# When Messages are Keys: Is HMAC a dual-PRF?

Matilda Backendal[1], Mihir Bellare[*2], Felix Günther[1], and Matteo Scarlata[1]

[1]Department of Computer Science, ETH Zurich, Zurich, Switzerland
[2]Department of Computer Science & Engineering, University of California San Diego, USA

June 7, 2023

**Abstract**

In Internet security protocols including TLS 1.3, KEMTLS, MLS and Noise, HMAC is being assumed to be a dual-PRF, meaning a PRF not only when keyed conventionally (through its first input), but also when "swapped" and keyed (unconventionally) through its second (message) input. We give the first in-depth analysis of the dual-PRF assumption on HMAC. For the swap case, we note that security does not hold in general, but completely characterize when it does; we show that HMAC is swap-PRF secure if and only if keys are restricted to sets satisfying a condition called feasibility, that we give, and that holds in applications. The sufficiency is shown by proof and the necessity by attacks. For the conventional PRF case, we fill a gap in the literature by proving PRF security of HMAC for keys of arbitrary length. Our proofs are in the standard model, make assumptions only on the compression function underlying the hash function, and give good bounds in the multi-user setting. The positive results are strengthened through achieving a new notion of variable key-length PRF security that guarantees security even if different users use keys of different lengths, as happens in practice.

A preliminary version of this paper appears in the proceedings of the *43rd International Cryptology Conference (CRYPTO 2023)*. This is the full version.

[*]Supported in part by NSF grant CNS-2154272 and KACST.

## Contents


**1 Introduction** **3**
- 1.1 Background . . . 3
- 1.2 Swap-PRF Security of HMAC . . . 4
- 1.3 PRF Security of HMAC . . . 6
- 1.4 Auxiliary Contributions and Technical Overview . . . 6

**2 Related Work** **8**

**3 Notation and Definitions** **9**
- 3.1 Notation and Conventions . . . 9
- 3.2 Standard Definitions . . . 10
- 3.3 New Definitions . . . 10

**4 Background and Modularization of HMAC** **12**

**5 Dual-PRF Security of NMAC** **16**
- 5.1 vkl-PRF Security of $\mathsf{NMAC}^{\leftrightarrows}$ . . . 16
- 5.2 Strong Multi-user PRF Security of NMAC . . . 20

**6 Dual-PRF Security of HMAC** **23**
- 6.1 Composition Theorem . . . 23
- 6.2 Analysis of HSKD . . . 27
- 6.3 vkl-PRF Security of HMAC . . . 33
- 6.4 vkl-PRF Security of $\mathsf{HMAC}^{\leftrightarrows}$ . . . 34

**7 Proofs for the rka-PRF Security of $\mathsf{h}^{\leftrightarrows}$** **35**
- 7.1 $\Phi$-rka-PRF Security of $\mathsf{h}^{\leftrightarrows}$ . . . 35
- 7.2 $\Phi$-rka-PRP Security of Block Ciphers in ICM . . . 37

**A Dual-PRF Assumptions on HMAC in Prior Work** **44**

**B Proofs for the Strong PRF Security of the Cascade** **46**
- B.1 Proof of Lemma 1 . . . 46
- B.2 Proof of Lemma 2 . . . 48
- B.3 Related Work and Techniques . . . 49

# 1 Introduction

HMAC [8] is a hash-function-based construct taking two inputs to return a fixed-length output. It was designed to be PRF-secure, a usage in which the first input is the key and the second is the message. It is standardized by the IETF [39] and NIST [46] and widely used, including in IPsec, SSH, and TLS. However, its conventional use as a PRF is now being supplemented with another type of use, as a key combiner, which (additionally) assumes that it is a swap-PRF, meaning a PRF when keyed by the second input, with the first input regarded as the message. This is happening in various Internet security protocols including TLS 1.3 [27, 24], KEMTLS [52], hybrid key-exchange designs [17, 53], post-quantum versions of WireGuard [35] and Noise [2], and Message Layer Security (MLS) [23]. Overall, then, HMAC is being assumed to be a dual-PRF; as defined in [5, 6] this means being both a PRF and a swap-PRF.

But is this assumption well-founded? The extensive real-world usage represented by the above protocols makes this question crucial. Yet, it has not been seriously investigated. And as an indication that the gap is more than academic, we note that there are concerns on *both* sides of the dual-PRF coin, meaning for both swap-PRF and conventional PRF security, as follows.

▷ HMAC is not swap-PRF secure, but may be for restricted inputs. Simple attacks (described below) show that HMAC is actually *not* swap-PRF secure. Luckily, due to the specific inputs used, these attacks do not endanger the protocols listed above. But that doesn't mean these usages are secure. We want to know for what inputs one can prove security and under what assumptions, and whether current usage assuming swap-PRF security can be validated with proofs.

▷ PRF security of HMAC has only been proved for one key length. Meanwhile, our work lead us to realize that there is a gap in the literature even with regard to the conventional PRF security of HMAC, namely that it has only been proven for keys of length equal to the block length [5, 6]. Yet, in practice, and in the above applications, HMAC is often used with keys shorter than the block length, and the standards allow keys of any length. We want to know whether proofs of PRF security are possible for all allowed key lengths, and under what assumptions.

Contributions in brief. Our contribution is to identify and fill the above gaps. To capture restrictions on inputs, we let HMAC[$\mathcal{S}$] denote HMAC restricted to keys being drawn from a subset $\mathcal{S}$ of its keyspace. As the more novel and interesting case, we first consider swap-PRF security, in which case $\mathcal{S}$ is a subset of the message space. We give necessary and sufficient conditions on $\mathcal{S}$ for HMAC[$\mathcal{S}$] to be swap-PRF secure, justifying the sufficiency by proof and the necessity by attack. Sets used in current applications meet our sufficient conditions, hence our proofs provide guarantees for these usages. Turning to the conventional PRF security of HMAC, we prove it for keys of *any* (sufficiently large) length. Our results hold even when different users use keys of different and adaptively-chosen lengths, as captured formally by a new definition, of a variable key-length PRF (vkl-PRF), that we give as an auxiliary contribution. In summary, we emerge with a full picture of the security of HMAC as a dual-PRF.

We stress that all our proofs for HMAC are in the standard model (no idealized assumptions) and make assumptions only on the compression function underlying the hash function. Also, for the first time for HMAC, we prove multi-user security with good bounds, for all of our results. We now look at all this in more detail. A summary of our and prior results is in Table 1; notation used there is explained below.

## 1.1 Background

The functions. The starting point is a compression function $\mathsf{h}\colon \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$, taking a $c$-bit chaining variable and $b$-bit block to return a $c$-bit result, where $c < b$. Briefly, one starts with the cascade $\mathsf{h}^*$ [9], which takes a $c$-bit key, and message whose length is a multiple of $b$, and iterates $\mathsf{h}$ to return a $c$-bit result. (See Figure 3 later in Section 4 for full details.)

The hash function $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^c$ associated to $\mathsf{h}$ by the **MD** transform [44, 25] is $\mathsf{H}(M) = \mathsf{h}^*(\mathtt{IV}, \overline{M})$ where $\mathtt{IV} \in \{0,1\}^c$ is a fixed initial vector and $\overline{M}$ pads $M$ out to a length that is a multiple of $b$ bits. (The details of the padding, which do matter, are in Section 4.) This is the design underlying the SHA-2 function family [47], including SHA-256 ($c = 256$, $b = 512$), SHA-384 ($c = 384$, $b = 1024$), and SHA-512 ($c = 512$, $b = 1024$).

The simplest version of HMAC is $\mathsf{HMAC}_b\colon \{0,1\}^b \times \{0,1\}^* \to \{0,1\}^c$, taking a $b$-bit key and defined by

$$\mathsf{HMAC}_b(K_b, M) = \mathsf{H}((K_b \oplus \mathtt{opad})\|\mathsf{H}((K_b \oplus \mathtt{ipad})\|M)),$$

where $\mathtt{ipad}, \mathtt{opad} \in \{0,1\}^b$ are distinct constants specified in the standards. The "full" $\mathsf{HMAC}\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^c$ as standardized [8, 39, 46] takes a key of arbitrary length and is defined by $\mathsf{HMAC}(K, M) = \mathsf{HMAC}_b(\mathsf{PoH}_b(K), M)$, where the "Pad or Hash" function $\mathsf{PoH}_b\colon \{0,1\}^* \to \{0,1\}^b$ is defined by

$$\mathsf{PoH}_b(K) = \begin{cases} K\|0^{b-|K|} & \text{if } |K| \le b \\ \mathsf{H}(K)\|0^{b-c} & \text{otherwise.} \end{cases}$$

To facilitate analysis, Section 4 unravels the above definitions of $\mathsf{HMAC}_b$ and HMAC to view the functions in a more modular way. The core is NMAC [8], which is applied to the message with keys derived from the given key via a subkey derivation function HSKD.

PRFs, swap-PRFs and dual-PRFs. Recall that the definition of a function family $\mathsf{F}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ being a PRF views the first input as a key and the second input as a message: we ask that oracles for $\mathsf{F}(X, \cdot)$ and a random function $g\colon \mathcal{Y} \to \mathcal{Z}$ be indistinguishable when $X \leftarrow_\$ \mathcal{X}$ is not known to the attacker [31]. The *swap* of F is the function family $\mathsf{F}^{\leftrightarrows}\colon \mathcal{Y} \times \mathcal{X} \to \mathcal{Z}$ defined by $\mathsf{F}^{\leftrightarrows}(Y, X) = \mathsf{F}(X, Y)$. We say that F is a swap-PRF if $\mathsf{F}^{\leftrightarrows}$ is a PRF. (That is, F is a PRF when keyed by the second, or message, input.) The concept of a dual PRF was introduced in [5, 6]. F is a dual-PRF if F is both a PRF *and* a swap-PRF. That is, it is a PRF when keyed as usual by the first input, but also if keyed by the second input.

Prior work. A primary line of work has focused on proving PRF security of HMAC-related functions in the standard model and making assumptions only on the compression function. First, BCK [8] showed that NMAC is PRF-secure if the compression function h is PRF-secure and collision resistant. Bellare [5, 6] showed that PRF security of h alone suffices for the conclusion, and better bounds were given by GPR [30]. Further [5, 6] showed that $\mathsf{HMAC}_b$ is PRF-secure assuming NMAC is PRF-secure and h is swap-PRF secure under a simple form of related-key attack. (This prior work is summarized in Table 1.) We briefly note some gaps that will be addressed below. Namely, there are no proofs (1) of swap-PRF security for any of $\mathsf{NMAC}, \mathsf{HMAC}_b, \mathsf{HMAC}$, (2) of PRF security of HMAC itself (only of $\mathsf{HMAC}_b$), (3) with good bounds in the multi-user setting.

Usage. Key derivation in some prominent Internet security protocols involves combining a pair of keys $K_1, K_2$ into a single key via $K \leftarrow \mathsf{HMAC}(K_1, K_2)$. The intent is that (1) if $K_1$ is good (uniformly random and unknown to the attacker) then so is $K$ regardless of $K_2$ and (2) vice versa. Requirement (1) is satisfied if HMAC is a PRF, and requirement (2) if HMAC is a swap-PRF. That is, jointly, the assumption is that HMAC is a dual-PRF. Specifically, this is happening in TLS 1.3 [27, 24], KEMTLS [52], hybrid key-exchange designs [17, 53], post-quantum versions of WireGuard [35] and Noise [2], and Message Layer Security (MLS) [23], representing a large number of real-world use cases. In light of this, it is crucial to evaluate the dual-PRF security of HMAC.

## 1.2 Swap-PRF Security of HMAC

Attacks. We first note that there are simple attacks showing HMAC is in fact *not* swap-PRF secure. The issue is the well-known fact that the "Pad or Hash" function $\mathsf{PoH}_b$ is not collision

| Function | Goal | Assumptions | Comments |
|---|---|---|---|
| $\mathsf{HMAC}_b$ | PRF | $\mathsf{h}$ is PRF, $\mathsf{h}^{\leftrightarrows}$ is $\Phi_{\mathsf{io}}$-rka-PRF | [5, 6] |
| $\mathsf{HMAC}$ | vkl-PRF | $\mathsf{h}$ is PRF, $\mathsf{h}^{\leftrightarrows}$ is PRF, $\mathsf{h}^{\leftrightarrows}$ is $\Phi_{\mathsf{zio},a}$-rka-PRF | Theorem 10 |
| $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$ (for $\mathcal{S}$ feasible) | vkl-PRF | $\mathsf{h}$ is PRF, $\mathsf{h}^{\leftrightarrows}$ is $\Phi_{\mathsf{pad},a}$-rka-PRF, $\mathsf{h}$ is CR | Theorem 11 |
| $\mathsf{NMAC}$ | PRF | $\mathsf{h}$ is PRF | [5, 6, 30], Theorem 5 |
| $\mathsf{NMAC}^{\leftrightarrows}$ | vkl-PRF | $\mathsf{h}$ is PRF, $\mathsf{h}^{\leftrightarrows}$ is $\Phi_{\mathsf{pad},a}$-rka-PRF | Theorem 4 |
| $\mathsf{h}^{\leftrightarrows}$ | $\Phi$-rka-PRF | $\mathsf{E}$ is $\Phi$-rka-PRP | Proposition 12 |

Table 1: Summary of prior results (in the single-user setting) and our (multi-user) results proving security for $\mathsf{HMAC}_b$, $\mathsf{HMAC}$, $\mathsf{HMAC}[\mathcal{S}]$, and $\mathsf{NMAC}$, as well as results supporting assumptions made for these.

---

resistant. (This is also the source of attacks on the indifferentiability of HMAC for arbitrary keys [26], and the reason for an erratum filed for the HMAC RFC [43, 29].) For example if $K_1, K_1'$ are keys of length strictly less than $b$ bits such that $K_1' = K_1 \| 0^\ell$ for some $\ell \geq 1$, then $\mathsf{PoH}_b(K_1) = K_1 \| 0^{b-|K_1|} = K_1' \| 0^{b-|K_1'|} = \mathsf{PoH}_b(K_1')$, so $\mathsf{HMAC}(K_1, K_2) = \mathsf{HMAC}(K_1', K_2)$ for any $K_2$. An adversary can now violate PRF security of $\mathsf{HMAC}^{\leftrightarrows}$ by using $K_1, K_1'$ as messages: it calls its oracle on $K_1$ and $K_1'$ and declares "real" if the outputs are the same and "random" otherwise.[1] Likewise, if $K_1$ has length more than $b$, then $\mathsf{HMAC}(K_1, K_2) = \mathsf{HMAC}(\mathsf{H}(K_1), K_2)$ for any $K_2$, leading again to an attack on the swap-PRF security of HMAC using $K_1, \mathsf{H}(K_1)$ as messages.

While this disallows an unqualified assumption of swap-PRF security on HMAC, it does not negatively impact the security of any of the aforementioned usages of HMAC as a swap-PRF. This is because, in those uses, the HMAC keys (messages for $\mathsf{HMAC}^{\leftrightarrows}$) are limited in some way, for example prescribed to be of a fixed length, and this precludes the trivial weak-key pairs exploited above. So the question of practical significance that emerges is whether we can prove security in these cases.

<u>Swap-PRF security of $\mathsf{HMAC}[\mathcal{S}]$.</u> To formalize the question, let $\mathsf{HMAC}[\mathcal{S}]\colon \mathcal{S} \times \{0,1\}^* \to \{0,1\}^c$ denote the restriction of HMAC to keys from a set $\mathcal{S} \subseteq \{0,1\}^*$. Then the question is, for what choices of $\mathcal{S}$ can we prove swap-PRF security of $\mathsf{HMAC}[\mathcal{S}]$? We answer this question by giving a complete characterization of the class of sets $\mathcal{S}$ for which swap-PRF security of $\mathsf{HMAC}[\mathcal{S}]$ holds. We restrict the attention to sets that are length closed, meaning if $L \in \mathcal{S}$ then $\{0,1\}^{|L|} \subseteq \mathcal{S}$, a natural condition desirable for applications. Now define $\mathcal{S}$ to be *feasible* if either (1) $\mathcal{S} = \{0,1\}^\ell$ for some $\ell \leq b$ or (2) $\mathcal{S}$ contains no strings of length $\leq b$. Then, in Section 6, we show:

$\mathsf{HMAC}[\mathcal{S}]$ is swap-PRF secure **if and only if** $\mathcal{S}$ is feasible.

The "if" is proven (Theorem 11), under assumptions on $\mathsf{h}$ discussed below, and the "only if" is shown by attacks (Proposition 9).

<u>Assumptions.</u> Indifferentiability results on HMAC [26] will imply swap-PRF security of $\mathsf{HMAC}[\mathcal{S}]$ for $\mathcal{S}$ the set of all keys of a fixed length $\ell < b$, but these results are in the random-oracle model, meaning they assume a truly random compression function. However, as noted above, prior

---

[1] For a running-code example of two distinct messages (one vs. two 0-bytes, as key input) colliding under the same key ($0^{256}$, as message input) for $\mathsf{HMAC}^{\leftrightarrows}$, type:

```
$ python3 -c 'import hmac, hashlib; print(
  hmac.new(b"\x00"    , b"\x00"*32, hashlib.sha256).hexdigest(), "=",
  hmac.new(b"\x00\x00", b"\x00"*32, hashlib.sha256).hexdigest())'
```

proofs [8, 5, 6, 30] have shown (conventional) PRF security of $\mathsf{NMAC}$ and $\mathsf{HMAC}_b$ in the *standard model* while making assumptions *only on the compression function* $\mathsf{h}$. Ideally, one would do the same in proofs of swap-PRF security. Our proofs for the "if" above meet this bar. For any feasible $\mathcal{S}$, Theorem 11 establishes swap-PRF security of $\mathsf{HMAC}[\mathcal{S}]$ under three assumptions on $\mathsf{h}$. The first two are from the prior works [8, 5, 6, 30] and can be considered standard, namely that $\mathsf{h}$ is PRF secure and collision resistant (CR). Now recall that the proof of PRF security of $\mathsf{HMAC}_b$ from [5, 6] assumes $\mathsf{h}^{\leftrightarrows}$ is a PRF under related-key attack (rka) as defined in [11], for a related-key-deriving (RKD) function we denote $\Phi_{\mathsf{io}}$. Our third assumption, in the same vein, also assumes rka-PRF security of $\mathsf{h}^{\leftrightarrows}$, but for a different RKD function $\Phi_{\mathsf{pad},a}$. Roughly, the function $\Phi_{\mathsf{pad},a}$ allows the adversary to overwrite a suffix of length at most $b-a$ bits of the $b$-bit key with padding. The assumption is new but we will validate it through auxiliary proofs. Intuitively it arises because key material of length $\ell < b$ can now be part of the message, which is padded to block length. We summarize the assumptions in Table 1.

## 1.3 PRF Security of $\mathsf{HMAC}$

We turn next to the other side of dual-PRF security, asking about the (conventional) PRF security of $\mathsf{HMAC}$.

<u>The gap.</u> There appears at first to be no question here; PRF security of $\mathsf{HMAC}$ is broadly assumed and seen as established in prior work [8, 5, 6, 30]. There is, however, a noteworthy gap in prior results, namely that they only prove security with keys of full block length. More precisely, in the notation introduced above, what is proved in [5, 6] is PRF security of $\mathsf{HMAC}_b$, not $\mathsf{HMAC}$ itself. Yet, in practice, $\mathsf{HMAC}$ is almost never used with $b$ bit keys. Take for example $\mathsf{HMAC\text{-}256}$, i.e., $\mathsf{HMAC}$ instantiated with $\mathsf{SHA\text{-}256}$: Its output is $c = 256$ bits long. When using it in a protocol for key derivation, it is hence natural to use 256-bit keys throughout. However, the block length of $\mathsf{SHA\text{-}256}$ is $b = 512$, meaning that 256-bit keys lead to $\mathsf{HMAC}$ being keyed with keys of length *less* than the block length. This is precisely what is done in protocols like TLS [49], meaning that even their usage of $\mathsf{HMAC}$ as a regular PRF falls outside of the security guarantees provided in the literature.

<u>Filling the gap.</u> Theorem 10 proves PRF security of $\mathsf{HMAC}$ with keys of *any* (sufficiently large, of course) length. With regard to assumptions, recall that those made for PRF security of $\mathsf{HMAC}_b$ were that $\mathsf{h}$ is PRF secure and $\mathsf{h}^{\leftrightarrows}$ is $\Phi_{\mathsf{io}}$-rka-PRF secure. We likewise assume that $\mathsf{h}$ is PRF secure and $\mathsf{h}^{\leftrightarrows}$ is $\Phi_{\mathsf{zio},a}$-rka-PRF secure, for a function $\Phi_{\mathsf{zio},a}$ that allows the adversary to overwrite an at most $b-a$ bits long suffix of the $b$-bit key with zeroes (cf. $\Phi_{\mathsf{pad},a}$ for our $\mathsf{HMAC}^{\leftrightarrows}$ result) and then to XOR with `ipad` or `opad`. This again arises because of key material of length $\ell < b$ bits, to which (in contrast to $\mathsf{HMAC}^{\leftrightarrows}$) the XOR is applied in $\mathsf{HMAC}$. As before we will validate the new assumption through auxiliary proofs. We clarify that unlike in the swap-PRF case, here there is no restriction of the keys or messages to some subset $\mathcal{S}$; for both keys and messages, any length is allowed, as per the definition of the full $\mathsf{HMAC}$.

## 1.4 Auxiliary Contributions and Technical Overview

We obtain the above results via a modular approach that treats swap and conventional PRF security in a unified way. Along the way we give some definitions and auxiliary results of independent interest.

<u>Vkl-PRFs, a new definition.</u> Recall that the definition of $\mathsf{F}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ being PRF secure picks a key $X \leftarrow_{\$} \mathcal{X}$ at random. But for $\mathsf{HMAC}\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^c$, the keyspace, $\{0,1\}^*$, contains keys of many different lengths and it is not clear under what distribution a random one would be chosen. The first and natural answer, and the one assumed above, is that one has fixed a key length $\ell$ and is drawing a key from $\{0,1\}^\ell$ at random. But this fails to capture different users using $\mathsf{HMAC}$ with keys of different lengths, which occurs in practice.

These considerations lead us to introduce (in Section 3) a new definition, of a *variable key-length* PRF (vkl-PRF). The definition is inherently in the multi-user setting. The adversary can, for each user, adaptively pick a key length, and the game initializes the key for that user to a uniformly random string of the chosen length. The rest is as one would expect from the usual multi-user PRF setting [9]. (A subtle point is that we cannot expect security for too-short keys. This is handled by having theorems assume a minimum key length.)

Our proof of PRF security for $\mathsf{HMAC}$ (Theorem 10) actually proves vkl-PRF security. For swap-PRF security for $\mathsf{HMAC}[\mathcal{S}]$, Theorem 11 likewise proves vkl-PRF security of $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}\colon \{0,1\}^* \times \mathcal{S} \to \{0,1\}^c$. This means that in both cases we give a guarantee that is strong and better models real-world usage, namely that security holds even when different users use keys of different and adaptively-chosen lengths.

<u>Results for $\mathsf{NMAC}$.</u> As in prior works, we start with $\mathsf{NMAC}$ (Section 5). Beginning with the swap case, we prove in Theorem 4 that $\mathsf{NMAC}^{\leftrightarrows}\colon \{0,1\}^* \times \{0,1\}^{2c} \to \{0,1\}^c$ is vkl-PRF secure. Unlike for $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$, this $\mathsf{NMAC}$ result involves no restrictions of inputs to any set $\mathcal{S}$, but rather holds for all inputs for which the function is defined. The assumptions made are PRF security of $\mathsf{h}$ and $\Phi_{\mathsf{pad},a}$-rka-PRF security of $\mathsf{h}^{\leftrightarrows}$, with the proof leveraging a lemma (Lemma 1) that we give on the strong multi-user PRF security of the 2-tier cascade defined in [7]. Due to targeting vkl-PRF security, the result is directly in the multi-user setting, and the bound in Theorem 4 is good. That is, it does not degrade with the number of users.

Turning now to the conventional PRF security of $\mathsf{NMAC}$, the proofs in prior works [8, 5, 6, 30] are for the single-user setting, and it has remained open if one can show security in the multi-user setting with a bound that does not degrade with the number of users. We resolve this and give such a proof (Theorem 5). We follow the approach used in GPR [30] to prove single-user security, but while the latter relied on a lemma on random systems from [42], we give a quite simple, self-contained game-playing proof, establishing multi-user security.

<u>A dual composition theorem.</u> To lift the above results to $\mathsf{HMAC}$, we give a dual composition theorem (Theorem 6, in Section 6). $\mathsf{HMAC}(X, Y)$ can be seen as deriving keys $X_i \| X_o \leftarrow \mathsf{HSKD}(X)$ for $\mathsf{NMAC}$ via a key-derivation function $\mathsf{HSKD}$ (shown in Figure 3) and then returning $\mathsf{NMAC}(X_i \| X_o, Y)$. We write this as $\mathsf{HMAC} = \mathbf{Comp}[\mathsf{NMAC}, \mathsf{HSKD}]$. Theorem 6 implies that

(1) if $\mathsf{NMAC}$ is PRF secure and $\mathsf{HSKD}$ is a variable seed-length (vsl) PRG, then $\mathsf{HMAC} = \mathbf{Comp}[\mathsf{NMAC}, \mathsf{HSKD}]$ is vkl-PRF secure, and

(2) if $\mathsf{NMAC}^{\leftrightarrows}$ is vkl-PRF secure, and the restriction $\mathsf{HSKD}[\mathcal{S}]$ of $\mathsf{HSKD}$ to inputs in set $\mathcal{S}$ is collision resistant, then $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows} = \mathbf{Comp}[\mathsf{NMAC}, \mathsf{HSKD}[\mathcal{S}]]^{\leftrightarrows}$ is vkl-PRF secure.

Let us explain. It is folklore understanding that the composition preserves PRF security assuming PRG security of the key-derivation. Result (1) casts this in our variable key-length setting, which involves introducing the definition of a vsl-PRG, but we see the result as expected. Result (2) is more interesting, saying that the composition equally well preserves swap-PRF security if we switch the assumption on key-derivation to collision resistance Put together, we get a simply-stated, unified result, saying that the composition preserves dual-PRF security assuming key-derivation is a collision-resistant PRG.

<u>Security of the $\mathsf{HSKD}$ key-derivation function.</u> To complete the circle we need to establish the security assumed of $\mathsf{HSKD}$ in the composition theorems, which we also do in Section 6. Proposition 7 shows vsl-PRG security of $\mathsf{HSKD}$ assuming PRF security of $\mathsf{h}$ and $\mathsf{h}^{\leftrightarrows}$, and $\Phi_{\mathsf{zio},a}$-rka-PRF security of $\mathsf{h}^{\leftrightarrows}$. The function is *not* collision resistant on its full domain (leading to the above-discussed attacks on the swap-PRF security of $\mathsf{HMAC}$), but Proposition 8 shows collision resistance of $\mathsf{HSKD}[\mathcal{S}]$ for all feasible $\mathcal{S}$, assuming collision resistance of $\mathsf{h}$. Putting all the above together gives our results on the PRF and swap-PRF security of $\mathsf{HMAC}$.

<u>Validation of rka-PRF assumptions on $\mathsf{h}^{\leftrightarrows}$.</u> $\mathsf{HMAC}$ is mainly used with the $\mathsf{SHA}$-256, $\mathsf{SHA}$-384, and $\mathsf{SHA}$-512 hash functions. Here the compression function is a Davies–Meyer one,

$\mathsf{h}(X, L) = \mathsf{E}(L, X) \oplus X$ for a block cipher $\mathsf{E}\colon \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$. Resistance to related-key attacks is a commonly studied goal for block ciphers, see e.g. [14, 38, 36, 11, 48, 34, 15, 28, 55, 16, 37, 19, 18, 4], so we ask if this assumption implies ours. Curiously, being in the swap setting helps us here and enables such a reduction. The reason is that when considering rka-PRF security of $\mathsf{h}^{\leftrightarrows}$, the key $L$, which is the message for $\mathsf{h}$, becomes the key for $\mathsf{E}$. Exploiting this, Proposition 12 (in Section 7) shows that if $\mathsf{E}$ is $\Phi$-rka-PRP secure, then $\mathsf{h}^{\leftrightarrows}$ is also $\Phi$-rka-PRF secure.

These results are strengthened by introducing (in Section 3) an extension of the single-user rka-PRF/PRP definitions of [11] to a multi-user setting; the novel element is that the RKD functions have a user-dependent input. Finally, one must be careful that, due to attacks from [11], $\Phi$-rka-PRP security of the block cipher $\mathsf{E}$ does not hold for all $\Phi$, so we need to verify it for the functions we introduce and use. This final step is done through an analysis that models $\mathsf{E}$ as an ideal cipher. We stress that idealized assumptions are made only on $\mathsf{E}$ and limited to this one step; all other security proofs are in the standard model.

### Organization

Section 2 provides a more extensive discussion of related work. Section 3 introduces notation and definitions (including our new vkl-PRF notion). Section 4 introduces the modularization of HMAC functions we use in our analysis, as well as some basic results on cascades of $\mathsf{h}$. We then turn to our main results. Section 5 studies the dual-PRF security of NMAC. Based on this, Section 6 takes on the dual-PRF security of HMAC, along the way establishing the necessary properties of the subkey derivation function HSKD, including the characterization of the feasible key sets for which HMAC is swap-PRF secure. Finally, in Section 7, we validate the assumptions on $\mathsf{h}^{\leftrightarrows}$, showing that it is an rka-PRF for the RKD functions $\Phi_{\mathsf{pad},a}$ and $\Phi_{\mathsf{zio},a}$.

## 2 Related Work

<u>Dual-PRF assumptions on HMAC.</u> Several works [27, 24, 52, 17, 53, 35, 23] (as indicated above) have explicitly or implicitly assumed dual-PRF security of HMAC. For example, for TLS 1.3, [27, Section 2.4] says "we however need to deploy stronger assumptions which we recap here. The first assumption is concerned with the use of HMAC as a dual PRF (cf. [5, 6])." While the cited paper [5, 6] introduces dual-PRF security, it does not establish swap-PRF security for HMAC, and as we noted, not even conventional PRF security for all key sizes. For KEMTLS, SSW [52], in the context of their proof of Theorem 4.1, perform game hops "under the PRF-security or dual-PRF-security [5, 6] of HKDF", the latter in this mode is HMAC. In Appendix A, we give an account of where dual-PRF assumptions on HMAC show up in prior work and when they are supported by our results.

<u>HKDF.</u> The extraction mode of Krawczyk's HKDF [40] uses HMAC with a salt as first input and key material as second input. This has computational extraction properties under certain assumptions, including that the first input (the salt) is random [40]. However, it does not justify key-combiner usage of HMAC in protocols, because here the first input to HMAC may be adversarially influenced. Swap-PRF security of HMAC, in contrast, allows the first input to be adversarially chosen. This is where swap and dual-PRF security shows up in protocols built on HKDF, motivating our study.

We note that in HKDF's extraction mode, the key-material in the second input may be a non-uniformly-distributed source of entropy such as a Diffie–Hellman shared secret in a key exchange protocol, and HMAC is supposed to return a computationally uniform key. Our results do not cover such usage; in dual-PRF security, keys are assumed to be uniformly distributed.

