Symmetric and Dual PRFs from Standard Assumptions: A Generic Validation of an HMAC Assumption

Mihir Bellare[1] Anna Lysyanskaya[2]

December 2015

Abstract

The security of HMAC is proven under the assumption that its compression function is a dual PRF, meaning a PRF when keyed by either of its two inputs. But, not only do we not know whether particular compression functions really are dual PRFs, we do not know if dual PRFs even exist. What if the goal is impossible? This paper addresses this with a foundational treatment of dual PRFs, giving constructions based on standard assumptions. This provides what we call a generic validation of the dual PRF assumption for HMAC. Our approach is to introduce and construct symmetric PRFs, which imply dual PRFs and may be of independent interest. We give a general construction of a symmetric PRF based on a function having a weak form of collision resistance coupled with a leakage hardcore function, a strengthening of the usual notion of hardcore functions we introduce. We instantiate this general construction in two ways to obtain a symmetric and dual PRF assuming (1) Any collision-resistant hash function, or (2) Any one-way permutation. A construction based on any one-way function evades us and is left as an intriguing open problem.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: mihir@eng.ucsd.edu. URL: http://cseweb.ucsd.edu/~mihir/. Supported in part by NSF grants CNS-1116800 and CNS-1228890. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467.

[2] Computer Science Department, Brown University, Providence, RI 02912, USA. Email: anna_lysyanskaya@brown.edu. URL: https://cs.brown.edu/people/anna/. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467.

Contents

1 Introduction

The PRF security of HMAC is proven under the assumption that its compression function is a *dual PRF*. But, not only do we not know whether particular compression functions really are dual PRFs, we do not know if dual PRFs even exist. What if the goal is impossible? We address this with a foundational treatment of dual PRFs, giving constructions based on standard assumptions. This is the first theoretical evidence that dual PRFs exist, and provides what we call a generic validation of the dual PRF assumption for HMAC. Tools that we introduce and use for our construction include leakage hardcore functions and symmetric PRFs.

PRFs. Let $\mathsf{F} : \mathsf{F.Keys} \times \mathsf{F.Inp} \to \mathsf{F.Out}$ be a function family taking a key $fk \in \mathsf{F.Keys}$ and an input $x \in \mathsf{F.Inp}$ to (deterministically) return the output $y = \mathsf{F}(fk, x) \in \mathsf{F.Out}$. We recall that F is a PRF [12] if an efficient adversary has negligible advantage in distinguishing whether its oracle is $\mathsf{F}(fk, \cdot)$ or a random function, where fk is chosen at random from $\mathsf{F.Keys}$. This well-known notion has seen an enormous number of applications in both theoretical and applied cryptography.

Dual PRFs. Let $\mathsf{S} : S_0 \times S_1 \to \mathsf{S.Out}$ be a function family. Let $\mathsf{S}^{\mathrm{swap}} : S_1 \times S_0 \to \mathsf{S.Out}$ be defined by $\mathsf{S}^{\mathrm{swap}}(a_0, a_1) = \mathsf{S}(a_1, a_0)$. That is, the key for $\mathsf{S}^{\mathrm{swap}}$ is the input for S and the input for $\mathsf{S}^{\mathrm{swap}}$ is the key for S. Both S and $\mathsf{S}^{\mathrm{swap}}$ are legitimate function families and we can ask if they are PRFs. We say that S is a dual PRF [3] if both S and $\mathsf{S}^{\mathrm{swap}}$ are PRFs. That is (1) an oracle for $\mathsf{S}(a_0, \cdot)$ is indistinguishable from an oracle for a random function when a_0 is chosen at random and, separately but also, (2) an oracle for $\mathsf{S}(\cdot, a_1)$ is indistinguishable from an oracle for a random function when a_1 is chosen at random. The question we consider in this paper is, do dual PRFs exist, and, if so, under what assumptions?

Context. Dual PRFs were introduced in [3] in order to prove security of HMAC. HMAC [4] is a cryptographic hash function based PRF implemented in TLS, SSL and many other places and used billions of times a day. The underlying primitive is the compression function h of the hash function. NMAC is obtained by iterating h and is proven PRF-secure assuming h is a PRF [3, 11]. HMAC is obtained from NMAC by putting the key directly in the input of the hash function H, which results in a swapping of roles: the first iteration of the compression function has the key as the second argument, while successive ones are keying h via its usual first argument. The PRF security of HMAC as proven in [3] thus relies on the assumption that the compression function h is a *dual* PRF.

Generic validation. The assumption that h is a dual PRF could fail for two reasons. One is generic, namely that *nothing* can be a dual PRF. Dual PRFs may simply not exist. The second reason is specific, namely that, although some functions may be dual PRFs, the particular h used in some particular hash function isn't.

