drop-in replacement for current heuristic key combiners in a range of protocols. Our construction follows a theoretical construction by Bellare and Lysyanskaya, and is based on concrete hardness assumptions, phrased in the spirit of one-wayness. Therefore, our construction provides security unless extremely strong attacks against the underlying cryptographic hash function are discovered. Moreover, since these assumptions are considered post-quantum secure, our constructions can safely be used in new hybrid protocols. From a practical perspective, our dual-PRF construction is highly efficient, adding only a negligible overhead of a few microseconds compared to currently used (heuristic) approaches. We believe that our approach exemplifies a perfect middle-ground for practically efficient constructions that are supported by realistic hardness assumptions.

Keywords: HKDF; Key Combiners; Hybrid Key Exchange.

1 Introduction

In modern cryptographic protocols, a paradigm has been established: users jointly execute a key exchange protocol and use the output shared secret keys in some arbitrary symmetric key protocol in order to achieve some cryptographic goal (e.g. building a secure channel). In practice, key exchange is usually based on Diffie-Hellman key agreement, and this is the basis of commonly used key exchange protocols on the Internet, such as TLS 1.3 and the Signal Double Ratchet protocols.

Significant work is now underway to elevate these important protocols to achieve *post-quantum security*, i.e., to protect against attackers with quantum computers, by including post-quantum cryptographic primitives. When extending these protocols, it is common to consider a hybrid approach, combining keys output

from classical key agreement protocols with post-quantum key agreement protocols and using these keys in symmetric-key primitives. Draft standards exist supporting this approach [38, 10] and there is a growing body of theoretical work on it [7, 14, 37]. Large-scale experiments have already been conducted to study the performance impact of taking such a hybrid approach in TLS [27, 26]. Similarly, ratcheted key exchange protocols, such as Signal, continuously derive new keying material, and combine with existing secret state to achieve strong notions such as forward secrecy and post-compromise security. In both settings – hybrid protocols and ratcheted key exchange – combining multiple secrets is done through the use of key derivation functions (KDFs).

In addition, key agreement primitives like Diffie-Hellman do not necessarily output uniformly random keys, which is required of the most common symmetric-key primitives. If the keying material output by the key agreement primitive is not uniformly random, first extraction must occur. This is the approach taken by HKDF [24], a hash-based KDF that is modularly divided into *extract* and *expand* phases, where the extract phase acts as a computational extractor, and the expand phase successively generates new keying material from the extracted entropy. HKDF is very widely used, including in TLS and Signal.

However, the use of a computational extractor as a key combiner is not necessarily a clean fit. In particular, we highlight that TLS, Signal, and recent ETSI[6] Hybrid Key Exchange protocols use computational extractors like HKDF.extract in heuristic ways that do not fit the threat model that it was analyzed under. New proposals for hybrid key exchange are currently being considered for standardization, and experience shows that once standardized, constructions are often used for decades. Therefore, it is of utmost importance that we fully understand the security guarantees of primitives like HKDF, and whether they are being used correctly (and indeed actually suitable for use at all) in the new standards that are under development, especially complex ones involving hybrid approaches. This brings us to the central questions that we address in this paper:

What are the security definitions required by key-combiners as they are currently used in practice, and do they satisfy them? Can we design an efficient and practical key-combiner that satisfies these security definitions under standard assumptions and that can be used as a drop-in replacement for the current heuristic approaches?

1.1 Our Contributions

To answer the first question, we being with a survey of several works that provide security proofs for TLS, Signal, and ETSI protocols. We show that these protocols use HKDF.extract as a key-combiner, and their security proofs require it to be modeled as a dual-PRF. However, to the best of our knowledge, HKDF.extract was never proven as a dual-PRF under standard assumptions. We also examine the proposal of [38]. This proposal does not come with formal proofs, but we use our previous protocol studies to infer what would be needed to prove its security. We highlight that it has several unanticipated weaknesses arising from the possibility of an attacker being able to control classical inputs to the key derivation steps in a post-quantum setting, specifically in situations where the collision-resistance of HKDF.extract is assumed to be broken.

Given this problematic picture, we then propose a concrete construction of a dual-PRF based on concrete hardness assumptions which are phrased in the spirit of one-wayness. The proposed construction follows a theoretical one from Bellare and Lysyanskaya [5], but requires significantly weaker assumptions. Further, we give a concrete construction, with concrete choices for underlying cryptographic components, along with a reference implementation. Since we use only assumptions and primitives that are considered quantum-secure, our results lift to a quantum setting.

Starting from the theoretical side, our core construction makes use of the following components: g is an injective one way function; F is a computational-extractor with respect to g; PRF is a standard PRF; H is an ϵ-regular function (all notions are defined in the main body). Then, the following procedure defines the operation of our dual-PRF on a pair of inputs K_1, K_2:

[6] ETSI is an independent, not-for-profit, standardization organization in the field of information and communications (see https://www.etsi.org).

1. $k_1 \leftarrow F(K_1)$, $k_2 \leftarrow F(K_2)$.
2. $u_1 \leftarrow g(K_1)$, $u_2 \leftarrow g(K_2)$.
3. $Y \leftarrow H(\mathsf{PRF}(k_1, 2||u_2) \oplus \mathsf{PRF}(k_2, 1||u_1))$.
4. Output Y.

Here, the intuition is that even if K_2 is adversarially chosen, the output of $\mathsf{PRF}(k_1, 2||u_2)$ will still be uniformly random as long as u_2 is unique. The same goes for adversarially chosen $K1$ and $\mathsf{PRF}(k_1, 2||u_2)$. As g is injective it is not possible to find two inputs that will collide on u_1 or on u_2.

We then show how this theoretical construction can be efficiently instantiated using the same cryptographic primitives as are used in the current heuristic HKDF.extract based solutions. We complete the paper by showing how our efficient dual-PRF can be used as a drop-in replacement key-combiner in TLS 1.3.

To summarize, our main contributions are:

- A study of existing security proofs for a range of key exchange protocols, highlighting how the assumptions they make concerning HKDF.extract are not fully supported by proven properties of HKDF.extract.
- The development of a suitable dual-PRF, proven secure under mild assumptions on its cryptographic components.
- An analysis of the implementation efficiency of this dual-PRF, showing that it comes with only small overhead compared to HKDF.extract.
- An explanation of how the proposed dual-PRF can be smoothly integrated into the TLS 1.3 handshake, focussing on its use to safely combine secrets coming from classical and post-quantum key exchanges.

Paper organization. In Section 2 we describe HKDF and HMAC, survey existing analysis in the literature, and point to problematic uses of these functions in deployed protocols. In Section 3 we give two constructions of *injective* one-way functions from realistic assumptions. This building block is then used in Section 4, where we present and formally analyze our suggested dual-PRF. In Section 5 we explain how to best instantiate the dual-PRF construction in practice, and provide implementation benchmarks. In Section 6 we show how to use the new construction in the TLS 1.3 key schedule. And lastly, in Section 7 we survey related work.

2 HKDF And Its Uses

Here, we describe HKDF, first by explaining its construction from underlying cryptographic primitives, and then surveying literature analyzing the HKDF and HMAC primitives. We then turn to the use of HKDF in real-world protocols, such as TLS 1.3 and the Signal protocol. We highlight the assumptions are made of HKDF when analysing these protocols, and how they do not match the existing literature. Finally, we examine in detail a recent IETF draft [38] that provides a construction for hybrid key exchange in the TLS 1.3 protocol, discussing various scenarios that are barriers for an analysis of this construction, including strong assumptions necessary to prove the construction's security.

2.1 HKDF – Construction and Analyses

HKDF is separated into two phases, HKDF.Extract and HKDF.Expand.

HKDF.Extract takes two inputs $(salt, ikm)$ and outputs a pseudorandom key prk. Here, $salt$ is either a uniformly random (but not secret) value or a constant, ikm is (secret) input keying material that has sufficient entropy (but not necessarily uniformly random) and prk is a secret, pseudorandom key that is indistinguishable from a uniformly random value (under certain assumptions, discussed below in detail).

In the expand phase, HKDF.Expand takes three inputs $(prk, info, L)$ and outputs keying material okm. Here prk is a pseudorandom key, $info$ is an optional context string, L is the length of the output keying material, and okm is some pseudorandom keying material of length L that is indistinguishable from a uniformly random value of the same length (again, under certain assumptions).

In this work, we focus on the extraction phase of HKDF, its use as a key combiner and computational extractor, and the assumptions made of the underlying cryptographic primitives. HKDF.Extract is built upon

HMAC; specifically, HKDF.Extract$(salt, ikm)$:= HMAC$(salt, ikm)$. Notice how the value *salt* here is used as a key to HMAC.

Since HKDF.Extract is essentially equivalent to HMAC, we now briefly focus the discussion on HMAC before returning to HKDF.

HMAC. HMAC is a pseudorandom function (PRF) family with:

$$\mathsf{HMAC}(K, m) := \mathsf{HASH}((K' \oplus opad) \| \mathsf{HASH}((K' \oplus ipad) \| m)),$$

where HASH is a cryptographic hash function with block size L, m is some message, $K' = \mathsf{HASH}(K)$ if $|K| > L$ and $K' = K$ otherwise, *opad* is the byte-string $\texttt{0x5C}_L$, and *ipad* is the byte-string $\texttt{0x36}_L$. Thus, HKDF.Extract is constructed as:

$$\mathsf{HKDF.Extract}(salt, ikm) := \mathsf{HASH}((salt' \oplus opad) \| \mathsf{HASH}((salt' \oplus ipad) \| ikm)).$$

The original proof of HMAC (due to Bellare, Canetti, and Krawczyk [4]) relied on the assumption that the hash function HASH was weakly collision-resistant (and also that the underlying compression function was a secure PRF). But, as Bellare highlighted [3], attacks on MD5 and SHA-1 demonstrated that these hash functions were not collision resistant. Widespread implementation of MD5 and SHA-1 meant that deprecating these primitives is a long, slow process that is still ongoing. Bellare improved upon the original security results for HMAC by providing a second proof of HMAC as a PRF, which only requires that the inner compression function h of the underlying hash function meet some PRF-like conditions. Specifically, the proof requires that h be a standard PRF, and its dual function $\overline{h}(message\_block, IV) = h(IV, message\_block)$ be a PRF under related-key attack, for a small and specific class of related keys. These assumptions are considered mild, to the extent that even broken hash functions like MD5 likely meet them. Therefore, HMAC is likely a PRF even when instantiated using MD5; indeed, no concrete attack has been published against HMAC using MD5. Notice, however, that this result applies for HMAC when used with a random key (whereas, as we already highlighted above, HKDF.Extract uses a salt as this input).