<u>Multi-user security.</u> The setting of protocols like TLS involves many users and it is well understood that this is better modeled by multi-user (mu) security than by single-user security

(su). For most primitives, su implies mu via a hybrid argument, but the adversary advantage grows by a factor of the number $u$ of users [41]. This has led to dedicated analyses and schemes for many primitives, aiming to show mu security with good bounds, meaning degradation of the advantage with $u$ is avoided [21, 33, 7]. Our work follows this, aiming for, and obtaining, good bounds in the mu setting. In particular we fill a gap in the literature by showing PRF security of NMAC with good mu bounds in Theorem 5. We also show vkl-prf of HMAC with good mu bounds in Theorem 10.

Uniform and non-uniform assumptions. The conference version of Bellare's work [5] showed (su) PRF security of NMAC and HMAC (with good bounds) assuming PRF security of the compression function against non-uniform adversaries. The journal version [6] added proofs under uniform assumptions but with a degraded bound. The gap was filled by Gaži, Pietrzak and Rybár [30], who showed (su) PRF security with bounds as good as in [5] but under uniform assumptions. All our results use uniform assumptions and are shown with blackbox reductions. The proofs explicitly build adversaries against the assumptions from the given adversary against the target.

Building new dual-PRFs. A new template for building dual-PRFs was given in BL [12]. It combines a computational extractor with a collision-resistant function and can be instantiated to obtain dual-PRFs assuming collision-resistant functions or One-Way Permutations (OWP)s. The BL template was extended in ADKPRY [3] to add an output transformation, leading to further instantiations. However, what is in use in the real world as a dual-PRF is HMAC and thus we focus on it rather than on new constructions. We note that determining the dual-PRF security of HMAC is stated as an open question in ADKPRY [3] and resolved in our work.

On the theoretical side, while the existence of One-Way Functions (OWF)s is known to imply the existence of PRFs [32], it is not known whether OWFs imply dual-PRFs. This is an intriguing open question.

# 3 Notation and Definitions

We recall some notation, including for game-playing, recap some standard definitions, and then give our new ones, namely variable key-length PRFs (vkl-PRFs), variable seed-length PRGs (vsl-PRGs) and multi-user rka-PRF security.

## 3.1 Notation and Conventions

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of non-negative integers. Let $\varepsilon$ be the empty string. For $n \in \mathbb{N}$ let $\{0,1\}^n$ be the set of all strings of length $n$, $\{0,1\}^{>n}$ the set of all strings of length greater than $n$, and let $\{0,1\}^*$ be the set of all strings of any length $n \geq 0$. For $b \in \mathbb{N}$ let $\{0,1\}^{b*} = \{X \in \{0,1\}^* : |X| \bmod b = 0\}$ be the set of all strings whose length is a multiple of $b$. If $a$ is a string, then $|a|$ denotes its length, $a[i]$ denotes its $i$-th bit and $a[i..j] = a[i] \ldots a[j]$. (The last is the empty string $\varepsilon$ when $j < i$.) Similarly, if $\mathbf{X}$ is a vector then $|\mathbf{X}|$ denotes its length (the number of its coordinates) and $\mathbf{X}[i]$ its $i$-th element. The empty vector has length 0 and is also written $\varepsilon$. We let $\mathbf{X}[1..i]$ denote $(\mathbf{X}[1], \ldots, \mathbf{X}[i])$ for $1 \leq i \leq |\mathbf{X}|$. If $i = 0$ then $\mathbf{X}[1..i] = \varepsilon$. For a string $A \in \{0,1\}^{b*}$ the operator $\stackrel{b}{\leftarrow}$ splits it into $b$-bit blocks and places them in a vector. For example, if $|A| \geq 2b$, then $\mathbf{X} \stackrel{b}{\leftarrow} A$ results in $\mathbf{X}[1] = A[1..b]$ and $\mathbf{X}[2] = A[b+1..2b]$, etc.

We say that a set $\mathcal{S} \subseteq \{0,1\}^*$ is *length closed* if for every $n \in \mathbb{N}$ it is the case that $\mathcal{S} \cap \{0,1\}^n$ is either $\{0,1\}^n$ or $\emptyset$. This condition is made on key and message spaces. For sets $\mathcal{S}_1$ and $\mathcal{S}_2$, let $\mathsf{FUNC}[\mathcal{S}_1, \mathcal{S}_2]$ denote the set of all functions $\mathsf{f} \colon \mathcal{S}_1 \to \mathcal{S}_2$ and $\mathsf{PERM}[\mathcal{S}_1]$ the set of all permutations on $\mathcal{S}_1$. The shorthand $\mathcal{S}_1 \stackrel{\cup}{\leftarrow} \mathcal{S}_2$ denotes $\mathcal{S}_1 \leftarrow \mathcal{S}_1 \cup \mathcal{S}_2$.

For a finite set $S$, we let $x \leftarrow\$ S$ denote sampling $x$ uniformly at random from $S$. We let $y \leftarrow A(x_1, \ldots; r)$ denote executing algorithm $A$ on inputs $x_1, \ldots$ and coins $r$ and letting $y$ be the result. We let $y \leftarrow\$ A(x_1, \ldots)$ be the result of picking $r$ at random and letting $y \leftarrow A(x_1, \ldots; r)$. Algorithms are randomized unless otherwise indicated. Running time is worst case.

We use the code-based game playing framework of [13]. (See Fig. 1 for an example.) Games have procedures, also called oracles, with INIT and FINALIZE being optional. In executing an adversary $\mathcal{A}$ with a game G, oracle INIT, if present, executes first. Then the adversary, given the outputs of INIT, can query other oracles at will. If FINALIZE is present, only one query to it is allowed, and this must be the last query the adversary makes. The output of the execution is defined as the output of FINALIZE if the latter is present and the output of the adversary otherwise. By $\mathrm{G}(\mathcal{A}) \Rightarrow y$ we denote the event that the execution of game G with adversary $\mathcal{A}$ results in output $y$. We write $\Pr[\mathrm{G}(\mathcal{A})]$ as shorthand for $\Pr[\mathrm{G}(\mathcal{A}) \Rightarrow 1]$, the probability that the execution returns 1. We write $\mathbf{Q}^{\mathrm{OR}}(\mathcal{A})$ for the number of queries made by adversary $\mathcal{A}$ to an oracle OR in the game with which $\mathcal{A}$ is executed. Our convention is that running time of an adversary executed with some game includes the time for the game procedures to respond to oracle queries.

In writing game or adversary pseudocode, it is assumed that Boolean variables are initialized to 0, integer variables are initialized to 0, and set-valued variables are initialized to the empty set $\emptyset$. The distinguished symbol $\perp$ stands for "undefined" and is used as a placeholder value for uninitialized variables and to signal errors. Table entries are assumed initialized to $\perp$. Let [[cond]] denote the boolean (1 or 0) result of evaluating condition cond. For example, $[[d^* = 1]]$ returns 1 if variable $d^*$ equals 1, and 0 otherwise.

### 3.2 Standard Definitions

<u>COLLISION RESISTANCE.</u> A collision for a function $\mathsf{F}\colon \mathcal{X} \to \mathcal{Y}$ is a pair of distinct points $X, X' \in \mathcal{X}$ such that $\mathsf{F}(X) = \mathsf{F}(X')$. Let game $\mathbf{G}^{\mathrm{CR}}_{\mathsf{F}}$ consist only of procedure FINALIZE that given $X, X' \in \mathcal{X}$, returns 1 if $X, X'$ are a collision for $\mathsf{F}$, and 0 otherwise. We define the advantage of an adversary $\mathcal{A}$ against the CR security of $\mathsf{F}$ as $\mathbf{Adv}^{\mathrm{CR}}_{\mathsf{F}}(\mathcal{A}) = \Pr\left[\,\mathbf{G}^{\mathrm{CR}}_{\mathsf{F}}(\mathcal{A})\,\right]$. The probability is over the coins of the adversary.

To accurately model cryptographic hash functions like SHA-256, SHA-384, SHA-512, which are keyless, our syntax is also keyless. The theoretical issue of existence of an adversary that violates CR security by hard-wiring a collision into its code is circumvented in the usual way. Namely, those of our results which assume CR of some $\mathsf{F}$ give explicit constructions of CR adversaries based on the given adversary, so a practical attack against the scheme leads to a practical attack against $\mathsf{F}$. This was popularized as the "human ignorance" approach in [50].

<u>PRFS.</u> Let $\mathsf{F}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function family, where $\mathcal{X}$ and $\mathcal{Y}$ are length closed sets. We recall the definition of it being a PRF via the games on the left of Figure 1. They are parameterized by a bit $d \in \{0, 1\}$, with $d = 1$ indicating the "real" game and $d = 0$ the "random" game. We let $\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{PRF}\text{-}1}_{\mathsf{F}}(\mathcal{A})] - \Pr[\mathbf{G}^{\mathrm{PRF}\text{-}0}_{\mathsf{F}}(\mathcal{A})]$ be its advantage. This definition is in the multi-user setting [9], with the original single-user setting [31] recovered by considering adversaries making only one NEW query.

### 3.3 New Definitions

<u>VKL-PRFS.</u> We extend the usual definition of PRF security to allow keys of variable, adversary-determined length, in the multi-user setting. Let $\mathsf{F}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function family as above, where $\mathcal{X}$ and $\mathcal{Y}$ are length closed sets. Consider games $\mathbf{G}^{\mathrm{vkl\text{-}PRF}}_{\mathsf{F}}$ in the middle of Figure 1, parameterized by $d \in \{0, 1\}$ as above. The advantage of an adversary $\mathcal{A}$ is defined as $\mathbf{Adv}^{\mathrm{vkl\text{-}PRF}}_{\mathsf{F}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{vkl\text{-}PRF}\text{-}1}_{\mathsf{F}}(\mathcal{A})] - \Pr[\mathbf{G}^{\mathrm{vkl\text{-}PRF}\text{-}0}_{\mathsf{F}}(\mathcal{A})]$. Let us now explain. The game is in the multi-user setting. By calling NEW with a length $\ell$, the adversary initializes a new user with a key whose length $\ell$ is determined by the adversary. The game checks that $\ell$ is in the set of *allowed key lengths* $\mathsf{F}.\mathrm{KL}$, which we define to be $\mathsf{F}.\mathrm{KL} = \{\ell : \{0,1\}^\ell \cap \mathcal{X} \neq \emptyset\}$. That is, $\mathsf{F}.\mathrm{KL}$ is the set of integers which appears as lengths of keys in the key space of $\mathsf{F}$. The FN

| Game $\mathbf{G}_{\mathsf{F}}^{\text{PRF-}d}$: | Game $\mathbf{G}_{\mathsf{F}}^{\text{vkl-PRF-}d}$: | Game $\mathbf{G}_{\mathsf{G}}^{\text{vsl-PRG-}d}$: |
|---|---|---|
| <u>NEW</u><br>$n \leftarrow n+1$<br>$X_n \leftarrow\!\!\$\; \mathcal{X}$<br><br><u>FN$(i, Y)$</u><br>If $\mathrm{T}[i, Y] = \perp$ then:<br>  If $d = 1$ then:<br>    $\mathrm{T}[i, Y] \leftarrow \mathsf{F}(X_i, Y)$<br>  Else $\mathrm{T}[i, Y] \leftarrow\!\!\$\; \mathcal{Z}$<br>Return $\mathrm{T}[i, Y]$ | <u>NEW$(\ell)$</u><br>Assert: $\ell \in \mathsf{F}.\mathrm{KL}$<br>$n \leftarrow n+1$<br>$\ell_n \leftarrow \ell$ ; $X_n \leftarrow\!\!\$\; \{0,1\}^{\ell}$<br><br><u>FN$(i, Y)$</u><br>If $\mathrm{T}[i, Y] = \perp$ then:<br>  If $d = 1$ then:<br>    $\mathrm{T}[i, Y] \leftarrow \mathsf{F}(X_i, Y)$<br>  Else $\mathrm{T}[i, Y] \leftarrow\!\!\$\; \mathcal{Z}$<br>Return $\mathrm{T}[i, Y]$ | <u>FN$(\ell)$</u><br>Assert $\ell \in \mathsf{G}.\mathrm{KL}$<br>$s \leftarrow\!\!\$\; \{0,1\}^{\ell}$; $r_1 \leftarrow \mathsf{G}(s)$<br>$r_0 \leftarrow\!\!\$\; \mathcal{R}$<br>Return $r_d$ |

Figure 1: **Left and middle:** PRF and variable key-length PRF security games ($d \in \{0,1\}$) for function family $\mathsf{F}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$. **Right:** Variable seed-length PRG games ($d \in \{0,1\}$) for PRG $\mathsf{G}\colon \mathcal{S} \to \mathcal{R}$.

oracle as usual allows the adversary to obtain either results of $\mathsf{F}$ on a chosen message under a previously initialized key, or random strings. Note that the lengths of keys can be determined by the adversary adaptively.

<u>MINIMAL AND MAXIMAL KEY LENGTH.</u> The vkl-PRF game allows keys of any length. In particular, it does not preclude even a one-bit key, as long as such keys are in the keyspace of $\mathsf{F}$. But clearly, if very short keys are allowed, no practically useful security is achieved. This is not a problem with the definition or to be dealt with here; it is handled, rather, in theorems. There, we will as usual consider various resource parameters for an adversary $\mathcal{A}$, such as the number of queries to the oracles. Now, additionally, there will be a *minimal key length* $\ell_{\mathsf{min}}(\mathcal{A})$, defined as the shortest $\ell$ for which it makes a NEW$(\ell)$ query. Bounds in the theorem statement will be a function of $\ell_{\mathsf{min}}(\mathcal{A})$. In practice, we would then ask that applications use key lengths for which the bounds are good, which in particular will dictate large-enough choices of minimal key length. We may also speak of a *maximal key length*, denoted $\ell_{\mathsf{max}}(\mathcal{A})$, of a given adversary, which affects the resources of adversaries constructed from it.

<u>VSL-PRGS.</u> A variable seed-length pseudorandom generator (PRG) is a function $\mathsf{G} : \mathcal{S} \to \mathcal{R}$. We assume that the seed space $\mathcal{S}$ is length closed and let the set of allowed seed lengths of $\mathsf{G}$ be $\mathsf{G}.\mathrm{KL} = \{\ell \,:\, \{0,1\}^{\ell} \cap \mathcal{S} \neq \emptyset\}$.

Analogously to vkl-PRF security, we extend the usual definition of PRG security [20, 54] to allow seeds of variable, adversary-determined length. The games defining this new notion are on the right of Figure 1, again for $d \in \{0,1\}$. We let the advantage of an adversary $\mathcal{A}$ against the vsl-PRG security of $\mathsf{G}$ be defined as $\mathbf{Adv}_{\mathsf{G}}^{\text{vsl-PRG}}(\mathcal{A}) = \Pr[\mathbf{G}_{\mathsf{G}}^{\text{vsl-PRG-}1}(\mathcal{A})] - \Pr[\mathbf{G}_{\mathsf{G}}^{\text{vsl-PRG-}0}(\mathcal{A})]$.

As for vkl-PRFs, the vsl-PRG security provided by $\mathsf{G}$ in some usage will depend crucially on the length of the shortest allowed seed. To quantify, we define the *minimal seed length* $\ell_{\mathsf{min}}(\mathcal{A})$ of an adversary $\mathcal{A}$ to be the shortest $\ell$ for which the adversary makes a FN$(\ell)$ query. Also the *maximal seed length* $\ell_{\mathsf{max}}(\mathcal{A})$, defined as the longest $\ell$ for which the adversary makes a FN$(\ell)$ query, will arise in resource considerations.

<u>RKA DEFINITIONS.</u> We extend the definitions of PRF and PRP security under related-key attacks [11] (rka-PRF resp. rka-PRP) to the multi-user setting, beginning with rka-PRF.

Let $\mathsf{F}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function family, and $\Phi : \Lambda_1 \times \Lambda_2 \times \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ a function. We call $\Phi$ the *related-key-deriving* function. It takes as input two parameters, which specify the transformation that will be applied to the key $X \in \mathcal{X}$. The multi-user aspect is captured by

| Game $\mathbf{G}^{\text{rka-PRF-}d}_{\Phi,\mathsf{F}}$: | Game $\mathbf{G}^{\text{rka-PRP-}d}_{\Phi,\mathsf{E}}$: |
|---|---|
| <u>New($\alpha$)</u> | <u>New($\alpha$)</u> |
| $n \leftarrow n+1$; $X_n \leftarrow\$ \mathcal{X}$; $\alpha_n \leftarrow \alpha$ | $n \leftarrow n+1$ ; $L_n \leftarrow\$ \mathcal{K}$; $\alpha_n \leftarrow \alpha$ |
| | For all $L \in \mathcal{K}$ do: |
| | $\quad \Pi_{n,L} \leftarrow\$ \mathsf{PERM}[\mathcal{X}]$ |
| <u>Fn($i, \beta, Y$)</u> | <u>Fn($i, \beta, X$)</u> |
| Assert: $\Phi_{\alpha_i,\beta}(X_i) \neq \perp$ | Assert: $\Phi_{\alpha_i,\beta}(L_i) \neq \perp$ |
| $X' \leftarrow \Phi_{\alpha_i,\beta}(X_i)$ // Derived key | $L' \leftarrow \Phi_{\alpha_i,\beta}(L_i)$ // Derived key |
| If $\mathrm{T}[i, X', Y] \neq \perp$ then: | If $d = 1$ then: |
| $\quad$ Return $\mathrm{T}[i, X', Y]$ | $\quad$ Return $\mathsf{E}(L', X)$ |
| If $d = 1$ then: $\mathrm{T}[i, X', Y] \leftarrow \mathsf{F}(X', Y)$ | Else return $\Pi_{i,L'}(X)$ |
| Else $\mathrm{T}[i, X', Y] \leftarrow\$ \mathcal{Z}$ | |
| Return $\mathrm{T}[i, X', Y]$ | |

Figure 2: **Left:** Related-key attack PRF security game ($d \in \{0, 1\}$) for function family $\mathsf{F}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ and RKD function $\Phi$. **Right:** Related-key attack PRP security game ($d \in \{0, 1\}$) for a block cipher $\mathsf{E}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ and RKD function $\Phi$.

---

the first parameter, $\alpha \in \Lambda_1$, which will depend on the user. The second parameter $\beta \in \Lambda_2$ in turn specifies the function within the function family belonging to each user. Mapping back to [11], for each $\alpha \in \Lambda_1$ and $\beta \in \Lambda_2$, we let $\Phi_{\alpha,\beta} : \mathcal{X} \to \mathcal{X}$ be the related-key deriving function $\Phi(\alpha, \beta, \cdot)$.

The rka-PRF security game $\mathbf{G}^{\text{rka-PRF-}d}_{\Phi,\mathsf{F}}$ ($d \in \{0, 1\}$) shown in Figure 2, provides the adversary with two oracles: New and Fn. Oracle New takes as input $\alpha \in \Lambda_1$, setting the RKD parameter $\alpha_i$ associated to the new user instance $i$. Oracle Fn takes as input a user index $i$, a $\beta \in \Lambda_2$, and a function input $Y \in \mathcal{Y}$. It returns $\mathsf{F}(\Phi_{\alpha_i,\beta}(X_i), Y)$ if $d = 1$ or a consistently sampled random string from $\mathcal{Z}$ if $d = 0$. We define the advantage of an adversary $\mathcal{A}$ against the PRF security under $\Phi$-restricted related-key attacks as $\mathbf{Adv}^{\text{rka-PRF}}_{\Phi,\mathsf{F}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{rka-PRF-1}}_{\Phi,\mathsf{F}}(\mathcal{A})] - \Pr[\mathbf{G}^{\text{rka-PRF-0}}_{\Phi,\mathsf{F}}(\mathcal{A})]$.

Analogously, we extend the single-user rka-PRP definition from [11] to the multi-user/key setting. For a block cipher $\mathsf{E}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$, the resulting game $\mathbf{G}^{\text{rka-PRP-}d}_{\Phi,\mathsf{E}}$ ($d \in \{0, 1\}$) is shown on the right in Figure 2, and the corresponding advantage of an adversary $\mathcal{A}$ defined as $\mathbf{Adv}^{\text{rka-PRP}}_{\Phi,\mathsf{E}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{rka-PRP-1}}_{\Phi,\mathsf{E}}(\mathcal{A})] - \Pr[\mathbf{G}^{\text{rka-PRP-0}}_{\Phi,\mathsf{E}}(\mathcal{A})]$.

## 4 Background and Modularization of HMAC

Our analyses rely on seeing HMAC as built from lower-level functions in a particular modular way. Here we present this view of HMAC. Then we state results on some tools that we will use in our proofs.

<u>Padding.</u> We fix integers $c, b$ representing the chaining-variable length and block length, respectively, and assume $b > c$. HMAC-related functions may pad inputs to lengths a multiple of $b$. We discuss the padding in some detail because it matters to ensure that the modular and standard definitions of HMAC are indeed the same. Also it needs to be handled explicitly in analyses of swap-PRF security. (This was not the case for prior analyses of PRF security [8, 5, 6, 30].)

We fix a padding function $\mathsf{pad}$, parameterized by a maximum encoding length $le \in \mathbb{N}$. Function $\mathsf{pad}$ takes as input a length $\ell < 2^{le}$ and returns a string $\mathsf{pad}(\ell)$ such that $\ell + |\mathsf{pad}(\ell)|$ is a positive multiple of $b$. The canonical method is $\mathsf{pad}(\ell) = 10^*\langle\ell\rangle$ where $\langle\ell\rangle$ is the encoding of integer $\ell$ in $le$ bits and $0^*$ denotes the minimum number $i$ of zeros to ensure that $\ell + 1 + i + le$ is a multiple of $b$, where the length encoding $le$ is another parameter. Thus $|\mathsf{pad}(\ell)|$ is always in the range $\{1 + le, \ldots, b + le\}$. For $X \in \{0, 1\}^*$ we denote by $\overline{X} = X \| \mathsf{pad}(|X|) \in \{0, 1\}^{b*}$.

$\underline{\mathsf{h}^*(L,S)}$ // **Cascade of compression function** $\mathsf{h}$

$\triangleright\ \mathsf{h}^*\colon \{0,1\}^c \times \{0,1\}^{b*} \to \{0,1\}^c$

$\mathbf{S} \xleftarrow{b} S$ ; $n \leftarrow |\mathbf{S}|$

$C_0 \leftarrow L$ ; For $i = 1$ to $n$ do: $C_i \leftarrow \mathsf{h}(C_{i-1}, \mathbf{S}[i])$

Return $C_n$

$\underline{\mathsf{H}(M)}$ // **Hash function** $\mathsf{H} = \mathbf{MD}[\mathsf{h}]$ [44, 25]

$\triangleright\ \mathsf{H}\colon \{0,1\}^* \to \{0,1\}^c$

Return $\mathsf{h}^*(\mathtt{IV}, M \,\|\, \mathsf{pad}(|M|))$

$\underline{\mathsf{NMAC}(K_i\|K_o, M)}$ // **NMAC** [8]

$\triangleright\ \mathsf{NMAC}\colon \{0,1\}^{2c} \times \{0,1\}^* \to \{0,1\}^c$

$X \leftarrow \mathsf{h}^*(K_i, M\|\mathsf{pad}(b + |M|))$ ; Return $\mathsf{h}(K_o, X \,\|\, \mathsf{pad}(b + c))$

$\underline{\mathsf{HSKD}_b(K_b)}$ // **Subkey derivation function of** $\mathsf{HMAC}_b$

$\triangleright\ \mathsf{HSKD}_b\colon \{0,1\}^b \to \{0,1\}^{2c}$

$K_i \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{ipad})$ ; $K_o \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{opad})$ ; Return $K_i\|K_o$

$\underline{\mathsf{HMAC}_b(K_b, M)}$ // **HMAC with $b$-bit keys** [8]

$\triangleright\ \mathsf{HMAC}_b\colon \{0,1\}^b \times \{0,1\}^* \to \{0,1\}^c$

$K_i\|K_o \leftarrow \mathsf{HSKD}_b(K_b)$ ; Return $\mathsf{NMAC}(K_i\|K_o, M)$

$\underline{\mathsf{PoH}_b(K)}$ // **Derive a $b$-bit key for** $\mathsf{HSKD}_b$ **from an** $\mathsf{HMAC}$ **key** $K \in \{0,1\}^*$

$\triangleright\ \mathsf{PoH}_b\colon \{0,1\}^* \to \{0,1\}^b$

If $|K| \le b$ then $K_b \leftarrow K \,\|\, 0^{b-|K|}$ else $K_b \leftarrow \mathsf{H}(K) \,\|\, 0^{b-c}$

Return $K_b$

$\underline{\mathsf{HSKD}(K)}$ // **Subkey derivation function of** $\mathsf{HMAC}$

$\triangleright\ \mathsf{HSKD}\colon \{0,1\}^* \to \{0,1\}^{2c}$

$K_b \leftarrow \mathsf{PoH}_b(K)$ ; $K_i\|K_o \leftarrow \mathsf{HSKD}_b(K_b)$ ; Return $K_i\|K_o$

$\underline{\mathsf{HMAC}(K, M)}$ // **"Full" HMAC** [39, 46]

$\triangleright\ \mathsf{HMAC}\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^c$

$K_i\|K_o \leftarrow \mathsf{HSKD}(K)$ ; Return $\mathsf{NMAC}(K_i\|K_o, M)$

Figure 3: HMAC and friends built from compression function $\mathsf{h}\colon \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$.

Concretely, for SHA-256, the parameters are $(c, b, le) = (256, 512, 64)$ and for SHA-384 and SHA-512 they are $(c, b, le) = (512, 1024, 128)$. We assume $b - c > le$, as is true for SHA-256, SHA-384, and SHA-512, so that if $|M| \bmod b \le c$ then for any $\ell$ its padded version $M\|\mathsf{pad}(\ell)$ is $\lceil |M|/b \rceil$ blocks long, meaning no extra block of padding is created. In particular, if $|M| \le c$ then $M\|\mathsf{pad}(\ell)$ is just one block, which is relevant in ensuring that the modular form of HMAC written below matches with the original.

<u>Message and key lengths.</u> The upper limit of $le$ on the binary length of an input $\ell$ to pad means that the maximum length of an input to the hash function H is $Le = 2^{le}$. This translates into restrictions on the maximum length of both keys and messages for HMAC. However in practice $Le$ is very large (at least $2^{64}$ in the above examples), making this a theoretical limitation. Thus, in the remainder of this paper we write $\{0,1\}^*$ for domains of functions that are actually limited to inputs of lengths $Le$. It is understood that in any theorems about these functions, there is an implicit assumption that adversary queries do not exceed the maximum allowed lengths.

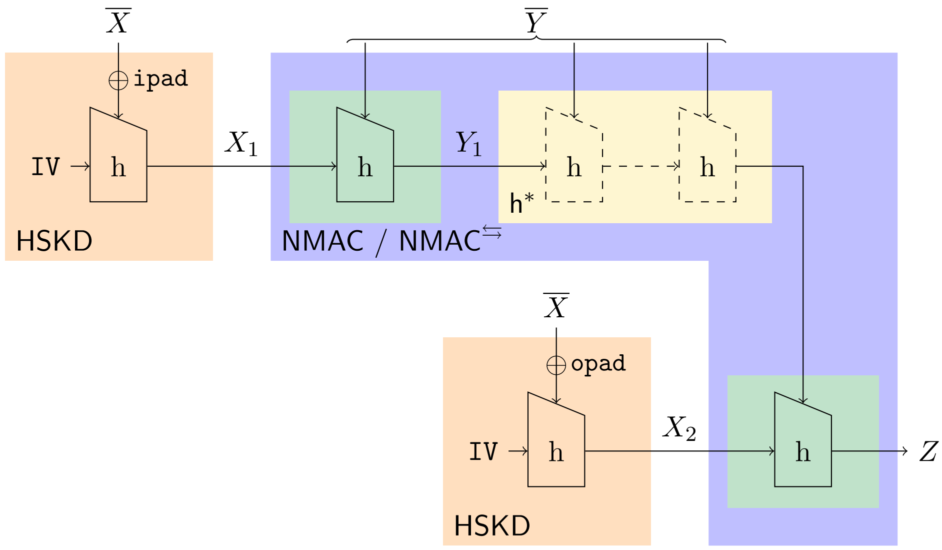



Figure 4: Illustration of how we modularize $\mathsf{HMAC}(X, Y)$ and its swap $\mathsf{HMAC}^{\leftrightarrows}(Y, X)$ as the composition of subkey derivation function $\mathsf{HSKD}$ (highlighted in orange) and $\mathsf{NMAC}(X_1 \,\|\, X_2, Y)$ resp. $\mathsf{NMAC}^{\leftrightarrows}(Y, X_1 \,\|\, X_2)$ (in blue). In turn, $\mathsf{NMAC}$ and $\mathsf{NMAC}^{\leftrightarrows}$ are modularized as the composition of $\mathsf{h}$ (in green) and $\mathsf{h}^*$ (in yellow). The dashed $\mathsf{h}$ invocations may be omitted depending on the length of $\overline{Y}$.

---

<u>MODULARIZATION OF HMAC FUNCTIONS.</u> The starting point is a compression function $\mathsf{h}\colon \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ taking a $c$-bit chaining variable and $b$-bit block to return a $c$-bit result. Figure 3 now specifies various functions that we will consider; for an illustration of how we modularize $\mathsf{HMAC}$, see also Figure 4.

The cascade $\mathsf{h}^*\colon \{0,1\}^c \times \{0,1\}^{b*} \to \{0,1\}^c$ [9] takes a $c$-bit key $L$ and an input $S$ whose length is assumed a multiple of $b$, and returns a $c$-bit output as shown. Note that $S = \varepsilon$ could be the empty string, in which case $n = 0$ in the code and thus what is returned is $C_0$, meaning $\mathsf{h}^*(L, \varepsilon) = L$.

Fixing an initial vector $\mathtt{IV} \in \{0,1\}^c$, one can then define the hash function $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^c$ as per the Merkle–Damgård transform [44, 25], $\mathsf{H}(M) = \mathsf{h}^*(\mathtt{IV}, M \,\|\, \mathsf{pad}(|M|))$.

Next is $\mathsf{NMAC}\colon \{0,1\}^{2c} \times \{0,1\}^{b*} \to \{0,1\}^c$ [8], an abstraction useful in the study of $\mathsf{HMAC}$. Its $2c$-bit key is viewed as split into an "inner" key $K_i$ and "outer" key $K_o$. Here $M \in \{0,1\}^*$ needs to be padded to length a multiple of $b$ to run $\mathsf{h}^*$. The obvious padding $\overline{M}$ would append $\mathsf{pad}(|M|)$. Instead, we define $\mathsf{NMAC}$ to pad $M$ with $\mathsf{pad}(b + |M|)$ and similarly pad $X$ with $\mathsf{pad}(b + c)$ rather than $\mathsf{pad}(c)$. As explained below, this is to accurately capture $\mathsf{HMAC}_b$ and $\mathsf{HMAC}$, where the $b$-bit keys are prepended to the message resp. $X$, adding to the overall hash function input length. We note that traditionally $\mathsf{NMAC}$ has been defined and analyzed as taking unpadded messages in $\{0,1\}^{b*}$ [8, 5, 6, 30], which is sufficient when considering PRF security, but for swap-PRF security the padding will matter.

The first, basic version of $\mathsf{HMAC}$ is $\mathsf{HMAC}_b\colon \{0,1\}^b \times \{0,1\}^* \to \{0,1\}^c$ which takes a $b$-bit key $K_b$. It is defined as $\mathsf{HMAC}_b(K_b, M) = \mathsf{H}((K_b \oplus \mathtt{opad}) \| \mathsf{H}((K_b \oplus \mathtt{ipad}) \| M))$, where $\mathtt{ipad}$, $\mathtt{opad} \in \{0,1\}^b$ are distinct constants defined in [8, 39, 46]. The modular rendition first applies the shown subkey derivation function $\mathsf{HSKD}_b\colon \{0,1\}^b \to \{0,1\}^{2c}$ to the key and then calls $\mathsf{NMAC}$ on the derived keys and the message, again as shown. It is in order for the main and modular forms here to indeed be the same function that $\mathsf{NMAC}$ is carefully defined to pad in a way that accommodates an extra first block.