Generic failure can be ruled out by showing that the security goal is achievable under standard assumptions. We call this *generic validation*. It has value because generic failure is not an idle fear. It has happened for several (attractive) goals, for example virtual blackbox obfuscation [13, 2] and commitment secure against selective opening [6] to name just a few.

Generic validation won't show that a particular candidate practical construct satisfies the assumption. This needs *dedicated validation*, which is ultimately cryptanalysis. But generic validation is the first step. In its absence, the goal may be just wishful thinking, and the candidate construct doomed. In its presence, the candidate is at least in principle plausible, and successful dedicated validation is a possibility. Generic validation is thus desirable for the security goal underlying any new assumption.

For (standard) PRFs, we have strong generic validation: classical foundational results say that

PRFs exist assuming only that one-way functions exist. (OWFs imply PRGs [15] which imply PRFs [12].) We also have constructions from many particular assumptions [18, 17, 1]. Dual PRFs, in contrast, have at this point *no generic validation*. Despite their having been introduced ten years ago [3], and despite their use as an assumption in supporting the security of the widely-used HMAC [3], there has been no construction under any (standard or not) assumption. This is the gap we fill.

SYMMETRIC PRFS. Our approach to construct dual PRFs is based on the notion we introduce of a symmetric PRF. Let $\mathsf{S} : S \times S \to \mathsf{S.Out}$ be a function family whose keyspace and input space are the same set, call it S. We say that S is *symmetric* if $\mathsf{S}(a_0, a_1) = \mathsf{S}(a_1, a_0)$ for all $a_0, a_1 \in S$. That is, S is unchanged if the order of its inputs is swapped. Then we make the following observation. Suppose S is (1) A PRF, and (2) is symmetric. Then it is a dual PRF. This is easy to see because the symmetry implies that $\mathsf{S}^{\mathrm{swap}} = \mathsf{S}$, namely $\mathsf{S}^{\mathrm{swap}}$ is in fact identical to S. So its PRF security follows directly from the fact that S is a PRF. We will construct symmetric PRFs. First let us step back to ask whether existing PRFs happen to be either dual or symmetric.

NEGATIVE RESULTS. One's first thought may be that every PRF S is also a dual PRF. It is easy to see that this is not true. For example suppose $\mathsf{S} : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ is a PRF with the property that $\mathsf{S}(0^k, a) = a$ for all a. This will not contradict PRF security of S because 0^k has negligible probability of being chosen as the key in the PRF game. However $\mathsf{S}^{\mathrm{swap}}$ is clearly not a PRF because $\mathsf{S}^{\mathrm{swap}}(a, 0^k) = \mathsf{S}(0^k, a) = a$ so an adversary can query its oracle at 0^k and it will get back the key a, using it which it can easily violate PRF security.

Thus we need special constructions. The next natural question is whether known constructions of PRFs are dual PRFs. But they are not. For example, take the classic GGM construction [12] of a PRF from a PRG. We show in Section 3 that there is a choice of the PRG under which the constructed PRF is not a dual PRF. Or take the Naor-Reingold PRF. We give in Section 3 a direct attack violating dual PRF security. The Dodis-Yampolskiy PRF [10] is promising because the formula adds the key and input, thereby seeming to give them symmetric roles, but security requires that the input comes from a much smaller space than the key, and this precludes being a symmetric PRF as per our definition. See Section 3 for more information.

SPRF. In Section 5 we give a general construction of a symmetric (hence dual) PRF S. Roughly the idea is to let R be a PRF with range $\{0,1\}^k$ and let $\mathsf{S} : D \times D \to \{0,1\}^k$ be defined via

$$\mathsf{S}(a_0, a_1) = \mathsf{R}(r_0, z_1) \oplus \mathsf{R}(r_1, z_0) \tag{1}$$

where $r_i = E(a_i)$ and $z_i = H(a_i)$ for certain functions E, H and $i \in \{0,1\}$, and D is some appropriate domain. Thus r_0, z_0 depend on the input a_0 while r_1, z_1 depend on the input a_1, and only in the application of R are the inputs "mixed." Two applications of R are used, the key being an r-value and the input the opposing z-value. Note that the use of this high-level structure with the xor already guarantees that S is symmetric, regardless of the choices of E, H.

Now we need to find choices of E, H under which S is a PRF. Intuitively, a difficulty in using the PRF security of R is that the construction does not use a key for R in a blackbox way. If we think of r_0 as the key, then z_0 is related information that is needed to simulate an attacker against S.