Modeling HMAC or HKDF as a dual-PRF. Generally speaking, HMAC is not provably a dual-PRF. Bellare [4] does assume that the *inner compression function* h of the hash function is a dual-PRF, but HMAC itself was not shown or assumed to be a dual-PRF in his work. We are not aware of any published proof that claims that HMAC as a whole is a dual-PRF under standard assumptions. Obviously, the same applies to HKDF.Extract.

Despite this, as we describe below, several works model HMAC, or equivalently HKDF.Extract, as a dual PRF.

Use and Analysis of HKDF in deployed protocols. In the remainder of this section, we describe the use of HKDF in practice and how it deviates from the proposed usage analysed by Krawczyk [25] in various real-world cryptographic protocols. In addition, we also highlight how various analyses of these protocols model HKDF: in particular, we look at the TLS 1.3 handshake protocol and its analysis by Dowling *et al.* [13], and the Signal Double Ratchet protocol and its analysis by Cohn-Gordon *et al.* [11]. Further, we look at the "Hybrid Key Exchange in TLS 1.3" proposed standard, likely to be deployed soon, and describe several attacks against it which arise in situations where the adversary is able to control some of the inputs to HKDF (for example, a classical Diffie-Hellman shared key which may be known to a quantum adversary). For each protocol, we give an overview of the exact assumptions used, leaving some details for Appendix A. Additionally in Appendix A, we examine the ETSI Hybrid Key Exchange protocols and their analysis by Campagna and Petcher [10].

2.2 HKDF in TLS 1.3

We begin by describing the TLS 1.3 key schedule, taken verbatim from Dowling [13] in Figure 2.2. Note that we focus here on the "cryptographic backbone" of the TLS key schedule - all other values (for instance, traffic keys) can be derived from the ES, HS or MS values by a party that knows the protocol transcript.

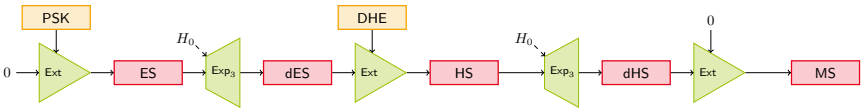

Fig. 1. The TLS 1.3 key schedule backbone. Input secrets PSK and DHE denote the preshared key and the Diffie-Hellman secret output, respectively - if they are not included in a given handshake variant, they are instead replaced with a zero-string. Ext denotes HKDF.Extract$(salt, ikm)$, where the left-hand input is the *salt* and the top input is the input keying material *ikm*. Exp denotes HKDF.Expand$(k, \texttt{Label} \| H_0)$, where the Label is a unique string for each derived value and H_0 is a hash of the zero-string.

HKDF as Modelled in Analyses of TLS 1.3 When analysing the full handshake, Dowling *et al.* describe a specific game hop where they replace the computation of $\mathsf{HS} \leftarrow \mathsf{HKDF.Extract}(\mathsf{dHS}, \mathsf{DHE})$ in the proof with a value HS', sampled uniformly at random from the output distribution of HKDF.Extract. Then, they argue that they can turn any algorithm capable of distinguishing this change into an adversary against the dual-snPRF-ODH security of the HKDF.Extract function, functionally modelling HKDF.Extract as a dual PRF. However, as we mentioned previously, this does not match any known proof of HKDF.Extract that can be found publicly. Full details of the assumption used in the analysis of TLS 1.3 Handshake protocol can be found in Appendix A.

In addition, Brendel *et al.* [9] demonstrate via algebraic reductions that it is unlikely that any variant of the PRF-ODH assumption can be proved under standard model assumptions, instead relying on modelling the PRF as a random oracle: this implies that proving HKDF.Extract can function as the underlying PRF in the dual-PRF-ODH assumption requires a random oracle assumption, which our construction avoids entirely.

HKDF as Used in TLS 1.3 The specification of TLS 1.3 [34] describes HKDF.Extract as a PRF, but is used throughout the protocol execution as a secret combiner or dual PRF, sometimes taking secret input from the *salt* input, or sometimes taking secret input from the *ikm* input. Recall that $\mathsf{HKDF.Extract}(salt, ikm) \rightarrow k$ takes a uniformly random *salt* (that does not need to be secret) and some secret keying material *ikm* (that may not be uniformly random) and outputs a uniformly random value *prk*. We highlight that combining the PSK and the DHE may not be safe if PSK is a value that has been generated by the adversary. This property is expected from the *ikm* input (which PSK is used as) in the analysis of HKDF.Extract when proving it as a computational extractor [25]. We note that by the TLS 1.3 standard [34], PSK can be established either in a previous TLS 1.3 connection, or in some out-of-band mechanism. The TLS 1.3 standard notes that "while using an out-of-band provisioned pre-shared secret, a critical consideration is using sufficient entropy during key generation," which implies non-uniform but sufficient entropy PSKs would be acceptable. Further, this assertion is made explicitly in the Guidance for External PSK Usage in TLS Internet Draft [20]: "... if a high entropy PSK is used, then PSK-only key establishment modes provide expected security properties for TLS...".

This matches the usage of PSK in generating the early secret ES (if the attacker did not control PSK, since *ikm* may be non-uniform with sufficient entropy), but misses a crucial point: if an adversary generates PSK, then it also controls the derivation of $\mathsf{dES} = \mathsf{HKDF.Expand}(\mathsf{HKDF.Extract}(0, \mathsf{PSK}), \texttt{Label} \| \mathsf{H}(0))$. The analysis of HKDF requires that *salt* is not controlled by an attacker, and thus the derivation of the handshake secret $\mathsf{HS} = \mathsf{HKDF.Extract}(\mathsf{dES}, \mathsf{DHE})$ does not match the requirements of the *salt* expected in the analysis of HKDF.Extract.

To address these concerns, in Section 4 we describe a dual PRF construction, and then in Section 6 we then propose a modified TLS 1.3 key schedule that achieves security even in these strong settings.

2.3 HKDF in Signal's Double Ratchet Protocol

We begin by describing the use of HKDF in the analysis of Signal's Double Ratchet protocol, and how it deviates from the proposed usage analysed by Krawczyk [25]. On a high-level, in Signal's asymmetric

ratchet users share some secret state (known as the root key rk_i), and the responder has a Diffie-Hellman public keyshare that the initiator knows the secret exponent of, which is used to initialise the protocol. The users then exchange Diffie-Hellman public keyshares (in a ping-pong fashion, see Figure 2.3) to successively generate new Diffie-Hellman secret values, which are combined with the previous root key rk_i to derive the new root key rk_{i+1}.

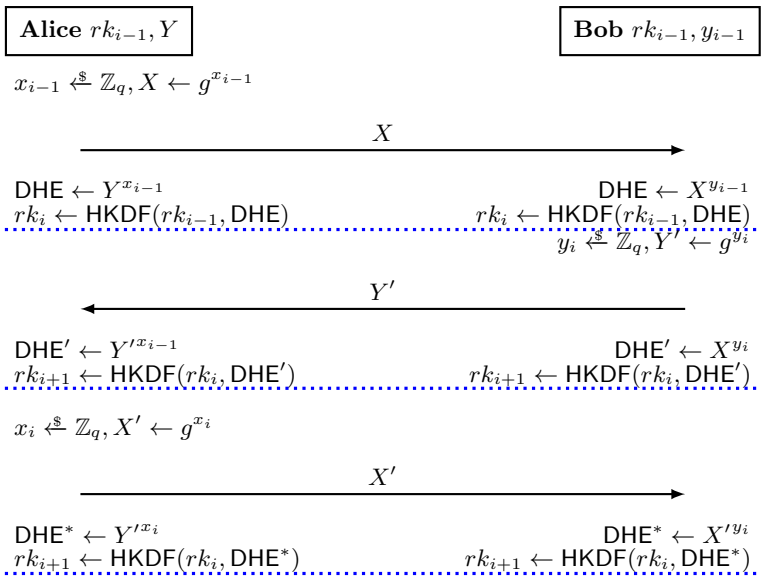

Fig. 2. A simplified depiction of the Signal Asymmetric Ratchet. Note that we have omitted details of further chain and message key derivation, focusing on the derivation of the root keys rk_i. Input secrets rk_{i-1}, y_{i-1} and Y denote the previous root key, Bob's previous Diffie-Hellman secret exponent and Bob's previous Diffie-Hellman public keyshare. HKDF denotes the full execution of HKDF, i.e. $\mathsf{HKDF.Expand}(\mathsf{HKDF.Extract}(rk_i, \mathsf{DHE}), \texttt{const})$, where `const` is some constant value.

The Double Ratchet protocol specification [31] recommends the use of HKDF as a key derivation function, so when deriving a new root key rk_i from the previous root key rk_{i-1} and a new Diffie-Hellman secret computation $Y^{x_{i-1}}$, the first step would be $tmp \leftarrow \mathsf{HKDF.Extract}(rk_{i-1}, Y^{x_{i-1}})$, the output of which is used as the keying input to HKDF.Expand. It can be seen that Signal, like TLS 1.3. is implicitly using HKDF.Extract as a secret combiner, or dual PRF.

HKDF as Modelled in Analyses of Signal Here we present the assumptions used to model HKDF in the recent computational analysis [11] of the Signal protocol. Similarly to the analysis of TLS 1.3, Cohn-Gordon *et al.* demonstrate that under certain assumptions of the underlying cryptographic primitives (including HKDF), that Signal's key exchange achieves key-indistinguishability. During the proof, the authors use a range of PRF-ODH assumptions to achieve an analysis of Signal in the standard model (see Appendix A for an example of a PRF-ODH assumption). This approach, much like the analysis of TLS 1.3's handshake protocol, implicitly models HKDF as a dual PRF. It is worth noting here that their previous analysis of the Signal protocols eschewed this approach, preferring to model HKDF as a random oracle, a much stronger assumption.

HKDF as Used in Signal Note that the specification assumes that both parties have been established with some preshared secret root key rk_1. The DR section "Security Considerations" [31] states that the DR protocol "is designed to recover security against a passive eavesdropper who observes encrypted messages

after compromising one (or both) of the parties to a session." Thus, an attacker that establishes a non-uniform root key is outside the proposed threat model.