Finally, to obtain "full" $\mathsf{HMAC}$, consider the subkey derivation function $\mathsf{HSKD}\colon \{0,1\}^* \to \{0,1\}^{2c}$. It first maps an arbitrary-length key $K$ down to a $b$-bit key $K_b$ via $\mathsf{PoH}_b$ and then maps $K_b$ to an $\mathsf{NMAC}$ key $K_i \| K_o$ via $\mathsf{HSKD}_b$. The definition of $\mathsf{HMAC}\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^c$

from [8, 39, 46] is $\mathsf{HMAC}(K, M) = \mathsf{HMAC}_b(\mathsf{PoH}_b(K), M) = \mathsf{H}((\mathsf{PoH}_b(K) \oplus \mathtt{opad}) \| \mathsf{H}((\mathsf{PoH}_b(K) \oplus \mathtt{ipad}) \| M))$. Figure 3 shows the modular form. Again, the choice of padding in the definition of NMAC is important to ensure that the modular and standard forms of HMAC are indeed the same function.

<u>Lemmas on cascade and 2-tier cascade.</u> Our proofs will use the 2-tier cascade, a generalization of the cascade from [7]. In addition to the function family $\mathsf{h}\colon \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ underlying $\mathsf{h}^*$, we have another function family $\mathsf{f}\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^c$. The 2-tier cascade associated to $\mathsf{f}, \mathsf{h}$ is the function family $\mathsf{2CSC}[\mathsf{f}, \mathsf{h}]\colon \mathcal{K} \times (\mathcal{X} \times \{0,1\}^{b*}) \to \{0,1\}^c$ defined as follows:

$\underline{\mathsf{2CSC}[\mathsf{f}, \mathsf{h}](J, (A, S))}$ // $J \in \mathcal{K}$ and $(A, S) \in \mathcal{X} \times \{0,1\}^{b*}$
$L \leftarrow \mathsf{f}(J, A)$ ; $X \leftarrow \mathsf{h}^*(L, S)$ ; Return $X$

Let $\mathcal{A}$ be a PRF adversary attacking $\mathsf{2CSC}[\mathsf{f}, \mathsf{h}]$. We say that it is prefix-free if for all $i$ there do not exist distinct $(A, S)$ and $(A', S')$ among the $\textsc{Fn}(i, \cdot)$ queries such that $A = A'$ and $S$ is a prefix of $S'$. (Note that there are no restrictions on queries across different users $i$.) This is a necessary condition for PRF security. The following says that it is also sufficient, and furthermore the result is in the multi-user setting with good bounds, meaning security does not degrade linearly with the number of users. The proof is a simple extension of a proof in [7] and is given in Appendix B.

**Lemma 1** (PRF security of 2CSC)**.** *Let* $\mathsf{f}\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^c$ *and* $\mathsf{h}\colon \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ *be function families. Let* $\mathsf{2CSC}\colon \mathcal{K} \times (\mathcal{X} \times \{0,1\}^{b*}) \to \{0,1\}^c$ *be the 2-tier cascade function family associated to* $\mathsf{f}, \mathsf{h}$ *as above. Let* $\mathcal{A}_{\mathsf{2CSC}}$ *be a prefix-free adversary against the PRF security of* 2CSC*. Assume the second components of each pair in the second argument of its queries to* $\textsc{Fn}$ *have at most* $n$ *blocks. Then we can construct adversaries* $\mathcal{A}_{\mathsf{f}}, \mathcal{A}_{\mathsf{h}}$ *such that*

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{2CSC}}(\mathcal{A}_{\mathsf{2CSC}}) \le \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{f}}(\mathcal{A}_{\mathsf{f}}) + n \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}). \tag{1}$$

*Adversary* $\mathcal{A}_{\mathsf{f}}$ *makes* $\mathbf{Q}^{\textsc{New}}(\mathcal{A}_{\mathsf{2CSC}})$ *queries to* $\textsc{New}$ *and* $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{2CSC}})$ *to* $\textsc{Fn}$*. Adversary* $\mathcal{A}_{\mathsf{h}}$ *makes at most* $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{2CSC}})$ *queries to* $\textsc{New}$ *and* $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{2CSC}})$ *to* $\textsc{Fn}$*. The running times of the constructed adversaries are about the same as that of* $\mathcal{A}_{\mathsf{2CSC}}$*.*

The cascade $\mathsf{h}^*$ itself is known to be a PRF under prefix-free queries assuming $\mathsf{h}$ is a PRF [9]. (And attacks show prefix-freeness is necessary.) This is in the single-user setting, which implies multi-user security with a loss in advantage that is a factor of the number of users. The following lemma gives instead a good bound, not degrading with the number of users. Here a PRF adversary $\mathcal{A}$ against $\mathsf{h}^*$ is said to be prefix-free if for all $i$ there do not exist distinct $X$ and $X'$ among the $\textsc{Fn}(i, \cdot)$ queries such that $X$ is a prefix of $X'$. This is obtained as a simply corollary of Lemma 1 using the technique of [7] of setting the first tier of the 2-tier cascade to a random function. For completeness details are in Appendix B, where we also discuss related work and techniques.

**Lemma 2** (Multi-user PRF security of $\mathsf{h}^*$)**.** *Let* $\mathsf{h}\colon \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^b$ *be a function family. Let* $\mathsf{h}^*\colon \{0,1\}^c \times \{0,1\}^{b*} \to \{0,1\}^c$ *be the cascade function family associated to* $\mathsf{h}$ *as per Figure 3. Let* $\mathcal{A}_{\mathsf{h}^*}$ *be a prefix-free adversary against the PRF security of* $\mathsf{h}^*$*. Assume that the second arguments of its queries to* $\textsc{Fn}$ *have at most* $n$ *blocks. Then we can construct an adversary* $\mathcal{A}_{\mathsf{h}}$ *such that*

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}^*}(\mathcal{A}_{\mathsf{h}^*}) \le n \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) \,. \tag{2}$$

*Adversary* $\mathcal{A}_{\mathsf{h}}$ *makes* $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{h}^*})$ *queries to* $\textsc{New}$ *and* $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{h}^*})$ *to* $\textsc{Fn}$*. The running time of* $\mathcal{A}_{\mathsf{h}}$ *is about the same as that of* $\mathcal{A}_{\mathsf{h}^*}$*.*

<u>Adversary merging lemma.</u> Sometimes we want to merge two PRF adversaries into a single PRF adversary whose advantage is a desired weighted sum of the advantages of the given adversaries. The following shows how this works.

**Lemma 3** (Merging Lemma)**.** *Let* $\mathsf{F}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ *be a function family. Let* $w_1, w_2 \geq 1$ *be integers. Let* $\mathcal{A}_1, \mathcal{A}_2$ *be PRF adversaries. Then we can build a PRF adversary* $\mathcal{A}$ *such that*

$$w_1 \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathcal{A}_1) + w_2 \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathcal{A}_2) = (w_1 + w_2) \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathcal{A}) \ . \tag{3}$$

*Resources translate as* $\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A}) = \max(\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A}_1), \mathbf{Q}^{\mathrm{Fn}}(\mathcal{A}_2))$ *and* $\mathbf{Q}^{\mathrm{New}}(\mathcal{A}) = \max(\mathbf{Q}^{\mathrm{New}}(\mathcal{A}_1), \mathbf{Q}^{\mathrm{New}}(\mathcal{A}_2))$*, and the running time of* $\mathcal{A}$ *is the maximum of the running times of* $\mathcal{A}_1, \mathcal{A}_2$ *plus overhead linear in the lengths of* $w_1, w_2$*.*

*Proof of Lemma 3.* Adversary $\mathcal{A}$ picks $v \leftarrow_{\$} \{1, \ldots, w_1 + w_2\}$. If $v \leq w_1$ then it lets $i \leftarrow 1$ and otherwise lets $i \leftarrow 2$. It now runs $\mathcal{A}_i$, responding to its oracle queries with its own oracles, and returns whatever $\mathcal{A}_i$ returns. Then for both $d = 0$ and $d = 1$ we have

$$\Pr[\mathbf{G}^{\text{PRF-}d}_{\mathsf{F}}(\mathcal{A})] = \frac{w_1}{w_1 + w_2} \cdot \Pr[\mathbf{G}^{\text{PRF-}d}_{\mathsf{F}}(\mathcal{A}_1)] + \frac{w_2}{w_1 + w_2} \cdot \Pr[\mathbf{G}^{\text{PRF-}d}_{\mathsf{F}}(\mathcal{A}_2)] \ .$$

Subtracting case $d = 0$ from $d = 1$, we get

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathcal{A}) = \frac{w_1}{w_1 + w_2} \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathcal{A}_1) + \frac{w_2}{w_1 + w_2} \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathcal{A}_2) \ .$$

Re-arranging terms yields Equation (3). $\square$

While the running time of $\mathcal{A}$ is, as stated, about the max of the individual running times of $\mathcal{A}_1$ and $\mathcal{A}_2$, a blackbox use of the lemma will result in its code (description) size being the sum of the individual ones. Put another way, if we use circuit size rather than running time as the metric, then the circuit size of $\mathcal{A}$ is the sum of those of $\mathcal{A}_1$ and $\mathcal{A}_2$, plus some overhead. This means that it isn't meaningful to apply Lemma 3 more than a constant number of times. In our usages, we will apply it at most twice. But also, in our usages, the doubling in code size is avoided because $\mathcal{A}_1, \mathcal{A}_2$ are both built from some underlying adversary $\mathcal{B}$ that is the same in both cases, and each adds little overhead to $\mathcal{B}$, so the code (or circuit) for the merged $\mathcal{A}$ can include just one copy of $\mathcal{B}$ and have size about the max of the sizes of $\mathcal{A}_1, \mathcal{A}_2$.

# 5 Dual-PRF Security of NMAC

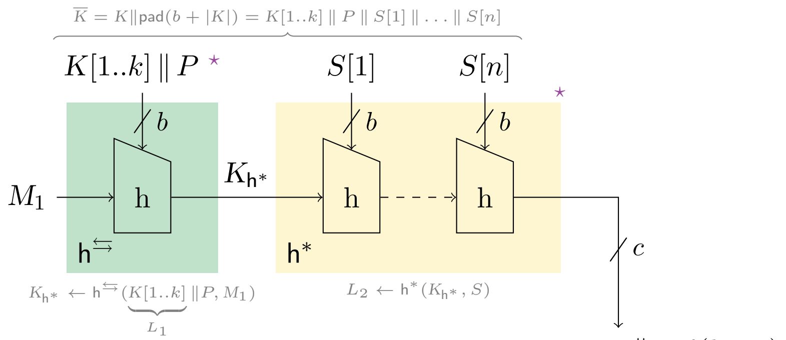
We now have all the definitions and modularizations in place to turn to our main results, on NMAC (in this section) and, based on these, HMAC (in the next). To understand the PRF security of $\mathsf{HMAC}^{\leftrightarrows}$ and HMAC, it is instructive to start with their (swapped) core, i.e., $\mathsf{NMAC}^{\leftrightarrows}$ and NMAC. Recall from Figure 3 that NMAC takes two keys $K_i \,\|\, K_o$ of length $c$ bits each and a message $M \in \{0,1\}^*$ as input. It first processes the padded message using the cascade $\mathsf{h}^*$ keyed with $K_i$, then applies $\mathsf{h}$ keyed with $K_o$ to the cascade's padded output.

## 5.1 vkl-PRF Security of $\mathsf{NMAC}^{\leftrightarrows}$

Swapping key and message input means that $\mathsf{NMAC}^{\leftrightarrows}$ takes as input a single arbitrary-length key $K$ and two (concatenated) messages $M_1 \,\|\, M_2$ of length $c$ bits each. The latter "key" the upper and the lower cascade, the former is padded to $K \| \mathsf{pad}(b + |K|)$ and enters the upper cascade through the block inputs.

It is conducive to single out the first and last invocation of $\mathsf{h}$: the first takes the leading (up to $b$) bits of the key $K$ (possibly padded), the last the (padded) output of the cascade, through the block input. In our security analysis, we will treat both as invocations of $\mathsf{h}^{\leftrightarrows}$ and rely on rka-PRF security to capture the padding. Figure 5 shows $\mathsf{NMAC}^{\leftrightarrows}$ modularized in this way.

<u>Security of $\mathsf{NMAC}^{\leftrightarrows}$.</u> We show that $\mathsf{NMAC}^{\leftrightarrows}$ is a vkl-PRF if $\mathsf{h}^{\leftrightarrows}$ is rka-PRF secure and $\mathsf{h}$ is a PRF. The RKD functions of interest for $\mathsf{h}^{\leftrightarrows}$ are those which allow the adversary to overwrite a




$\underline{\mathsf{NMAC}^{\leftrightarrows}(K, M_1 \| M_2)}$ // $K \in \{0,1\}^*$ and $M_1, M_2 \in \{0,1\}^c$

1 $k \leftarrow \min(|K|, b)$ ; $\overline{K} \leftarrow K \| \mathsf{pad}(b + |K|)$
2 $L_1 \leftarrow K[1..k]$ ; $P \leftarrow \overline{K}[k{+}1..b]$ ; $S \leftarrow \overline{K}[b{+}1..|\overline{K}|]$
3 $K_{\mathsf{h}^*} \leftarrow \mathsf{h}^{\leftrightarrows}(L_1 \| P, M_1)$
4 $L_2 \leftarrow \mathsf{h}^*(K_{\mathsf{h}^*}, S)$
5 $Z \leftarrow \mathsf{h}^{\leftrightarrows}(L_2 \| \mathsf{pad}(b + c), M_2)$
6 Return $Z$

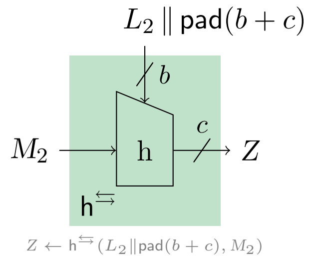



Figure 5: Illustration of and code for $\mathsf{NMAC}^{\leftrightarrows}$, viewed as the composition of $\mathsf{h}^{\leftrightarrows}$ (highlighted in green) and $\mathsf{h}^*$ (in yellow). The purple asterisks ⋆ indicate that the $P$ part of the key padding may be empty (namely, if $k = b$) or that the cascade G may be omitted (if $|\overline{K}| = b$).

---

suffix of the full, block-length key with padding. In the reduction, we will use this to simulate the padding in lines 3 and 5 of Figure 5.

We define $\Phi_{\mathsf{pad},a} : \{0, \ldots, b\} \times \{\varepsilon\} \times \{0,1\}^b \to \{0,1\}^b \cup \{\perp\}$, on input $\ell, \varepsilon, K$, to return $\perp$ if $\ell < a$ and $K[1..\ell] \parallel \mathsf{pad}(b + \ell)[1..b - \ell]$ otherwise. In more detail:

$\underline{\Phi_{\mathsf{pad},a}(\ell, \varepsilon, K)}$
If $\ell < a$ then return $\perp$
$P \leftarrow \mathsf{pad}(b + \ell)[1..b - \ell]$ ; $L \leftarrow K[1..\ell] \parallel P$
Return $L$

That is, $\Phi_{\mathsf{pad},a}(\ell, \varepsilon, K)$ overwrites the suffix of $K$ with the first $b - \ell$ bits of padding specified by $\mathsf{pad}(b + \ell)$, where pad is defined as per Section 4. The function requires that $\ell \geq a$, otherwise the output is $\perp$. That is, at least $a$ bits of key prefix are left intact in the transformation. Note that the second input to $\Phi_{\mathsf{pad},a}$ is empty, meaning that for each user, there is only a single RKD function; namely, the one applying the appropriate-length padding.

We justify the assumption that $\mathsf{h}^{\leftrightarrows}$ is a PRF under related-key attacks when restricted to $\Phi_{\mathsf{pad},a}$ in Section 7.

**Theorem 4** (vkl-PRF security of $\mathsf{NMAC}^{\leftrightarrows}$)**.** *Let $c$, $b$, and* pad *be as in Section 4, let* $\mathsf{h}$, $\mathsf{h}^*$ *be defined as in Figure 3. Let $\mathcal{A}$ be an adversary against the vkl-PRF security of* $\mathsf{NMAC}^{\leftrightarrows}$ *whose* New *queries have minimal and maximal key length $\ell_{\mathsf{min}}(\mathcal{A})$ resp. $\ell_{\mathsf{max}}(\mathcal{A})$. Let $n = \lceil \ell_{\mathsf{max}}(\mathcal{A})/b \rceil$ be the block length of the maximum-length key used by $\mathcal{A}$ and let $a = \min(c, \ell_{\mathsf{min}}(\mathcal{A}))$. Then we can construct adversaries $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}, \mathcal{A}_{\mathsf{h}}$ such that*

$$\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{NMAC}^{\leftrightarrows}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{pad},a}, \mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) + n \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) \,. \tag{4}$$

*Adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ makes at most $\max(\mathbf{Q}^{\text{New}}(\mathcal{A}), \mathbf{Q}^{\text{Fn}}(\mathcal{A}))$ to oracle* New *and at most $\mathbf{Q}^{\text{Fn}}(\mathcal{A})$ to oracle* Fn*. $\mathcal{A}_{\mathsf{h}}$ makes $\mathbf{Q}^{\text{Fn}}(\mathcal{A})$ queries to each of oracle* New *and* Fn*. The running times of both adversaries are approximately that of $\mathcal{A}$.*

Games $\mathrm{G}_0$–$\mathrm{G}_5$:

$\underline{\textsc{New}(\ell)}$ // Game $\mathrm{G}_0$

$n \leftarrow n + 1$

// Pre-computation of $\mathsf{NMAC}^{\leftrightarrows}$ key values; cf. Figure 5, lines 1–2

$K_n \leftarrow\!\!{\scriptstyle\$}\ \{0,1\}^{\ell}$ ; $k \leftarrow \min(\ell, b)$ ; $\overline{K}_n \leftarrow K_n \,\|\, \mathsf{pad}(b+\ell)$

$L_{n,1} \leftarrow K_n[1..k]$ ; $P_n \leftarrow \overline{K}_n[k+1..b]$ ; $S_n \leftarrow \overline{K}_n[b+1..|\overline{K}_n|]$

$\underline{\textsc{New}(\ell)}$ // Games $\mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3$

$n \leftarrow n + 1$ ; $k \leftarrow \min(\ell, b)$

$K_n \leftarrow\!\!{\scriptstyle\$}\ \{0,1\}^{\ell}$ ; $\overline{K}_n \leftarrow K_n \,\|\, \mathsf{pad}(b+\ell)$ ; $S_n \leftarrow \overline{K}_n[b+1..|\overline{K}_n|]$

$\underline{\textsc{New}(\ell)}$ // Games $\mathrm{G}_4, \mathrm{G}_5$

$n \leftarrow n + 1$

Figure 6: The New oracles for games in the proof of Theorem 4.

---

*Proof of Theorem 4.* We use a sequence of games $\mathrm{G}_0$–$\mathrm{G}_5$. All are executed with $\mathcal{A}$, and hence provide the oracles named in game $\mathbf{G}^{\text{vkl-PRF-}d}_{\mathsf{NMAC}^{\leftrightarrows}}$. The New oracles of the games are shown in Figure 6. This allows further descriptions to focus on the Fn oracle.

▷ *Replacing the first invocation of* $\mathsf{h}^{\leftrightarrows}$ *with a random function.* We begin with game $\mathrm{G}_0$ being $\mathbf{G}^{\text{vkl-PRF-1}}_{\mathsf{NMAC}^{\leftrightarrows}}(\mathcal{A})$. That is, $\mathrm{G}_0$ is the "real" vkl-PRF game.

The first game hop, to game $\mathrm{G}_1$, replaces the first evaluation of $\mathsf{h}^{\leftrightarrows}$ in $\mathsf{NMAC}^{\leftrightarrows}$ (on line 3 of Figure 5) by consistent random sampling. Consequently, $L_{n,1}$ is not used anymore and the corresponding first $k$ bits of $K_n$ become superfluous in the New oracle (Figure 6). The Fn oracles of games $\mathrm{G}_0, \mathrm{G}_1$ reflecting this change are in Figure 7. They differ only in that line 4 is only in $\mathrm{G}_0$ and line 5 is only in $\mathrm{G}_1$. By standard equation rewriting

$$\Pr[\mathbf{G}^{\text{vkl-PRF-1}}_{\mathsf{NMAC}^{\leftrightarrows}}(\mathcal{A})] = \Pr[\mathrm{G}_0(\mathcal{A})] = \Pr[\mathrm{G}_1(\mathcal{A})] + (\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]). \tag{5}$$

We build an adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},1}$ such that

$$\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})] \le \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{pad},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows},1}). \tag{6}$$

Adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},1}$ simulates $\mathrm{G}_0$ for $\mathcal{A}$, with two changes: First, it does not sample $K_n$ itself in the New oracle of $\mathrm{G}_0$, but instead issues a $\textsc{New}(k)$ query to its $\mathbf{G}^{\text{rka-PRF}}_{\Phi_{\mathsf{pad},a},\mathsf{h}^{\leftrightarrows}}$ game, where $k = \min(\ell, b)$. This initializes a $b$-bit key for $\mathsf{h}^{\leftrightarrows}$ and fixes the RKD parameter for user $n$ to $\alpha_n = k$. If $\ell > b$, adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},1}$ then samples the remaining $\ell - b$ bits of $K_n$ uniformly at random. It then applies the (remaining) padding to compute $S_n$. Second, instead of computing $\mathsf{h}^{\leftrightarrows}$ in line 4 of $\mathrm{G}_0$ (Figure 7) in response to a query $\textsc{Fn}(i, M_1\|M_2)$ from $\mathcal{A}$, adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},1}$ queries its Fn oracle on $(i, \varepsilon, M_1)$ and uses the result for $\mathrm{T}_{\mathsf{h}^{\leftrightarrows},1}[i, M_1]$. Note that for each user index $i$, the same RKD function $\Phi_{\mathsf{pad},a}(\alpha_i, \varepsilon, \cdot)$ is always applied. Hence it is sufficient to index the table by $(i, M_i)$, as these uniquely determine the output from oracle Fn in the rka-PRF game. Furthermore, since $\mathsf{h}^{\leftrightarrows}(L_{i,1}\|P_i, M_1)$ on line 4 of Figure 7 is computed only once per pair $(i, M_1)$, $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},1}$ makes at most one Fn query for each of the $\mathbf{Q}^{\text{Fn}}(\mathcal{A})$ Fn queries made by $\mathcal{A}$. Its New queries are all with an input $k \ge a$, since the minimal $k$ on which New is called is $k = \min(\ell_{\mathsf{min}}(\mathcal{A}), b) \ge \min(\ell_{\mathsf{min}}(\mathcal{A}), c) = a$. This is by virtue of $\ell_{\mathsf{min}}(\mathcal{A})$ being the smallest key length $\ell$ queried by $\mathcal{A}$, and the assumption that $b > c$ (see Section 4). Adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},1}$ soundly simulates $\mathrm{G}_{(1-d)}$ when playing game $\mathbf{G}^{\text{rka-PRF-}d}_{\Phi_{\mathsf{pad},a},\mathsf{h}^{\leftrightarrows}}$, for $d \in \{0,1\}$, which yields the bound in Equation (6).

$\underline{\text{FN}(i, M)}$ // Games $G_0$, $G_1$

1 If $T[i, M] \neq \bot$ then return $T[i, M]$
2 $M_1 \| M_2 \leftarrow M$ // $|M_1| = |M_2| = c$
3 If $T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1] = \bot$ then
4 $\quad T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1] \leftarrow \mathsf{h}^{\leftrightarrows}(L_{i,1} \| P_i, M_1)$ // $G_0$
5 $\quad T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1] \leftarrow\!\!\$\ \{0,1\}^c$ // $G_1$
6 $L_{i,2} \leftarrow \mathsf{h}^*(T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1], S_i)$
7 $T[i, M] \leftarrow \mathsf{h}^{\leftrightarrows}(L_{i,2} \| \mathsf{pad}(b + c), M_2)$ ; Return $T[i, M]$

$\underline{\text{FN}(i, M)}$ // Games $G_2$, $G_3$

1 If $T[i, M] \neq \bot$ then return $T[i, M]$
2 $M_1 \| M_2 \leftarrow M$ // $|M_1| = |M_2| = c$
3 If $T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1] = \bot$ then $T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1] \leftarrow\!\!\$\ \{0,1\}^c$
4 If $T_{\mathsf{h}^*}[i, M_1] = \bot$ then
5 $\quad T_{\mathsf{h}^*}[i, M_1] \leftarrow \mathsf{h}^*(T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1], S_i)$ // $G_2$
6 $\quad T_{\mathsf{h}^*}[i, M_1] \leftarrow\!\!\$\ \{0,1\}^c$ // $G_3$
7 $T[i, M] \leftarrow \mathsf{h}^{\leftrightarrows}(T_{\mathsf{h}^*}[i, M_1] \| \mathsf{pad}(b + c), M_2)$ ; Return $T[i, M]$

$\underline{\text{FN}(i, M)}$ // Games $G_4$, $G_5$

1 If $T[i, M] \neq \bot$ then return $T[i, M]$
2 $M_1 \| M_2 \leftarrow M$ // $|M_1| = |M_2| = c$
3 If $T_{\mathsf{h}^*}[i, M_1] = \bot$ then $T_{\mathsf{h}^*}[i, M_1] \leftarrow\!\!\$\ \{0,1\}^c$
4 If $T_{\mathsf{h}^{\leftrightarrows},2}[i, M_1, M_2] = \bot$ then
5 $\quad T_{\mathsf{h}^{\leftrightarrows},2}[i, M_1, M_2] \leftarrow \mathsf{h}^{\leftrightarrows}(T_{\mathsf{h}^*}[i, M_1] \| \mathsf{pad}(b + c), M_2)$ // $G_4$
6 $\quad T_{\mathsf{h}^{\leftrightarrows},2}[i, M_1, M_2] \leftarrow\!\!\$\ \{0,1\}^c$ // $G_5$
7 $T[i, M] \leftarrow T_{\mathsf{h}^{\leftrightarrows},2}[i, M_1, M_2]$ ; Return $T[i, M]$

Figure 7: The FN oracles for games $G_0$–$G_5$ in the proof of Theorem 4.

▷ *Replacing* $\mathsf{h}^*$ *with a random function.* We simplify the FN oracle from $G_1$ to $G_2$. Note that storing the values $L_{i,2}$ in $T_{\mathsf{h}^*}[i, M_1]$ and computing them only once (lines 4 and 5 of $G_2$) is sound since $S_i$ is fixed per key $i$, so for each $T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1]$, the inputs to $\mathsf{h}^*$ are fixed.

In $G_3$ (Figure 7), we replace the computation of $\mathsf{h}^*(T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1], S_i)$, originally in line 4 of Figure 5, by (consistent) random sampling. By rewriting,

$$\Pr[G_1(\mathcal{A})] = \Pr[G_2(\mathcal{A})] = \Pr[G_3(\mathcal{A})] + (\Pr[G_2(\mathcal{A})] - \Pr[G_3(\mathcal{A})]). \tag{7}$$

We construct an adversary $\mathcal{A}_{\mathsf{h}^*}$ such that

$$\Pr[G_2(\mathcal{A})] - \Pr[G_3(\mathcal{A})] \leq \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}^*}(\mathcal{A}_{\mathsf{h}^*}). \tag{8}$$

Observe that in game $G_2$, values $T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1]$ in the FN oracle are random $c$-bit strings. Adversary $\mathcal{A}_{\mathsf{h}^*}$ simulates $G_2$, except for two differences: Instead of sampling $T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1] \leftarrow\!\!\$\ \{0,1\}^c$ in line 3 of $G_2$ (Figure 7), $\mathcal{A}_{\mathsf{h}^*}$ issues a NEW query to set up a new key with index $j$ in its $\mathbf{G}^{\mathrm{PRF}\text{-}d}_{\mathsf{h}^*}$ game and stores $T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1] \leftarrow j$. Further, instead of computing $\mathsf{h}^*(T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1], S_i)$ itself in line 5 of $G_2$, it queries $(T_{\mathsf{h}^{\leftrightarrows},1}[i, M_1], S_i)$ to its FN oracle, using the result as $T_{\mathsf{h}^*}[i, M_1]$. It thus makes at most $\mathbf{Q}^{\mathrm{FN}}(\mathcal{A})$ queries to its NEW oracle, and one FN query under each key (since each $T_{\mathsf{h}^*}[i, M_1]$ is only computed once), each of whose second component is at most $n = \lceil \ell_{\mathsf{max}}(\mathcal{A})/b \rceil$ blocks long. (Note that $n + 1$ is an upper bound on the number of blocks of the *padded* key $\overline{K}$

of $\mathsf{NMAC}^{\leftrightarrows}$. The input $S$ to the cascade in any $\textsc{Fn}$ query is hence at most $n$ blocks.) Depending on the challenge bit $d$ in $\mathbf{G}^{\text{PRF-}d}_{\mathsf{h}^*}$, $\mathcal{A}_{\mathsf{h}^*}$ soundly simulates either $\mathrm{G}_2$ or $\mathrm{G}_3$, yielding Equation (8).

▷ *Replacing the second* $\mathsf{h}^{\leftrightarrows}$ *with a random function.* We again first simplify the $\textsc{Fn}$ oracle from $\mathrm{G}_3$ to $\mathrm{G}_4$ (cf. Figure 7), sampling $\mathrm{T}_{\mathsf{h}^*}[i, M_1]$ directly and omitting the no longer needed table $\mathrm{T}_{\mathsf{h}^{\leftrightarrows},1}$. Next, in $\mathrm{G}_5$ (also Figure 7), we sample the output of the second call to $\mathsf{h}^{\leftrightarrows}$ (Figure 5, line 5), stored in $\mathrm{T}_{\mathsf{h}^{\leftrightarrows},2}[i, M_1, M_2]$, (consistently) at random instead of computing it as $\mathsf{h}^{\leftrightarrows}(\mathrm{T}_{\mathsf{h}^*}[i, M_1] \| \mathsf{pad}(b + c), M_2)$.

Rewriting equations, we get

$$\Pr[\mathrm{G}_3(\mathcal{A})] = \Pr[\mathrm{G}_4(\mathcal{A})] = \Pr[\mathrm{G}_5(\mathcal{A})] + (\Pr[\mathrm{G}_4(\mathcal{A})] - \Pr[\mathrm{G}_5(\mathcal{A})]). \tag{9}$$

We bound the introduced probability difference via an adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},2}$ as

$$\Pr[\mathrm{G}_4(\mathcal{A})] - \Pr[\mathrm{G}_5(\mathcal{A})] \leq \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{pad},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows},2}). \tag{10}$$

Adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},2}$ acts as the challenger in $\mathrm{G}_4$ with the following two modifications: First, for any $\mathrm{T}_{\mathsf{h}^*}[i, M_1]$ value sampled in line 3 of Figure 7, $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},2}$ queries $\textsc{New}(c)$ and stores the key index in a table $\mathrm{T}_{\mathsf{h}^*}[i, M_1]$. Second, instead of computing $\mathsf{h}^{\leftrightarrows}(\mathrm{T}_{\mathsf{h}^*}[i, M_1] \| \mathsf{pad}(b + c), M_2)$ in line 5, $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},2}$ queries its $\textsc{Fn}$ oracle on $(\mathrm{T}_{\mathsf{h}^*}[i, M_1], \varepsilon, M_2)$. Note that $\Phi_{\mathsf{pad},a}$ will never return $\perp$ here since by definition $a = \min(\ell_{\mathsf{min}}(\mathcal{A}), c)$, so $c \geq a$. This results in at most $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ queries to each of $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},2}$'s oracles $\textsc{New}$ and $\textsc{Fn}$. Depending on bit $d$ in $\mathbf{G}^{\text{rka-PRF-}d}_{\Phi_{\mathsf{pad},a},\mathsf{h}^{\leftrightarrows}}$, $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},2}$ soundly simulates either $\mathrm{G}_4$ $(d = 1)$ or $\mathrm{G}_5$ $(d = 0)$, yielding Equation 10.