Very roughly, we want E to extract hardcore bits, and we want H to provide some kind of collision resistance. In the proof that S is a PRF we would first use the security of E to move to a game in which r_0 is random. Then we would use the PRF security of R to replace $\mathsf{R}(r_0, \cdot)$ with a random function R. Finally we would use the security of H to say that the z_1 values do not repeat, which means in each xor the first component, and hence the whole, is random.

However getting this to work requires some care. We strive to make the conditions on E, H as weak and general as possible so as to allow the maximum flexibility in instantiation and the ability to instantiate under assumptions as weak as possible. In this spirit one choice we make is to allow both E and H to be keyed. Both the key and the input would be derived from the single input a_i above. Now the main difficulty is that no standard notion of hardcore function security suffices for E. Instead we introduce the notion of a leakage hardcore function for H. Roughly —the formal definition is in Section 4— this means that E with a target key applied to a hidden x_0 continues to look random even given an oracle that can get the results of H at x_0 under other, different keys of its choice. For H we choose as weak a notion of collision resistance as possible. We show that its being computationally almost universal (cau) [3] suffices. See Section 5 for the full construction and Theorem 5.2 for the formal claim and proof of PRF security.

Instantiations. To obtain constructions of symmetric (and hence dual) PRFs under specific, standard assumptions, we instantiate the primitives in our general SPRF construction under the assumption in question. In Section 6 we give two corresponding results, one under one-way permutations (OWPs) and the other under collision-resistant (CR) hash functions, meaning either of these assumptions now yield symmetric and dual PRFs. The OWP instantiation uses the Blum-Micali-Yao (BMY) PRG [8, 21] to instantiate the leakage hardcore function E and an iterated OWP to instantiate H. The CR hash function instantiation uses the latter to instantiate H and uses a strong randomness extractor to instantiate E.

Discussion and open questions. The main open question that evades us is a construction of a symmetric and dual PRF from any one-way function (OWF). The first question is whether one can instantiate our SPRF construction under a OWF. If not, the next question is whether there is some other, different construction.

We note that while our result about SPRF has striven to make as general and weak-as-possible assumptions on the component E, H functions, we have not, in our instantiations, found a way to take full advantage of this. The only way we have found to get a leakage hardcore function E for H is to make H keyless, in which case Lemma 4.1 says that a standard hardcore function suffices. Non-trivial constructions of leakage hardcore functions for H that is cau are thus a direction via which one might make progress.

2 Basic definitions

Our treatment is concrete rather than asymptotic. For any security goal for a primitive, for example prf security of a function family, we define an advantage metric, in this case the prf advantage of an adversary against the function family, which is a number. There is no explicit security parameter. For a function family to be a PRF means, informally, that "efficient" adversaries have "negligible" prf advantage. Theorems are made formal by giving the concrete security of reductions. Discussion surrounding theorems will clarify what they mean qualitatively. The concrete treatment makes notation somewhat simpler, allows us to see the quantitative security of reductions, and is more in keeping with the motivating setting of HMAC, where there are no asymptotics.

Notation and conventions. We let ε denote the empty string. If y is a string then $|y|$ denotes its length and $y[i]$ denotes its i-th coordinate for $1 \leq i \leq |y|$. If X is a finite set, we let $x \leftarrow\!\!\$\, X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. Running time is worst case. If A is an algorithm, we let $y \leftarrow A(x_1, \ldots; r)$ denote running A with random coins r on inputs $x_1, \ldots$ and assigning the output to y. We let $y \leftarrow\!\!\$\, A(x_1, \ldots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \ldots; r)$. We

Game $\mathbf{G}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A})$	Game $\mathbf{G}^{\mathsf{ow}}_{\mathsf{F}}(\mathcal{A})$
$fk \leftarrow\!\!\$\ \mathsf{F.Keys}$	$fk \leftarrow\!\!\$\ \mathsf{F.Keys}$
$c \leftarrow\!\!\$\ \{0,1\}$; $c' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{FN}}$	$x \leftarrow\!\!\$\ \mathsf{F.Inp}$; $y \leftarrow \mathsf{F}(fk, x)$
Return $(c = c')$	$x' \leftarrow\!\!\$\ \mathcal{A}(fk, y)$
$\underline{\mathrm{FN}(x)}$	Return $(\mathsf{F}(fk, x') = y)$
If $T[x] = \perp$ then	
If $(c = 1)$ then	Game $\mathbf{G}^{\mathsf{cau}}_{\mathsf{H}}(\mathcal{A})$
$T[x] \leftarrow \mathsf{F}(fk, x)$	$(x_0, x_1) \leftarrow\!\!\$\ \mathcal{A}$
Else $T[x] \leftarrow\!\!\$\ \mathsf{F.Out}$	$hk \leftarrow\!\!\$\ \mathsf{H.Keys}$
Return $T[x]$	If $(x_0 = x_1)$ then return false
	Return $(\mathsf{H}(hk, x_0) = \mathsf{H}(hk, x_1))$
	Game $\mathbf{G}^{\mathsf{cr}}_{\mathsf{H}}(\mathcal{A})$
	$hk \leftarrow\!\!\$\ \mathsf{H.Keys}$
	$(x_0, x_1) \leftarrow\!\!\$\ \mathcal{A}(hk)$
	If $(x_0 = x_1)$ then return false
	Return $(\mathsf{H}(hk, x_0) = \mathsf{H}(hk, x_1))$