Regardless, it is clear to see that the Signal asymmetric ratchet protocol is not robust against an attacker that can cause the initial root secret to be non-uniform if using HKDF, since the KDF guarantees of HKDF.Extract do not cover this scenario. A problem that is more relevant to the proposed threat model and the guarantees of HKDF.Extract is the dependence of secret values into the HKDF.Extract input. From the Krawczyk analysis of HKDF [25]:

> *"More significantly, however, is the need for independence between the source Σ and the salt value used by the KDF. Allowing the attacker to influence Σ after seeing the salt may result in a completely insecure KDF."*

After an attacker compromises the root key rk_i, it is able to inject Diffie-Hellman public values between the two parties, and thus the guarantees of the KDF no longer hold. It may be possible for the attacker to cause the next update to the root key rk_{i+1} to become non-uniform, and thus break the guarantees for the KDF chain in the future, after the attacker has become passive once again.

2.4 HKDF in Hybrid Key Exchange in TLS 1.3

In this section we apply our observations to case of the IETF informational standard "Hybrid Key Exchange in TLS 1.3" [38]. The draft is concerned primarily on combining secrets derived from standard TLS 1.3 Diffie-Hellman key exchange with post-quantum key exchange mechanisms. We begin with a description of the proposed draft, and depict the proposed key schedule. Note that for ease of understanding, we have depicted HKDF Extract operations down into its component HMAC calls.

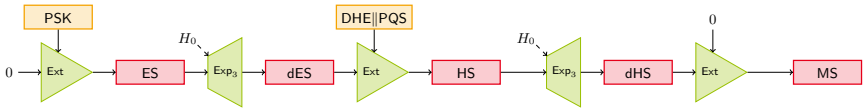

Fig. 3. The proposed Hybrid TLS 1.3 key schedule backbone. Input secrets PSK, PQS and DHE denote the preshared key, the post-quantum secret and the Diffie-Hellman secret output, respectively - if they are not included in a given handshake variant, they are instead replaced with a zero-string. Ext denotes HKDF.Extract$(salt, ikm)$, where the left-hand input is the *salt* and the top input is the input keying material ikm. Exp denotes HKDF.Expand$(k, \texttt{Label} \| H_0)$, where the `Label` is a unique string for each derived value and H_0 is a hash of the zero-string.

Again, we focus on the cryptographic backbone of the proposed key schedule. We note that as opposed to including PQS as a key input in the final HKDF.Extract step, PQS is instead concatenated with DHE and given as keyed input to the second HKDF.Extract step. Since PQS may be the output of a KEM, it may be that an attacker may not know the output of DHE, but can control the output of PQS. In particular, HKDF.Extract is not proven to provide security in this setting under standard assumptions - we theorise that it would be necessary to model HKDF.Extract as a random oracle to achieve security as a result. Finally, much like standard TLS 1.3, we believe that even if we assume that the attacker does not control PQS or DHE, a dual PRF assumption would be required to prove security in the proposed key schedule, which again is not proven for HKDF.Extract.

To address these problems, after we describe a dual PRF construction in Section 4, we then propose a Hybrid TLS 1.3 key schedule (see Section 6) that achieves security even in these strong settings, and under weaker assumptions, completing the circle on the use of dual PRFs in TLS 1.3 and Hybrid TLS 1.3.

Making an extractor robust against an attacker's ability to influence the source of randomness Σ is known as ϵ-resilient extractors. In particular, it is not an aspect of HKDF Extract that is proven.

Data: $\mathsf{PSK}, \mathsf{DHE}$
Result: MS

$\mathsf{ES} = \mathsf{HMAC}(0, \mathsf{PSK})$;
$\mathsf{dES} = \mathsf{HMAC}(\mathsf{ES}, \mathtt{Label}_{\mathsf{dES}} \| \mathsf{HASH}(0))$;
$\mathsf{HS} = \mathsf{HMAC}(\mathsf{dES}, \mathsf{DHE})$;
$\mathsf{dHS} = \mathsf{HMAC}(\mathsf{HS}, \mathtt{Label}_{\mathsf{dHS}} \| \mathsf{HASH}(0))$;
$\mathsf{MS} = \mathsf{HMAC}(\mathsf{dHS}, 0)$;

Data: $\mathsf{PSK}, \mathsf{DHE}, \mathsf{PQS}$
Result: MS

$\mathsf{ES} = \mathsf{HMAC}(0, \mathsf{PSK})$;
$\mathsf{dES} = \mathsf{HMAC}(\mathsf{ES}, \mathtt{Label}_{\mathsf{dES}} \| \mathsf{HASH}(0))$;
$\mathsf{HS} = \mathsf{HMAC}(\mathsf{dES}, \mathsf{DHE} \| \mathsf{PQS}$;
$\mathsf{dHS} = \mathsf{HMAC}(\mathsf{HS}, \mathtt{Label}_{\mathsf{dHS}} \| \mathsf{HASH}(0))$;
$\mathsf{MS} = \mathsf{HMAC}(\mathsf{dHS}, 0)$;

Fig. 4. On the left we depict the "standard" TLS 1.3 key schedule, and on the right we depict the proposed Hybrid TLS 1.3 key schedule. Note that we use 0 to denote a zero-string of L_H bits, where L_H is the output length of the underlying hash function.

3 Two Forms of One-Wayness and Injectivity

We present two types of assumptions, closer in spirit to one-wayness, and prove that they imply functions with desired properties: *one-wayness* and *injectivity*. One notion that we consider is that of exponentially strong one-way functions. Here, we consider one-way functions where the guessing chance of any attacker is exponentially small (i.e., almost the best possible). We observe that such functions can be efficiently converted into one-way functions (which are still exponentially secure) that are also *injective*. The construction has the following form:

$$\mathsf{inj}F(x) = F(h(x)),$$

where F is strong enough one-way function and h is a simple combinatorial hash function (i.e., 2 universal). The downside of this construction is that the security reduction is rather costly in practice, and therefore the resulting parameters are not that great. (We provide numerical examples below.)

This motivates our second approach. This approach asserts that one-wayness holds even if multiple evaluations (on the same input) are given, i.e., several applications of one-way functions on identical inputs behaves like a one-way function. At the same time, we need the corresponding family of functions to be 2-universal (namely any two distinct inputs do not collide on a randomly chosen function from the family). We show how to turn such a family (satisfying both properties), to a one way function which is injective with high probability. The construction has the following form:

$$\mathsf{inj}F(x) = F_1(x) \| \ldots \| F_k(x),$$

where the F_i's come from a family of functions which are one-way given same-input evaluations of other functions in the family and at the same time they satisfy a simple combinatorial property (i.e., being 2 universal). The advantage of this approach is that there is essentially no loss in parameters, although the assumptions are somewhat less standard. Still, we believe that it serves as a reasonable trade-off between assumptions and practicality.

We start by presenting the more practically-efficient approach (i.e., the second approach) and then present the one based on more standard assumptions but less efficient (i.e., the first approach).

3.1 A practically-efficient approach

We first give the necessary definitions and then state the result.

Definition 1 (2-universal). *Let $\mathcal{F} = \{f\colon \{0,1\}^n \to \{0,1\}^m\}$ be a family of functions. We say that $\mathcal{F}$ is ϵ-2-universal if for every distinct $x_1, x_2 \in \{0,1\}^n$, it holds that $\Pr_{f \leftarrow F}[f(x_1) = f(x_2)] \leq \epsilon$.*

Definition 2 (Same-input one-way). *Let $\mathcal{F} = \{f\colon \{0,1\}^n \to \{0,1\}^m\}$ be a family of functions. We say that $\mathcal{F}$ is (s, δ)-same-input one-way if for every size-s circuit $\mathcal{A}$, it holds that*

$$\Pr[\forall i \in [k]\colon f_i(x') = f_i(x)] \leq \delta(\lambda),$$

where the probability is over the choice of $f_1, \ldots, f_k \leftarrow \mathcal{F}$ and over $x \leftarrow \{0,1\}^n$, and where x' is the output of A given $(f_1(x), \ldots, f_k(x), f_1, \ldots, f_k)$.

The above definition, when $k = 1$, is just the usual definition of (s, δ)-one-way functions. Note that this definition is very related to previously defined notions, e.g., [35, 18]. In particular, Hemenway, Lu, Ostrovsky [18] show that Definition 2 is existentially equivalent to the existence of one-way functions. However, the novel aspect of our work is that we will need the same family of functions to be 2-universal (Definition 1) at the same time.

We prove that for a natural range of parameters the function that just concatenates the outputs of sufficiently many randomly sampled f's must be injective.

Lemma 1. *Assume that there is a family $\mathcal{F} = \{f \colon \{0,1\}^n \to \{0,1\}^m\}$ which is both ϵ-2-universal and (s, δ)-same-input one-way. Then, the function family that contains functions of the form $h \colon \{0,1\}^n \to \{0,1\}^{mk}$ such that $h(x) = f_1(x) \| \ldots \| f_k(x)$ is an* injective *(s, δ)-one-way function with probability $1 - 2^{2n} \cdot \epsilon(n)^k$ (over the choice of $f_1, \ldots, f_k \leftarrow \mathcal{F}$).*

Proof. First, observe that the (s, δ)-one-wayness of h follows directly from the same-input product one-wayness of $\mathcal{F}$. Therefore, we only need to prove that h is injective with high probability.

Let N be a random variable corresponding to the number of distinct pairs (x_1, x_2) that collide relative to h. By Markov's inequality,

$$\Pr_{f_1,\ldots,f_k \leftarrow \mathcal{F}}[h \text{ is not injective}] = \Pr_{f_1,\ldots,f_k \leftarrow \mathcal{F}}[N \geq 1] \leq \mathop{\mathbb{E}}_{f_1,\ldots,f_k \leftarrow \mathcal{F}}[N].$$

By definition of h,

$$\mathop{\mathbb{E}}_{f_1,\ldots,f_k \leftarrow \mathcal{F}}[N] = \sum_{x_1,x_2 \in \{0,1\}^n} \mathop{\mathbb{E}}_{f_1,\ldots,f_k \leftarrow \mathcal{F}}[\forall i \in [k] \colon f_i(x_1) = f_i(x_2)].$$

Since the f_i's are chosen independently and they are coming from a pairwise family, for every $x_1, x_2 \in \{0,1\}^n$, it holds that

$$\begin{aligned} \mathop{\mathbb{E}}_{f_1,\ldots,f_k \leftarrow \mathcal{F}}[\forall i \in [k] \colon f_i(x_1) = f_i(x_2)] &\leq \left(\Pr_{f \leftarrow \mathcal{F}}[f(x_1) = f(x_2)] \right)^k \\ &\leq \epsilon(n)^k. \end{aligned}$$

Therefore,

$$\Pr_{f_1,\ldots,f_k \leftarrow \mathcal{F}}[h \text{ is not injective}] \leq 2^{2n} \cdot \epsilon(n)^k.$$

3.2 A foundational approach

In this section we show that an extreme setting of parameters can be used to turn any one-way function into an injective one.