Finally we claim that

$$\Pr[\mathrm{G}_5(\mathcal{A})] = \Pr[\mathbf{G}^{\text{vkl-PRF-0}}_{\mathsf{NMAC}^{\leftrightarrows}}(\mathcal{A})], \tag{11}$$

which results from observing that $\mathrm{T}_{\mathsf{h}^*}[i, M_1]$ is not used anymore in $\mathrm{G}_5$, and the $\textsc{Fn}$ oracle in $\mathrm{G}_5$ responds with (consistently-sampled) random strings. Hence $\mathrm{G}_5$ equals $\mathbf{G}^{\text{vkl-PRF-0}}_{\mathsf{NMAC}^{\leftrightarrows}}$.

Combining Equations (5)-(11) gives

$$\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{NMAC}^{\leftrightarrows}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{pad},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) + \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}^*}(\mathcal{A}_{\mathsf{h}^*})\ , \tag{12}$$

where $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ is formed by picking $\gamma \leftarrow\!\!\$\ \{1, 2\}$ and running $\mathcal{A}_{\mathsf{h}^{\leftrightarrows},\gamma}$. Adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ makes at most $\max(\mathbf{Q}^{\textsc{New}}(\mathcal{A}), \mathbf{Q}^{\textsc{Fn}}(\mathcal{A}))$ to oracle $\textsc{New}$ and at most $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ to oracle $\textsc{Fn}$.

▷ *Applying Lemma 2.* Lastly, we invoke Lemma 2 on the multi-user PRF security of the cascade. This gives us an adversary $\mathcal{A}_{\mathsf{h}}$ such that

$$\mathbf{Adv}^{\text{PRF}}_{\mathsf{h}^*}(\mathcal{A}_{\mathsf{h}^*}) \leq n \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}})\ , \tag{13}$$

where $\mathcal{A}_{\mathsf{h}}$ makes $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{h}^*}) = \mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ queries to each of oracle $\textsc{New}$ and $\textsc{Fn}$. □

### 5.2 Strong Multi-user PRF Security of NMAC

PRF security for NMAC was already established in [8, 5, 6, 30] for the single user setting. By the usual hybrid argument [9] one can conclude multi-user PRF security with advantage degraded by the number of $\textsc{New}$ queries. We are interested in strong multi-user PRF security, meaning bounds that do not suffer such degradation and instead are as good as in the single-user setting.

Let us first review the proof of single-user PRF security of GPR [30]. Here the starting point is the proof of BCK [8] which first exploits PRF security of the outer application of $\mathsf{h}$ to reduce to the hidden-key collision-resistance of $\mathsf{h}^*$. Now, if $\mathsf{h}^*$ was a general PRF, it would imply it is also hidden-key collision resistant. But it is only a prefix-free PRF. The technique of GPR [30] is to exploit a random systems lemma by Maurer [42] that allows us to first reduce to a non-adaptive setting. Then, one can use a trick of appending a new block to colliding messages. This preserves collisions but breaks prefix-ness.

Games $\mathrm{G}_0$, $\mathrm{G}_1$:

<u>NEW</u>

1 $n \leftarrow n+1$ ; $K_{n,1} \leftarrow\!\!\$\ \{0,1\}^c$

2 $K_{n,2} \leftarrow\!\!\$\ \{0,1\}^c$ // $\mathrm{G}_0$

<u>FN$(i, M)$</u>

3 $X \leftarrow \mathsf{h}^*(K_{i,1}, M \| \mathsf{pad}(b+|M|))$

4 If $\mathrm{T}[i,X] = \perp$ then

5 $\quad \mathrm{T}[i,X] \leftarrow \mathsf{h}(K_{i,2}, X \| 0^{b-c})$ // $\mathrm{G}_0$

6 $\quad \mathrm{T}[i,X] \leftarrow\!\!\$\ \{0,1\}^c$ // $\mathrm{G}_1$

7 Return $\mathrm{T}[i,X]$

Games $\boxed{\mathrm{G}_2}$, $\mathrm{G}_3$:

<u>NEW</u>

1 $n \leftarrow n+1$ ; $K_{n,1} \leftarrow\!\!\$\ \{0,1\}^c$

<u>FN$(i, M)$</u>

2 $X \leftarrow \mathsf{h}^*(K_{i,1}, M \| \mathsf{pad}(b+|M|))$

3 $Y \leftarrow\!\!\$\ \{0,1\}^c$

4 If $\mathrm{T}[i,X] \neq \perp$ then

5 $\quad \mathsf{bad} \leftarrow 1$ ; $\boxed{Y \leftarrow \mathrm{T}[i,X]}$

6 $\mathrm{T}[i,X] \leftarrow Y$ ; Return $\mathrm{T}[i,X]$

Game $\mathrm{G}_4$:

<u>NEW</u>

1 $n \leftarrow n+1$

<u>FN$(i, M)$</u>

2 $m_i \leftarrow m_i + 1$ ; $\mathrm{M}_{i,m_i} \leftarrow M \| \mathsf{pad}(b+|M|)$

3 $Y \leftarrow\!\!\$\ \{0,1\}^c$ ; Return $Y$

<u>FINALIZE$(d^*)$</u>

4 For $i = 1, \ldots, n$ do

5 $\quad K_{i,1} \leftarrow\!\!\$\ \{0,1\}^c$

6 $\quad$ For $j = 1, \ldots, m_i$ do

7 $\quad\quad \mathrm{C}_{i,j} \leftarrow \mathsf{h}^*(K_{i,1}, \mathrm{M}_{i,j})$

8 $\quad$ If $(\exists\, \alpha \neq \beta : \mathrm{C}_{i,\alpha} = \mathrm{C}_{i,\beta})$ then $\mathsf{bad} \leftarrow 1$

9 Return $\mathsf{bad}$

Figure 8: Games for the proof of Theorem 5. The boxed code in line 5 is only in $\mathrm{G}_2$ and not in $\mathrm{G}_3$.

---

We give a simple, direct proof that establishes strong multi-user security with good bounds: The bound in Equation (14) below shows no degradation with the number of NEW queries. We do not use the random systems lemma. Instead we directly use game playing to move to a non-adaptive game. Finally we exploit our Lemma 2 showing strong multi-user PRF security of the cascade. The proof is quite simple and it is surprising this was not noted before.

**Theorem 5.** *Let* $\mathsf{h}\colon \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ *and let* $\mathsf{NMAC}$ *be as defined in Figure 3. Let* $\mathcal{A}$ *be an adversary against the PRF security of* $\mathsf{NMAC}$*. Assume each of its* FN *queries has length at most* $L$*, and let* $m = 1 + \lceil L/b \rceil$*. Let* $q_f = \mathbf{Q}^{\mathrm{FN}}(\mathcal{A})$ *and assume* $m \cdot q_f < 2^b$*. Then we can construct an adversary* $\mathcal{A}_{\mathsf{h}}$ *such that*

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{NMAC}}(\mathcal{A}) \leq (m+2) \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) + \frac{q_f(q_f-1)}{2^{c+1}} . \tag{14}$$

*Adversary* $\mathcal{A}_{\mathsf{h}}$ *makes* $\mathbf{Q}^{\mathrm{FN}}(\mathcal{A})$ *queries to* FN *and* $\max(\mathbf{Q}^{\mathrm{NEW}}(\mathcal{A}), \mathbf{Q}^{\mathrm{FN}}(\mathcal{A}))$ *to* NEW*, and its running time is about the same as that of* $\mathcal{A}$*.*

*Proof of Theorem 5.* Consider the games in Figure 8. Then

$$\begin{aligned}\Pr[\mathbf{G}^{\mathsf{PRF\text{-}1}}_{\mathsf{NMAC}}(\mathcal{A})] &= \Pr[\mathrm{G}_0(\mathcal{A})] \\ &= \Pr[\mathrm{G}_1(\mathcal{A})] + (\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]) .\end{aligned}$$

It is easy to build an adversary $\mathcal{A}_{\mathsf{h},1}$ such that

$$\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})] \leq \mathbf{Adv}_{\mathsf{h}}^{\mathrm{PRF}}(\mathcal{A}_{\mathsf{h},1}) \; . \tag{15}$$

Namely, $\mathcal{A}_{\mathsf{h},1}$ initializes $n \leftarrow 0$ and runs $\mathcal{A}$. When $\mathcal{A}$ makes a $\textsc{New}$ query, $\mathcal{A}_{\mathsf{h},1}$ lets $n \leftarrow n+1$, lets $K_{n,1} \leftarrow\!\!\$ \{0,1\}^c$ and queries its own $\textsc{New}$ oracle. When $\mathcal{A}$ makes a $\textsc{Fn}(i, M)$ query, $\mathcal{A}_{\mathsf{h},1}$ lets $X \leftarrow \mathsf{h}^*(K_{i,1}, M \| \mathsf{pad}(b + |M|))$, queries its own $\textsc{Fn}$ oracle with $i, X$ and returns the response to $\mathcal{A}$. Eventually $\mathcal{A}$ halts with some output, and $\mathcal{A}_{\mathsf{h},1}$ returns the same as its own output. When $\mathcal{A}_{\mathsf{h},1}$ is playing game $\mathbf{G}_{\mathsf{h}}^{\mathrm{PRF}\text{-}1}$, the key chosen for user $i$ plays the role of $K_{i,2}$ in game $\mathrm{G}_0$, hence $\Pr[\mathbf{G}_{\mathsf{h}}^{\mathrm{PRF}\text{-}1}(\mathcal{A}_{\mathsf{h},1})] = \Pr[\mathrm{G}_0(\mathcal{A})]$. Also $\Pr[\mathbf{G}_{\mathsf{h}}^{\mathrm{PRF}\text{-}0}(\mathcal{A}_{\mathsf{h},1})] = \Pr[\mathrm{G}_1(\mathcal{A})]$. This justifies Eq. (15). Next

$$\Pr[\mathrm{G}_1(\mathcal{A})] = \Pr[\mathrm{G}_2(\mathcal{A})] = \Pr[\mathrm{G}_3(\mathcal{A})] + (\Pr[\mathrm{G}_2(\mathcal{A})] - \Pr[\mathrm{G}_3(\mathcal{A})]) \; .$$

Now we have

$$\Pr[\mathrm{G}_3(\mathcal{A})] = \Pr[\mathbf{G}_{\mathsf{NMAC}}^{\mathrm{PRF}\text{-}0}(\mathcal{A})] \; .$$

Also games $\mathrm{G}_2$ and $\mathrm{G}_3$ are identical-until-$\mathsf{bad}$, so by the fundamental lemma of game playing [13]

$$\Pr[\mathrm{G}_2(\mathcal{A})] - \Pr[\mathrm{G}_3(\mathcal{A})] \leq \Pr[\mathrm{G}_3(\mathcal{A}) \text{ sets } \mathsf{bad}] \; .$$

We construct game $\mathrm{G}_4$ such that

$$\Pr[\mathrm{G}_4(\mathcal{A})] = \Pr[\mathrm{G}_3(\mathcal{A}) \text{ sets } \mathsf{bad}] \; ,$$

where, thanks to the hop to $\mathrm{G}_3$, the responses to $\textsc{Fn}$ queries from adversary $\mathcal{A}$ are now all independent random strings, assuming (without loss of generality) that the queries $\mathcal{A}$ makes to its $\textsc{Fn}$ oracle are all distinct. That is, we assume that no pair $i, M$ is repeated. Then, the responses from oracle $\textsc{Fn}$ are non-adaptive, meaning they do not depend on the inputs from $\mathcal{A}$. This allows us to apply the trick of adding a block to messages which cause collisions for the cascade, to circumvent prefix queries. Then, we can build an adversary $\mathcal{A}_{\mathsf{h}^*}$ such that

$$\Pr[\mathrm{G}_4(\mathcal{A})] \leq \mathbf{Adv}_{\mathsf{h}^*}^{\mathrm{PRF}}(\mathcal{A}_{\mathsf{h}^*}) + \frac{q_f(q_f - 1)}{2^{c+1}} \; . \tag{16}$$

The important thing is to ensure that $\mathcal{A}_{\mathsf{h}^*}$ is prefix free, meaning, for any user, no query is a prefix of another. This needs to be true even though such a condition may not be true of the $\textsc{Fn}$ queries $\mathcal{A}$ makes. We will then conclude by applying Lemma 2 (on the multi-user PRF security of $\mathsf{h}^*$).

Adversary $\mathcal{A}_{\mathsf{h}^*}$ can be seen as having two phases. In its first phase, it runs $\mathcal{A}$, replying to $\textsc{New}, \textsc{Fn}$ queries as per game $\mathrm{G}_4$, meaning at random, to obtain the messages $\mathrm{M}_{i,j}$ that $\mathcal{A}$ queries. In this phase, it makes no queries to its own $\textsc{New}$ or $\textsc{Fn}$ oracles. In more detail, $\mathcal{A}_{\mathsf{h}^*}$ initializes $n \leftarrow 0$. When $\mathcal{A}$ queries $\textsc{New}$, adversary $\mathcal{A}_{\mathsf{h}^*}$ lets $n \leftarrow n+1$ and initializes $m_n \leftarrow 0$. When $\mathcal{A}$ queries $\textsc{Fn}(i, M)$, adversary $\mathcal{A}_{\mathsf{h}^*}$ lets $m_i \leftarrow m_i + 1$ and records the queried message as $\mathrm{M}_{i,m_i} \leftarrow M$. Then it picks $Y \leftarrow\!\!\$ \{0,1\}^c$ and returns $Y$ to $\mathcal{A}$ as the response to its query. Note that the $\mathrm{M}_{i,j}$ messages are distributed exactly as in game $\mathrm{G}_4$, regardless of whether $\mathcal{A}_{\mathsf{h}^*}$ is playing in its own real or random game.

Once $\mathcal{A}$ has halted, $\mathcal{A}_{\mathsf{h}^*}$ enters its second phase. It starts by making $n$ queries to its own $\textsc{New}$ oracle. It is now useful, for a string $X \in \{0,1\}^{b*}$, to let $\mathrm{Bl}(X)$ be the set of all $b$-bit blocks in $X$; formally $\mathrm{Bl}(X) = \{X[1], \ldots, X[m]\}$ where $X[1] \ldots X[m] \xleftarrow{b} X$. Then $\mathcal{A}_{\mathsf{h}^*}$ does the following:

Adversary $\mathcal{A}_{\mathsf{h}^*}$, second phase

$d \leftarrow 0$
For $i = 1, \ldots, n$ do
  $\mathcal{M}_i \leftarrow \mathrm{Bl}(\mathrm{M}_{i,1}) \cup \cdots \cup \mathrm{Bl}(\mathrm{M}_{i,m_i})$
  // $\mathcal{M}_i$ is the set of all $b$-bit blocks in messages $\mathrm{M}_{i,1}, \ldots, \mathrm{M}_{i,m_i}$
  Pick $X_i \in \{0,1\}^b \setminus \mathcal{M}_i$
  // The assumption $m \cdot q_f < 2^b$ means $\{0,1\}^b \setminus \mathcal{M}_i \neq \emptyset$
  For $j = 1, \ldots, m_i$ do
    $\mathrm{P}_{i,j} \leftarrow \mathrm{M}_{i,j} \| X_i$
    $\mathrm{C}^*_{i,j} \leftarrow \mathrm{F}\textsc{n}(i, \mathrm{P}_{i,j})$ // Here $\mathcal{A}_{\mathsf{h}^*}$ queries its own $\mathrm{F}\textsc{n}$ oracle
  If $(\exists\, \alpha \neq \beta \,:\, \mathrm{C}^*_{i,\alpha} = \mathrm{C}^*_{i,\beta})$ then $d \leftarrow 1$
Return $d$

Due to the choice of $X_i$, the queries $i, \mathrm{P}_{i,j}$ that adversary $\mathcal{A}_{\mathsf{h}^*}$ makes to its $\mathrm{F}\textsc{n}$ oracle are prefix free. Suppose $\mathcal{A}_{\mathsf{h}^*}$ is playing in its "real" PRF game, namely $\mathbf{G}^{\text{PRF-1}}_{\mathsf{h}^*}$. If $\mathsf{h}^*(K_{i,1}, \mathrm{M}_{i,\alpha}) = \mathsf{h}^*(K_{i,1}, \mathrm{M}_{i,\beta})$, then also $\mathsf{h}^*(K_{i,1}, \mathrm{P}_{i,\alpha}) = \mathsf{h}^*(K_{i,1}, \mathrm{P}_{i,\beta})$, where $K_{i,1}$ represents the key chosen for user $i$ in game $\mathbf{G}^{\text{PRF-1}}_{\mathsf{h}^*}$. Thus $\Pr[\mathbf{G}^{\text{PRF-1}}_{\mathsf{h}^*}(\mathcal{A})] \geq \Pr[\mathrm{G}_4(\mathcal{A})]$. On the other hand, because $\mathcal{A}$'s queries to its $\mathrm{F}\textsc{n}$ oracle were assumed distinct, we have $\Pr[\mathbf{G}^{\text{PRF-0}}_{\mathsf{h}^*}(\mathcal{A})] \leq q_f(q_f - 1) \cdot 2^{-c-1}$ by the birthday bound. The bound applies since the $\mathrm{C}^*_{i,j}$ values are uniformly random strings of length $c$ in this case. This gives Eq. (16). Now, the number of blocks in $\mathrm{P}_{i,j}$ is at most $m+1$. (It is one more than the number in $\mathrm{M}_{i,j}$, and the latter is at most $m$.) So any $\mathrm{F}\textsc{n}$ query of $\mathcal{A}_{\mathsf{h}^*}$ is at most $m+1$ blocks long. Now Lemma 2 gives us an adversary $\mathcal{A}_{\mathsf{h},2}$ such that

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}^*}(\mathcal{A}_{\mathsf{h}^*}) \leq (m+1) \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},2}) . \tag{17}$$

Finally we merge adversaries $\mathcal{A}_{\mathsf{h},1}, \mathcal{A}_{\mathsf{h},2}$ using Lemma 3 with weights $w_1 = 1$ and $w_2 = m+1$ to get an adversary $\mathcal{A}_{\mathsf{h}}$ such that

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},1}) + (m+1) \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},2}) \leq (m+2) \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) .$$

Putting together all the above concludes the proof. □

# 6 Dual-PRF Security of HMAC

Here we obtain what we consider our main result, namely to settle the dual-PRF security of the full HMAC via two, complementary results. (1) In the swap-PRF case, we give necessary and sufficient conditions on the set $\mathcal{S}$ of keys (messages for $\mathsf{HMAC}^{\leftrightarrows}$) for which $\mathsf{HMAC}^{\leftrightarrows}$ is vkl-PRF secure. (2) For the PRF case, we prove vkl-PRF security of HMAC for all (long enough) key lengths.

As illustrated in Figure 4, we can see HMAC (resp. $\mathsf{HMAC}^{\leftrightarrows}$) as the composition of a subkey derivation function HSKD deriving the keys for NMAC (resp. $\mathsf{NMAC}^{\leftrightarrows}$). We leverage this to reduce the security of HMAC to that of NMAC and HSKD. The first step below is a general composition theorem. The second step below is to analyze HSKD to show the attributes asked for by the composition theorem. Then we can conclude by applying our results for NMAC from Section 5.

For the PRF security of HMAC, this approach is not new; it is the same taken in [8, 5, 6]. But we will see that it is also handy for analyzing $\mathsf{HMAC}^{\leftrightarrows}$. See Table 2 for an overview and Figure 9 for an illustration of the details for $\mathsf{HMAC}^{\leftrightarrows}$.

## 6.1 Composition Theorem

Let $\mathsf{MAC}\colon \{0,1\}^{2c} \times \{0,1\}^* \to \{0,1\}^c$ be a given function family. (In our application it will be NMAC.) Let $\mathcal{F}$ be a length closed-set and let $\mathsf{SKD}\colon \mathcal{F} \to \{0,1\}^{2c}$ be a function that we call the

| Function | Goal | Assumption | Comments |
|---|---|---|---|
| $\mathbf{Comp}[\mathsf{MAC},\mathsf{SKD}]$ ($\mathsf{HMAC}_b = \mathbf{Comp}[\mathsf{NMAC},\mathsf{HSKD}_b]$) | PRF | $\mathsf{MAC}$ is PRF, $\mathsf{SKD}$ is PRG | [6] |
| $\mathbf{Comp}[\mathsf{MAC},\mathsf{SKD}]$ ($\mathsf{HMAC} = \mathbf{Comp}[\mathsf{NMAC},\mathsf{HSKD}]$) | vkl-PRF | $\mathsf{MAC}$ is vkl-PRF, $\mathsf{SKD}$ is vsl-PRG | Theorem 6.1 |
| $\mathbf{Comp}[\mathsf{MAC},\mathsf{SKD}]$ ($\mathsf{HMAC}^{\leftrightarrows} = \mathbf{Comp}[\mathsf{NMAC}^{\leftrightarrows},\mathsf{HSKD}]$) | vkl-PRF | $\mathsf{MAC}^{\leftrightarrows}$ is vkl-PRF, $\mathsf{SKD}$ is CR | Theorem 6.2 |
| $\mathsf{HSKD}$ | vsl-PRG | $\mathsf{h}$ is PRF, $\mathsf{h}^{\leftrightarrows}$ is PRF, $\mathsf{h}^{\leftrightarrows}$ is $\Phi_{\mathsf{zio},a}$-rka-PRF | Proposition 7 |
| $\mathsf{HSKD}$ | CR | $\mathsf{h}$ is CR | Proposition 8 |

Table 2: Overview of the composition results for $\mathsf{HMAC}_b$, $\mathsf{HMAC}$, and $\mathsf{HMAC}^{\leftrightarrows}$, modularly obtained by viewing them as composition of a function family $\mathsf{MAC}$ and a subkey derivation function $\mathsf{SKD}$.

---

subkey derivation function. We define their composition $\mathsf{F} = \mathbf{Comp}[\mathsf{MAC},\mathsf{SKD}]\colon \mathcal{F} \times \{0,1\}^* \to \{0,1\}^c$ by

$\underline{\mathsf{F}(K, M)}$ // $K \in \mathcal{F}$ and $M \in \{0,1\}^*$
$K_i \| K_o \leftarrow \mathsf{SKD}(K)$ // Derive a $2c$-bit subkey
Return $\mathsf{MAC}(K_i \| K_o, M)$

The following allows us to deduce both normal and swap-PRF security of $\mathsf{F}$ from the corresponding security of $\mathsf{MAC}$. What differs is the assumption on the subkey derivation function $\mathsf{SKD}$: (1) If $\mathsf{SKD}$ is a vsl-PRG then the transform preserves PRF security and (2) if $\mathsf{SKD}$ is CR then the transform preserves swap-PRF security. Notably, we can show both results for *variable-key length* (swap-)PRF security. This is made precise in Theorem 6. Through (1), our result fills a gap left in prior work, namely $\mathsf{HMAC}$'s security for keys of non–full block length.

We first state the composition result and then study the assumptions on $\mathsf{SKD}$ in Section 6.2.

**Theorem 6** (Composition theorem)**.** *Let* $\mathsf{MAC}\colon \{0,1\}^{2c} \times \{0,1\}^* \to \{0,1\}^c$ *be a function family and* $\mathsf{SKD}\colon \mathcal{F} \to \{0,1\}^{2c}$ *a function. Let* $\mathsf{F} = \mathbf{Comp}[\mathsf{MAC},\mathsf{SKD}]\colon \mathcal{F} \times \{0,1\}^* \to \{0,1\}^c$ *be their composition as above.*

1. **[Composition transfers vkl-PRF security if $\mathsf{SKD}$ is a vsl-PRG]** *Let $\mathcal{A}_{\mathsf{F}}$ be an adversary against the vkl-PRF security of $\mathsf{F}$. Then we can construct adversaries $\mathcal{A}_{\mathsf{MAC}}, \mathcal{A}_{\mathsf{SKD}}$ such that*
$$\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{F}}(\mathcal{A}_{\mathsf{F}}) \leq \mathbf{Adv}^{\text{PRF}}_{\mathsf{MAC}}(\mathcal{A}_{\mathsf{MAC}}) + \mathbf{Adv}^{\text{vsl-PRG}}_{\mathsf{SKD}}(\mathcal{A}_{\mathsf{SKD}})\ . \tag{18}$$
*Also $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{SKD}}) = \mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{F}})$ and $\mathcal{A}_{\mathsf{SKD}}$ has minimal (resp. maximal) seed length $\ell_{\mathsf{min}}(\mathcal{A}_{\mathsf{F}})$ (resp. $\ell_{\mathsf{max}}(\mathcal{A}_{\mathsf{F}})$).*

2. **[Composition transfers swap-vkl-PRF security if $\mathsf{SKD}$ is CR]** *Let $\mathcal{A}_{\mathsf{F}}$ be an adversary against the vkl-PRF security of $\mathsf{F}^{\leftrightarrows}$. Then we can construct adversaries $\mathcal{A}_{\mathsf{MAC}}, \mathcal{A}_{\mathsf{SKD}}$ such that*
$$\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{F}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{F}}) \leq \mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{MAC}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{MAC}}) + \mathbf{Adv}^{\text{CR}}_{\mathsf{SKD}}(\mathcal{A}_{\mathsf{SKD}})\ . \tag{19}$$

*In both cases, adversary $\mathcal{A}_{\mathsf{MAC}}$ has the same query counts and in 2. the same minimal/maximal key lengths as $\mathcal{A}_{\mathsf{F}}$. The constructed adversaries have about the same running time as $\mathcal{A}_{\mathsf{F}}$.*

We prove each sub-theorem separately.

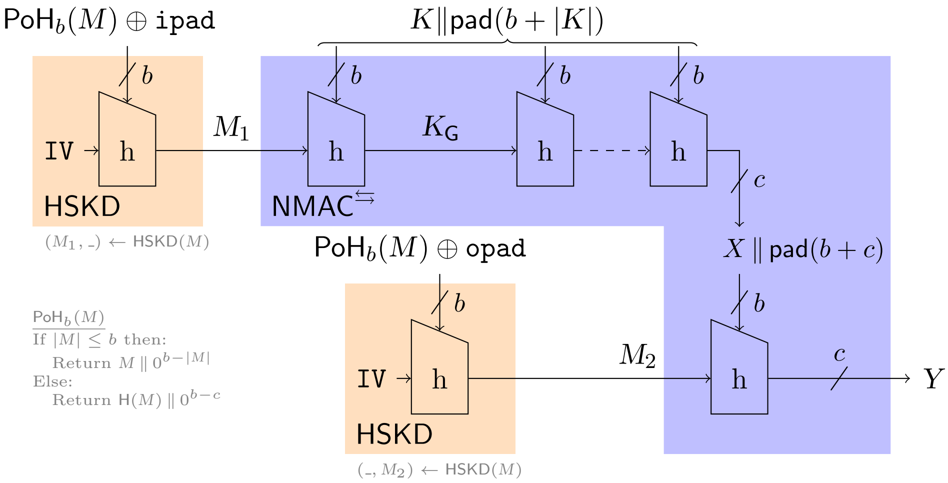



Figure 9: Illustration of how $\mathsf{HMAC}^{\leftrightarrows}(K, M)$ can be seen as the composition of $\mathsf{HSKD}$ (highlighted in orange) and $\mathsf{NMAC}^{\leftrightarrows}$ (in blue).

| Games $G_0$, $G_1$, $G_2$: | Adversary $\mathcal{A}_{\mathsf{SKD}}$: | Adversary $\mathcal{A}_{\mathsf{MAC}}$: |
|---|---|---|
| <u>$\textsc{New}(\ell)$</u> | <u>$\textsc{New}^*(\ell)$</u> | <u>$\textsc{New}^*(\ell)$</u> |
| 1 Assert: $\ell \in \mathsf{F.KL}$ | 1 Assert: $\ell \in \mathsf{F.KL}$ | 1 Assert: $\ell \in \mathsf{F.KL}$ |
| 2 $n \leftarrow n+1$ ; $\ell_n \leftarrow \ell$ | 2 $n \leftarrow n+1$ ; $\ell_n \leftarrow \ell$ | 2 $\textsc{New}()$ |
| 3 $s \leftarrow\$ \{0,1\}^\ell$ | 3 $K_n \leftarrow \textsc{Fn}(\ell)$ | <u>$\textsc{Fn}^*(i, M)$</u> |
| 4 $K_n \leftarrow \mathsf{SKD}(s)$ // $G_0$ | <u>$\textsc{Fn}^*(i, M)$</u> | 3 Return $\textsc{Fn}(i, M)$ |
| 5 $K_n \leftarrow\$ \{0,1\}^{2c}$ // $G_1$,$G_2$ | 4 If $\mathrm{T}[i, M] \neq \bot$ then: | |
| <u>$\textsc{Fn}(i, M)$</u> | 5 Return $\mathrm{T}[i, M]$ | |
| 6 If $\mathrm{T}[i, M] \neq \bot$ then: | 6 $\mathrm{T}[i, M] \leftarrow \mathsf{MAC}(K_i, M)$ | |
| 7 Return $\mathrm{T}[i, M]$ | 7 Return $\mathrm{T}[i, M]$ | |
| 8 $\mathrm{T}[i, M] \leftarrow \mathsf{MAC}(K_i, M)$ // $G_0$,$G_1$ | | |
| 9 $\mathrm{T}[i, M] \leftarrow\$ \{0,1\}^c$ // $G_2$ | | |
| 10 Return $\mathrm{T}[i, M]$ | | |

Figure 10: Games and adversaries for the proof of Theorem 6.1.

---

*Proof of Theorem 6.1.* The proof proceeds by a sequence of games $G_0$–$G_2$, shown in Figure 10. We begin with $G_0$, which runs adversary $\mathcal{A}_\mathsf{F}$, providing oracles $\textsc{New}$ and $\textsc{Fn}$ of the vkl-PRF game. Upon the $n$th query $\textsc{New}(\ell)$ from adversary $\mathcal{A}_\mathsf{F}$, the game samples a random seed $s$ of length $\ell$ from $\mathcal{F}$ and then applies $\mathsf{SKD}(s)$ on line 4 to compute a key $K_n$ for $\mathsf{MAC}$. Oracle $\textsc{Fn}(i, M)$ computes $\mathsf{MAC}(K_i, M)$. Since $\mathsf{SKD}$ is a deterministic function, the fact that it is computed once per seed in $\textsc{New}$ instead of for each $\textsc{Fn}$ query makes no difference. This means that $G_0$ is equivalent to $\mathbf{G}^{\text{vkl-PRF-1}}_{\mathsf{F}}(\mathcal{A}_\mathsf{F})$.