Figure 1: **Games for defining PRF and OWF security of a function family F, cau security of a function family H and HC being a hardcore function family for H.**

let $[A(x_1, \ldots)]$ denote the set of all possible outputs of A when invoked with inputs $x_1, \ldots$. We use the code based game playing framework of [7]. (See Fig. **??** for an example.) By $\Pr[\mathrm{G}]$ we denote the event that the execution of game G results in the game returning true. We adopt the convention that the running time of an adversary refers to the worst-case execution time of the game with the adversary, so that the time for the execution of oracles to compute replies to oracle queries is included. This means that usually in reductions, adversary running time is roughly maintained.

Function families. A function family $\mathsf{F} : \mathsf{F.Keys} \times \mathsf{F.Inp} \to \mathsf{F.Out}$ is a 2-argument function taking a key fk in the keyspace F.Keys and an input x in the input space F.Inp to return an output $\mathsf{F}(fk, x)$ in the output space F.Out. For $fk \in \mathsf{F.Keys}$ we let $\mathsf{F}_{fk} : \mathsf{F.Inp} \to \mathsf{F.Out}$ be defined by $\mathsf{F}_{fk}(x) = \mathsf{F}(fk, x)$ for all $x \in \mathsf{F.Inp}$. We say that F is a permutation family if $\mathsf{F.Inp} = \mathsf{F.Out}$ and F_{fk} is a permutation for every $fk \in \mathsf{F.Keys}$. We say that F is keyless if $\mathsf{F.Keys} = \{\varepsilon\}$ consists only of the empty string. (It is tempting in this case to just drop the key in the notation but it makes it harder to pattern-match with the definitions and so, somewhat pedantically, we tend to explicitly write ε as the key when dealing with keyless families.) The reason to consider such families is that some notions of security, such as one-wayness, hold just as well for them. (For others, like prf-security, keying is crucial.)

Pseudo-random functions. The security of function family F as a PRF is defined via game $\mathbf{G}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A})$ of Fig. 1 associated to F and adversary $\mathcal{A}$. Table T is assumed initially $\perp$ everywhere. The prf advantage of $\mathcal{A}$ is

$$\begin{aligned}\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A}) &= 2\Pr[\mathbf{G}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A})] - 1 \\ &= \Pr[\,\mathbf{G}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A}) \,|\, c = 1\,] - \Big(1 - \Pr[\,\mathbf{G}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A}) \,|\, c = 0\,]\Big)\ . \qquad (2)\end{aligned}$$

The first equation is the definition, while the second is an alternative representation known to the

equal by a standard conditioning argument.

ONE-WAY FUNCTION FUNCTIONS. The security of function family F as a OWF is defined via game $\mathbf{G}^{\mathsf{ow}}_{\mathsf{F}}(\mathcal{A})$ of Fig. 1 associated to F and adversary $\mathcal{A}$. The point x' returned by the latter is required to be in $\mathsf{F.Inp}$. The owf advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{\mathsf{ow}}_{\mathsf{F}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A})]$. In this case, F may or may not be keyed. A one-way permutation (OWP) is simply a family of permutations that is a OWF.

UNIVERSAL AND CAU FUNCTIONS. Consider game $\mathbf{G}^{\mathsf{cau}}_{\mathsf{H}}(\mathcal{A})$ of Fig. 1 associated to H and adversary $\mathcal{A}$. The points x_0, x_1 returned by the latter are required to be in $\mathsf{H.Inp}$. The cau advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{\mathsf{cau}}_{\mathsf{H}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathsf{cau}}_{\mathsf{H}}(\mathcal{A})]$. We say that H is universal if $\mathbf{Adv}^{\mathsf{cau}}_{\mathsf{H}}(\mathcal{A}) = 1/|\mathsf{H.Out}|$ for all adversaries $\mathcal{A}$, regardless of their computing time. Computational au is a computational relaxation from [3] in which the advantage is treated as a computational metric in the usual way and adversaries may be computationally bounded.