Lemma 2. *Assume that $f \colon \{0,1\}^\lambda \to \{0,1\}^m$ is (s, δ)-one-way function and $\mathcal{G} = \{g \colon \{0,1\}^n \to \{0,1\}^\lambda\}$ is a pairwise hash function. The function*

$$h(\cdot) = f(g(\cdot))$$

is an injective *(s', δ')-one-way function with probability $1 - 2^{2n}\delta(\lambda)$ (over the choice of $g \leftarrow \mathcal{G}$). Further, $s' = s - O(\lambda)$ and $\delta'(n) = 2^{\lambda - n}\delta(\lambda) + 2^{2n-\lambda}$.*

Setting of parameters. We want to get a function which is both injective with very high probability and also (strong) one-way. Set $\lambda = 7n/2$ and $\delta(\lambda) = 2^{-6\lambda/7} = 2^{-3n}$. Start with an (s, δ)-one-way function f, we get that the function h is injective with probability $1-2^{-n}$ and it is (s', δ')-one-way for $s' \approx s$ and $\delta'(n) = 2^{-n/2}$.

Proof (of Lemma 2). We first show that h is injective with high probability, following the high level idea of Lemma 1. For every $x_1 \neq x_2$, the pairwise independence of g implies that

$$\Pr_{g \leftarrow \mathcal{G}}[f(g(x_1)) = f(g(x_2))] = \Pr_{w_1, w_2 \leftarrow \{0,1\}^\lambda}[f(w_1) = f(w_2)].$$

Let N be a random variable corresponding to the number of distinct pairs (x_1, x_2) that collide relative to h. By Markov's inequality,

$$\begin{aligned}\Pr_{g \leftarrow \mathcal{G}}[h \text{ is not injective}] &= \Pr_{g \leftarrow \mathcal{G}}[N \geq 1] \leq \mathop{\mathbb{E}}_{g \leftarrow \mathcal{G}}[N] \\ &= \binom{2^n}{2} \Pr_{w_1, w_2 \leftarrow \{0,1\}^\lambda}[f(w_1) = f(w_2)].\end{aligned}$$

Since f is strong one-way, it holds that $\Pr_{X \leftarrow \{0,1\}^\lambda}[f(X') = f(X)] \leq \delta(\lambda)$. In particular, it must be that $\Pr_{w_1, w_2 \leftarrow \{0,1\}^\lambda}[f(w_1) = f(w_2)] \leq \delta(\lambda)$ (otherwise there is a non trivial way to invert). Thus,

$$\Pr_{g \leftarrow \mathcal{G}}[h \text{ is not injective}] \leq \binom{2^n}{n} \delta(\lambda).$$

We proceed with the proof of one-wayness of h. First, note that by a similar calculation,

$$\Pr_{g \leftarrow \mathcal{G}}[g \text{ is not injective}] \leq 2^{2n-\lambda}.$$

Now, assume that there is a size $s' = s'(\lambda)$ inverter $\mathcal{A}$ for h that succeeds with probability $\delta'(\lambda)$. The procedure $\mathcal{A}$ gets as input y and g such that g is sampled from the pairwise hash function family and y is computed by first sampling $x \in \{0,1\}^n$, then computing $w = g(x)$, and finally $y = f(w)$. We define $\mathcal{B}$ that inverts f and works as follows. It gets y as input, sampled by choosing $w \in \{0,1\}^\lambda$ and then computing $f(w)$, then it sampled $g \leftarrow \mathcal{G}$ and submits to $\mathcal{A}$ the pair y, g. Upon reply x' from $\mathcal{A}$, it computes $w = g(x')$ and outputs this value. The running time of $\mathcal{B}$ is bounded by the running time of $\mathcal{A}$ plus sampling and evaluating g exactly once which can be done in time $O(\lambda)$.

Conditioned on g being injective, the inverter $\mathcal{B}$ succeeds in inverting f at least when $f^{-1}(x) \cap \mathsf{Im}(g) \neq \emptyset$ and $\mathcal{A}$ succeeds. Since g is injective its image size is 2^n but its range is 2^λ, and so the aforementioned intersection is non-empty with probability least $2^n/2^\lambda$. This means that the probability that $\mathcal{B}$ succeeds in inverting f is at least $2^{\lambda-n} \cdot \delta(\lambda) + 2^{2n-\lambda}$.

Remark 1. The construction and proof above can be viewed as a variant of a step in the result of [19] showing how to use a strong variant of one-wayness to get collision resistance. Here, we only require injective one-wayness and we achieve it from a weaker variant of one-wayness than [19].

4 Our dual-PRF Construction

In this section, we describe our new theoretical dual-PRF construction and prove its security.

4.1 Preliminary Definitions

We recall the standard definition of pseudorandom functions [17] and dual pseudorandom functions [3, 5].

Definition 3 (Pseudorandom function (PRF)). *Let $\mathcal{F} = \{f_\lambda : K_\lambda \times X_\lambda \to Y_\lambda\}_{\lambda\in\mathbb{N}}$ be a function family ensemble and let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda\in\mathbb{N}} = \{g_\lambda : X_\lambda \to Y_\lambda\}_{\lambda\in\mathbb{N}}$ be the set of all functions mapping X_λ to Y_λ. We say that $\mathcal{F}$ is a (t,ϵ)-pseudorandom function family (PRF) if no t-size adversary can distinguish $f_\lambda(k,\cdot)$ from $g_\lambda(\cdot)$ for k chosen uniformly from the set K_λ and where g_λ is chosen uniformly from $\mathcal{G}_\lambda$ with probability better than ϵ. More precisely, for an adversary $A = \{A_\lambda\}_{\lambda\in\mathbb{N}}$, we define*

$$Adv^{prf}_{f,A}(\lambda) = \Pr_{k\leftarrow K_\lambda}[A_\lambda^{f_\lambda(k,\cdot)}(1^\lambda) = 1] - \Pr_{g_\lambda\leftarrow\mathcal{G}_\lambda}[A_\lambda^{g_\lambda(\cdot)}(1^\lambda) = 1].$$

The family $\mathcal{F}$ is a (t,ϵ)-PRF if for every size-s adversary A it holds that $Adv^{prf}_{f,A}(\lambda) \le \epsilon(\lambda)$.

A dual PRF is a two-input function which is a PRF when either one of them is viewed as the key [3, 5].

Definition 4 (Dual PRF). *Let $\mathcal{F} = \{f_\lambda : X^0_\lambda \times X^1_\lambda \to Y_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble of functions. Let $\bar{\mathcal{F}} = \{\bar{f}_\lambda : X^1_\lambda \times X^0_\lambda \to Y_\lambda\}_{\lambda\in\mathbb{N}}$ be the ensemble of dual functions such that each $\bar{f} \in \bar{\mathcal{F}}$ is defined as $\bar{f}(y,x) = f(x,y)$. We say $\mathcal{F}$ is a (t,ϵ)-*dual *PRF if both $\mathcal{F}$ and $\bar{\mathcal{F}}$ are (t,ϵ)-PRFs.*

Our construction uses a PRF that is secure even when the adversary is given an image of a one-way function of the key.

Definition 5 (Computational extractor). *We say that a function $F\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a (t,ϵ)-computational-extractor with respect to a function $g\colon \{0,1\}^\lambda \to \{0,1\}^m$ if for any circuit A of size at most t it holds that*

$$|\Pr_K[A(g(K), F(K))] - \Pr_{K,R}[A(g(K), R)]| \le \epsilon \ ,$$

where R and K are sampled uniformly over $\{0,1\}^\lambda$.

Definition 6 (Statistical Distance). *The statistical distance between two random variables X, Y is defined by*

$$\Delta(X,Y) := \frac{1}{2}\cdot\sum_x |\Pr[X = x] - \Pr[Y = x]|$$

Definition 7 (ϵ-Regular). *Let U_ℓ be the uniform distribution over n bits. A function $H\colon \{0,1\}^n \to \{0,1\}^m$ is ϵ-regular if $\Delta(H(U_n), U_m) \le \epsilon$, where Δ is the statistical distance measure.*

4.2 Our construction

Theorem 1. *Assume that g is an injective (t,ϵ)-one way function, that F is a (t,ϵ)-computational-extractor with respect to g, that* PRF *is a standard (t,ϵ)-PRF, and that H is ϵ-regular.*

Then, the following construction is a $(t, 3\epsilon)$-dual PRF. On input K_1, K_2 compute the following

1. $k_1 \leftarrow F(K_1)$, $k_2 \leftarrow F(K_2)$.
2. $u_1 \leftarrow g(K_1)$, $u_2 \leftarrow g(K_2)$.
3. $Y \leftarrow H(\mathsf{PRF}(k_1, 2||u_2) \oplus \mathsf{PRF}(k_2, 1||u_1))$.
4. *Output Y.*

Proof. The proof of security works the same for both directions of showing dual PRF (when K_1 is uniform and K_2 is malicious and vice versa). Thus, we assume that K_1 is chosen uniformly at random and the adversary performs queries while choosing values for K_2. We give four additional hybrid constructions and then claim their indistinguishability (Construction 1 through Construction 4 below). For each construction we mark in red the difference from the previous construction. Fix some adversary A that runs in time t. Let ϵ_0 be its advantage in the original construction. Similarly, let ϵ_i be the advantage of A when given construction i. We bound the difference between ϵ_0 and ϵ_4 and prove that $\epsilon_4 = 0$.

We use the notation $x \leftarrow U$ to denote the uniform distribution over x (the length of x will be clear from the context).

Construction 1 :
1. $k_1 \leftarrow U$, $k_2 \leftarrow F(K_2)$.
2. $u_1 \leftarrow g(K_1)$, $u_2 \leftarrow g(K_2)$.
3. $Y \leftarrow H(\mathsf{PRF}(k_1, 2||u_2) \oplus \mathsf{PRF}(k_2, 1||u_1))$.
4. Output Y.