Game $G_1$ only differs from $G_0$ in oracle $\textsc{New}$, where line 4 is only in $G_0$ and line 5, which instead samples a new $2c$-bit key uniformly at random, is only in game $G_1$. We construct an adversary $\mathcal{A}_{\mathsf{SKD}}$ such that

$$\Pr[G_0(\mathcal{A}_\mathsf{F})] - \Pr[G_1(\mathcal{A}_\mathsf{F})] \le \mathbf{Adv}^{\text{vsl-PRG}}_{\mathsf{SKD}}(\mathcal{A}_{\mathsf{SKD}}). \tag{20}$$

Adversary $\mathcal{A}_{\mathsf{SKD}}$, shown in the middle of Figure 10, acts as the challenger in game $G_0$, except that instead of computing $K_n \leftarrow \mathsf{SKD}$ in response to the $n$th $\textsc{New}(\ell)$ query, adversary $\mathcal{A}_{\mathsf{SKD}}$ forwards the query to the its vsl-PRG game (see Figure 1) to obtain $K_n \leftarrow \textsc{Fn}(\ell)$. This ensures that $\mathcal{A}_{\mathsf{SKD}}$ soundly simulates $\mathrm{G}_{1-d}$ for $\mathcal{A}_{\mathsf{F}}$ when playing game $\mathbf{G}^{\text{vsl-PRG-}d}_{\mathsf{SKD}}$, where $d \in \{0,1\}$, establishing the bound in Equation (20).

Next, game $\mathrm{G}_2$ is identical to game $\mathrm{G}_1$, except that in oracle $\textsc{Fn}$, the response to each query is a consistent, randomly sampled string in $\{0,1\}^c$ instead of the output of $\mathsf{MAC}$. We construct an adversary $\mathcal{A}_{\mathsf{MAC}}$ such that

$$\Pr[\mathrm{G}_1(\mathcal{A}_{\mathsf{F}})] - \Pr[\mathrm{G}_2(\mathcal{A}_{\mathsf{F}})] \leq \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{MAC}}(\mathcal{A}_{\mathsf{MAC}}). \tag{21}$$

Adversary $\mathcal{A}_{\mathsf{MAC}}$, shown on the right in Figure 10, simulates game $\mathrm{G}_1$ for $\mathcal{A}_{\mathsf{F}}$, acting as the challenger, except in two ways. First, instead of sampling a key itself in oracle $\textsc{New}$, adversary $\mathcal{A}_{\mathsf{MAC}}$ issues a query to oracle $\textsc{New}()$ in its PRF game. Second, instead of computing $\mathsf{MAC}$ in response to a $\textsc{Fn}$ query, adversary $\mathcal{A}_{\mathsf{MAC}}$ forwards the query to its own $\textsc{Fn}$ oracle. This way, $\mathcal{A}_{\mathsf{MAC}}$ simulates $\mathrm{G}_1$ for $\mathcal{A}_{\mathsf{F}}$ when playing game $\mathbf{G}^{\text{PRF-1}}_{\mathsf{MAC}}$ and $\mathrm{G}_2$ when playing game $\mathbf{G}^{\text{PRF-0}}_{\mathsf{MAC}}$. This yields Equation (21).

We note that game $\mathrm{G}_2$ is equivalent to $\mathbf{G}^{\text{vkl-PRF-0}}_{\mathsf{F}}(\mathcal{A}_{\mathsf{F}})$, and hence by definition of the advantage and the established equations we reached the claim

$$\begin{aligned}\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{F}}(\mathcal{A}_{\mathsf{F}}) &= \Pr[\mathrm{G}_0(\mathcal{A}_{\mathsf{F}})] - \Pr[\mathrm{G}_2(\mathcal{A}_{\mathsf{F}})] \\ &= \Pr[\mathrm{G}_0(\mathcal{A}_{\mathsf{F}})] - \Pr[\mathrm{G}_1(\mathcal{A}_{\mathsf{F}})] + \Pr[\mathrm{G}_1(\mathcal{A}_{\mathsf{F}})] - \Pr[\mathrm{G}_2(\mathcal{A}_{\mathsf{F}})] \\ &\leq \mathbf{Adv}^{\text{vsl-PRG}}_{\mathsf{SKD}}(\mathcal{A}_{\mathsf{SKD}}) + \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{MAC}}(\mathcal{A}_{\mathsf{MAC}}).\end{aligned}$$ □

*Proof of Theorem 6.2.* The proof proceeds by a sequence of games $\mathrm{G}_0$–$\mathrm{G}_3$, shown in Figure 11. We begin with game $\mathrm{G}_0$ being $\mathbf{G}^{\text{vkl-PRF-1}}_{\mathsf{F}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{F}})$. That is, $\mathrm{G}_0$ is the "real" vkl-PRF game, with the functionality of $\mathsf{F}^{\leftrightarrows} = \mathbf{Comp}[\mathsf{MAC}^{\leftrightarrows}, \mathsf{SKD}]$ made explicit. The game keeps a table $\mathrm{T}$ for the $\textsc{Fn}$ oracle query results, indexed by $M_1 \| M_2 = \mathsf{SKD}(M)$ (Figure 11, line 3).

The first game hop, to game $\mathrm{G}_1$, replaces the evaluation of $\mathsf{MAC}^{\leftrightarrows}$ in line 4 of Figure 11 by consistent random sampling. Games $\mathrm{G}_0, \mathrm{G}_1$ in Figure 11 only differ in that line 4 is only in $\mathrm{G}_0$ and line 5 is only in $\mathrm{G}_1$. We build an adversary $\mathcal{A}_{\mathsf{MAC}}$ such that

$$\Pr[\mathrm{G}_0(\mathcal{A}_{\mathsf{F}})] - \Pr[\mathrm{G}_1(\mathcal{A}_{\mathsf{F}})] \leq \mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{MAC}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{MAC}}). \tag{22}$$

Adversary $\mathcal{A}_{\mathsf{MAC}}$ simulates $\mathrm{G}_0$ for $\mathcal{A}_{\mathsf{F}}$ by forwarding the queries from adversary $\mathcal{A}_{\mathsf{F}}$ to the corresponding oracles in its $\mathbf{G}^{\text{vkl-PRF}}_{\mathsf{MAC}^{\leftrightarrows}}$ game, replacing $M$ in $\textsc{Fn}$ queries by $M_1 \| M_2$ obtained by evaluating $\mathsf{SKD}(M)$. Adversary $\mathcal{A}_{\mathsf{MAC}}$ soundly simulates $\mathrm{G}_{(1-d)}$ when playing game $\mathbf{G}^{\text{vkl-PRF-}d}_{\mathsf{MAC}^{\leftrightarrows}}$, for $d \in \{0,1\}$, which yields the bound in Equation (22).

We now rewrite $\mathrm{G}_1$ as $\mathrm{G}_2$. In $\mathrm{G}_2$, the table $\mathrm{T}$ is indexed by the input messages $M$ (Figure 11 line 2). An additional table $\mathrm{T}_{\mathsf{SKD}}$, indexed by $M_1 \| M_2$, keeps track of collisions in the output of $\mathsf{SKD}$. For already seen values of $M$, the game returns early, while for new values of $M$ the game ensures consistent random sampling by only sampling a new value in $\mathrm{T}_{\mathsf{SKD}}$ if one was not set before (line 5). Additionally, game $\mathrm{G}_2$ sets a $\mathsf{bad}$ flag whenever a collision in the output of $\mathsf{SKD}$ occurs (line 5).

Since $\mathrm{G}_1$ and $\mathrm{G}_2$ are semantically equivalent, we have

$$\Pr[\mathrm{G}_1(\mathcal{A}_{\mathsf{F}})] = \Pr[\mathrm{G}_2(\mathcal{A}_{\mathsf{F}})]. \tag{23}$$

In the final game hop to $\mathrm{G}_3$, we always sample new values in $\mathrm{T}_{\mathsf{SKD}}$, whether a collision in the output of $\mathsf{SKD}$ occurred or not. Games $\mathrm{G}_2, \mathrm{G}_3$ in Figure 11 only differ when the $\mathsf{bad}$ flag is set (line 5). By the fundamental lemma of game playing [13], we have

$$\begin{aligned}\Pr[\mathrm{G}_2(\mathcal{A}_{\mathsf{F}})] &= \Pr[\mathrm{G}_3(\mathcal{A}_{\mathsf{F}})] + (\Pr[\mathrm{G}_2(\mathcal{A}_{\mathsf{F}})] - \Pr[\mathrm{G}_3(\mathcal{A}_{\mathsf{F}})]) && (24)\\ &\leq \Pr[\mathrm{G}_3(\mathcal{A}_{\mathsf{F}})] + \Pr[\mathrm{G}_2(\mathcal{A}_{\mathsf{F}}) \text{ sets } \mathsf{bad}]. && (25)\end{aligned}$$

$\underline{\text{FN}(i, M)}$ // Games $G_0$, $G_1$

1 Assert: $M \in \mathcal{F}$

2 $M_1 \| M_2 \leftarrow \mathsf{SKD}(M)$ // $|M_1| = |M_2| = c$

3 If $T[i, M_1\|M_2] \neq \perp$ then return $T[i, M_1\|M_2]$

4 $T[i, M_1\|M_2] \leftarrow \mathsf{MAC}^{\leftrightarrows}(K_i, M_1\|M_2)$ // $G_0$

5 $T[i, M_1\|M_2] \leftarrow\\$ \{0,1\}^c$ // $G_1$

6 Return $T[i, M_1\|M_2]$

$\underline{\text{FN}(i, M)}$ // Games $G_2$, $\boxed{G_3}$

1 Assert: $M \in \mathcal{F}$

2 If $T[i, M] \neq \perp$ then return $T[i, M]$

3 $M_1 \| M_2 \leftarrow \mathsf{SKD}(M)$

4 If $T_{\mathsf{SKD}}[i, M_1\|M_2] \neq \perp$ then:

// $T[i, M] = \perp$, but $T_{\mathsf{SKD}}[i, M_1\|M_2] \neq \perp$: collision for $\mathsf{SKD}$

5 $\quad \mathsf{bad} \leftarrow 1$; $\boxed{T_{\mathsf{SKD}}[i, M_1\|M_2] \leftarrow\\$ \{0,1\}^c}$

6 Else: $T_{\mathsf{SKD}}[i, M_1\|M_2] \leftarrow\\$ \{0,1\}^c$

7 $T[i, M] \leftarrow T_{\mathsf{SKD}}[i, M_1\|M_2]$

8 Return $T[i, M]$

$\underline{\text{NEW}(\ell)}$

1 Assert: $\ell \in \mathsf{MAC}^{\leftrightarrows}.\mathrm{KL}$

2 $n \leftarrow n + 1$ ; $\ell_n \leftarrow \ell$ ; $K_n \leftarrow\\$ \{0,1\}^{\ell}$

Figure 11: The FN and NEW oracles for games $G_0$–$G_3$ in the proof of Theorem 6.2. The boxed code is only in game $G_3$.

---

We can now build an adversary $\mathcal{A}_{\mathsf{SKD}}$ such that:

$$\Pr[G_2(\mathcal{A}_{\mathsf{F}}) \text{ sets } \mathsf{bad}] \leq \mathbf{Adv}^{\mathrm{CR}}_{\mathsf{SKD}}(\mathcal{A}_{\mathsf{SKD}}), \tag{26}$$

based on the following observation: $\mathsf{bad}$ is only set to true in line 5 if in an FN query, $(i, M)$ is new, but $(i, M_1, M_2)$ is not. (Otherwise the oracle returns already in line 2).) Hence, the only way to set $\mathsf{bad}$ is via two queries $(i, M)$ and $(i, M')$ such that $M \neq M'$, yet $\mathsf{SKD}(M) = M_1\|M_2 = \mathsf{SKD}(M')$. In other words, setting $\mathsf{bad}$ amounts to a collision for function $\mathsf{SKD}$. Adversary $\mathcal{A}_{\mathsf{SKD}}$ acts as the challenger in $G_3$. Additionally, $\mathcal{A}_{\mathsf{SKD}}$ keeps track of queried messages in a table by setting $T_{Coll}[i, M_1, M_2] \leftarrow M$ at the end of each query $\text{FN}(i, M)$ from $\mathcal{A}_{\mathsf{F}}$. If adversary $\mathcal{A}_{\mathsf{F}}$ makes a query $\text{FN}(i, M^*)$ which sets $\mathsf{bad}$, $\mathcal{A}_{\mathsf{SKD}}$ computes $M_1^*\|M_2^* \leftarrow \mathsf{SKD}(M^*)$ and outputs $M^*$ and $T_{Coll}[i, M_1^*, M_2^*]$ as the collision for $\mathsf{SKD}$, establishing Equation (26).

We observe that $G_3$ is the "random" vkl-PRF game $\mathbf{G}^{\text{vkl-PRF-0}}_{\mathsf{F}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{F}})$. Combining the equations, we obtain the bound in the theorem statement. $\square$

## 6.2 Analysis of HSKD

In order to deduce security for $\mathsf{HMAC}$ and $\mathsf{HMAC}^{\leftrightarrows}$ via the composition result from Theorem 6, we need to analyze the properties of $\mathsf{HMAC}$'s subkey derivation function $\mathsf{HSKD}$ shown in Figure 3. We rewrite it, unfolding $\mathsf{PoH}_b$ and $\mathsf{HSKD}_b$:

$\underline{\mathsf{HSKD}(K)}$ $\triangleright$ $\mathsf{HSKD}\colon \{0,1\}^* \to \{0,1\}^{2c}$

If $|K| \leq b$ then $K_b \leftarrow K \,\|\, 0^{b-|K|}$ else $K_b \leftarrow \mathsf{H}(K) \,\|\, 0^{b-c}$ // $\mathsf{PoH}_b$

$K_i \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{ipad})$ ; $K_o \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{opad})$ // $\mathsf{HSKD}_b$

Return $K_i\|K_o$

We will next justify $\mathsf{HSKD}$ both as a variable seed-length PRG (for $\mathsf{HMAC}$) and as a collision-resistant function under restricted inputs (for $\mathsf{HMAC}^{\leftrightarrows}$).

<u>$\mathsf{HSKD}$ is a vsl-PRG.</u> To prove that $\mathsf{HSKD}$ is a variable-seed-length PRG, we rely on properties of its two building blocks, $\mathsf{H}$ and $\mathsf{h}$. First, $\mathsf{H}$ is the MD hash function defined as $\mathsf{H}(K) = \mathsf{h}^*(\mathtt{IV}, K \,\|\, \mathsf{pad}(|K|))$, which, given that it is only applied in the case that $|K| > b$, can be viewed as an instantiation of the 2-tier cascade $\mathsf{2CSC}$, where the first tier is $\mathsf{h}^{\leftrightarrows}$. Viewed this way, we have

$\underline{\mathsf{H}(K)}$ $\triangleright$ $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^c$
$\overline{K} \leftarrow K \,\|\, \mathsf{pad}(|K|)$; $S \leftarrow \overline{K}[b+1..|\overline{K}|]$
Return $\mathsf{2CSC}[\mathsf{h}^{\leftrightarrows}, \mathsf{h}](K[1..b], (\mathtt{IV}, S))$

We will use the result from Lemma 1 to bound the probability of distinguishing the replacement of the output of $\mathsf{H}(K)$ by a random $c$-bit string by the advantage of an adversary against the PRF security of $\mathsf{h}^{\leftrightarrows}$ and $\mathsf{h}$, respectively.

Next, we study the remaining invocations of $\mathsf{h}$ in $K_i \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{ipad}) = \mathsf{h}^{\leftrightarrows}(K_b \oplus \mathtt{ipad}, \mathtt{IV})$ and $K_o \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{opad}) = \mathsf{h}^{\leftrightarrows}(K_b \oplus \mathtt{opad}, \mathtt{IV})$. Here, we reduce the distinguishing probability of replacing $K_i$ and $K_o$ by uniformly sampled random strings in $\{0,1\}^c$ to the advantage of an adversary against the PRF security of $\mathsf{h}^{\leftrightarrows}$ under related-key attacks.

The RKD function of interest is $\Phi_{\mathsf{zio},a} : \{0,\dots,b\} \times \{\mathtt{ipad}, \mathtt{opad}\} \times \{0,1\}^b \to \{0,1\}^b \cup \{\bot\}$ defined as follows: on inputs $\ell, \mathsf{io}, K$, it returns $\bot$ if $\ell < a$ and $(K[1..\ell] \,\|\, 0^{b-\ell}) \oplus \mathsf{io}$ otherwise. In more detail:

$\underline{\Phi_{\mathsf{zio},a}(\ell, \mathsf{io}, K)}$
If $\ell < a$ then return $\bot$
$K' \leftarrow K[1..\ell] \,\|\, 0^{b-\ell}$
$L \leftarrow K' \oplus \mathsf{io}$ ; Return $L$

That is, $\Phi_{\mathsf{zio},a}$ is the related-key-deriving function that overwrites a $b-\ell$-bit long suffix of the key with zeros, and then XORs the result with $\mathtt{ipad}$ or $\mathtt{opad}$. The parameter $a$ specifies the length of the shortest prefix of the key which must be left unchanged before the XOR, otherwise the function returns $\bot$.

PRF security of $\mathsf{h}^{\leftrightarrows}$ under related-key attacks, together with regular PRF security of $\mathsf{h}^{\leftrightarrows}$ and $\mathsf{h}$ then yield the following result.

**Proposition 7** (vsl-PRG security of $\mathsf{HSKD}$)**.** *Let* $\mathsf{HSKD}\colon \{0,1\}^* \to \{0,1\}^{2c}$ *be as defined in Figure 3. Let* $\mathcal{A}$ *be an adversary against the* vsl-PRG *security of* $\mathsf{HSKD}$*. Let* $n = \lceil \ell_{\mathsf{max}}(\mathcal{A})/b \rceil$ *be the block length of the maximal-length seed queried by* $\mathcal{A}$ *and let* $a = \min(c, \ell_{\mathsf{min}}(\mathcal{A}))$*. Then we can construct adversaries* $\mathcal{A}_{\mathsf{f}}$*,* $\mathcal{A}_{\mathsf{h}}$ *and* $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ *such that*

$$\mathbf{Adv}^{\text{vsl-PRG}}_{\mathsf{HSKD}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{f}}) + n \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) + \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{zio},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) \,. \tag{27}$$

*Adversaries* $\mathcal{A}_{\mathsf{f}}$ *and* $\mathcal{A}_{\mathsf{h}}$ *each make at most* $\mathbf{Q}^{\text{Fn}}(\mathcal{A})$ *queries to* New *and at most* $\mathbf{Q}^{\text{Fn}}(\mathcal{A})$ *queries to* Fn*. Adversary* $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ *makes* $\mathbf{Q}^{\text{Fn}}(\mathcal{A})$ *queries to oracle* New *and* $2 \cdot \mathbf{Q}^{\text{Fn}}(\mathcal{A})$ *queries to oracle* Fn*.*

*Proof of Proposition 7.* The proof uses a sequence of games, $\mathrm{G}_0$-$\mathrm{G}_3$, all shown in Figure 12.

$\triangleright$ *Replacing* $\mathsf{2CSC}$ *with a random function.* The first game $\mathrm{G}_0$, is equivalent to the real vsl-PRG game $\mathbf{G}^{\text{vsl-PRG-1}}_{\mathsf{HSKD}}$. Game $\mathrm{G}_1$ only differs from $\mathrm{G}_0$ in that the invocation of $\mathsf{2CSC}[\mathsf{h}^{\leftrightarrows}, \mathsf{h}]$ on line 5 in Figure 12 is replaced by random sampling from $\{0,1\}^c$ on line 6.

We construct an adversary $\mathcal{A}_{\mathsf{2CSC}}$ such that

$$\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})] \leq \mathbf{Adv}^{\text{PRF}}_{\mathsf{2CSC}}(\mathcal{A}_{\mathsf{2CSC}}) \,. \tag{28}$$

Adversary $\mathcal{A}_{\mathsf{2CSC}}$, shown in Figure 12, simulates game $\mathrm{G}_0$ for $\mathcal{A}$, acting as the challenger, except in two ways. First, if $\mathcal{A}$ issues a $\textsc{Fn}(\ell)$ query with $\ell > b$, then instead of sampling the whole $\ell$-bit key itself, adversary $\mathcal{A}_{\mathsf{2CSC}}$ calls oracle $\textsc{New}$ in the $\mathbf{G}^{\text{PRF-}d}_{\mathsf{2CSC}}$ game to initialize a new $b$-bit key for $\mathsf{2CSC}$. It then samples the remaining $\ell - b$ bits of the key for $\mathsf{HSKD}$, applies the padding function and sets $S$ to be the padded suffix of the key. Next, instead of computing $\mathsf{2CSC}$ itself, adversary $\mathcal{A}_{\mathsf{2CSC}}$ queries its own $\textsc{Fn}$ oracle under the newly created key on input $\mathtt{IV}$ and $S$.

This way, adversary $\mathcal{A}_{\mathsf{2CSC}}$ simulates game $\mathrm{G}_{(d-1)}$ ($d \in \{0,1\}$) for $\mathcal{A}$ when playing game $\mathbf{G}^{\text{PRF-}d}_{\mathsf{2CSC}}$ itself. This gives the bound in Equation (28). $\mathcal{A}_{\mathsf{2CSC}}$ makes at most $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ $\textsc{New}$ queries and at most one $\textsc{Fn}$ query under each key (hence they are all trivially prefix free). The second component $S$ in each $\textsc{Fn}$ query is at most $n = \lceil \ell_{\mathsf{max}}(\mathcal{A})/b \rceil$ blocks, since $S$ consists of all but the first block of the padded key $\overline{K}$ and $|\overline{K}|/b \le \lceil \ell_{\mathsf{max}}(\mathcal{A})/b \rceil + 1$, since $|K| \le \ell_{\mathsf{max}}(\mathcal{A})$ and the padding adds at most one extra block.

Hence, by definition of $\mathrm{G}_0$ and applying Equation (28), we have

$$
\begin{aligned}
\Pr[\mathbf{G}^{\text{vsl-PRG-1}}_{\mathsf{HSKD}}(\mathcal{A})] &= \Pr[\mathrm{G}_0(\mathcal{A})] \\
&= (\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]) + \Pr[\mathrm{G}_1(\mathcal{A})] \\
&\le \mathbf{Adv}^{\text{PRF}}_{\mathsf{2CSC}}(\mathcal{A}_{\mathsf{2CSC}}) + \Pr[\mathrm{G}_1(\mathcal{A})]\ . \qquad (29)
\end{aligned}
$$

▷ *Replacing* $\mathsf{h}^{\leftrightarrows}$ *with a random function.* Next, we simplify $\mathrm{G}_1$ to the semantically equivalent $\mathrm{G}_2$, shown in Figure 12. Game $\mathrm{G}_3$, also in Figure 12, is identical to $\mathrm{G}_2$, except that lines 3 and 4 are replaced by line 5. We construct an adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ such that

$$
\Pr[\mathrm{G}_2(\mathcal{A})] - \Pr[\mathrm{G}_3(\mathcal{A})] \le \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{zio},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}})\ . \qquad (30)
$$

Adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$, shown in Figure 12, runs adversary $\mathcal{A}$, acting as the challenger in $\mathrm{G}_2$, except in two ways. First, instead of sampling $X$ on its own as on lines 1 and 2, adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ calls oracle $\textsc{New}$ in game $\mathbf{G}^{\text{rka-PRF}}_{\Phi_{\mathsf{zio},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}})$ on input $\ell$ if $\ell \le b$ and else on $c$. Then, instead of invoking $\mathsf{h}^{\leftrightarrows}$ to create $K_i$ and $K_o$ as on lines 3 and 4 in Figure 12, $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ calls its $\textsc{Fn}$ oracle under the new key and second RKD parameter `ipad` and `opad`, respectively.

When adversary $\mathcal{A}$ halts and returns, adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ also halts and returns the same output. This way, adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ soundly simulates $\mathrm{G}_2$ and $\mathrm{G}_2$ when playing game $\mathbf{G}^{\text{rka-PRF-}d}_{\Phi_{\mathsf{zio},a},\mathsf{h}^{\leftrightarrows}}$, with $d = 1$ and $d = 0$, respectively. Here, $a = \min(c, \ell_{\mathsf{min}}(\mathcal{A}))$ is the smallest value on which $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ calls oracle $\textsc{New}$ while simulating the game for $\mathcal{A}$. This justifies the claimed bound in Equation (30).

Furthermore, game $\mathrm{G}_3$ is equivalent to the "random" vsl-PRG game $\mathbf{G}^{\text{vsl-PRG-0}}_{\mathsf{HSKD}}(\mathcal{A})$. Hence, putting the above together, we have

$$
\mathbf{Adv}^{\text{vsl-PRG}}_{\mathsf{HSKD}}(\mathcal{A}) \le \mathbf{Adv}^{\text{PRF}}_{\mathsf{2CSC}}(\mathcal{A}_{\mathsf{2CSC}}) + \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{zio},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}})\ . \qquad (31)
$$

▷ *Applying Lemma 1.* We now apply Lemma 1 to bound $\mathbf{Adv}^{\text{PRF}}_{\mathsf{2CSC}}(\mathcal{A}_{\mathsf{2CSC}})$ in Equation (31) by the advantages of two adversaries $\mathcal{A}_{\mathsf{f}}$ and $\mathcal{A}_{\mathsf{h}}$, both constructed as in the proof of Lemma 1.

This gives

$$
\mathbf{Adv}^{\text{PRF}}_{\mathsf{2CSC}}(\mathcal{A}_{\mathsf{2CSC}}) \le \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{f}}) + n \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}). \qquad (32)
$$

Adversary $\mathcal{A}_{\mathsf{f}}$ makes $\mathbf{Q}^{\textsc{New}}(\mathcal{A}_{\mathsf{2CSC}}) \le \mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ queries to $\textsc{New}$ and $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{2CSC}}) \le \mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ to $\textsc{Fn}$. Adversary $\mathcal{A}_{\mathsf{h}}$ makes at most $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{2CSC}}) \le \mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ queries to $\textsc{New}$ and $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{2CSC}}) \le \mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ to $\textsc{Fn}$.

Combining Equation (31) and Equation (32) gives the claimed bound in Equation (27) of the Theorem statement. □

<u>HSKD IS COLLISION RESISTANT FOR FEASIBLE INPUT SPACES.</u> As already pointed out in the introduction, it is easy to find colliding messages for $\mathsf{HMAC}^{\leftrightarrows}$, due to how short key inputs (now

$\underline{\textsc{Fn}(\ell)}$ // Games $\mathrm{G}_0$, $\mathrm{G}_1$

$K \leftarrow_{\$} \{0,1\}^\ell$
If $\ell \le b$ then $K_b \leftarrow K \,\|\, 0^{b-\ell}$
Else:
  $\overline{K} \leftarrow K \| \mathsf{pad}(\ell)$; $S \leftarrow \overline{K}[b+1..|\overline{K}|]$
  $X \leftarrow \mathsf{2CSC}[\mathsf{h}^{\leftrightarrows}, \mathsf{h}](K[1..b], (\mathtt{IV}, S))$ // $\mathrm{G}_0$
  $X \leftarrow_{\$} \{0,1\}^c$ // $\mathrm{G}_1$
  $K_b \leftarrow X \,\|\, 0^{b-c}$
$K_i \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{ipad})$ ; $K_o \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{opad})$
Return $K_i \| K_o$

$\underline{\textsc{Fn}(\ell)}$ // Games $\mathrm{G}_2$, $\mathrm{G}_3$

If $\ell \le b$ then $X \leftarrow_{\$} \{0,1\}^\ell$
Else $X \leftarrow_{\$} \{0,1\}^c$
$K_i \leftarrow \mathsf{h}^{\leftrightarrows}(X \,\|\, 0^{b-|X|} \oplus \mathtt{ipad}, \mathtt{IV})$ // $\mathrm{G}_2$
$K_o \leftarrow \mathsf{h}^{\leftrightarrows}(X \,\|\, 0^{b-|X|} \oplus \mathtt{opad}, \mathtt{IV})$ // $\mathrm{G}_2$
$K_i \| K_o \leftarrow_{\$} \{0,1\}^{2c}$ // $\mathrm{G}_3$
Return $K_i \| K_o$

Adversary $\mathcal{A}_{\mathsf{2CSC}}()$:

$\underline{\textsc{Fn}^*(\ell)}$

$n \leftarrow n+1$; $K \leftarrow_{\$} \{0,1\}^\ell$
If $\ell \le b$ then $K_b \leftarrow K \,\|\, 0^{b-\ell}$
Else:
  $\textsc{New}()$ // Initialize $b$-bit key for 2CSC
  $\overline{K} \leftarrow K \| \mathsf{pad}(\ell)$; $S \leftarrow \overline{K}[b+1..|\overline{K}|]$
  $X \leftarrow \textsc{Fn}(n, (\mathtt{IV}, S))$
  $K_b \leftarrow X \,\|\, 0^{b-c}$
$K_i \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{ipad})$
$K_o \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{opad})$
Return $K_i \| K_o$

Adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}()$:

$\underline{\textsc{Fn}^*(\ell)}$

$n \leftarrow n+1$
If $\ell \le b$ then $\textsc{New}(\ell)$
Else $\textsc{New}(c)$
$K_i \leftarrow \textsc{Fn}(n, \mathtt{ipad}, \mathtt{IV})$
$K_o \leftarrow \textsc{Fn}(n, \mathtt{opad}, \mathtt{IV})$
Return $K_i \| K_o$

Figure 12: Games and adversaries for the proof of Proposition 7. Both adversary $\mathcal{A}_{\mathsf{2CSC}}$ (left) and $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ (right) run adversary $\mathcal{A}$, simulating access to oracle $\textsc{Fn}$ through subroutines $\textsc{Fn}^*$ and returning with the output of $\mathcal{A}$. Functions $\mathsf{h}^{\leftrightarrows}$ and $\mathsf{h}^*$ are the swapped function and cascade construction, respectively, associated to the compression function $\mathsf{h}\colon \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$, as defined in Figure 3. $\mathtt{IV}$, $\mathtt{ipad}$ and $\mathtt{opad}$ are public constants.

---

messages, under adversarial control) are padded with zeros and long key inputs are hashed down (then padded). Indeed, these trivial collisions are instances of bigger classes of collisions that exist for $\mathsf{HSKD}$, which for $\mathsf{HMAC}^{\leftrightarrows}$ is applied to the message input. Since such collisions in $\mathsf{HSKD}$ immediately yield PRF attacks on $\mathsf{HMAC}^{\leftrightarrows}$, the natural question thus is: for which input spaces is $\mathsf{HSKD}$ collision resistant?

Let $\mathsf{HSKD}[\mathcal{S}]\colon \mathcal{S} \to \{0,1\}^{2c}$ be the restriction of $\mathsf{HSKD}$ to the input space $\mathcal{S}$, corresponding to $\mathsf{HMAC}[\mathcal{S}]$, i.e., restricting the key space of $\mathsf{HMAC}$ to $\mathcal{S}$. Focusing on length-closed sets, we now give a complete characterization of the class of *feasible* sets for which we will show $\mathsf{HSKD}[\mathcal{S}]$ to be collision resistant (enabling swap-PRF security of $\mathsf{HMAC}[\mathcal{S}]$). For all other sets, we will give attacks against swap-PRF security of $\mathsf{HMAC}[\mathcal{S}]$. Together with our attack results on $\mathsf{HMAC}^{\leftrightarrows}$ below, this will establish that $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$ is vkl-PRF secure *if and only if* $\mathcal{S}$ is feasible.

We can write any finite length-closed set as $\mathcal{S} = \{0,1\}^{\ell_1} \cup \cdots \cup \{0,1\}^{\ell_n}$, $\ell_1 < \cdots < \ell_n$, for $n, \ell_1, \ldots, \ell_n \in \mathbb{N}$. Then $\mathcal{S}$ belongs to (exactly) one of the following classes:

1. $\mathcal{S} \subseteq \{0,1\}^{>b}$: The class of *feasible* sets of only "long" keys (of lengths $> b$), denoted $\mathcal{S}_L$.