CR FUNCTIONS. The security of function family H as a collision-resistant (CR) function is defined via game $\mathbf{G}^{\mathsf{cr}}_{\mathsf{H}}(\mathcal{A})$ of Fig. 1 associated to H and adversary $\mathcal{A}$. The points x_0, x_1 returned by the latter are required to be in $\mathsf{H.Inp}$. The cr advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{\mathsf{cr}}_{\mathsf{H}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathsf{cr}}_{\mathsf{H}}(\mathcal{A})]$. Practical CR hash functions such as SHA-1 are keyless. A CR function family is cau, giving an easy way to get the latter.

EXTRACTORS. Let X, Y be random variables. We define the statistical distance between X and Y, the min-entropy of X and the min-entropy of X given Y, via:

$$\begin{aligned}
\mathbf{SD}(X,Y) &= \frac{1}{2}\sum_z |\Pr[X=z] - \Pr[Y=z]| \\
2^{-\mathbf{H}_\infty(X)} &= \max_x \Pr[X=x] \\
2^{-\mathbf{H}_\infty(X|Y)} &= \sum_y \Pr[Y=y] \cdot \max_x \Pr[\,X=x \,|\, Y=y\,]\,.
\end{aligned}$$

Recall, paraphrasing the definition above, that a function family $\mathsf{Ext}\colon \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ is universal if for every distinct $x_1, x_2 \in \{0,1\}^n$ we have $\Pr[\mathsf{Ext}(sk, x_1) = \mathsf{Ext}(sk, x_2)] = 2^{-m}$ where the probability is over $sk \leftarrow\!\!\$\, \{0,1\}^s$. The following is a generalized version of the Leftover Hash Lemma (LHL) [15, 9].

Lemma 2.1 *Let $\mathsf{Ext}\colon \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ be a function family that is universal. Let X be a random variable over $\{0,1\}^n$. Let U_s, U_m be random variables distributed uniformly over $\{0,1\}^s$ and $\{0,1\}^m$, respectively, and let Y be a random variable. Assume the three random variables $(X,Y), U_s, U_m$ are independent. Then*

$$\mathbf{SD}((U_s, \mathsf{Ext}(U_s, X), Y), (U_s, U_m, Y)) \le \frac{1}{2}\sqrt{2^{m-\mathbf{H}_\infty(X|Y)}}\,. \tag{3}$$

SYMMETRIC PRFS. Let $\mathsf{S}\colon S_0 \times S_1 \to \mathsf{S.Out}$ be a function family. Let $\mathsf{S}^{\mathrm{swap}}\colon S_1 \times S_0 \to \mathsf{S.Out}$ be defined by $\mathsf{S}^{\mathrm{swap}}(a_0, a_1) = \mathsf{S}(a_1, a_0)$. We say that S is a dual PRF if both S and $\mathsf{S}^{\mathrm{swap}}$ are PRFs. We say that S is *symmetric* if $S_0 = S_1$ and $\mathsf{S}(a_0, a_1) = \mathsf{S}(a_1, a_0)$ for every $a_0, a_1 \in S_1$. If S is symmetric then $\mathsf{S}^{\mathrm{swap}} = \mathsf{S}$. Thus if S is symmetric and a PRF, it is automatically a dual PRF. We will accordingly target the stronger notion of a symmetric PRF and obtain a dual PRF as a consequence.

3 Dual PRF security of existing PRF constructions

If we seek dual PRFs, the first and natural question is whether existing constructions of PRFs might happen to already be dual. Here we look at a few popular ones and show this is not the case.

GGM. Let $\mathsf{F}_1 : \{0,1\}^k \times \{0,1\} \to \{0,1\}^k$ be a PRF with input space $\{0,1\}$. The GGM construction [12] builds from it the PRF $\mathsf{GGM} : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ defined as follows.

$$\begin{array}{l}\underline{\text{Function family } \mathsf{GGM}(x,y)} \\ \text{For } i = 1,\ldots,k \text{ do } x \leftarrow \mathsf{F}_1(x, y[i]) \\ \text{Return } x\end{array}$$

Suppose F_1 has the property that $\mathsf{F}_1(0^k, 0) = \mathsf{F}_1(0^k, 1) = 0^k$. It could still be a PRF and in particular if PRFs exist we can easily build a PRF F_1 with this property. But then $\mathsf{GGM}^{\mathrm{swap}}(y, 0^k) = \mathsf{GGM}(0^k, y) = 0^k$ so $\mathsf{GGM}^{\mathrm{swap}}$ is certainly not a PRF. Thus GGM is not a dual PRF. This shows that the GGM construction does not in general yield a dual PRF.