Claim. $|\epsilon_0 - \epsilon_1| \leq \epsilon$.

Proof. This follows directly from the definition of (t, ϵ)-computational extractor with respect to the function g. An adversary for g gets as input $u_1 \leftarrow g(K_1)$, and k_1, where k_1 is either random or $k_1 \leftarrow F(K_1)$. Thus, it can simulate the rest of the construction:
1. sample K_2.
2. compute $k_2 \leftarrow F(K_2)$, and $u_2 \leftarrow g(K_2)$.
3. computes $K \leftarrow H(\mathsf{PRF}(k_1, 1||u_2) \oplus \mathsf{PRF}(k_2, 1||u_1))$.

This is a perfect simulation, and thus the probability of distinguishing in the PRF game is the same as the probability of distinguishing between construction 0 and construction 1, which is at most ϵ.

Construction 2 :
1. $k_1 \leftarrow U$, $k_2 \leftarrow F(K_2)$.
2. $u_1 \leftarrow g(K_1)$, $u_2 \leftarrow g(K_2)$.
3. $Y \leftarrow H(U \oplus \mathsf{PRF}(k_2, 1||u_1))$.
4. Output Y.

Claim. $|\epsilon_1 - \epsilon_2| \leq \epsilon$.

Proof. Assume, without loss of generality, that A performs unique queries, $q_1, \ldots, q_t$. Since g is injective, we have that $y_1 = g(q_1), \ldots, y_t = g(q_t)$ are distinct. Thus, the claim follows directly from the security of PRF. Since k_1 is a uniform key that is used solely in the PRF F, and since it is always queried by distinct inputs, we have that its output is pseudorandom (with error ϵ).

Construction 3 :
1. $k_1 \leftarrow U$, $k_2 \leftarrow F(K_2)$.
2. $u_1 \leftarrow g(K_1)$, $u_2 \leftarrow g(K_2)$.
3. $Y \leftarrow H(U)$.
4. Output Y.

Claim. $|\epsilon_2 - \epsilon_3| = 0$.

Proof. Since U is completely uniform (for each query of the adversary), we have that U is exactly distributed as $U \oplus \mathsf{PRF}(k_2, 1||u_1)$.

Construction 4 :
1. $k_1 \leftarrow U$, $k_2 \leftarrow F(K_2)$.
2. $u_1 \leftarrow g(K_1)$, $u_2 \leftarrow g(K_2)$.
3. $Y \leftarrow U$.
4. Output Y.

Claim. $|\epsilon_3 - \epsilon_4| \leq \epsilon$.

Proof. This follows directly from the fact that H is an ϵ-regular.

Claim. $\epsilon_4 = 0$

Proof. This follows since this construction is in fact a completely random oracle.

Taking into account all the errors from the hybrids, we get that the advantage of A is at most $\epsilon_0 \leq 3\epsilon$.

```
Data: Expansion factor exp, input x
Result: u
x_1, x_2...x_n ← SplitToBlocks(x);
u ← Empty ;
for i from 1 to n do
    for j from 0 to exp − 1 do
        u ← u||H_j(x_i)
    end
end
```

Fig. 5. Instantiation of the one-way injective function $g(k, x)$

5 From Theory to Practice

In this section, we show how we can instantiate the theoretical dual-PRF construction described in Sections 3 and 4 to provide a practical, efficient, and provably secure key combiner. As we have discussed in Section 2.1, HKDF.extract function [25] is sometimes misused for combining two keys. Our combiner can be used as a drop-in replacement to the HKDF.extract function using only the same efficient underlying primitive, i.e., a standard hash function.

5.1 Practical One Way Injection Function

Our dual-PRF construction requires an injective one-way function g. We instantiate this function using only a standard hash function. We take the practical approach from Section 3.1. Similar to HMAC construction, we define a family of hash functions $H_1, H_2, \ldots$, by replacing the function's fixed IV. The IV is replaced by simply prepending a separate fixed public value to the input:

$$H_i(x) = \mathsf{HASH}(B_i||x)$$

Where B_i are fixed public values, and $|B_i| = \mathsf{BlockSize}$. $\mathsf{BlockSize}$ is the hash function's block size (e.g., 512 bits for SHA256 and SHA512).

We now define the function $g'(exp, x)$, where exp is the expansion factor exp defined in Section 3.1, and x is the input. We assume $|x| \leq \mathsf{BlockSize}$:

$$g'(k, x) = H_1(x)||H_2(x)||...||H_{exp-1}(x)$$

The pseudocode for our instantiation of function $g(k, x)$ is given in Fig. 5. If $|x|$ is larger than the block size of the hash function, we simply split x to blocks $x_1, x_2, .., x_n$ of length $\mathsf{BlockSize}$ each (with the last block possibly shorter). We then apply the function g' on each block separately and concatenate the outputs.

Reiterating Section 3.1, for our construction to be secure, we only need to make the following two assumptions on our hash function:

1. The resulting family of keyed hash functions is 2-universal. That is, we merely require that there is no single block collision on the inner compression function that will hold on a large fraction of the possible IVs.
2. The resulting family of keyed hash functions is same-input one-way, which means that it is hard to find a preimage even given multiple images of the compression function on the same input value but with different IVs.

We argue that these assumptions are much weaker than collision resistance for suitable values of the expansion factor exp. Moreover, we believe that they hold even for weak hash functions that are known to have single block collisions such as MD5 [39].

Data: Keys $K1, K2$ and optional $salt$ with default value of all zero bytes
Result: K

$k_1 \leftarrow \mathsf{HMAC}(salt, K_1), u_1 \leftarrow g(K_1)$;
$k_2 \leftarrow \mathsf{HMAC}(salt, K_2), u_2 \leftarrow g(K_2)$;
$K \leftarrow \mathsf{HASH}(\mathsf{HMAC}(k_1, 2||u_2) \oplus \mathsf{HMAC}(k_2, 1||u_1))$;

Fig. 6. Instantiation of KeyCombine $(K_1, K_2, salt)$

For a modern hash function such as SHA256, taking $k = 3$ causes the function to be *expanding*, from 512 bits to 756. We argue that such a choice will very likely mean that our function is computationally injective even given an efficient algorithm for finding collisions in SHA256: Since breaking injectivity under these parameters is so far outside the capabilities of current cryptanalysis, we analyze past cryptanalysis of weaker functions to arrive at this conclusion. That is, we assume that through future developments, cryptanalysis against SHA-256 achieves the same capabilities as the current state of MD5 cryptanalysis. Even in such an unlikely event, breaking injectivity would be outside of cryptographic knowledge. Taking even larger values for exp, e.g. $exp = 5$ will mean our construction is injective with overwhelming probability, while preserving its efficiency.

5.2 Practical dual-PRF

We recall the dual-PRF construction from Section 4:
1. $k_1 \leftarrow F(K_1), u_1 \leftarrow g(K_1), k_2 \leftarrow F(K_2), u_2 \leftarrow g(K_2)$.
2. $K \leftarrow H(F'(k_1, 1||u_2) \oplus F'(k_2, 2||u_1))$.
3. Output K.

We have described how to practically instantiate the one-way injective function $g(x)$ in Section 5.1 (here we omit the expansion factor exp for brevity). We will now describe instantiating the functions F, F', and $H(k)$.

F(K) is an extractor. We propose to follow the approach taken in HKDF.extract, and instantiate it with the HMAC function using an optional salt value, using the salt as key, and the input K as the message:

$$F(K, salt) = \mathsf{HMAC}(salt, K)$$

As in HKDF.extract we can set the default value of salt to all zero bytes. The approach taken by HKDF.extract, of using HMAC as a computational extractor, is well-established, and enjoys a security proof: If the compression function underlying the hash function is a good extractor, then HMAC is a good extractor [12].

$F'(K, U)$ is a PRF that can be simply instantiated with HMAC, and $H(U)$ can be implemented using the hash function:

$$F'(K, U) = \mathsf{HMAC}(K, U)$$
$$H(U) = \mathsf{HASH}(U)$$

The pseudocode for our instantiation of KeyCombine from a dual-PRF is given in Fig. 6.Our combiner receives two keys $K1, K2$, and an optional salt value and returns a combined key K that is indistinguishable from random, even if one of the keys was chosen maliciously.

Security Analysis As discussed in Section 1.1, the goal of our KeyCombine construction is to output a key K that is indistinguishable from random. This requirement should be fulfilled even if one of the input keys was adversarially chosen, as long as the other key has sufficiently high min-entropy. We have already proved that $g(K)$ is a one-way injective function, and we can safely assume that HASH is an ϵ-Regular function as defined in Section 4.

```
Data: Keys K_1, K_2, ..., K_N and optional salt with default value of all zero bytes
Result: K
for i from 1 to n do
  | k_i ← HMAC(salt, K_i), u_i ← g(K_i);
end
K' ← 0;
for i from 1 to n do
  | input ← i;
  | for j from 1 to n do
  |   | if i ≠ j then
  |   |   | input ← input||u_j
  |   | end
  | end
  | K' ← K' ⊕ HMAC(k_i, input)
end
K ← HASH(K'));
```

Fig. 7. Instantiation of multi-key KeyCombine $(K_1, K_2, .., K_N, salt)$

It therefore remains to show that $\mathsf{HMAC}(salt, K_1)$ is a computational extractor with respect to our instantiation of the function g. Krawczyk [25] showed that $\mathsf{HMAC}(salt, K)$ with a random public salt is a computational extractor with respect to K chosen from a non-uniform distribution with high enough min-entropy. In our definition, if we model the advantage gained by the attacker due to learning $g(K)$ as defining a non-uniform distribution, we can use the same argument to show that $\mathsf{HMAC}(salt, K)$ is also a computational extractor with respect to $g(x)$. Using this construction also allows us to handle non-uniform keys as in HKDF.extract.

We note that as in HKDF.extract, using a random salt is preferable but might not always be possible. For example, in TLS 1.3, the default salt value of all zero bytes is used, as there is no authenticated source of randomness. Moreover, if the salt or part of it is chosen adversarially, we cannot prove the security of the resulting construction.

5.3 Multi-Key Variant

Our construction can be easily extended to combine any number of keys. The pseudocode for a multi-key KeyCombine is given in Fig. 7.

The same hybrid argument given in Section 4 can be used to show that the resulting key is indistinguishable from random even if all keys but one are adversarially chosen.