2. $\mathcal{S} \subseteq \{0,1\}^{\le b}$: We distinguish between

(2a) $n = 1$: The class of <u>*feasible* sets of the form $\{0,1\}^\ell$ (for some $\ell \le b$) of "short", fixed-length keys</u>, denoted $\mathcal{S}_S$.

(2b) $n > 1$: We give <u>*0-padding* attacks</u> for these sets.

3. $\exists i, j \in [1, n] : \ell_i \le b \wedge \ell_j > b$: We distinguish between

(3a) $c \le \ell_i \le b$: We give <u>*hash-confusion* attacks</u> for these sets.

(3b) $\ell_i < c$: We give <u>*hash-suffix* attacks</u> for these sets. (Note that a key length smaller than the $c$-bit output length of the compression function is an unnatural choice of keys. We nevertheless discuss attacks.)

We now first establish that $\mathsf{HSKD}[\mathcal{S}]$ is collision resistant for the classes of *feasible* sets of "short", fixed-length keys $\mathcal{S}_S = \{0,1\}^\ell$ (for some $\ell \le b$) and only "long" keys $\mathcal{S}_L \subseteq \{0,1\}^{>b}$, assuming collision resistance of the compression function $\mathsf{h}$. Essentially, keys from feasible sets colliding under $\mathsf{HSKD}$ requires a collision under one of the internal $\mathsf{h}$ calls. Notably, our results for feasible sets mean that $\mathsf{HSKD}$ is collision-resistant whenever *fixed-length* keys are used (no matter the length). This in particular covers the dual-PRF usage of HMAC in practice, since protocols like TLS 1.3 [49], KEMTLS [52] and MLS [23] use fixed-length key inputs; see Appendix A for a detailed discussion.

In Proposition 9, we then complete this characterization by giving attacks on the PRF security of $\mathsf{HMAC}^{\leftrightarrows}$ for all other (*infeasible*) input sets, which generically emerge through finding collisions in $\mathsf{HSKD}$.

**Proposition 8** (CR security of $\mathsf{HSKD}[\mathcal{S}]$)**.** *Let* $\mathsf{h}$ *and* $\mathsf{HSKD}$ *be defined as in Figure 3. Let* $\mathsf{HSKD}[\mathcal{S}] \colon \mathcal{S} \to \{0,1\}^{2c}$ *be the restriction of* $\mathsf{HSKD}$ *to a feasible set* $\mathcal{S}$ *of type* $\mathcal{S}_S$ *or* $\mathcal{S}_L$ *defined above. Let* $\mathcal{A}$ *be a CR adversary for* $\mathsf{HSKD}[\mathcal{S}]$*. Then we can construct an adversary* $\mathcal{A}_{\mathsf{h}}$ *such that*

$$\mathbf{Adv}^{\mathrm{CR}}_{\mathsf{HSKD}[\mathcal{S}]}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{CR}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) \ . \tag{33}$$

*The running time of the constructed adversary is about the same as that of* $\mathcal{A}$*.*

*Proof of Proposition 8.* Recall the definition of $\mathsf{HSKD}$ in Figure 3, unfolding $\mathsf{PoH}_b$ and $\mathsf{HSKD}_b$. For inputs $K$ from $\mathcal{S}_S$, we are in the case $|K| \le b$, for inputs from $\mathcal{S}_L$, we are in the case $|K| > b$, so:

$\underline{\mathsf{HSKD}[\mathcal{S}](K)}$
If $\mathcal{S} = \mathcal{S}_S$ then $K_b \leftarrow K \,\|\, 0^{b-|K|}$ // $\mathsf{PoH}_b$
If $\mathcal{S} = \mathcal{S}_L$ then $K_b \leftarrow \mathsf{H}(K) \,\|\, 0^{b-c}$ // $\mathsf{PoH}_b$
$K_i \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{ipad})$ ; $K_o \leftarrow \mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{opad})$ // $\mathsf{HSKD}_b$
Return $K_i \| K_o$

▷ *Case* $\mathcal{S}_S$. We construct $\mathcal{A}_{\mathsf{h}}$ as follows: $\mathcal{A}_{\mathsf{h}}$ simulates the CR game for $\mathcal{A}$. When $\mathcal{A}$ outputs $(K, K')$ such that $\mathsf{HSKD}[\mathcal{S}_S](K) = \mathsf{HSKD}[\mathcal{S}_S](K')$, $\mathcal{A}_{\mathsf{h}}$ computes $K_b = (K \| 0^{b-\ell}) \oplus \mathtt{ipad}$, $K'_b = (K' \| 0^{b-\ell}) \oplus \mathtt{ipad}$, and outputs $((\mathtt{IV}, K_b), (\mathtt{IV}, K'_b))$. Since $K, K' \in \mathcal{S}_S = \{0,1\}^\ell$ (for fixed $\ell \le b$), both keys get padded with the same number of zero bits. Hence, if $K \ne K'$, then $K_b \ne K'_b$, and $\mathbf{Adv}^{\mathrm{CR}}_{\mathsf{HSKD}[\mathcal{S}]}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{CR}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}})$.

▷ *Case* $\mathcal{S}_L$. If $\mathcal{A}$ outputs a collision $(K, K')$ such that $\mathsf{HSKD}[\mathcal{S}_L](K) = \mathsf{HSKD}[\mathcal{S}_L](K')$, it follows that $\mathsf{h}(\mathtt{IV}, K_b \oplus \mathtt{ipad}) = \mathsf{h}(\mathtt{IV}, K'_b \oplus \mathtt{ipad})$, and likewise for $\mathtt{opad}$. Since $K, K' \in \mathcal{S}_L \subseteq \{0,1\}^{>b}$, both keys get hashed and then padded to obtain $K_b \leftarrow \mathsf{H}(K) \,\|\, 0^{b-c}$ resp. $K'_b \leftarrow \mathsf{H}(K') \,\|\, 0^{b-c}$. This means that either $K_b \ne K'_b$, and there was a collision in the outputs of $\mathsf{h}$, or that $K_b = K'_b$, and there was a collision in the outputs of $\mathsf{H}$.

We construct $\mathcal{A}_{\mathsf{h}}$ as follows: Run $\mathcal{A}$ to obtain an $\mathsf{HSKD}$ collision $(K, K')$, and compute $K_b \leftarrow \mathsf{PoH}_b(K)$ and $K'_b \leftarrow \mathsf{PoH}_b(K')$. If $K_b \ne K'_b$, output $((\mathtt{IV}, K_b), (\mathtt{IV}, K'_b))$ as a collision for $\mathsf{h}$. Otherwise, following the proof of CR for $\mathsf{H}$ as the Merkle–Damgård [45, 25] iteration of $\mathsf{h}$, trace backwards through the computation of $\mathsf{H}(K) = \mathsf{H}(K')$ to find the compression function calls that collide on the output for distinct input values in $\mathsf{H}(K) = \mathsf{H}(K')$ and output those colliding values. (Such collision must exist since $K \neq K'$.) Hence, $\mathcal{A}_{\mathsf{h}}$ outputs a collision for $\mathsf{h}$ when $\mathcal{A}$ outputs one for $\mathsf{HSKD}$, justifying Equation (33). □

**Proposition 9** (PRF-insecurity of $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$ for infeasible $\mathcal{S}$)**.** *Consider* $\mathsf{HMAC}^{\leftrightarrows}[\mathcal{S}_i] \colon \{0,1\}^* \times \mathcal{S}_i \to \{0,1\}^c$ *for the following infeasible* $\mathcal{S}_i$ *from the characterization above:*

1. $\mathcal{S}_1$ *(case 2b), with* $\{0,1\}^{\ell_i} \cup \{0,1\}^{\ell_j} \subseteq \mathcal{S}_1$ *for* $\ell_i \neq \ell_j$ *and* $\ell_i < \ell_j \leq b$*,*
2. $\mathcal{S}_2$ *(case 3a), with* $\{0,1\}^{\ell_i} \cup \{0,1\}^{\ell_j} \subseteq \mathcal{S}_2$ *for* $c \leq \ell_i \leq b$ *and* $\ell_j > b$*, and*
3. $\mathcal{S}_3$ *(case 3b), with* $\{0,1\}^{\ell_i} \cup \{0,1\}^{\ell_j} \subseteq \mathcal{S}_3$ *for* $\ell_i < c$ *and* $\ell_j > b$*.*

*Then* $\mathsf{HMAC}^{\leftrightarrows}[\mathcal{S}_i]$ *is* <u>*not*</u> *PRF secure. Concretely, we construct adversaries* $\mathcal{A}_i$ *such that*

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{HMAC}^{\leftrightarrows}[\mathcal{S}_1]}(\mathcal{A}_1) = \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{HMAC}^{\leftrightarrows}[\mathcal{S}_2]}(\mathcal{A}_2) = 1 - 2^{-c}\ , \ \text{and} \tag{34}$$

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{HMAC}^{\leftrightarrows}[\mathcal{S}_3]}(\mathcal{A}_3) = 2^{\ell_i - c} - 2^{-c}\ , \tag{35}$$

*which make one* New *and two* Fn *queries and run in small, constant time. The bound for* $\mathcal{A}_3$ *conservatively assumes* $\mathsf{H}$ *behaves like a random oracle.*

Note that for the rather unnatural set $\mathcal{S}_3$ (case 3b), the adversary $\mathcal{A}_3$ we construct has advantage proportional to $2^{\ell_i - c}$. If $|\ell_i - c|$ is small, then the adversary has high probability of winning the PRF game. If $|\ell_i - c|$ is large, this means $\ell_i$ is much smaller than the compression function output length, in which case a better strategy for attacking the PRF security via exhaustive key search (taking about $2^{\ell_i}$ time for keys of length $\ell_i$) might come in reach.

*Proof of Proposition 9.* Recall that $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows} = \mathbf{Comp}[\mathsf{NMAC}^{\leftrightarrows}, \mathsf{HSKD}[\mathcal{S}]]$.

We start by observing that an adversary $\mathcal{A}_{\mathsf{SKD}}$ against the collision resistance of $\mathsf{HSKD}[\mathcal{S}]$ can directly be used to build an adversary $\mathcal{A}$ against the PRF security of $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$. Adversary $\mathcal{A}$ runs $\mathcal{A}_{\mathsf{SKD}}$ to obtain a collision $(M, M')$—that is, $M \neq M'$ with $\mathsf{HSKD}[\mathcal{S}](M) = \mathsf{HSKD}[\mathcal{S}](M')$. It then issues a New() query and two Fn queries, $Y \leftarrow \mathrm{Fn}(1, M)$ and $Y' \leftarrow \mathrm{Fn}(1, M')$, to its PRF game, and outputs 1 if $Y = Y'$. In the real world ($b = 1$), we always have $Y = \mathsf{HMAC}^{\leftrightarrows}(K, M) = \mathsf{HMAC}^{\leftrightarrows}(K, M') = Y'$; in the random world ($b = 0$), $Y, Y'$ are drawn at random from $\{0,1\}^c$, colliding with probability $2^{-c}$. Hence,

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{CR}}_{\mathsf{HSKD}[\mathcal{S}]}(\mathcal{A}_{\mathsf{SKD}}) - 2^{-c}. \tag{36}$$

Let us now examine the three choices of $\mathcal{S}_i$ separately, and describe the collision-finding strategy for $\mathsf{HSKD}[\mathcal{S}_i]$ in each case.

▷ *0-padding attacks in* $\mathcal{S}_1$. Define $\mathcal{A}_{\mathsf{SKD}}$ to pick any $M \in \{0,1\}^{\ell_i}$, set $M' = M \,\|\, 0^{\ell_j - \ell_i}$, and output $(M, M')$. Due to $\mathsf{PoH}_b$ zero-padding both messages to $b$ bits, $(M, M')$ is a collision for $\mathsf{HSKD}$ with probability 1. Via Eq. (36), this yields the claim for $\mathcal{S}_1$.

▷ *Hash-confusion attacks in* $\mathcal{S}_2$. Define $\mathcal{A}_{\mathsf{SKD}}$ to pick any $M \in \{0,1\}^{\ell_j}$, set $M' = \mathsf{H}(M) \,\|\, 0^{b-c}$, and output $(M, M')$. Due to $\mathsf{PoH}_b$ hashing down and then zero-padding keys longer than $b$ bits, $(M, M')$ is a collision for $\mathsf{HSKD}$ with probability 1. Via Eq. (36), this yields the claim for $\mathcal{S}_2$.

▷ *Hash-suffix attacks in* $\mathcal{S}_3$. As for $\mathcal{S}_2$, let $\mathcal{A}_{\mathsf{SKD}}$ pick any $M \in \{0,1\}^{\ell_j}$, set $M' = \mathsf{H}(M) \,\|\, 0^{b-c}$, and output $(M, M')$. Due to $\mathsf{PoH}_b$ hashing down and then zero-padding keys longer than $b$ bits, $(M, M')$ is a collision for $\mathsf{HSKD}$ if the last $c - \ell_i$ bits of $\mathsf{H}(M)$ are zeros. Conservatively assuming $\mathsf{H}$ behaves like a random oracle, the probability for this is $2^{c-\ell_i}$. Via Eq. (36), this yields the claim for $\mathcal{S}_3$. □

### 6.3 vkl-PRF Security of $\mathsf{HMAC}$

Existing proofs of PRF security of $\mathsf{HMAC}$ assume a $b$-bit key [8, 5, 6], i.e., they actually only hold for $\mathsf{HMAC}_b$. As standardized [39, 46], $\mathsf{HMAC}$ however allows keys of varying length. Combining Theorem 6.1 with our results on multi-user PRF security of $\mathsf{NMAC}$ (Theorem 5) and vsl-PRG security of $\mathsf{HSKD}$ (Proposition 7), we establish vkl-PRF security for $\mathsf{HMAC}$ through the composition $\mathsf{HMAC} = \mathbf{Comp}[\mathsf{NMAC}, \mathsf{HSKD}]$, assuming dual-PRF security of $\mathsf{h}$ and rka-PRF security of $\mathsf{h}^{\leftrightarrows}$.

**Theorem 10** (vkl-PRF security of $\mathsf{HMAC}$)**.** *Let $c$, $b$, and* $\mathsf{pad}$ *be as in Section 4, and functions* $\mathsf{h}$, $\mathsf{HMAC}$ *be as defined in Figure 3. Let $\mathcal{A}$ be an adversary against the vkl-PRF security of* $\mathsf{HMAC}$*. Assume each of its* $\textsc{Fn}$ *queries has length at most $L$, and let $m = 1 + \lceil L/b \rceil$. Let $q_f = \mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ and assume $m \cdot q_f < 2^b$. Assume $\mathcal{A}$'s queries to* $\textsc{New}$ *have minimal and maximal key length $\ell_{\mathsf{min}}(\mathcal{A})$ resp. $\ell_{\mathsf{max}}(\mathcal{A})$. Let $n = \lceil \ell_{\mathsf{max}}(\mathcal{A})/b \rceil$, $a = \min(c, \ell_{\mathsf{min}}(\mathcal{A}))$, and $\Phi_{\mathsf{zio},a}$ be the "zero-pad-then-xor" RKD function defined in Section 6.2. Then we can construct adversaries $\mathcal{A}_{\mathsf{h}}$, $\mathcal{A}_{\mathsf{f}}$ and $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ such that*

$$\begin{aligned}\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{HMAC}}(\mathcal{A}) \leq (m+2+n) \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) + \frac{q_f(q_f-1)}{2^{c+1}} \\ + \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{f}}) + \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{zio},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) \; . \end{aligned} \tag{37}$$

*Adversary $\mathcal{A}_{\mathsf{h}}$ makes* $\max(\mathbf{Q}^{\textsc{New}}(\mathcal{A}), \mathbf{Q}^{\textsc{Fn}}(\mathcal{A}))$ *queries to each of* $\textsc{New}$ *and* $\textsc{Fn}$*. Adversaries $\mathcal{A}_{\mathsf{f}}$, $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ make $\mathbf{Q}^{\textsc{New}}(\mathcal{A})$ queries to* $\textsc{New}$*, $\mathcal{A}_{\mathsf{f}}$ makes at most $\mathbf{Q}^{\textsc{New}}(\mathcal{A})$ queries to* $\textsc{Fn}$*, and $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ makes $2 \cdot \mathbf{Q}^{\textsc{New}}(\mathcal{A})$ to* $\textsc{Fn}$*. The running times of adversaries $\mathcal{A}_{\mathsf{h}}, \mathcal{A}_{\mathsf{f}}$ and $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ are about the same as that of $\mathcal{A}$.*

*Proof of Theorem 10.* Applying Theorem 6.1 to $\mathsf{HMAC} = \mathbf{Comp}[\mathsf{NMAC}, \mathsf{HSKD}]$ gives us adversaries $\mathcal{A}_{\mathsf{MAC}}$ and $\mathcal{A}_{\mathsf{SKD}}$ such that

$$\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{HMAC}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{PRF}}_{\mathsf{NMAC}}(\mathcal{A}_{\mathsf{MAC}}) + \mathbf{Adv}^{\text{vsl-PRG}}_{\mathsf{HSKD}}(\mathcal{A}_{\mathsf{SKD}}) \; .$$

In terms of oracle queries, we have $\mathbf{Q}^{\textsc{New}}(\mathcal{A}_{\mathsf{MAC}}) = \mathbf{Q}^{\textsc{New}}(\mathcal{A})$ and $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{MAC}}) = \mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$, while $\mathcal{A}_{\mathsf{SKD}}$ makes one $\textsc{Fn}$ query for each of the $\mathbf{Q}^{\textsc{New}}(\mathcal{A})$ users and has minimal and maximal seed length $\ell_{\mathsf{min}}(\mathcal{A})$ resp. $\ell_{\mathsf{max}}(\mathcal{A})$.

We next apply Theorem 5 to obtain

$$\mathbf{Adv}^{\text{PRF}}_{\mathsf{NMAC}}(\mathcal{A}_{\mathsf{MAC}}) \leq (m+2) \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},1}) + \frac{q_f(q_f-1)}{2^{c+1}}$$

for $q_f = \mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{MAC}}) = \mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$ and an adversary $\mathcal{A}_{\mathsf{h},1}$ with $\mathbf{Q}^{\textsc{New}}(\mathcal{A}_{\mathsf{h},1}) = \mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{h},1}) = \mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{MAC}}) = \mathbf{Q}^{\textsc{Fn}}(\mathcal{A})$.

We then apply Proposition 7, yielding

$$\mathbf{Adv}^{\text{vsl-PRG}}_{\mathsf{HSKD}}(\mathcal{A}_{\mathsf{SKD}}) \leq \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{f}}) + n \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},2}) + \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{zio},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}})$$

for adversaries $\mathcal{A}_{\mathsf{f}}$, $\mathcal{A}_{\mathsf{h},2}$, $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ and terms $n = \lceil \ell_{\mathsf{max}}(\mathcal{A})/b \rceil$ (the block length of the maximal-length key) and $a = \min(c, \ell_{\mathsf{min}}(\mathcal{A}_{\mathsf{SKD}})) = \min(c, \ell_{\mathsf{min}}(\mathcal{A}))$ (the shortest RKA key-prefix).

In terms of oracle queries, we have

$$\begin{aligned}&\mathbf{Q}^{\textsc{New}}(\mathcal{A}_{\mathsf{f}}) = \mathbf{Q}^{\textsc{New}}(\mathcal{A}_{\mathsf{h},2}) = \mathbf{Q}^{\textsc{New}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) = \mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{SKD}}) = \mathbf{Q}^{\textsc{New}}(\mathcal{A}) \; , \\ &\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{f}}), \mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{h},2}) \leq \mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{SKD}}) = \mathbf{Q}^{\textsc{New}}(\mathcal{A}), \text{ and} \\ &\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) = 2 \cdot \mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{SKD}}) = 2 \cdot \mathbf{Q}^{\textsc{New}}(\mathcal{A}) \; .\end{aligned}$$

Finally we merge adversaries $\mathcal{A}_{\mathsf{h},1}, \mathcal{A}_{\mathsf{h},2}$ using Lemma 3 with weights $w_1 = m+2$ and $w_2 = n$ to get an adversary $\mathcal{A}_{\mathsf{h}}$ such that

$$(m+2) \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},1}) + n \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},2}) \leq (m+2+n) \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) \; .$$

with oracle query counts $\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}_{\mathsf{h}}) = \mathbf{Q}^{\textsc{New}}(\mathcal{A}_{\mathsf{h}}) = \max(\mathbf{Q}^{\textsc{Fn}}(\mathcal{A}), \mathbf{Q}^{\textsc{New}}(\mathcal{A}))$.

Summing the terms and collecting the adversary resources yields the claim. □

## 6.4 vkl-PRF Security of $\mathsf{HMAC}^{\leftrightarrows}$

We finally turn to the swap-PRF security of $\mathsf{HMAC}$. Letting $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}\colon \{0,1\}^* \times \mathcal{S} \to \{0,1\}^c$ denote the restriction of $\mathsf{HMAC}^{\leftrightarrows}$ to keys from a set $\mathcal{S}$, we establish vkl-PRF security of $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$ for the *feasible* key input spaces of the forms $\mathcal{S}_S = \{0,1\}^\ell$ (for some $\ell \leq b$) and $\mathcal{S}_L \subseteq \{0,1\}^{>b}$.

Again, we obtain this result via Theorem 6.2 for the composition $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows} = \mathbf{Comp}[\mathsf{NMAC}^{\leftrightarrows}, \mathsf{HSKD}[\mathcal{S}]]$. Leveraging our vkl-PRF security result of $\mathsf{NMAC}^{\leftrightarrows}$ (Theorem 4) and collision resistance of $\mathsf{HSKD}$ for feasible $\mathcal{S}$ (Proposition 8), we establish vkl-PRF security for $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$ assuming PRF security of $\mathsf{h}$, rka-PRF security of $\mathsf{h}^{\leftrightarrows}$, and collision resistance of $\mathsf{h}$.

One might note that CR of $\mathsf{h}$ is *not* assumed in the proofs of PRF security for $\mathsf{HMAC}_b$ [5, 30]. So why extra assumptions for $\mathsf{HMAC}^{\leftrightarrows}$? The answer is that collisions in the hash function $\mathsf{H}$ lead to attacks violating PRF security of $\mathsf{HMAC}^{\leftrightarrows}$, as mentioned before and detailed in Section 6.2. Collision resistance of $\mathsf{h}$ rules this out.

**Theorem 11** (vkl-PRF security of $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$)**.** *Let $c$, $b$, and $\mathsf{pad}$ be as in Section 4, and functions $\mathsf{h}$, $\mathsf{HMAC}^{\leftrightarrows}$ be as defined in Figure 3. Let $\mathcal{S}$ be a feasible key input space as defined in Section 6.2 and $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}\colon \{0,1\}^* \times \mathcal{S} \to \{0,1\}^c$ be the swap of $\mathsf{HMAC}$ restricted to keys from $\mathcal{S}$. Let $\mathcal{A}$ be an adversary against the vkl-PRF security of $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$ whose* New *queries have minimal and maximal key length $\ell_{\mathsf{min}}(\mathcal{A})$ resp. $\ell_{\mathsf{max}}(\mathcal{A})$. Let $n = \lceil \ell_{\mathsf{max}}(\mathcal{A})/b \rceil$, $a = \min(c, \ell_{\mathsf{min}}(\mathcal{A}))$, and $\Phi_{\mathsf{pad},a}$ be the "padding" RKD function defined in Section 5.1. Then we can construct adversaries $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$, $\mathcal{A}_{\mathsf{h},1}$, $\mathcal{A}_{\mathsf{h},2}$ such that*

$$\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{pad},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) + n \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},1}) + \mathbf{Adv}^{\text{CR}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},2}) \,. \tag{38}$$

*Adversary $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ makes at most $\max(\mathbf{Q}^{\text{New}}(\mathcal{A}), \mathbf{Q}^{\text{Fn}}(\mathcal{A}))$ queries to* New *and at most $\mathbf{Q}^{\text{Fn}}(\mathcal{A})$ to* Fn*. $\mathcal{A}_{\mathsf{h},1}$ makes $\mathbf{Q}^{\text{Fn}}(\mathcal{A})$ queries to each of* New *and* Fn*. The running times of the constructed adversaries are about the same as that of $\mathcal{A}$.*

*Proof of Theorem 11.* We apply Theorem 6.2 to $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows} = \mathbf{Comp}[\mathsf{NMAC}^{\leftrightarrows}, \mathsf{HSKD}[\mathcal{S}]]$ which yields adversaries $\mathcal{A}_{\mathsf{MAC}}$ and $\mathcal{A}_{\mathsf{SKD}}$ such that

$$\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{NMAC}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{MAC}}) + \mathbf{Adv}^{\text{CR}}_{\mathsf{HSKD}[\mathcal{S}]}(\mathcal{A}_{\mathsf{SKD}}) \,. \tag{39}$$

In terms of oracle queries, we have $\mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{MAC}}) = \mathbf{Q}^{\text{New}}(\mathcal{A})$ and $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{MAC}}) = \mathbf{Q}^{\text{Fn}}(\mathcal{A})$.

We next apply Theorem 4 to obtain

$$\mathbf{Adv}^{\text{vkl-PRF}}_{\mathsf{NMAC}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{MAC}}) \leq 2 \cdot \mathbf{Adv}^{\text{rka-PRF}}_{\Phi_{\mathsf{pad},a},\mathsf{h}^{\leftrightarrows}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) + n \cdot \mathbf{Adv}^{\text{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},1}) \,. \tag{40}$$

for adversaries $\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}$ and $\mathcal{A}_{\mathsf{h},1}$ with

$$\mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) = \max(\mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{MAC}}), \mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{MAC}})) = \max(\mathbf{Q}^{\text{New}}(\mathcal{A}), \mathbf{Q}^{\text{Fn}}(\mathcal{A}))$$

and $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^{\leftrightarrows}}) \leq \mathbf{Q}^{\text{Fn}}(\mathcal{A})$, and adversary $\mathcal{A}_{\mathsf{h},1}$ with

$$\mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{h},1}) = \mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h},1}) = \mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{MAC}}) = \mathbf{Q}^{\text{Fn}}(\mathcal{A}).$$

We then apply Proposition 8 to obtain, for any feasible key input space $\mathcal{S}$,

$$\mathbf{Adv}^{\text{CR}}_{\mathsf{HSKD}[\mathcal{S}]}(\mathcal{A}_{\mathsf{SKD}}) \leq \mathbf{Adv}^{\text{CR}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h},2}) \,. \tag{41}$$

Summing the terms and collecting the adversary resources yields the claim. $\square$

# 7 Proofs for the rka-PRF Security of $\mathsf{h}^{\leftrightarrows}$

We study the assumption of rka-PRF security of $\mathsf{h}^{\leftrightarrows}$, needed for the vkl-PRF security of $\mathsf{NMAC}^{\leftrightarrows}$ (Theorem 4) and the vsl-PRG security of $\mathsf{HSKD}$ (Proposition 7) when $\mathsf{h}$ is the Davies-Meyer hash function. That is, we let $\mathsf{h}(X, L) = \mathsf{E}(L, X) \oplus X$, where $\mathsf{E}\colon \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$ is the underlying block cipher with $b$-bit key and $c$-bit input and output.

The assumption that $\mathsf{h}^{\leftrightarrows}$ is rka-PRF secure has already been made [5, 6] in the single-user setting for what, in our notation is the RKD function $\Phi_{\mathsf{io}}\colon \{\varepsilon\} \times \{\mathtt{ipad}, \mathtt{opad}\} \times \{0,1\}^b \to \{0,1\}^b$ defined by $\Phi_{\mathsf{io}}(\varepsilon, \mathsf{io}, L) := L \oplus \mathsf{io}$ for $\mathsf{io} \in \{\mathtt{ipad}, \mathtt{opad}\}$. (Here we recast the single-user definition of [11] in our multi-user notation and define the RKD function as a multi-variable function, where the second variable defines the behavior of the function, rather than define a set of RKD functions which each only take the key as input.)

As we noted in the introduction, Section 1.4, the PRF security of $\mathsf{h}^{\leftrightarrows}$ is arguably a *milder* assumption than the PRF security of $\mathsf{h}$ itself. This is because in the Davies-Meyer construction, the key $X$ for $\mathsf{h}$ is the message for $\mathsf{E}$. But if we consider $\mathsf{h}^{\leftrightarrows}(L, X) = \mathsf{E}(L, X) \oplus X$ as a PRF, the key $L$ is actually the block cipher key. Thus (rka-)PRF security follows from (rka-)PRP security of $\mathsf{E}$. This is not true for $\mathsf{h}$.

In the following, we show that for any RKD function $\Phi$, the $\Phi$-rka-PRF security of $\mathsf{h}^{\leftrightarrows}$ follows from the $\Phi$-rka-PRP security of $\mathsf{E}$. Then, in Section 7.2, we justify the assumption that $\mathsf{E}$ is rka-PRP secure for the RKD functions $\Phi_{\mathsf{pad},a}$ and $\Phi_{\mathsf{zio},a}$ needed for our results.

## 7.1 $\Phi$-rka-PRF Security of $\mathsf{h}^{\leftrightarrows}$

**Proposition 12** (rka-PRF security of $\mathsf{h}^{\leftrightarrows}$ under rka-PRP assumptions)**.** *Let* $\mathsf{E}\colon \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$ *be a block cipher. Let the* $\mathsf{h}^{\leftrightarrows}\colon \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$ *be defined by* $\mathsf{h}^{\leftrightarrows}(L, X) = \mathsf{E}(L, X) \oplus X$*. Let* $\mathcal{A}$ *be an adversary against the* $\Phi$*-rka-PRF security of* $\mathsf{h}^{\leftrightarrows}$*. Then*

$$\mathbf{Adv}^{\text{rka-PRF}}_{\Phi,\mathsf{h}^{\leftrightarrows}}(\mathcal{A}) \leq \frac{\mathbf{Q}^{\text{FN}}(\mathcal{A})^2}{2^c} + \mathbf{Adv}^{\text{rka-PRP}}_{\Phi,\mathsf{E}}(\mathcal{A}_{\mathsf{E}}) \tag{42}$$

*for an adversary* $\mathcal{A}_{\mathsf{E}}$ *making* $\mathbf{Q}^{\text{NEW}}(\mathcal{A})$ *queries to oracle* New *and* $\mathbf{Q}^{\text{FN}}(\mathcal{A})$ *queries to oracle* Fn*.*

The idea of the proof is to first replace the block cipher invocation in $\mathsf{h}^{\leftrightarrows}$ by random permutations, a hop which is bounded by the advantage of adversary $\mathcal{A}_{\mathsf{E}}$ against the rka-PRP security of $\mathsf{E}$. Then, the permutations are replaced by random functions, incurring the additive term in the bound (cf. the multi-user PRP/PRF switching lemma in [10]).

*Proof of Proposition 12.* The proof uses a sequence of games $\mathrm{G}_0 - \mathrm{G}_3$, shown in Figure 13. The games are all executed with adversary $\mathcal{A}$.