Naor Reingold. Let $\mathbb{G}$ be prime-order group in which the DDH problem is hard, and let $g \in \mathbb{G}$ be a generator of $\mathbb{G}$. Let $q = |\mathbb{G}|$. The Naor-Reingold PRF [18] $\mathsf{NR} : \mathbb{Z}_q^{n+1} \times \{0,1\}^n \to \mathbb{G}$ is defined by

$$\begin{array}{l}\underline{\text{Function family } \mathsf{NR}(\mathbf{a}, x)} \\ b \leftarrow \mathbf{a}[0] \cdot \prod_{i=1}^{n} \mathbf{a}[i]^{x[i]} \bmod q \\ y \leftarrow g^b \\ \text{Return } y\end{array}$$

Here the key $\mathbf{a}$ is a $(n+1)$-vector over $\mathbb{G}$ and its i-th component is denoted $\mathbf{a}[i] \in \mathbb{G}$, with the components indexed from 0 to n. Let $1_{\mathbb{G}}$ denote the identity element of $\mathbb{G}$ and let $\mathbf{0} = (0,\ldots,0) \in \mathbb{G}^{n+1}$ denote the $(n+1)$-vector all of whose components equal 0. Then $\mathsf{NR}^{\mathrm{swap}}(x, \mathbf{0}) = \mathsf{NR}(\mathbf{0}, x) = g^0 = 1_{\mathbb{G}}$ for all $x \in \{0,1\}^n$. Thus $\mathsf{NR}^{\mathrm{swap}}$ cannot be a PRF and NR is not a dual PRF. This is true for all choices of $\mathbb{G}, g$.

Some variants of NR [5] restrict the keyspace to $(\mathbb{Z}_q^*)^{n+1}$, which would preclude the above attack on $\mathsf{NR}^{\mathrm{swap}}$. However, $\mathsf{NR}^{\mathrm{swap}}$ is still subject to attack by setting $\mathbf{a}$ to all 1s.

Dodis Yampolskiy. Let $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-degenerate bilinear map, where groups $\mathbb{G}, \mathbb{G}_T$ have prime order p. Let g be a generator of $\mathbb{G}$ and $S \subseteq \mathbb{Z}_p$ a set of size N. Then the Dodis Yampolskiy PRF [10] $\mathsf{DY} : \mathbb{Z}_p \times S \to \mathbb{G}_T$ is defined by

$$\begin{array}{l}\underline{\text{Function family } \mathsf{DY}(a, x)} \\ \text{If } (a+x) \bmod p = 0 \text{ then } b \leftarrow 1 \text{ else } b \leftarrow (a+x)^{-1} \bmod p \\ y \leftarrow \mathbf{e}(g,g)^b \\ \text{Return } y\end{array}$$

This construction is promising because the roles of a and x are symmetric, so we may think we can swap them and have a symmetric PRF. The difficulty is that for security the input x must come from a much smaller space than the key, meaning $N = |S|$ is much less than p. This is because security is based on the q-BDHI assumption, and as per [10, Theorem 2], security of the PRF requires $q = N$ and security of q-BDHI for adversaries with running time more than N. In particular, the construction is not shown secure when $S = \mathbb{Z}_p$. But to meet our definition of a symmetric PRF from Section 2, the key-space and domain must be the same set. We note that this asymmetry in the key and input for DY, and how it precludes some applications, has been pointed out before in several contexts, including in BC [5] for security against related-key attack.

Game $\mathbf{G}^{\mathsf{hc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A})$	Game $\mathbf{G}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A})$
$hk_0 \leftarrow\!\!\$\ \mathsf{H.Keys}$	$hk_0 \leftarrow\!\!\$\ \mathsf{H.Keys}$
$hck_0 \leftarrow\!\!\$\ \mathsf{HC.Keys}$	$hck_0 \leftarrow\!\!\$\ \mathsf{HC.Keys}$
$x_0 \leftarrow\!\!\$\ \mathsf{H.Inp}$	$x_0 \leftarrow\!\!\$\ \mathsf{H.Inp}$
$w_0 \leftarrow \mathsf{H}(hk_0, x_0)$	$s_1 \leftarrow \mathsf{HC}(hck_0, (hk_0, x_0))$
$s_1 \leftarrow \mathsf{HC}(hck_0, (hk_0, x_0))$	$s_0 \leftarrow\!\!\$\ \mathsf{HC.Out}$
$s_0 \leftarrow\!\!\$\ \mathsf{HC.Out}$	$c \leftarrow\!\!\$\ \{0,1\}$
$c \leftarrow\!\!\$\ \{0,1\}$	$c' \leftarrow\!\!\$\ \mathcal{A}^{\text{LK}}(hk_0, hck_0, s_c)$
$c' \leftarrow\!\!\$\ \mathcal{A}(hk_0, hck_0, w_0, s_c)$	Return $(c = c')$
Return $(c = c')$	$\underline{\text{LK}(hk_1)}$
	$w_0 \leftarrow \mathsf{H}(hk_1, x_0)$
	Return w_0