5.4 Benchmarks

We now provide benchmarks for our construction. Our aim is to measure the computational cost of the construction, especially compared to the cost of a full hybrid key exchange protocol. As we show, the construction introduces modest overhead, of several microseconds of computation time. Moreover, this overhead is negligible from the perspective of connection latency, where the median of approximately 50 *milliseconds* is 4 orders of magnitude greater than our overhead [26].

We have implemented our key combiner in C, using the OpenSSL cryptographic library [7]. Our code runs both our key combiner and HKDF.Extract in a loop, for a duration of over a second, while counting the number of completed calls to each function. The average time per call is then the number of completed calls divided by the total duration. We focus on the case where both keys to be combined are 256 bits long. This is the expected case for hybrid key exchange (see below). For our key combiner, we combine the two

[7] Please find the reference implementation here: `https://github.com/nimia/kdf_reference_implementation`

keys with a salt consisting of all zeroes, of length 256 bits. For HKDF.Extract, we concatenate the keys and feed them to the function, with the same choice of salt. In these conditions, our construction requires on average 7.13 microseconds per call, whereas HKDF.Extract requires on average 1.35 microseconds per call. (All experiments in this section were performed on a workstation with an Intel i7 CPU, at a 3.60GHz clock frequency.) Hence our construction adds an overhead of 5.78 microseconds.

The benchmarks presented here for our new construction are conservative, since our implementation forgoes straightforward optimizations such as pre-generation and reuse of hash states for $Hash(B_i)$.

As a reference point, we examine an actual deployment of hybrid key exchange in TLS, the CECPQ2 experiment by Google and Cloudflare [26, 27]. This experiment paired ECDH over the x25519 curve with NTRU-HRSS [22] [8]. In addition to the key exchange algorithms, TLS handshakes require signature generation and verification. Using ECDSA over the secp256r1 curve is a popular choice for the signature algorithm, used e.g. when connecting from an up-to-date browser to Google's servers.

Table 1 summarizes the cost of running the hybrid protocol using the above choices for algorithms. As can be seen, our construction induces but a small part of the computational requirements, dwarved by the overall connection establishment time.

Primitive	Operation	Time (μsec)	Source
Our Construction	Key Combination	7.13	Our benchmarks
HKDF.Extract	Equivalent Key Processing	1.35	Our benchmarks
ECDH over x25519	Exponentiation	44.71	openssl speed
NTRU-HRSS	Encapsulation	11.24	Our benchmarks
NTRU-HRSS	Decapsulation	17.63	Our benchmarks
ECDSA over secp256r1	Sign	24.62	openssl speed
ECDSA over secp256r1	Verify	79.04	openssl speed
CECPQ2	Handshake duration	54,000 (median)	[26]

Table 1. Running times for the primitives used in the CECPQ2 experiment, as well as our construction. The "Time" column gives average times, except for the last row, which gives the median connection establishment time for the entire protocol, from the server's point of view. For NTRU-HRSS, we benchmarked the published software [10].

6 Application to TLS

In this section we explain how to apply our KeyCombine construction to the key schedule of TLS 1.3. We introduce two variants of the key schedule in TLS 1.3:

1. The first preserves the existing structure of the TLS 1.3 key schedule (for completeness, we have included this as the top layer of Figure 6), which we denote *structure-preserving TLS 1.3*. This allows us to use our KeyCombine construction as a drop-in replacement for HKDF.Extract as it is currently used. Structure-Preserving TLS 1.3 can be found on the middle layer in Figure 6.
2. The second deviates from the existing structure of the TLS 1.3 key schedule, which we denote *break-the-chain TLS 1.3*. This approach re-includes all secret values input to the key schedule directly in the derivations of ES, HS and MS when available. This modification is less of a drop-in replacement for HKDF.Extract as it is currently used, but allows us to achieve better guarantees of the key schedule as a result. Break-the-Chain TLS 1.3 can be found on the bottom layer in Figure 6.

[8] We focus on the "CECPQ2" flavor of the experiment; an additional flavor, "CECPQ2b", paired ECDH over x25519 with SIKE. That flavor requires more computational effort, so we focus on the faster flavor.
[9] https://ntru-hrss.org/software.shtml
[10] https://ntru-hrss.org/software.shtml

Note that our variants include a post-quantum secret PQS in the key schedule – this value can be replaced with a zero-string when it is not available.

Our Structure-Preserving TLS 1.3 key schedule replaces the execution of $\mathsf{HKDF.Extract}(0, \mathsf{PSK})$, $\mathsf{HKDF.Extract}(\mathsf{dES}, \mathsf{DHE})$ and $\mathsf{HKDF.Extract}(\mathsf{dHS}, 0)$ with $\mathsf{KeyCombine}(\mathsf{PSK}, 0, 0)$, $\mathsf{KeyCombine}(\mathsf{DHE}, \mathsf{dES}, 0)$, $\mathsf{KeyCombine}(\mathsf{PQS}, \mathsf{dHS}, 0)$.
Our Break-the-Chain TLS 1.3 replaces the execution of $\mathsf{HKDF.Extract}(0, \mathsf{PSK})$, $\mathsf{HKDF.Extract}(\mathsf{dES}, \mathsf{DHE})$ and $\mathsf{HKDF.Extract}(\mathsf{dHS}, 0)$ with $\mathsf{KeyCombine}(\mathsf{PSK}, 0, 0)$, $\mathsf{KeyCombine}(\mathsf{DHE}, \mathsf{PSK}, 0)$, $\mathsf{KeyCombine}_3(\mathsf{PQS}, \mathsf{DHE}, \mathsf{PSK}, 0$, where $\mathsf{KeyCombine}_3(\mathsf{PQS}, \mathsf{DHE}, \mathsf{PSK}, 0)$ is our multi-key variant, taking three keyed inputs and a single salt input.

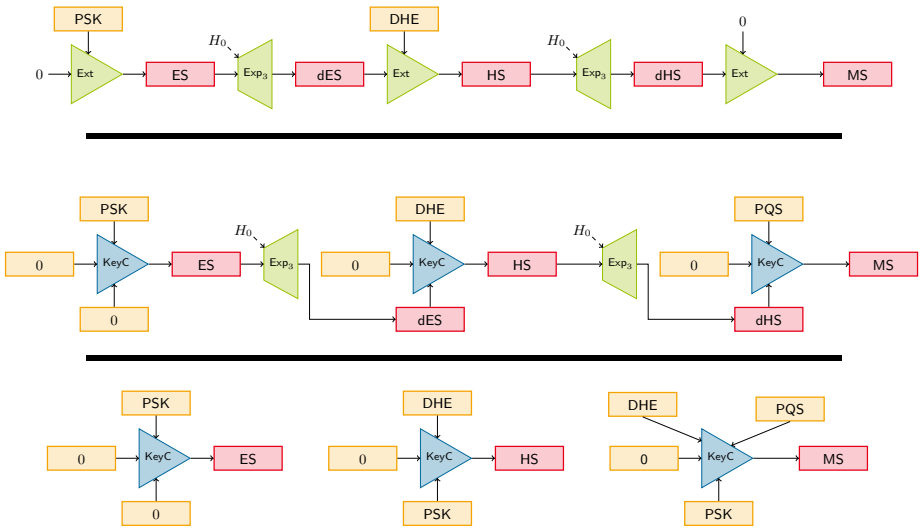

Fig. 8. The original TLS 1.3 key schedule (above), a modified TLS 1.3 (hybrid) key schedule backbone using our new KeyCombine construction that preserves the structure of original TLS 1.3 (middle), and a modified TLS 1.3 (hybrid) key schedule backbone that deviates from the structure of original TLS 1.3 (below). Note that KeyC is our KeyCombine construction, taking keying inputs from the top and bottom, and *salt* value from the left-hand side.

In the case of Structure-Preserving TLS 1.3, this change has little impact on the analysis of the TLS 1.3 handshake protocol that we discuss in Section 2.2 [13]. In the analysis of the different TLS 1.3 handshakes, we substitute game-hops in the analysis that replace the execution of HKDF.Extract with various KeyCombine executions, that we describe below.

This change affects **Game B.2**, **Game B.5** of the proof of Theorem 5.2 (Multi-Stage security of `TLS1.3-full-1RTT`), which replaces:

- $\mathsf{HKDF.Extract}(\mathsf{dES}, \mathsf{DHE})$ with the execution of $\mathsf{KeyCombine}(\mathsf{DHE}, \mathsf{dES}, 0)$, requiring no change to the dual-sn-PRF-ODH assumption.
- $\mathsf{HKDF.Extract}(\mathsf{dHS}, 0)$ with the execution of $\mathsf{KeyCombine}(0, \mathsf{dHS}, \mathsf{nonces})$, requiring no change to the PRF assumption.

This also affects **Game 4**, **Game 6**, and **Game 9** of the proof of Theorem 6.2 (Multi-Stage security of `TLS1.3-PSK-0RTT`), which replaces:

- $\mathsf{HKDF.Extract}(0, \mathsf{PSK})$ with the execution of $\mathsf{KeyCombine}(\mathsf{PSK}, 0, 0)$, requiring no change to the dual-PRF assumption.
- $\mathsf{HKDF.Extract}(\mathsf{dES}, 0)$ with the execution of $\mathsf{KeyCombine}(0, \mathsf{dES}, 0)$, requiring no change to the PRF assumption.

– HKDF.Extract(dHS, 0) with the execution of KeyCombine(0, dHS, nonces), requiring no change to the PRF assumption.

Finally, this affects **Game A.4**, **Game A.6**, **Game B.3**, **Game C.2**, and **Game C.5** of the proof of Theorem 6.4 (Multi-Stage security of `TLS1.3-PSK-(EC)DHE-0RTT`), which replaces:

– HKDF.Extract(0, PSK) with the execution of KeyCombine(PSK, 0, 0), requiring no change to the dual-PRF assumption.
– HKDF.Extract(dES, DHE) with the execution of KeyCombine(DHE, dES, 0), requiring no change to the PRF assumption.
– HKDF.Extract(0, PSK) with the execution of KeyCombine(PSK, 0, 0), requiring no change to the dual-PRF assumption.
– HKDF.Extract(dES, DHE) with the execution of KeyCombine(DHE, dES, 0), requiring no change to the dual-sn-PRF-ODH assumption.
– HKDF.Extract(dHS, 0) with the execution of KeyCombine(0, dHS, nonces), requiring no change to the PRF assumption.