▷ *Replacing* $\mathsf{E}$ *with random permutations.* We begin with game $\mathrm{G}_0$ being $\mathbf{G}^{\text{rka-PRF-1}}_{\Phi,\mathsf{h}^{\leftrightarrows}}(\mathcal{A})$. That is, $\mathrm{G}_0$ is the "real" rka-PRF game. The first game hop, to game $\mathrm{G}_1$, replaces the evaluation of the blockcipher $\mathsf{E}$ in oracle Fn by the evaluation of a random permutation. That is, $\mathrm{G}_0$ and $\mathrm{G}_1$ differ only in oracle Fn, where line 6 is only in $\mathrm{G}_0$ and line 7 is only in $\mathrm{G}_1$. By standard equation rewriting

$$\begin{aligned}\Pr[\mathbf{G}^{\text{rka-PRF-1}}_{\Phi,\mathsf{h}^{\leftrightarrows}}(\mathcal{A})] &= \Pr[\mathrm{G}_0(\mathcal{A})] \\ &= \Pr[\mathrm{G}_1(\mathcal{A})] + (\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]).\end{aligned} \tag{43}$$

We build an adversary $\mathcal{A}_{\mathsf{E}}$ such that

$$\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})] \leq \mathbf{Adv}^{\text{rka-PRP}}_{\Phi,\mathsf{E}}(\mathcal{A}_{\mathsf{E}}). \tag{44}$$

Game $G_0$, $G_1$:

$\underline{\textsc{New}(\alpha)}$

$n \leftarrow n+1$; $L_n \leftarrow\!\!\$\ \{0,1\}^b$; $\alpha_n \leftarrow \alpha$

For all $L \in \{0,1\}^b$ do:

  $\Pi_{n,L} \leftarrow\!\!\$\ \mathsf{PERM}[\{0,1\}^c]$

$\underline{\textsc{Fn}(i, \beta, X)}$

Assert: $\Phi_{\alpha_i,\beta}(L_i) \neq \perp$

$L_i' \leftarrow \Phi_{\alpha_i,\beta}(L_i)$

$Z \leftarrow \mathsf{E}(L_i', X)$ // $G_0$

$Z \leftarrow \Pi_{i,L_i'}(X)$ // $G_1$

Return $Z \oplus X$

Game $G_2$, $G_3$:

$\underline{\textsc{New}(\alpha)}$

$n \leftarrow n+1$; $L_n \leftarrow\!\!\$\ \{0,1\}^b$; $\alpha_n \leftarrow \alpha$

$\underline{\textsc{Fn}(i, \beta, X)}$

Assert: $\Phi_{\alpha_i,\beta}(L_i) \neq \perp$

$L_i' \leftarrow \Phi_{\alpha_i,\beta}(L_i)$

If $\mathrm{T}[i, L_i', X] \neq \perp$ then:

  Return $\mathrm{T}[i, L_i', X]$

$Z \leftarrow\!\!\$\ \{0,1\}^c$

If $Z \in \mathrm{T}[i, L_i', \cdot].\mathcal{R}$ then:

  $\mathsf{bad} \leftarrow 1$

  $Z \leftarrow\!\!\$\ \{0,1\}^c \setminus \mathrm{T}[i, L_i', \cdot].\mathcal{R}$ // $G_2$

$\mathrm{T}[i, L_i', X] \leftarrow Z$

Return $\mathrm{T}[i, L_i', X] \oplus X$

Figure 13: Games $G_0$-$G_3$ for the proof of Proposition 12. $\mathrm{T}[i, L_i', \cdot].\mathcal{R}$ is the set of all $Z \in \{0,1\}^c$ such that $Z = \mathrm{T}[i, L_i', X]$ for some $X$.

---

Adversary $\mathcal{A}_{\mathsf{E}}$ runs $\mathcal{A}$, acting as the challenger in $G_0$, except that it relays New and Fn queries from $\mathcal{A}$ to its own corresponding oracles in the $\mathbf{G}^{\text{rka-PRP}}_{\Phi,\mathsf{E}}$ game. Hence, adversary $\mathcal{A}_{\mathsf{E}}$ makes one New query for each of the $\mathbf{Q}^{\text{New}}(\mathcal{A})$ New queries made by $\mathcal{A}$ and one Fn query for each of the $\mathbf{Q}^{\text{Fn}}(\mathcal{A})$ Fn queries made by $\mathcal{A}$. This way, adversary $\mathcal{A}_{\mathsf{E}}$ soundly simulates $G_{(1-d)}$ when playing game $\mathbf{G}^{\text{rka-PRP-}d}_{\Phi,\mathsf{E}}$, for $d \in \{0,1\}$. When adversary $\mathcal{A}$ halts, $\mathcal{A}_{\mathsf{E}}$ halts and returns the same output. This yields the bound in Equation (44).

▷ *Replacing the random permutations with random functions.* We rewrite game $G_1$ as $G_2$ on the right in Figure 13, replacing the permutations sampled in oracle New by consistent lazy sampling in the Fn oracle. For each new query $\textsc{Fn}(i, \beta, X)$, the oracle initially samples $Z \leftarrow\!\!\$\ \{0,1\}^c$. If the sampled value violates the permutation condition, meaning that $Z$ was already sampled in response to an earlier Fn query under the same derived key for some $X' \neq X$, then a flag $\mathsf{bad}$ is set to 1 and $Z$ is sampled again from the set of unused strings in $\{0,1\}^c$. Since $G_2$ is syntactically different, but semantically equivalent to $G_1$, we have $\Pr[G_1(\mathcal{A})] = \Pr[G_2(\mathcal{A})]$, giving

$$\begin{aligned}\Pr[\mathbf{G}^{\text{rka-PRF-1}}_{\Phi,\mathsf{h}^{\leftrightarrows}}(\mathcal{A})] &\leq \Pr[G_2(\mathcal{A})] + \mathbf{Adv}^{\text{rka-PRP}}_{\Phi,\mathsf{E}}(\mathcal{A}_{\mathsf{E}}) \\ &= \Pr[G_3(\mathcal{A})] + (\Pr[G_2(\mathcal{A})] - \Pr[G_3(\mathcal{A})]) + \mathbf{Adv}^{\text{rka-PRP}}_{\Phi,\mathsf{E}}(\mathcal{A}_{\mathsf{E}}).\end{aligned} \tag{45}$$

by combining Equation (43) and Equation (44) and rewriting.

Next, game $G_3$ is identical to game $G_2$, except that $Z$ is not resampled after the $\mathsf{bad}$ flag is set to 1. That is, we replace the lazily sampled random permutations in $G_2$ by random functions in $G_3$. Because $G_2$ and $G_3$ are identical-until-$\mathsf{bad}$, we have

$$\Pr[G_2(\mathcal{A})] - \Pr[G_3(\mathcal{A})] \leq \Pr[G_3 \text{ sets } \mathsf{bad}] \tag{46}$$

by the fundamental lemma of game playing [13]. Furthermore, $G_3$ is equivalent to $\mathbf{G}^{\text{rka-PRF-0}}_{\Phi,\mathsf{h}^{\leftrightarrows}}(\mathcal{A})$, since each Fn query with a new $(i, L_i', X)$-tuple is responded to by $Z \oplus X$, where $Z$ is a randomly sampled string in $\{0,1\}^c$ and XOR preserves the uniform probability distribution.

Therefore, by combining Equation (45) and Equation (46) we have

$$\mathbf{Adv}^{\text{rka-PRF}}_{\Phi,\mathsf{h}^{\leftrightarrows}}(\mathcal{A}) \leq \Pr[G_3 \text{ sets } \mathsf{bad}] + \mathbf{Adv}^{\text{rka-PRP}}_{\Phi,\mathsf{E}}(\mathcal{A}_{\mathsf{E}}). \tag{47}$$

We claim that $\Pr[\mathrm{G}_3 \text{ sets } \mathsf{bad}] \le (\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A}))^2/2^c$, where $\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A})$ is the number of Fn queries made by adversary $\mathcal{A}$, yielding the bound in Proposition 12. Next we proceed to prove the claim.

We are trying to bound the collision probability of randomly sampled $c$-bit strings within each set $\mathrm{T}[i, L'_i, \cdot].\mathcal{R}$ for every $(i, L'_i)$ pair. Put differently, we want to apply a multi-user version of the PRP/PRF switching lemma, where each user corresponds to a pair $(i, L'_i)$, with the goal of bounding the probability that an adversary can distinguish the switch from random permutations to random functions.

At any point in time, the maximum size of $\mathrm{T}[i, L'_i, \cdot].\mathcal{R}$ is at most $\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A})$ for any $(i, L'_i)$, since each table entry is the result of a query to oracle Fn. Hence, when sampling $Z \leftarrow\$ \{0,1\}^c$, the probability that $Z \in \mathrm{T}[i, L'_i, \cdot].\mathcal{R}$ is at most $\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A})/2^c$, and this is also an upper bound on the probability that this query sets $\mathsf{bad}$ to 1. Since there are at most $\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A})$ queries to oracle Fn which can set $\mathsf{bad}$ in total, this gives $\Pr[\mathrm{G}_3 \text{ sets } \mathsf{bad}] \le (\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A}))^2/2^c$, by the union bound.

Note that this bound is somewhat loose, but gets close to the actual probability in the case when all Fn queries by adversary $\mathcal{A}$ are to the same $(i, L'_i)$ pair. The bound then would be the usual $(\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A}))^2/2^{c+1}$ from the PRP/PRF switching lemma.

□

### 7.2 Φ-rka-PRP Security of Block Ciphers in ICM

To validate the assumption on the block cipher $\mathsf{E}$ in Proposition 12, we prove the $\Phi$-rka-PRP security of $\mathsf{E}$ modeled as an ideal cipher for the RKD functions $\Phi_{\mathsf{pad},a}$ and $\Phi_{\mathsf{zio},a}$.

<u>Discussion.</u> BK [11] give conditions on an RKD function which they show suffice for rka-PRP security of an ideal cipher. (An extension of their result is given by AFPW [1].) So one approach that may be suggested for our goal would be to show that $\Phi_{\mathsf{pad},a}$ and $\Phi_{\mathsf{zio},a}$ meet these conditions and then invoke the BK result. However, the BK and AFPW results are in the single-user (su) setting and our definitions are in our, new, multi-user (mu) setting. We could ask if a hybrid argument would reduce mu security to su security, but it is not clear how to do this in an effective way because we allow the RKD functions to be different across users. Instead, we give direct proofs that give explicit bounds.

So far, the parameter $a$ to the RKD functions has been carried around through the proofs, seemingly without any effect. However, since $a$ corresponds to the minimal key length of the adversaries against our higher level primitives, e.g. in the proofs of vkl-PRF security of $\mathsf{HMAC}$ (Thm. 10) and $\mathsf{HMAC}^{\leftrightarrows}$ (Thm. 11), it clearly cannot be arbitrarily small. Now, it finally shows up in the following Proposition statements, as a factor in the rka-PRP advantage. Indeed, the advantage bounds directly show that security relies on $a$ being large enough.

<u>Definitions.</u> We first extend our multi-user definition of rka-PRP security of a block cipher $\mathsf{E}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ to the ideal cipher model by adding oracles E and $\mathrm{E}^{-1}$ to the game in Figure 2, giving the adversary access to the ideal cipher (and its inverse). Additionally, the game replaces the use of $\mathsf{E}$ in oracle Fn by a call to oracle E. For each $K \in \mathcal{K}$, a permutation $\Pi_k$ is drawn at random from $\mathsf{PERM}[\mathcal{X}]$. Oracle E takes as input a pair $(k, x) \in \mathcal{K} \times \mathcal{X}$ and returns $\Pi_k(x)$, while oracle $\mathrm{E}^{-1}$ takes as input a pair $(k, y) \in \mathcal{K} \times \mathcal{X}$ and returns $\Pi_k^{-1}(y)$. The resulting game $\mathbf{G}^{\text{rka-PRP}}_{\mathcal{K},\mathcal{X}}$, parameterized by $\mathcal{K}, \mathcal{X}$ and $d \in \{0,1\}$ is shown in Figure 14. We define the advantage of an adversary $\mathcal{A}$ by

$$\mathbf{Adv}^{\text{rka-PRP}}_{\Phi,\mathcal{K},\mathcal{X}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{rka-PRP-1}}_{\Phi,\mathcal{K},\mathcal{X}}(\mathcal{A})] - \Pr[\mathbf{G}^{\text{rka-PRP-0}}_{\Phi,\mathcal{K},\mathcal{X}}(\mathcal{A})].$$

<u>$\Phi_{\mathsf{pad},a}$-rka-PRP security of ideal ciphers.</u> Recall from Section 5.1 that $\Phi_{\mathsf{pad},a} : \{0,\ldots,b\} \times \{\varepsilon\} \times \{0,1\}^b \to \{0,1\}^b \cup \{\bot\}$ takes input $\ell, \varepsilon, K$ to return $\bot$ if $\ell < a$ and $K[1..\ell] \,\|\, \mathsf{pad}(b{+}\ell)[1..b{-}\ell]$ otherwise. Note that the bound in Eq. (48) in the following Proposition does not depend on the number $\mathbf{Q}^{\mathrm{Fn}}(\mathcal{A})$ of Fn queries of the adversary $\mathcal{A}$.

| Game $\mathbf{G}^{\text{rka-PRP-}d}_{\Phi,\mathcal{K},\mathcal{X}}$: | $\underline{\text{New}(\alpha)}$ |
|---|---|
| $\underline{\text{Init}()}$ | $n \leftarrow n+1$ ; $L_n \leftarrow\$\ \mathcal{K}$; $\alpha_n \leftarrow \alpha$ |
| For each $k \in \mathcal{K}$ do: | For all $L \in \mathcal{K}$ do: |
| $\quad \Pi_k \leftarrow\$\ \mathsf{PERM}[\mathcal{X}]$ | $\quad \Pi_{n,L} \leftarrow\$\ \mathsf{PERM}[\mathcal{X}]$ |
| $\underline{\mathrm{E}(k,x)}$ | $\underline{\text{Fn}(i,\beta,X)}$ |
| Return $\Pi_k(x)$ | Assert: $\Phi_{\alpha_i,\beta}(L_i) \neq \perp$ |
| $\underline{\mathrm{E}^{-1}(k,y)}$ | $L' \leftarrow \Phi_{\alpha_i,\beta}(L_i)$ // Derived key |
| Return $\Pi_k^{-1}(y)$ | If $d=1$ then return $\mathrm{E}(L',X)$ |
| | Else return $\Pi_{i,L'}(X)$ |

Figure 14: Related-key attack PRP security games ($d \in \{0,1\}$) in the ICM for block cipher $\mathsf{E}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ and RKD function $\Phi$.

---

**Proposition 13** ($\Phi_{\mathsf{pad},a}$-rka-PRP security in the ideal cipher model)**.** *Let $a, c < b$ be integers. Let $\mathcal{K} = \{0,1\}^b$ and $\mathcal{X} = \{0,1\}^c$. Let $\mathcal{A}$ be an ICM rka-PRP adversary. Let $q_{ic} = \mathbf{Q}^{\mathrm{E}}(\mathcal{A}) + \mathbf{Q}^{\mathrm{E}^{-1}}(\mathcal{A})$ and let $u = \mathbf{Q}^{\text{New}}(\mathcal{A})$. Then*

$$\mathbf{Adv}^{\text{rka-PRP}}_{\Phi_{\mathsf{pad},a},\mathcal{K},\mathcal{X}}(\mathcal{A}) \leq \frac{u \cdot q_{ic}}{2^a} + \frac{u(u-1)}{2^{a+1}} \ . \tag{48}$$

*Proof of Proposition 13.* Let us call $\Pi$ a family of permutations if $\Pi_k \in \mathsf{PERM}[\mathcal{X}]$ for each $k \in \mathcal{K}$. We visualize $\Pi$ as a table with $|\mathcal{K}|$ rows and $|\mathcal{X}|$ columns, the row $k$ column $x$ entry in the table being $\Pi_k(x)$. Our proof will consist entirely of manipulations of table rows. Manipulations include rows being swapped across different tables, or resampled.

Proceeding, consider the games of Figure 15. It is assumed that a $\text{New}(\ell)$ query satisfies $\ell \geq a$ and that the second input to oracle Fn is always $\varepsilon$, allowing us to drop the "Assert" conditions. Game $\mathrm{G}_0$ rewrites $\mathbf{G}^{\text{rka-PRP-1}}_{\Phi,\mathcal{K},\mathcal{X}}$, moving as many steps as possible from oracles into $\text{Init}()$. We have

$$\Pr[\mathbf{G}^{\text{rka-PRP-1}}_{\Phi,\mathcal{K},\mathcal{X}}(\mathcal{A})] = \Pr[\mathrm{G}_0(\mathcal{A})] \ . \tag{49}$$

The intent now is to move to game $\mathrm{G}_3$. Here, two families of permutations, $\Delta$ and $\Gamma$, are sampled. Queries to E and $\mathrm{E}^{-1}$ are answered via $\Delta$, and queries to Fn via $\Gamma$, with the claim that the adversary cannot distinguish the change, except with small probability. To formalize and prove this claim we consider games $\mathrm{G}_1, \mathrm{G}_2$, where the former includes the boxed code and the latter does not. The game starts out aiming to answer queries to E and $\mathrm{E}^{-1}$ via $\Delta$, and queries to Fn via $\Gamma$. The boxed code however, conflates rows where a difference would be visible to the adversary. Thus in $\mathrm{G}_1$, for each key $k$, either $\Delta_k$ or $\Gamma_k$ answers all queries involving key $k$. Even though the choice between the two is dynamically determined, it is static once made, so that a single random permutation answers all queries for row $k$, just as in $\mathrm{G}_0$. So we have

$$\Pr[\mathrm{G}_0(\mathcal{A})] = \Pr[\mathrm{G}_1(\mathcal{A})]. \tag{50}$$

Games $\mathrm{G}_1, \mathrm{G}_2$ are identical-until-$\mathsf{bad}$ so by the Fundamental Lemma of Game Playing [13] we have

$$\begin{aligned}\Pr[\mathrm{G}_1(\mathcal{A})] &= \Pr[\mathrm{G}_2(\mathcal{A})] + (\Pr[\mathrm{G}_1(\mathcal{A})] - \Pr[\mathrm{G}_2(\mathcal{A})]) & (51)\\ &\leq \Pr[\mathrm{G}_2(\mathcal{A})] + \Pr[\mathrm{G}_2(\mathcal{A}) \text{ sets } \mathsf{bad}] \ . & (52)\end{aligned}$$

The probability that $\mathsf{bad}$ is set at lines 5 or 7 is at most $uq_{ic} \cdot 2^{-a}$, so

$$\Pr[\mathrm{G}_2(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{u \cdot q_{ic}}{2^a} \ . \tag{53}$$

**Game $\mathrm{G}_0$:**

$\underline{\textsc{Init}()}$

For each $k \in \mathcal{K}$ do:

  $\Pi_k \leftarrow\$\ \mathsf{PERM}[\mathcal{X}]$

For $i = 1, \ldots, u$ do:

  $L_i \leftarrow\$\ \mathcal{K}$

$\underline{\mathrm{E}(k, x)}$

Return $\Pi_k(x)$

$\underline{\mathrm{E}^{-1}(k, x)}$

Return $\Pi_k^{-1}(x)$

$\underline{\textsc{New}(\ell)}$

$n \leftarrow n + 1$

$L'_n \leftarrow L_n[1..\ell] \| \mathsf{pad}(b + \ell)[1..b - \ell]$

$\underline{\textsc{Fn}(i, \varepsilon, X)}$

Return $\Pi_{L'_i}(X)$

**Games $\boxed{\mathrm{G}_1}$, $\mathrm{G}_2$:**

$\underline{\textsc{Init}()}$

For each $k \in \mathcal{K}$ do:

  $\Delta_k \leftarrow\$\ \mathsf{PERM}[\mathcal{X}]$ ; $\Gamma_k \leftarrow\$\ \mathsf{PERM}[\mathcal{X}]$

For $i = 1, \ldots, u$ do:

  $L_i \leftarrow\$\ \mathcal{K}$ ; $J_i \leftarrow L_i[1..a]$

$\underline{\mathrm{E}(k, x)}$

If $(\exists\, i : k[1..a] = J_i)$ then $\mathsf{bad} \leftarrow 1$ ; $\boxed{\Delta_k \leftarrow \Gamma_k}$

Return $\Delta_k(x)$

$\underline{\mathrm{E}^{-1}(k, y)}$

If $(\exists\, i : k[1..a] = J_i)$ then $\mathsf{bad} \leftarrow 1$ ; $\boxed{\Delta_k \leftarrow \Gamma_k}$

Return $\Delta_k^{-1}(y)$

$\underline{\textsc{New}(\ell)}$

$n \leftarrow n + 1$ ; $L'_n \leftarrow L_n[1..\ell] \| \mathsf{pad}(b + \ell)[1..b - \ell]$

$\underline{\textsc{Fn}(i, \varepsilon, X)}$

Return $\Gamma_{L'_i}(X)$

**Game $\mathrm{G}_3$:**

$\underline{\textsc{Init}()}$

For each $k \in \mathcal{K}$ do:

  $\Delta_k \leftarrow\$\ \mathsf{PERM}[\mathcal{X}]$ ; $\Gamma_k \leftarrow\$\ \mathsf{PERM}[\mathcal{X}]$

For $i = 1, \ldots, u$ do: $L_i \leftarrow\$\ \mathcal{K}$

$\underline{\mathrm{E}(k, x)}$

Return $\Delta_k(x)$

$\underline{\mathrm{E}^{-1}(k, y)}$

Return $\Delta_k^{-1}(y)$

$\underline{\textsc{New}(\ell)}$

$n \leftarrow n + 1$

$L'_n \leftarrow L_n[1..\ell] \| \mathsf{pad}(b + \ell)[1..b - \ell]$

$\underline{\textsc{Fn}(i, \varepsilon, X)}$

Return $\Gamma_{L'_i}(X)$

**Games $\boxed{\mathrm{G}_4}$, $\mathrm{G}_5$:**

$\underline{\textsc{Init}()}$

For each $k \in \mathcal{K}$ do:

  $\Delta_k \leftarrow\$\ \mathsf{PERM}[\mathcal{X}]$

For $i = 1, \ldots, u$ do: $L_i \leftarrow\$\ \mathcal{K}$

$\underline{\mathrm{E}(k, x)}$

Return $\Delta_k(x)$

$\underline{\mathrm{E}^{-1}(k, y)}$

Return $\Delta_k^{-1}(y)$

$\underline{\textsc{New}(\ell)}$

$n \leftarrow n + 1$; $L'_n \leftarrow L_n[1..\ell] \| \mathsf{pad}(b + \ell)[1..b - \ell]$

$\gamma_n \leftarrow \mathsf{PERM}[\mathcal{X}]$

If $(\exists\, j < n : L'_j = L'_n)$ then

  $\mathsf{bad} \leftarrow 1$ ; $\boxed{\gamma_n \leftarrow \gamma_j}$

$\underline{\textsc{Fn}(i, \varepsilon, X)}$

Return $\gamma_i(X)$

Figure 15: Games for proof of Proposition 13.

---

Game $\mathrm{G}_3$ is obtained by dropping unused code in $\mathrm{G}_2$, so

$$\Pr[\mathrm{G}_2(\mathcal{A})] = \Pr[\mathrm{G}_3(\mathcal{A})]. \tag{54}$$

Now we want to move from $\mathrm{G}_3$ to game $\mathbf{G}^{\text{rka-PRP-0}}_{\Phi,\mathcal{K},\mathcal{X}}$. The difference is that the former uses the same family of permutations $\Gamma$ to answer $\textsc{Fn}$ queries for all users, while the latter uses a different family for each user. The difference is however only detectable if there is a key collision, meaning $L'_i = L'_j$ for some $i \neq j$. To formalize this and conclude we consider games $\mathrm{G}_4, \mathrm{G}_5$, where the former includes the boxed code and the latter does not. Family $\Gamma$ is no longer picked. Instead, New picks a random permutation $\gamma_n$ for the relevant row on line 7. In game $\mathrm{G}_4$, if $L'_n = L'_j$, then the boxed code modifies the line 7 choice to ensure $\gamma_n = \gamma_j$. The result is that $\gamma_i$ plays the role of $\Gamma'_{L_i}$ in $\mathrm{G}_3$, so

$$\Pr[\mathrm{G}_3(\mathcal{A})] = \Pr[\mathrm{G}_4(\mathcal{A})] \; . \tag{55}$$

In $\mathrm{G}_5$ however, the boxed code is absent, so $\gamma_1, \ldots, \gamma_u$ are random, independent permutations. This implies that

$$\Pr[\mathrm{G}_5(\mathcal{A})] = \Pr[\mathbf{G}^{\text{rka-PRP-0}}_{\Phi,\mathcal{K},\mathcal{X}}(\mathcal{A})] \; . \tag{56}$$

Games $\mathrm{G}_4, \mathrm{G}_5$ are identical-until-bad so by the Fundamental Lemma of Game Playing [13] we have

$$\Pr[\mathrm{G}_4(\mathcal{A})] = \Pr[\mathrm{G}_5(\mathcal{A})] + (\Pr[\mathrm{G}_4(\mathcal{A})] - \Pr[\mathrm{G}_5(\mathcal{A})]) \tag{57}$$

$$\leq \Pr[\mathrm{G}_5(\mathcal{A})] + \Pr[\mathrm{G}_5(\mathcal{A}) \text{ sets } \mathsf{bad}] \; . \tag{58}$$

Finally

$$\Pr[\mathrm{G}_5(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{u(u-1)}{2^{a+1}} \; , \tag{59}$$

since each derived key $L'_j$ has a uniformly random prefix of at least $a$ bits, and the probability that $\mathrm{G}_5$ sets bad is bounded from above by the probability that any out of the at most $u$ many such prefixes collide. Putting the above together concludes the proof. □

$\Phi_{\mathsf{zio},a}$-RKA-PRP SECURITY OF IDEAL CIPHERS. Let $\mathtt{ipad}, \mathtt{opad} \in \{0,1\}^b$ be distinct strings. Recall from Section 6.2 that $\Phi_{\mathsf{zio},a} : \{0,\ldots,b\} \times \{\mathtt{ipad}, \mathtt{opad}\} \times \{0,1\}^b \to \{0,1\}^b \cup \{\perp\}$ takes input $\ell, \mathsf{io}, K$ to return $\perp$ if $\ell < a$ and $(K[1..\ell] \,\|\, 0^{b-\ell}) \oplus \mathsf{io}$ otherwise.

**Proposition 14** ($\Phi_{\mathsf{zio},a}$-rka-PRP security in the ideal cipher model)**.** *Let $a, c < b$ be integers. Let $\mathcal{K} = \{0,1\}^b$ and $\mathcal{X} = \{0,1\}^c$. Let $\mathcal{A}$ be an ICM rka-PRP adversary. Let $q_{ic} = \mathbf{Q}^{\mathrm{E}}(\mathcal{A}) + \mathbf{Q}^{\mathrm{E}^{-1}}(\mathcal{A})$ and let $u = \mathbf{Q}^{\mathrm{New}}(\mathcal{A})$. Then*

$$\mathbf{Adv}^{\text{rka-PRP}}_{\Phi_{\mathsf{zio},a},\mathcal{K},\mathcal{X}}(\mathcal{A}) \leq \frac{2u \cdot q_{ic}}{2^a} + \frac{4u(u-1)}{2^{a+1}} \; . \tag{60}$$

The proof is similar to that of Proposition 13 and is omitted. Briefly the idea is that for each $i$ there are now two related keys $L'_i$ and $L''_i$ and the rest is the same. This leads to additional factors of 2 in the bound.

## References


[1] Martin R. Albrecht, Pooya Farshim, Kenneth G. Paterson, and Gaven J. Watson. On cipher-dependent related-key attacks in the ideal-cipher model. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 128–145. Springer, Heidelberg, February 2011.

[2] Yawning Angel, Benjamin Dowling, Andreas Hülsing, Peter Schwabe, and Florian Weber. Post quantum noise. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 97–109. ACM Press, November 2022.

[3] Nimrod Aviram, Benjamin Dowling, Ilan Komargodski, Kenneth G. Paterson, Eyal Ronen, and Eylon Yogev. Practical (post-quantum) key combiners from one-wayness and applications to TLS. Cryptology ePrint Archive, Report 2022/065, 2022. https://eprint.iacr.org/2022/065.

[4] Manuel Barbosa and Pooya Farshim. The related-key analysis of Feistel constructions. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 265–284. Springer, Heidelberg, March 2015.

[5] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Heidelberg, August 2006.

[6] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision resistance. *Journal of Cryptology*, 28(4):844–878, October 2015.

[7] Mihir Bellare, Daniel J. Bernstein, and Stefano Tessaro. Hash-function based PRFs: AMAC and its multi-user security. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 566–595. Springer, Heidelberg, May 2016.

[8] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15. Springer, Heidelberg, August 1996.

[9] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *37th FOCS*, pages 514–523. IEEE Computer Society Press, October 1996.

[10] Mihir Bellare and Viet Tung Hoang. Efficient schemes for committing authenticated encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 845–875. Springer, Heidelberg, May / June 2022.

[11] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, May 2003.

[12] Mihir Bellare and Anna Lysyanskaya. Symmetric and dual PRFs from standard assumptions: A generic validation of an HMAC assumption. Cryptology ePrint Archive, Report 2015/1198, 2015. https://eprint.iacr.org/2015/1198.

[13] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[14] Eli Biham. New types of cryptanalytic attacks using related keys (extended abstract). In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 398–409. Springer, Heidelberg, May 1994.

[15] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key impossible differential attacks on 8-round AES-192. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 21–33. Springer, Heidelberg, February 2006.

[16] Eli Biham, Orr Dunkelman, and Nathan Keller. A simple related-key attack on the full SHACAL-1. In Masayuki Abe, editor, *CT-RSA 2007*, volume 4377 of *LNCS*, pages 20–30. Springer, Heidelberg, February 2007.

[17] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 206–226. Springer, Heidelberg, 2019.

[18] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 1–18. Springer, Heidelberg, December 2009.

[19] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer, Heidelberg, August 2009.

[20] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[21] Priyanka Bose, Viet Tung Hoang, and Stefano Tessaro. Revisiting AES-GCM-SIV: Multi-user security, faster key derivation, and better bounds. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 468–499. Springer, Heidelberg, April / May 2018.

[22] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: Relations, instantiations, and impossibility results. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 651–681. Springer, Heidelberg, August 2017.

[23] Chris Brzuska, Eric Cornelissen, and Konrad Kohbrok. Security analysis of the MLS key derivation. In *2022 IEEE Symposium on Security and Privacy*, pages 2535–2553. IEEE Computer Society Press, May 2022.

[24] Chris Brzuska, Antoine Delignat-Lavaud, Christoph Egger, Cédric Fournet, Konrad Kohbrok, and Markulf Kohlweiss. Key-schedule security for the TLS 1.3 standard. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 621–650. Springer, Heidelberg, December 2022.

[25] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, August 1990.

[26] Yevgeniy Dodis, Thomas Ristenpart, John P. Steinberger, and Stefano Tessaro. To hash or not to hash again? (In)differentiability results for $H^2$ and HMAC. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 348–366. Springer, Heidelberg, August 2012.

[27] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol. *Journal of Cryptology*, 34(4):37, October 2021.

[28] Orr Dunkelman, Nathan Keller, and Jongsung Kim. Related-key rectangle attack on the full SHACAL-1. In Eli Biham and Amr M. Youssef, editors, *SAC 2006*, volume 4356 of *LNCS*, pages 28–44. Springer, Heidelberg, August 2007.

[29] Stephen Farrell. [Cfrg] erratum for hmac what do we think... IRTF Crypto Forum Research Group mailing list. https://mailarchive.ietf.org/arch/msg/cfrg/hxj9UM2LdBy2eipAJX2idjQuxhk/, February 2017.

[30] Peter Gaži, Krzysztof Pietrzak, and Michal Rybár. The exact PRF-security of NMAC and HMAC. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 113–130. Springer, Heidelberg, August 2014.