Figure 2: **Games for defining security of HC as a standard and leakage hardcore function for H.**

Discussion. Although this should be obvious, we should nonetheless clarify that the above attacks do not represent any bugs or critiques. These constructions were not designed or claimed to be dual PRFs. But the first question one should ask in seeking dual PRFs is whether existing constructions of PRFs happen to be dual PRFs. The above indicates that this is not the case and one must seek new constructions.

4 Leakage hardcore functions

For our construction we will introduce an extension of the standard notion of a hardcore function. We call it a leakage hardcore function. To understand it, it is useful to begin by recalling the usual notion.

Hardcore functions. Suppose H is a function family. A *hardcore function for* H is a function family $\mathsf{HC} : \mathsf{HC.Inp} \times (\mathsf{H.Keys} \times \mathsf{H.Inp}) \to \mathsf{HC.Out}$, so that an input is a pair (hk, x) consisting of a key for H and an input for H. We say that HC is a hardcore predicate for H if $\mathsf{HC.Out} = \{0,1\}$. Recall that security considers an adversary given a key hk_0 defining the function $\mathsf{H}(hk_0, \cdot)$, a key hck_0 for the hardcore function, and the result $w \leftarrow \mathsf{H}(hk_0, x)$ of evaluating the function at $x_0 \leftarrow\!\!\$\ \mathsf{H.Inp}$. Now the adversary gets s_c for a challenge bit c where $s_1 = \mathsf{HC}(hck_0, (hk_0, x_0))$ is the output of the hardcore function on x_0 and s_0 is a random string of the same length, and it should have a hard time figuring out c. Formally the security of HC as a hardcore function for H is defined via game $\mathbf{G}^{\mathsf{hc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A})$ of Fig. 2 associated to H, HC and adversary $\mathcal{A}$. The hcf advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{\mathsf{hc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}) = 2\Pr[\mathbf{G}^{\mathsf{hc}}_{\mathsf{H}}(\mathcal{A})] - 1$.

Leakage hardcore functions. A *leakage hardcore function for* H is again a function family $\mathsf{HC} : \mathsf{HC.Inp} \times (\mathsf{H.Keys} \times \mathsf{H.Inp}) \to \mathsf{HC.Out}$, so that an input is a pair (hk, x) consisting of a key for H and an input for H. Again we say that HC is a leakage hardcore predicate for H if $\mathsf{HC.Out} = \{0,1\}$. The new element in a leakage hardcore function is that the adversary has an oracle LK via which it can obtain "leakage" about x_0. This leakage has a very particular form, namely the adversary can obtain the value of the same function family H on x_0 under any key $hk_1 \in \mathsf{H.Keys}$ of its choice,

meaning Lk takes input hk_1 and returns $\mathsf{H}(hk_1, x_0)$, the result of evaluating H on the given key under the hidden input x_0. The requirement is that figuring out the challenge bit remains hard. The formalization uses game $\mathbf{G}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A})$ of Fig. 2 associated to H, HC and adversary $\mathcal{A}$. The lhc advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}) = 2\Pr[\mathbf{G}^{\mathsf{lhc}}_{\mathsf{H}}(\mathcal{A})] - 1$. Since $\mathcal{A}$ could in particular call its oracle on hk_0, we omit giving it $\mathsf{H}(hk_0, x_0)$ as input as in the standard game.

Obtaining leakage hardcore functions. Towards obtaining a leakage hardcore function for a given function family H, one simple observation is that if H is keyless then a standard hardcore function is leakage hardcore. This is captured by the following lemma.