Thus, our KeyCombine construction can act as a drop-in replacement for the use of HKDF.Extract of TLS 1.3, with little additional computational overhead, with a minor adaption of existing proofs of the protocol, but now modeled with assumptions that accurately match the existing analysis.

Breaking the Chain: Next, on the bottom layer of Figure 6 we introduce our Break-the-Chain TLS variant. This change provides stronger guarantees for the derivation of secret values throughout the key schedule. To give some intuition as to why this is the case, consider an attacker that is capable of choosing distinct PSK, PSK′ values for a client in two different sessions. When deriving ES (in all key schedules depicted in Figure 6), since the attacker has generated the only keying inputs PSK (PSK′ for the second session, respectively), and all other values are constant: thus, HKDF.Extract and KeyCombine provide no guarantees on the security of the output ES (resp. ES). Thus, it is possible that ES = ES′, even if PSK ≠ PSK. Now, if the client re-uses the Diffie-Hellman secret value DHE in both sessions, in the original and structure-preserving key schedules, then HS will collide with HS′, even if the attacker does not know DHE. However, HS will not collide with HS′ in the Break-the-Chain variant of the key schedule.

On the choice of salts: Since (like HKDF.Extract) our construction cannot be proven secure if the salt is chosen by the attacker, we instead use a constant zero-string for salting our KeyCombine construction in the key combination steps, similar to the first HKDF.Extract operation deriving the ES in the original TLS 1.3 key schedule.

7 Related Work

Bellare and Lysyanskaya [5] proved it is possible to build dual-PRFs from standard assumptions. Their theoretical constructions are based either on collision-resistant hash functions or one-way permutations. However, they leave the construction of a dual-PRF from the much weaker assumption of any one-way function as an open problem. We partly answer this open problem by providing a construction based on assumptions that are close in spirit to one-wayness; our construction therefore shares a similar structure to theirs. Further, their work is entirely theoretical, focusing on proving that dual PRFs exist, and (reasonably) disregarding the question of how one would instantiate such a dual PRF from standard, widely-used cryptographic components. However, the purpose of this work is to provide a readily implementable such construction.

Brendel *et al.* [8] have analyzed the resilience of hybrid protocols to a breakdown of one of the key exchange algorithms. Their proposed solution requires running two full instances of the protocol in parallel, essentially doubling the communication requirements.

In [16], Giacon *et al.* have introduced several KEM combiners. Their work investigates a higher-level primitive - KEMs and how to combine them - whereas we would like to combine keys in general, not necessarily

encapsulated keys. For example, their KEM combiner does not readily allow combining a preshared key. Moreover, using their KEM combiner for hybrid key exchange in TLS 1.3 would involve feeding ciphertexts as input to the key-mixing component of the key schedule, which would require intrusive re-engineering for most TLS implementations.

The work by Bindel *et al.* [7] examines hybrid key encapsulation mechanisms and AKE protocols. Their point of focus is therefore also much higher-level than key combiners. Moreover, two of their constructions assume a dual-PRF, for which we provide an efficient and practical instantiation.

Huguenin-Dumittan and Vaudenay [21] have recently proposed a novel KEM combiner for quantum-resistant KEMs. Their work therefore deals with a much higher-level primitive than ours. Further, their scheme is only proven secure under the much stronger assumptions of (Quantum) Random Oracle Model (QROM and ROM models).

Acknowledgements. We thank Douglas Stebila for many fruitful discussions.

N. Aviram's research is supported in part by The Blavatnik ICRC. I. Komargodski's research is supported in part by an Alon Young Faculty Fellowship, by a JPM Faculty Research Award, by a grant from the Israel Science Foundation (ISF Grant No. 1774/20), and by a grant from the US-Israel Binational Science Foundation and the US National Science Foundation (BSF-NSF Grant No. 2020643). The work of Paterson is supported in part by a gift from VMware. E. Ronen is supported in part by Len Blavatnik and the Blavatnik Family foundation, the Blavatnik ICRC, and Robert Bosch Technologies Israel Ltd. He is a member of CPIIS.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle diffie-hellman assumptions and an analysis of dhies. In: Cryptographers Track at the RSA Conference. pp. 143–158. Springer (2001)
2. Alwen, J., Coretti, S., Dodis, Y.: The double ratchet: security notions, proofs, and modularization for the signal protocol. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 129–158 (2019)
3. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Annual International Cryptology Conference. pp. 602–619. Springer (2006)
4. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Annual international cryptology conference. pp. 1–15. Springer (1996)
5. Bellare, M., Lysyanskaya, A.: Symmetric and Dual PRFs from Standard Assumptions: A Generic Validation of an HMAC Assumption. (2015)
6. Bhargavan, K., Lavaud, A.D., Fournet, C., Pironti, A., Strub, P.Y.: Triple handshakes and cookie cutters: Breaking and fixing authentication over tls. In: 2014 IEEE Symposium on Security and Privacy. pp. 98–113. IEEE (2014)
7. Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., Stebila, D.: Hybrid key encapsulation mechanisms and authenticated key exchange. In: International Conference on Post-Quantum Cryptography. pp. 206–226. Springer (2019)
8. Brendel, J., Fischlin, M., Günther, F.: Breakdown resilience of key exchange protocols: NewHope, TLS 1.3, and Hybrids. In: European Symposium on Research in Computer Security. pp. 521–541. Springer (2019)
9. Brendel, J., Fischlin, M., Günther, F., Janson, C.: Prf-odh: Relations, instantiations, and impossibility results. In: Annual International Cryptology Conference. pp. 651–681. Springer (2017)
10. Campagna, M., Petcher, A.: Security of hybrid key encapsulation. IACR Cryptology ePrint Archive. (2020), `https://eprint.iacr.org/2020/1364.pdf`
11. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. Journal of Cryptology **33**(4), 1914–1983 (2020)
12. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness extraction and key derivation using the cbc, cascade and hmac modes. In: Annual International Cryptology Conference. pp. 494–510. Springer (2004)
13. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the tls 1.3 handshake protocol. Journal of Cryptology **34**(4), 1–69 (2021)
14. Dowling, B., Hansen, T.B., Paterson, K.G.: Many a mickle makes a muckle: A framework for provably quantum-secure hybrid key exchange. In: Ding, J., Tillich, J. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12100, pp. 483–502. Springer (2020). https://doi.org/10.1007/978-3-030-44223-1_26, `https://doi.org/10.1007/978-3-030-44223-1\_26`

15. Fischlin, M., Günther, F.: Multi-stage key exchange and the case of Google's QUIC protocol. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 1193–1204 (2014)
16. Giacon, F., Heuer, F., Poettering, B.: KEM combiners. In: IACR International Workshop on Public Key Cryptography. pp. 190–218. Springer (2018)
17. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
18. Hemenway, B., Lu, S., Ostrovsky, R.: Correlated product security from any one-way function. In: Public Key Cryptography - PKC. pp. 558–575 (2012)
19. Holmgren, J., Lombardi, A.: Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In: 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018. pp. 850–858 (2018)
20. Housley, R., Hoyland, J., Sethi, M., Wood, C.A.: Guidance for External PSK Usage in TLS. Internet-Draft draft-ietf-tls-external-psk-guidance-05, Internet Engineering Task Force (Jan 2022), https://datatracker.ietf.org/doc/html/draft-ietf-tls-external-psk-guidance-05, work in Progress
21. Huguenin-Dumittan, L., Vaudenay, S.: Fo-like combiners and hybrid post-quantum cryptography. Cryptology ePrint Archive, Report 2021/1288 (2021), https://ia.cr/2021/1288
22. Hülsing, A., Rijneveld, J., Schanck, J., Schwabe, P.: High-speed key encapsulation from NTRU. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 232–252. Springer (2017)
23. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of tls-dhe in the standard model. In: Annual Cryptology Conference. pp. 273–293. Springer (2012)
24. Krawczyk, H., Eronen, P.: Hmac-based extract-and-expand key derivation function (hkdf) (May 2010), http://tools.ietf.org/rfc/rfc5869.txt, rFC5869
25. Krawczyk, H.: Cryptographic extraction and key derivation: The HKDF scheme. Cryptology ePrint Archive, Report 2010/264 (2010), https://ia.cr/2010/264
26. Kwiatkowski, K., Valenta, L.: The TLS Post-Quantum Experiment (2019), https://blog.cloudflare.com/the-tls-post-quantum-experiment/
27. Langley, A.: Real-world measurements of structured-lattices and supersingular isogenies in TLS (2019), https://www.imperialviolet.org/2019/10/30/pqsivssl.html
28. Layton, W.: Post quantum cryptography and national security systems. International Cryptographic Module Conference (ICMC) (2021)
29. Leurent, G.: Practical key-recovery attack against apop, an md5-based challenge-response authentication. International Journal of Applied Cryptography **1**(1), 32–46 (2008)
30. Paterson, K.G., van der Merwe, T.: Reactive and proactive standardisation of tls. In: International Conference on Research in Security Standardisation. pp. 160–186. Springer (2016)
31. Perrin, T., Marlinspike, M.: The double ratchet algorithm (2016), https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf
32. Ray, M.: Authentication gap in TLS renegotiation (2009), archived at {https://web.archive.org/web/20091228061844/http://extendedsubset.com/?p=8}
33. Rescorla, E.: Understanding the TLS renegotiation attack (2009), archived at https://web.archive.org/web/20091231034700/http://www.educatedguesswork.org/2009/11/understanding_the_tls_renegoti.html
34. Rescorla, E., Dierks, T.: The Transport Layer Security (tls) Protocol version 1.3 (2018)
35. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. SIAM J. Comput. **39**(7), 3058–3088 (2010)
36. Sasaki, Y., Yamamoto, G., Aoki, K.: Practical password recovery on an md5 challenge and response. IACR Cryptol. ePrint Arch. **2007**, 101 (2007)
37. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020. pp. 1461–1480. ACM (2020). https://doi.org/10.1145/3372297.3423350, https://doi.org/10.1145/3372297.3423350
38. Stebila, D., Fluhrer, S., Gueron, S.: Hybrid key exchange in TLS 1.3. Internet-Draft draft-ietf-tls-hybrid-design-03, Internet Engineering Task Force (Jul 2021), https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-03, work in Progress
39. Xie, T., Feng, D.: Construct md5 collisions using just a single block of message. IACR Cryptol. ePrint Arch. **2010**, 643 (2010)

Supplementary Materials

A Detailed Assumptions on HKDF

Here we present the assumptions used to model HKDF and HMAC in the computational analyses of various cryptographic protocols.