[31] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[32] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[33] Viet Tung Hoang, Stefano Tessaro, and Aishwarya Thiruvengadam. The multi-user security of GCM, revisited: Tight bounds for nonce randomization. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1429–1440. ACM Press, October 2018.

[34] Seokhie Hong, Jongsung Kim, Sangjin Lee, and Bart Preneel. Related-key rectangle attacks on reduced versions of SHACAL-1 and AES-192. In Henri Gilbert and Helena Handschuh, editors, *FSE 2005*, volume 3557 of *LNCS*, pages 368–383. Springer, Heidelberg, February 2005.

[35] Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R. Zimmermann. Post-quantum WireGuard. In *2021 IEEE Symposium on Security and Privacy*, pages 304–321. IEEE Computer Society Press, May 2021.

[36] John Kelsey, Bruce Schneier, and David Wagner. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *ICICS 97*, volume 1334 of *LNCS*, pages 233–246. Springer, Heidelberg, November 1997.

[37] Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-key rectangle attacks on reduced AES-192 and AES-256. In Alex Biryukov, editor, *FSE 2007*, volume 4593 of *LNCS*, pages 225–241. Springer, Heidelberg, March 2007.

[38] Lars R. Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 196–208. Springer, Heidelberg, December 1993.

[39] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational), February 1997. Updated by RFC 6151.

[40] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648. Springer, Heidelberg, August 2010.

[41] Atul Luykx, Bart Mennink, and Kenneth G. Paterson. Analyzing multi-key security degradation. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 575–605. Springer, Heidelberg, December 2017.

[42] Ueli M. Maurer. Indistinguishability of random systems. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 110–132. Springer, Heidelberg, April / May 2002.

[43] Erdem Memisyazici. RFC Erratum on RFC 2104, "HMAC: Keyed-Hashing for Message Authentication". RFC Errata, Errata ID: 4809. https://www.rfc-editor.org/errata_search.php?rfc=2104&eid=4809, September 2016.

[44] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer, Heidelberg, August 1990.

[45] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 428–446. Springer, Heidelberg, August 1990.

[46] National Institute of Standards and Technology. The keyed-hash message authentication code (HMAC). Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 198-1, U.S. Department of Commerce, Washington, D.C., 2008.

[47] National Institute of Standards and Technology. Secure hash standard (SHS). Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 180-4, U.S. Department of Commerce, Washington, D.C., 2015.

[48] Raphael Chung-Wei Phan. Related-key attacks on triple-DES and DESX variants. In Tatsuaki Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 15–24. Springer, Heidelberg, February 2004.

[49] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Proposed Standard), August 2018.

[50] Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 211–228. Springer, Heidelberg, September 2006.

[51] John M. Schanck and Douglas Stebila. A Transport Layer Security (TLS) Extension For Establishing An Additional Shared Secret – draft-schanck-tls-additional-keyshare-00. https://datatracker.ietf.org/doc/html/draft-schanck-tls-additional-keyshare-00, April 2017.

[52] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1461–1480. ACM Press, November 2020.

[53] Douglas Stebila, Scott Fluhrer, and Shay Gueron. Hybrid key exchange in TLS 1.3 – draft-ietf-tls-hybrid-design-05. https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-05, August 2022.

[54] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982.

[55] Wentao Zhang, Wenling Wu, Lei Zhang, and Dengguo Feng. Improved related-key impossible differential attacks on reduced-round AES-192. In Eli Biham and Amr M. Youssef, editors, *SAC 2006*, volume 4356 of *LNCS*, pages 15–27. Springer, Heidelberg, August 2007.

# A Dual-PRF Assumptions on $\mathsf{HMAC}$ in Prior Work

In many analyses of real-world protocols, $\mathsf{HMAC}$ is assumed to be a secure dual (or swap) PRF, including works on TLS 1.3 [27, 24], KEMTLS [52], hybrid key-exchange designs [17, 53], post-quantum versions of WireGuard [35] and Noise [2], and Message Layer Security (MLS) [23]. Here, we give an account of where the assumption shows up, and when it is supported by our results.

Generally speaking, our results support the dual-PRF assumption on $\mathsf{HMAC}$ when used with uniformly random keys in the aforementioned analyses (as keys in these real-world protocols have fixed length and, hence, are feasible). Note, however, that since our results only cover uniformly random PRF keys, they do not say anything about using $\mathsf{HMAC}$ to extract uniform keys from non-uniform entropy inputs (following, for example, the $\mathsf{HKDF}$ paradigm [40]). In this setting, dual-PRF(-like) assumptions have also been made in the cited, and other, works.

**TLS 1.3 [27].** In the proofs of multi-stage key exchange security of TLS 1.3's full handshake (Theorem 6.2, Game 4), and pre-shared key handshake (Theorem 6.4, Games A.4 and B.3) in [27], the handshake's early secret ES $\leftarrow \mathsf{HKDF.Extract}(0, \mathrm{PSK})$ is replaced with a random value, given that PSK is uniformly random, assuming "the dual PRF security of $\mathsf{HKDF.Extract}$." Since 0 here denotes a string of zero bytes of (fixed) length of the hash function's output, treating $\mathsf{HKDF.Extract}$ as $\mathsf{HMAC}^{\leftrightarrows}$ keyed with PSK, our result from Theorem 11 applies.

The analysis in [27] also assumes a dual version of the PRF-ODH assumption [22] on $\mathsf{HKDF.Extract}$; our results do not cover this assumption.

**TLS 1.3 [24].** The main theorem for the key schedule security of TLS 1.3 in [24] "assumes [...] dual pseudorandomness $\mathtt{Gpr}^{\mathsf{f}\text{-}alg,b}$ with $\mathsf{f}\text{-}alg = \mathsf{xtr}^{\dagger}\text{-}alg$" [24, Section 3.2], similarly to [27] when arguing about the early secret in the TLS 1.3 handshake. Here, $\mathsf{xtr}^{\dagger}\text{-}alg$ denotes $\mathsf{HMAC}^{\leftrightarrows}$ in the agile sense of $alg$ determining the underlying hash function $\mathsf{H}$.

**KEMTLS [52].** In [52], the proof of Theorem 4.1 showing multi-stage key exchange security of KEMTLS involves a "sequence of game hops which, one-by-one, replace derived secrets and stage keys with random values, under the PRF-security or dual-PRF-security [5, 6] of HKDF (dual-PRF-security arises since the TLS 1.3 and KEMTLS key schedules sometime use secrets in the "salt" argument of HKDF.Extract, rather than the "input keying material argument")." Each of the dual-PRF steps applies to an $\mathsf{HKDF.Extract} = \mathsf{HMAC}$ computation with two fixed-length keys as inputs (out of which the second is leveraged as the random key input), our result from Theorem 11 applies.

**Hybrid KEMs [17].** Studying hybrid key exchange, [17] introduces several KEM combiners, including $\mathsf{dualPRF}$ which "relies on the existence of dual pseudorandom functions [8, 5, 12] which provide security if either the key material or the label carries entropy", and $\mathsf{N}$, "a nested variant of the dual-PRF combiner inspired by the key derivation procedure in TLS 1.3 and the proposal how to augment it for hybrid schemes in [51]", suggesting "The HKDF key derivation function is, for example, based on this dual principle." [17, Section 3]. In their proof of security of the $\mathsf{dualPRF}$ and $\mathsf{N}$ combiners, dual-PRF security is assumed for a generic function $\mathsf{dPRF}$ which takes two keys as input. Reasonably assuming that KEM key outputs have fixed lengths, these keys qualify as feasible for our results on $\mathsf{HMAC}$ and $\mathsf{HMAC}[\mathcal{S}]^{\leftrightarrows}$ from Theorems 10 and 11; however [17] also consider (partially) quantum adversaries.

**Post-quantum WireGuard [35].** As component of a post-quantum proposal for the WireGuard VPN protocol, a "dual-PRF appears in the form of a key-derivation function $\mathsf{KDF}(X, Y) = Z$." [35, Section II.D] In the key exchange security proof, both PRF and swap-PRF assumptions are made on $\mathsf{KDF}$ (which in WireGuard is based on $\mathsf{HMAC}$) when used with fixed-length inputs, so our results from Theorems 10 and 11 apply.

**Post-quantum Noise [2].** In the analysis of a post-quantum version of the Noise protocol, [2] introduces a "Noise Hash Object" as an abstraction, whose security relies on $\mathsf{HMAC}$ being a dual-PRF in Theorem 1. In their instantiation, $\mathsf{HMAC}$ in the swapped case is using an empty string or a prior output as "state" [2, Algorithm 2]; applying our results from Theorem 11 would require encoding both in same-length bitstrings.

**MLS [23].** Analyzing the key derivation in the Message Layer Security (MLS) protocol, [23] assumes "that the Extract function in Krawczyk's HKDF design [40] is a dual pseudorandom function and thus, we assume that HKDF is a dual KDF." HKDF is modeled as taking two key inputs of fixed length [23, Section V], making it amenable to our results from Theorems 10 and 11.

| Games $G_0$, $G_1$, $G_2$: | Adversary $\mathcal{A}_{\mathsf{f}}()$ |
|---|---|
| $\underline{\text{New}()}$ | 1 $b \leftarrow\$ \mathcal{A}_{\mathsf{2CSC}}^{\text{New},\text{Fn}^*}$ |
| 1 $n \leftarrow n+1$; $J_n \leftarrow\$ \mathcal{K}$ | 2 Return $b$ |
| $\underline{\text{Fn}(i,(A,S))}$ | $\underline{\text{Fn}^*(i,(A,S))}$ |
| 2 If $\mathrm{T}[i,(A,S)] = \perp$ then: | 3 $L \leftarrow \text{Fn}(i,A)$ |
| 3 If $\mathrm{T}_{\mathsf{f}}[i,A] = \perp$ then: | 4 Return $\mathsf{h}^*(L,S)$ |
| 4 $\mathrm{T}_{\mathsf{f}}[i,A] \leftarrow \mathsf{f}(J_i,A)$ // $G_0$ | |
| 5 $\mathrm{T}_{\mathsf{f}}[i,A] \leftarrow\$ \{0,1\}^c$ // $G_1$ | |
| 6 $\mathrm{T}[i,(A,S)] \leftarrow \mathsf{h}^*(\mathrm{T}_{\mathsf{f}}[i,A],S)$ // $G_0, G_1$ | |
| 7 $\mathrm{T}[i,(A,S)] \leftarrow\$ \{0,1\}^c$ // $G_2$ | |
| 8 Return $\mathrm{T}[i,(A,S)]$ | |

Figure 16: Games $G_0$-$G_2$ and adversary $\mathcal{A}_{\mathsf{f}}$ for the proof of Lemma 1.

# B Proofs for the Strong PRF Security of the Cascade

## B.1 Proof of Lemma 1

We use a sequence of games $G_0$–$G_2$ and a set of hybrid games $H_0$–$H_n$, as shown in Figures 16 and 17. All are executed with $\mathcal{A}_{\mathsf{2CSC}}$, and hence provide the oracles named in game $\mathbf{G}_{\mathsf{2CSC}}^{\text{PRF-}d}$.

▷ *Replacing* $\mathsf{f}$ *with a random function.* We begin with game $G_0$ being the "real" PRF game $\mathbf{G}_{\mathsf{2CSC}}^{\text{PRF-1}}(\mathcal{A}_{\mathsf{2CSC}})$. The first game hop, to game $G_1$, replaces the evaluation of $\mathsf{f}$ in line 4 of $G_0$ by (consistent) random sampling in line 5. By standard equation rewriting

$$\begin{aligned}\Pr[\mathbf{G}_{\mathsf{2CSC}}^{\text{PRF-1}}(\mathcal{A}_{\mathsf{2CSC}})] &= \Pr[G_0(\mathcal{A}_{\mathsf{2CSC}})] \\ &= \Pr[G_1(\mathcal{A}_{\mathsf{2CSC}})] + (\Pr[G_0(\mathcal{A}_{\mathsf{2CSC}})] - \Pr[G_1(\mathcal{A}_{\mathsf{2CSC}})]).\end{aligned} \tag{61}$$

We build an adversary $\mathcal{A}_{\mathsf{f}}$, such that

$$\Pr[G_0(\mathcal{A}_{\mathsf{2CSC}})] - \Pr[G_1(\mathcal{A}_{\mathsf{2CSC}})] \le \mathbf{Adv}_{\mathsf{f}}^{\text{PRF}}(\mathcal{A}_{\mathsf{f}}). \tag{62}$$

Adversary $\mathcal{A}_{\mathsf{f}}$ simulates $G_0$ for $\mathcal{A}_{\mathsf{2CSC}}$, with two changes: when adversary $\mathcal{A}_{\mathsf{2CSC}}$ makes its calls to oracle New, adversary $\mathcal{A}_{\mathsf{f}}$ also calls its New oracle. Then, when adversary $\mathcal{A}_{\mathsf{2CSC}}$ makes a query $\text{Fn}(i,(A,S))$, $\mathcal{A}_{\mathsf{f}}$ issues query $L \leftarrow \text{Fn}(i,A)$ to its own oracle and returns $\mathsf{h}^*(L,S)$ to $\mathcal{A}_{\mathsf{2CSC}}$. That is, $\mathcal{A}_{\mathsf{f}}$ simulates the function oracle by getting the key $L$ for the cascade from the Fn oracle in game $\mathbf{G}_{\mathsf{f}}^{\text{PRF-}d}$ and then applies the cascade to the input $S$. Hence, $\mathcal{A}_{\mathsf{f}}$ makes one New query and one Fn query respectively for each of the $\mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{2CSC}})$ New and $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{2CSC}})$ Fn queries made by $\mathcal{A}_{\mathsf{2CSC}}$.

When adversary $\mathcal{A}_{\mathsf{2CSC}}$ halts and returns, adversary $\mathcal{A}_{\mathsf{f}}$ also halts and returns the same output. With this strategy, adversary $\mathcal{A}_{\mathsf{f}}$ soundly simulates $G_{(1-d)}$ when playing game $\mathbf{G}_{\mathsf{f}}^{\text{PRF-}d}$, for $d \in \{0,1\}$, yielding the bound in Equation (62).

▷ *Replacing* $\mathsf{h}^*$ *with a random function.*

The second game hop, to game $G_2$, replaces the invocation of the cascade on line 6 by random sampling in line 7.

Again by standard equation rewriting

$$\Pr[G_1(\mathcal{A}_{\mathsf{2CSC}})] = \Pr[G_2(\mathcal{A}_{\mathsf{2CSC}})] + (\Pr[G_1(\mathcal{A}_{\mathsf{2CSC}})] - \Pr[G_2(\mathcal{A}_{\mathsf{2CSC}})]).$$

Using a hybrid argument, we build an adversary $\mathcal{A}_{\mathsf{h}}$ against the PRF security of the compression function $\mathsf{h}$, such that

$$\Pr[G_1(\mathcal{A}_{\mathsf{2CSC}})] - \Pr[G_2(\mathcal{A}_{\mathsf{2CSC}})] \le n \cdot \mathbf{Adv}_{\mathsf{h}}^{\text{PRF}}(\mathcal{A}_{\mathsf{h}}), \tag{63}$$

Game $\mathrm{H}_s$ // $0 \le s \le n$:

<u>$\textsc{New}()$</u>

$\nu \leftarrow \nu + 1$

<u>$\textsc{Fn}(i, (A, S))$</u>

If $\mathrm{T}[i, (A, \mathbf{S})] \neq \bot$ then:
 Return $\mathrm{T}[i, (A, \mathbf{S})]$
$\mathbf{S} \xleftarrow{b} S$; $N \leftarrow |\mathbf{S}|$
If $N \le s$ then:
 $\mathrm{T}[i, (A, \mathbf{S})] \leftarrow\!\!\$\ \{0,1\}^c$
Else:
 If $\mathrm{T}_L[i, (A, \mathbf{S}[1..s])] = \bot$ then:
  $\mathrm{T}_L[i, (A, \mathbf{S}[1..s])] \leftarrow\!\!\$\ \{0,1\}^c$
 $C[s] \leftarrow \mathrm{T}_L[i, (A, \mathbf{S}[1..s])$
 For $j = s+1$ to $N$ do:
  $C[j] \leftarrow \mathsf{h}(C[j-1], \mathbf{S}[j])$
 $\mathrm{T}[i, (A, \mathbf{S})] \leftarrow C[N]$
Return $\mathrm{T}[i, (A, \mathbf{S})]$

<u>Adversary $\mathcal{A}_\mathsf{h}()$</u>

$\omega \leftarrow\!\!\$\ \{1, \ldots, n\}$
$b \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{2CSC}}^{\textsc{New}^*, \textsc{Fn}^*}()$
Return $b$

<u>$\textsc{New}^*()$</u>

$\nu \leftarrow \nu + 1$

<u>$\textsc{Fn}^*(i, (A, S))$</u>

If $\mathrm{T}[i, (A, \mathbf{S})] \neq \bot$ then:
 Return $\mathrm{T}[i, (A, \mathbf{S})]$
$\mathbf{S} \xleftarrow{b} S$; $N \leftarrow |\mathbf{S}|$
If $N < \omega$ then:
 $\mathrm{T}[i, (A, \mathbf{S})] \leftarrow\!\!\$\ \{0,1\}^c$
Else: (if $N \ge \omega$)
 // If $i, (A, \mathbf{S}[1..\omega-1])$ is new, init new key:
 If $\mathrm{T}^*_L[i, (A, \mathbf{S}[1..\omega-1])] = \bot$
  $ctr \leftarrow ctr + 1$; $\textsc{New}()$
  $\mathrm{T}^*_L[i, (A, \mathbf{S}[1..\omega-1])] \leftarrow ctr$
 // Get key index:
 $u \leftarrow \mathrm{T}^*_L[i, (A, \mathbf{S}[1..\omega-1])]$
 $C[\omega] \leftarrow \textsc{Fn}(u, \mathbf{S}[\omega])$
 For $j = \omega+1$ to $N$ do:
  $C[j] \leftarrow \mathsf{h}(C[j-1], \mathbf{S}[j])$
 $\mathrm{T}[i, (A, \mathbf{S})] \leftarrow C[N]$
Return $\mathrm{T}[i, (A, \mathbf{S})]$

Figure 17: Hybrid games $\mathrm{H}_0$-$\mathrm{H}_n$ and adversary $\mathcal{A}_\mathsf{h}$ for the proof of Lemma 1.

where $n$ is the maximum block length of the $S$ component in any $\textsc{Fn}(i, (A, S))$ query by $\mathcal{A}_{\mathsf{2CSC}}$. To this end, we construct $n+1$ hybrid games $\mathrm{H}_0 - \mathrm{H}_n$ as shown in Figure 17. In game $\mathrm{H}_s$, the first $s$ invocations of $\mathsf{h}$ in the cascade are replaced by random sampling, the remainder are computed as before. Hence, game $\mathrm{H}_0$ is equivalent to $\mathrm{G}_1$ and $\mathrm{H}_n$ to $\mathrm{G}_2$, giving

$$\Pr[\mathrm{G}_1(\mathcal{A}_{\mathsf{2CSC}})] - \Pr[\mathrm{G}_2(\mathcal{A}_{\mathsf{2CSC}})] = \Pr[\mathrm{H}_0(\mathcal{A}_{\mathsf{2CSC}})] - \Pr[\mathrm{H}_n(\mathcal{A}_{\mathsf{2CSC}})]. \tag{64}$$

Adversary $\mathcal{A}_\mathsf{h}$ begins by sampling $\omega \leftarrow\!\!\$\ \{1, \ldots, n\}$ and then acts as the challenger in game $\mathrm{H}_\omega$, with the following differences. When adversary $\mathcal{A}_{\mathsf{2CSC}}$ makes a new query $\textsc{Fn}(i, (A, S))$, and the number of blocks $N$ of $S$ is less than $\omega$, then adversary $\mathcal{A}_\mathsf{h}$ samples a (consistent) random string in $\{0,1\}^c$, stores the result in a table $\mathrm{T}[i, (A, S)]$ for consistency and returns the result to $\mathcal{A}_{\mathsf{2CSC}}$.

If instead the number of blocks $N$ is at least $\omega$, then adversary $\mathcal{A}_\mathsf{h}$ checks if the prefix $(A, \mathbf{S}[1..\omega-1])$ is per-user new. If so, it calls oracle $\textsc{New}$ to initialize a new key, and stores the index of the key in a table $\mathrm{T}^*_L[i, (A, \mathbf{S}[1..\omega-1])]$. Otherwise it fetches the key index $u$ from the table. Adversary $\mathcal{A}_\mathsf{h}$ then issues query $C[\omega] \leftarrow \textsc{Fn}(u, \mathbf{S}[\omega])$. If $N > \omega$, it then applies the cascade to the remaining blocks of $S$. It stores $\mathsf{h}^*(C[\omega], \mathbf{S}[\omega+1..N])$ in table $\mathrm{T}[i, (A, S)]$ and returns the result to $\mathcal{A}_{\mathsf{2CSC}}$. When adversary $\mathcal{A}_{\mathsf{2CSC}}$ halts and returns, adversary $\mathcal{A}_\mathsf{h}$ also halts and returns the same output.

This way, adversary $\mathcal{A}_\mathsf{h}$ simulates $\mathrm{H}_{\omega-d}$ when playing game $\mathbf{G}_\mathsf{h}^{\mathrm{PRF}\text{-}d}$, for $d \in \{0,1\}$, making at most $\mathbf{Q}^{\text{FN}}(\mathcal{A}_{\mathsf{2CSC}})$ queries to oracle NEW and $\mathbf{Q}^{\text{FN}}(\mathcal{A}_{\mathsf{2CSC}})$ queries to FN. Note that the assumption that the FN queries of $\mathcal{A}_{\mathsf{2CSC}}$ are prefix-free guarantees the soundness of the simulation, as it will never be the case that $\mathcal{A}_{\mathsf{2CSC}}$ sees both the response to a query $\text{FN}(i, (A, S))$ with $|\mathbf{S}| = \omega - 1$ (which will be a random string sampled by $\mathcal{A}_{\mathsf{h}}$) and the response to $\text{FN}(i, (A, S \,\|\, S'))$ for some non-empty $S'$ (which will be computed by $\mathcal{A}_{\mathsf{h}}$ using its FN oracle on a key index corresponding to $(A, \mathbf{S}[1..\omega - 1])$) for user $i$. That is, it will not be the case that $\mathcal{A}_{\mathsf{h}}$ is forced to both pick a random value at level $\omega - 1$, and call oracle NEW to initialize a key which should correspond to this value (something which $\mathcal{A}_{\mathsf{h}}$ cannot guarantee).

By the law of total probability and a telescoping sum, we have

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) &= \Pr\left[\,\mathbf{G}^{\mathrm{PRF}\text{-}1}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}})\,\right] - \Pr\left[\,\mathbf{G}^{\mathrm{PRF}\text{-}0}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}})\,\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\Pr\left[\,\mathrm{H}_{i-1}(\mathcal{A}_{\mathsf{2CSC}}) \,\middle|\, \omega = i\,\right] - \Pr\left[\,\mathrm{H}_{i}(\mathcal{A}_{\mathsf{2CSC}}) \,\middle|\, \omega = i\,\right]\right) \\
&= \frac{1}{n}\cdot\left(\Pr[\mathrm{H}_0(\mathcal{A}_{\mathsf{2CSC}})] - \Pr[\mathrm{H}_n(\mathcal{A}_{\mathsf{2CSC}})]\right).
\end{aligned}$$

Combined with Equation (64), this gives the bound in Equation (63). □

## B.2 Proof of Lemma 2

The proof uses the insight from BBT [7] that the multi-user security of the cascade corresponds to the single-user prf-security of the 2-tier cascade when the first tier is a random function. To explain: if we let the first tier of the 2-tier cascade be the family $\mathsf{f}$ of *all functions* with domain $\mathcal{X} = \{1, \ldots, u\}$ and range $\{0,1\}^c$, where $u \geq \mathbf{Q}^{\text{NEW}}(\mathcal{A}_{\mathsf{h}^*})$ is an upper bound on the number of users initialized by adversary $\mathcal{A}_{\mathsf{h}^*}$, then choosing a key for $\mathsf{f}$ corresponds to choosing a random function $\mathsf{kf} : \mathcal{X} \to \{0,1\}^c$. That is, the key space $\mathcal{K}$ of $\mathsf{f}$ is the set of all functions from $\mathcal{X}$ to $\{0,1\}^c$, and $\mathsf{f}(\mathsf{kf}, i) = \mathsf{kf}(i)$.

Sampling a new key $L \in \{0,1\}^c$ for the cascade then corresponds to evaluating $\mathsf{kf}$ at a new index. Hence we can instantiate the cascade as

$$\mathsf{h}^*(\mathsf{kf}(i), S) = \mathsf{2CSC}(\mathsf{kf}, (i, S)), \tag{65}$$

and view each new invocation of $\mathsf{h}^*$ under a new key as an evaluation of $\mathsf{2CSC}$ on the single-user instance keyed by $\mathsf{kf}$. This allows us to apply the result from lemma 1 to show the prf-security of $\mathsf{h}^*$.

To proceed, we construct an intermediate adversary $\mathcal{A}_{\mathsf{2CSC}}$ against the single-user prf-security of $\mathsf{2CSC}$ which works as follows.

Adversary $\mathcal{A}_{\mathsf{2CSC}}^{\text{NEW},\text{FN}}()$

NEW()

$b \leftarrow\!\$\; \mathcal{A}_{\mathsf{h}^*}^{\text{NEW}^*,\text{FN}^*}()$

Return $b$

$\text{NEW}^*()$

$\text{FN}^*(i, S)$

$X \leftarrow \text{FN}(1, (i, S))$

Return $X$

Adversary $\mathcal{A}_{\mathsf{2CSC}}$ begins by calling oracle NEW to initialize the (single) key for $\mathsf{2CSC}$. Since $\mathsf{2CSC}$ is instantiated with $\mathsf{f}$ and $\mathsf{h}$, as described above, this samples a random function $\mathsf{kf} : \{1, \ldots, u\} \to \{0,1\}^c$. Then, adversary $\mathcal{A}_{\mathsf{2CSC}}$ runs $\mathcal{A}_{\mathsf{h}^*}$, simulating access to oracles NEW and FN via $\text{NEW}^*$ and $\text{FN}^*$, respectively. When $\mathcal{A}_{\mathsf{h}^*}$ makes one of its $\mathbf{Q}^{\text{NEW}}(\mathcal{A}_{\mathsf{h}^*})$ many queries to oracle $\text{NEW}^*$, adversary $\mathcal{A}_{\mathsf{2CSC}}$ does nothing. When $\mathcal{A}_{\mathsf{h}^*}$ queries $\text{FN}^*(i, S)$, adversary $\mathcal{A}_{\mathsf{2CSC}}$ responds by issuing the query $\text{FN}(1, (i, S))$ to its own oracle under the single key with index 1, and forwards the response to $\mathcal{A}_{\mathsf{h}^*}$. When adversary $\mathcal{A}_{\mathsf{h}^*}$ halts and returns a guess for the hidden bit, $\mathcal{A}_{\mathsf{2CSC}}$ does the same.

This way, adversary $\mathcal{A}_{\mathsf{2CSC}}$ perfectly simulates game $\mathbf{G}_{\mathsf{h}^*}^{\text{PRF-}d}$ when playing game $\mathbf{G}_{\mathsf{2CSC}}^{\text{PRF-}d}$ itself, and

$$\mathbf{Adv}_{\mathsf{h}^*}^{\text{PRF}}(\mathcal{A}_{\mathsf{h}^*}) \leq \mathbf{Adv}_{\mathsf{2CSC}}^{\text{PRF}}(\mathcal{A}_{\mathsf{2CSC}}). \tag{66}$$

Combining Equation (66) with Equation (1) from Lemma 1 now yields

$$\mathbf{Adv}_{\mathsf{h}^*}^{\text{PRF}}(\mathcal{A}_{\mathsf{h}^*}) \leq \mathbf{Adv}_{\mathsf{f}}^{\text{PRF}}(\mathcal{A}_{\mathsf{f}}) + n \cdot \mathbf{Adv}_{\mathsf{h}}^{\text{PRF}}(\mathcal{A}_{\mathsf{h}}), \tag{67}$$

for adversaries $\mathcal{A}_{\mathsf{f}}$ and $\mathcal{A}_{\mathsf{h}}$ as specified in the proof of Lemma 1. (Note that the assumption that Fn oracle queries from $\mathcal{A}_{\mathsf{h}^*}$ are prefix-free guarantees that adversary $\mathcal{A}_{\mathsf{2CSC}}$ only makes prefix-free queries, as needed for Lemma 1.) However, since $\mathsf{f}$ is the family of *all* functions mapping $\{1,\ldots,u\}$ to keys for $\mathsf{h}$ in $\{0,1\}^c$, the advantage of *any* adversary against the prf-security of $\mathsf{f}$ is 0. (Any instance of $\mathsf{f}$ is a truly random function, and hence indistinguishable from the lazily sampled random function in $\mathbf{G}_{\mathsf{f}}^{\text{PRF-0}}$ in Figure 1.) Hence Equation (67) simplifies to Equation (2).

Adversary $\mathcal{A}_{\mathsf{h}}$ works just like in Lemma 1. That is, it makes at most $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^*})$ queries to New and $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^*})$ queries to oracle Fn. Note that this means that the maximum number of users $u$ does not show up anywhere, neither in the advantage bound nor in the resources used by adversary $\mathcal{A}_{\mathsf{h}}$. How can this be? The explanation is that $\mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{h}^*})$ is subsumed by $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^*})$. Indeed, in each layer of the cascade except for the first, the compression function $\mathsf{h}$ is invoked with at most $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^*})$ different keys, which is reflected in the proof by the (at most) $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^*})$ many queries to oracle New by $\mathcal{A}_{\mathsf{h}}$.

In the first layer, $\mathsf{h}$ is invoked with at most $u \geq \mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{h}^*})$ distinct keys, but since each invocation is prompted by one of the $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^*})$ many queries to oracle Fn, the actual number of keys used is at most $\mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^*})$. That is, a query to oracle New to initialize a key index which is subsequently never used can be safely ignored, and is in fact ignored in the reduction (since adversary $\mathcal{A}_{\mathsf{2CSC}}$ does nothing in response to a New oracle query). Put differently: we could write a reduction in which $\mathcal{A}_{\mathsf{h}}$ makes at most $\max(\mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{h}^*}), \mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^*}))$ queries to oracle New, but w.l.o.g. we may also assume that adversary $\mathcal{A}_{\mathsf{h}^*}$ does not make any "unnecessary" queries to oracle New, so that $\mathbf{Q}^{\text{New}}(\mathcal{A}_{\mathsf{h}^*}) \leq \mathbf{Q}^{\text{Fn}}(\mathcal{A}_{\mathsf{h}^*})$.

This concludes the proof. □

### B.3 Related Work and Techniques

In their study of AMAC, BBT [7] introduce the 2-tier cascade and augmented 2-tier cascade. They show single-user PRF security of the augmented 2-tier cascade and deduce strong multi-user PRF security for the augmented cascade. The augmentation is the application of an unkeyed output function. Setting the latter to the identity function results in the cascade, so one might think that the results of BBT [7] imply ours as a special case. But this is not true. In fact, the results of BBT [7] do not hold when the output function is the identity. Their assumption on the compression function is PRF security under leakage of the output function, which is not true when the latter is the identity. Indeed, in their result, there is no restriction on queries, meaning no requirement on prefix-freeness, an indication that the result cannot hold for the cascade itself, where we know that security will not hold without a restriction on the queries: they must be prefix-free.