A.1 TLS 1.3

We begin with the analysis [13] of the TLS 1.3 handshake protocol. On a high-level, the handshake protocol is a key exchange protocol, establishing secure symmetric keys between the client and the server executing the handshake protocol together. Dowling *et al.* demonstrate that under certain assumptions of the underlying cryptographic primitives (including HMAC and HKDF.Extract), the TLS 1.3 handshake protocol achieves a standard key exchange property known as key-indistinguishability - this captures the notion that the session key derived by both parties is indistinguishable from a uniformly random key drawn from the same distribution. During the proof, the authors use a dual-PRF-ODH assumption to model the use of Diffie-Hellman in the HKDF.Extract function.

The PRF-ODH assumption family was introduced by Jager *et al.* [23] for the analysis of TLS 1.2. The authors explain that it is an extension of the ODH assumption introduced by Abdalla *et al.* [1], and has been used in the analysis of many modern cryptographic protocols. We give a definition for the snPRF-ODH and dual-snPRF-ODH assumptions (the specific assumptions used in the analysis of TLS 1.3) below, formalised as a game played between a challenger and an adversary $\mathcal{A}$.

Definition 8 (snPRF-ODH and dual-snPRF-ODH assumptions).
Let $\lambda \in \mathbb{N}$, G be a cyclic group of prime order q with generator g and $\mathsf{PRF} : G \times \{0,1\}^* \to \{0,1\}^\lambda$ be a pseudo-random function. We define the snPRF-ODH *security game as follows:*

1. *The challenger samples $b \xleftarrow{\$} \{0,1\}$, $u, v \xleftarrow{\$} \mathbb{Z}_q$, and provides G, g, g^u, and g^v to $\mathcal{A}$, who responds with a challenge label x.*
2. *The challenger computes $y_0 = \mathsf{PRF}(g^{uv}, x)$ and samples $y_1 \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random, providing y_b to $\mathcal{A}$.*
3. *$\mathcal{A}$ may query a pair (S, x), on which the challenger first ensures that $S \in G$ and $(S, x) \neq (g^v, x)$ and, if so, returns $y \leftarrow \mathsf{PRF}(S^u, x)$.*
4. *Eventually, $\mathcal{A}$ stops and outputs a guess $b' \in \{0,1\}$.*

We define the snPRF-ODH *advantage function as $\mathsf{Adv}^{\mathsf{snPRF\text{-}ODH}}_{\mathsf{PRF},G,\mathcal{A}} := 2 \cdot \Pr[b' = b] - 1$. We define the dual variant of the assumption,* dual-snPRF-ODH*, as the* snPRF-ODH *assumption for a function $\mathsf{PRF} : \{0,1\}^* \times G \to \{0,1\}^\lambda$ with swapped inputs, keyed with a group element in the second input and taking the label as first input.*

When analysing the full handshake, Dowling *et al.* describe a specific game hop where they replace the computation of $\mathsf{HS} \leftarrow \mathsf{HKDF.Extract}(\mathsf{dHS}, \mathsf{DHE})$ in the proof with a value HS', sampled uniformly at random from the output distribution of HKDF.Extract. Then, they argue that they can turn any algorithm capable of distinguishing this change into an adversary against the dual-snPRF-ODH security of the HKDF.Extract function, functionally modelling HKDF.Extract as a dual PRF. However, as we mentioned previously this does not match any known proof of HKDF.Extract that can be found publicly.

A.2 The Signal Key Exchange Protocol

We now present the assumptions used to model HKDF in the computational analyses [31] of the Signal key exchange protocol. As in the TLS 1.3 analysis, Cohn-Gordon *et al.* describe a particular game hop where they replace the computation of the root key $rk_i \leftarrow \mathsf{HKDF}(rk_{i-1}, \mathsf{DHE}, \texttt{const})$ in the proof with a value rk'_i,

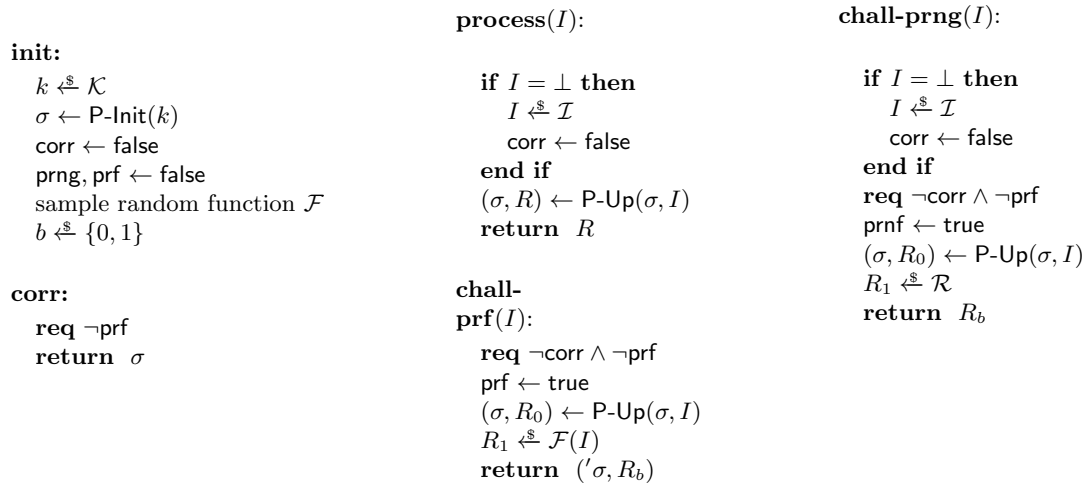

Fig. 9. The PRF-PRNG game used by Alwen *et al.* to model the security of a HKDF chain.

sampled uniformly at random from the output distribution of HKDF. Specifically, they argue that they can turn any algorithm capable of distinguishing this change into an adversary against the dual-lrPRF-ODH security of the HKDF.Extract function, formally modelling HKDF.Extract as a dual PRF. Unlike the analysis of TLS 1.3, however, the authors require not a single PRF-ODH assumption, but a range of PRF-ODH assumptions that are parameterized by how many times the challenger will generate $\mathsf{PRF}(g^{uv'}, x)$ and $\mathsf{PRF}(g^{u'v}, x)$ values upon being queried for a "left" or "right" (thus dual-lrPRF-ODH) oracle $(g^{v'}, x)$ or $(g^{u'}, x)$ (where $g^{u'}$ and $g^{v'}$ are not one of the DH challenge values (g^u, g^v) given by the challenger). Since some game hops in the analysis require that the root key rk_{i-1} act as the secret value, and some game hops require that the Diffie-Hellman output DHE act as the secret value, and it is the HKDF.Extract phase in HKDF that combines these secrets, functionally this means modelling HKDF.Extract as a dual PRF. However, as we mentioned previously this does not match any known proof of HKDF.Extract that can be found publicly.

In a work capturing the security of the Signal protocol, Alwen *et al.* [2] re-create the Signal protocol from generic cryptographic primitives. In particular, they note that Signalinstantiate the HKDF chain used by Signal as a PRF-PRNG. As they describe: "*A* PRF-PRNG *resembles both a pseudo-random function (PRF) and a pseudorandom number generator with input (PRNG)... On the one hand, as a PRNG would, a PRF-PRNG (1) repeatedly accepts inputs I and uses them to refresh its state σ and (2) occasionally uses the state, provided it has sufficient entropy, to derive a pseudo-random pair of otuput R and new state."* We give the PRF-PRNG security game described by Alwen of the PRF-PRNG in Figure A.2.

Commenting on whether HKDF could be used to instantiate a PRF-PRNG, Alwen *et al.* merely state: *"Being a PRF-PRNG is a property the HKDF function used by Signal is assumed to have... This paper therefore merely reduces to the security of the presented schemes to the PRF-PRNG security of whatever function is used to instantiate it."*

A.3 The ETSI Hybrid Key Exchange Drafts

Recently, ETSI proposed two constructions for combining secrets from hybrid key exchange protocols: concatenate KDF (CatKDF, see Figure 10) and cascade KDF (CasKDF). We describe the CatKDF mechanism below, and then discuss the assumptions made of the underlying cryptographic primitives used in the analysis of this construction. We note that CatKDF specifically uses HKDF as its underlying key derivation function.

We note that the assumptions for the ETSI proofs do not consider the setting we have proposed, but are significantly stronger. Recall that we assume the adversary gets to control some part of the secret concatenation (e.g. when KEMs are used), and that honest parties may re-use secrets (e.g. when static ECDH is used). In the analysis of the ETSI CatKDF draft, the adversary only gets to receive the secrets

Data: $PSK, k_1, \ldots, k_n, MA, MB, lbl, ctxt$
Result: k

$s \leftarrow PSK \| k_1 \| \ldots \| k_n$;
$f_{val} = H(ctxt \| MA \| MB)$;
$k \leftarrow \mathsf{HKDF}(s, lbl, f_{val}, len)$;

Fig. 10. The CatKDF construction proposed by ETSI.

associated with all but a single KEM, but is not able to inject KEM ciphertexts or secrets. This type of passive adversary is signficantly weaker that the threat model we consider.

In addition, the analysis of the CatKDF construction models HKDF either as a random oracle, or as a weakly secure KDF. In either case, we argue that this is not an appropriate threat model in our setting where an adversary can inject KEM ciphertexts and control the encapsulated secret, nor where the underlying hash function is not collision-resistant:

1. HKDF is modelled as a random oracle, which is inherently collision-resistant. Specifically, each random oracle takes a query tuple $(k, salt, ctxt)$, where k is keying material, $salt$ is salt and $ctxt$ is context values, and returns a random bit string if the query is not entirely equal to a previous query value. The HKDF paper does not claim security in the presence of attacker-controlled entropy [25], so this assumption seems rather strong.
2. HKDF is modelled as a weakly secure KDF. In this assumption, both the secret value k and the salt $salt$ are randomly selected, and $(salt, \sigma)$ is given to the adversary, where σ is a description of the secret space that k is drawn from. The adversary is allowed to specify a context value $ctxt$ and an output length, and has to distinguish the output from the KDF from a random string of the same length. The adversary is not allowed to query the KDF more than once. Note that for the purposes of CatKDF, the secret value k is the concatenation of all secrets output from the hybrid KEM, and they seem to assume one of which is IND-CPA secure.