Lemma 4.1 *Suppose* H *is a keyless function family and* $\mathsf{HC} : \mathsf{HC.Inp} \times (\{\varepsilon\} \times \mathsf{H.Inp}) \to \mathsf{HC.Out}$ *is a function family. Let* $\mathcal{A}$ *be a lhc-adversary. Then the proof constructs a hc-adversary* $\mathcal{A}_0$ *such that*

$$\mathbf{Adv}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{hc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}) \ .$$

Adversary $\mathcal{A}_0$ *has about the same running time as adversary* $\mathcal{A}$*.*

Proof of of Lemma 4.1: Adversary $\mathcal{A}_0$ gets inputs hk_0, hck_0, w_0, s_c and runs $\mathcal{A}$ on inputs hk_0, hck_0, s_c. A query hk_1 made by $\mathcal{A}$ to it Lk oracle must be in $\mathsf{H.Keys} = \{\varepsilon\}$ and thus $\mathcal{A}_1$ can simulate this oracle, returning w_0 as the response. Eventually $\mathcal{A}$ outputs a bit c', and $\mathcal{A}_1$ outputs the same bit. ▌

Our construction of a symmetric PRF will need a cau function family that has a leakage hardcore function which outputs lots of bits. In Section 5 we will assume it. Later we will give various constructions from various assumptions.

5 The SPRF construction

We provide our general SPRF construction of a symmetric, and hence dual, PRF.

Ingredients. Our construction of a symmetric PRF has the following ingredients:

- A cau function family $\mathsf{H} : \mathsf{H.Keys} \times \mathsf{H.Inp} \to \mathsf{H.Out}$
- A leakage hardcore function family $\mathsf{HC} : \mathsf{HC.Keys} \times (\mathsf{H.Keys} \times \mathsf{H.Inp}) \to \mathsf{HC.Out}$ for H.
- A PRF $\mathsf{R} : \mathsf{HC.Out} \times \mathsf{R.Inp} \to \mathsf{R.Out}$ such that $\mathsf{H.Out} \times \mathsf{H.Keys} \times \mathsf{HC.Keys} \subseteq \mathsf{R.Inp}$ and the range $\mathsf{R.Out}$ is a commutative group whose operation we denote $*$. Thus a key for R is an output of HC while a triple consisting of an output of H, a key for H and a key for HC is a valid input for R.

We refer to a triple $(\mathsf{H}, \mathsf{HC}, \mathsf{R})$ of function families satisfying the above conditions as a *suite*. The simplest case for the group is that $\mathsf{R.Out} = \{0,1\}^{\mathsf{R.ol}}$ is the set of all strings of some length $\mathsf{R.ol}$, and $y_1 * y_2 = y_1 \oplus y_2$, but the existence of efficient PRFs with algebraic ranges [18] motivates being more general.

SPRF construction. Our construction associates to any suite $(\mathsf{H}, \mathsf{HC}, \mathsf{R})$ as above the function family $\mathsf{S} = \mathbf{SPRF}[\mathsf{H}, \mathsf{HC}, \mathsf{R}]$ defined as follows. It has $\mathsf{S.Keys} = \mathsf{S.Inp} = \mathsf{H.Inp} \times \mathsf{H.Keys} \times \mathsf{HC.Keys}$, meaning a key or input is a triple $a = (x, hk, hck)$ consisting of a point $x \in \mathsf{H.Inp}$, a key hk for the cau family H and a key hck for the hardcore function family HC. It has range the group $\mathsf{S.Out} = \mathsf{R.Out}$. The function family is then defined as shown in Fig. 3.

Proposition 5.1 *Let* $(\mathsf{H}, \mathsf{HC}, \mathsf{R})$ *be a suite of function families. Let* $\mathsf{S} = \mathbf{SPRF}[\mathsf{H}, \mathsf{HC}, \mathsf{R}]$ *be the function family associated to them as above. Then* S *is symmetric.*

Function family $\mathsf{S}(a_0, a_1)$

$(x_0, hk_0, hck_0) \leftarrow a_0$; $(x_1, hk_1, hck_1) \leftarrow a_1$

$r_0 \leftarrow \mathsf{HC}(hck_0, (hk_0, x_0))$; $r_1 \leftarrow \mathsf{HC}(hck_1, (hk_1, x_1))$

$w_0 \leftarrow \mathsf{H}(hk_1, x_0)$; $w_1 \leftarrow \mathsf{H}(hk_0, x_1)$

$z_0 \leftarrow (w_0, hk_0, hck_0)$; $z_1 \leftarrow (w_1, hk_1, hck_1)$

$y_0 \leftarrow \mathsf{R}(r_0, z_1)$; $y_1 \leftarrow \mathsf{R}(r_1, z_0)$

$y \leftarrow y_0 * y_1$

Return y

Figure 3: **Our SPRF construction.**

Proof of Proposition 5.1: The first condition, that the keyspace and input space of S are the same set, is met by definition. For a_0, a_1 in this common set we now need to show that $\mathsf{S}(a_0, a_1) = \mathsf{S}(a_1, a_0)$. This follows from the symmetry in the description of S and the assumption that the group $\mathsf{R.Out}$ is commutative. ▮

PRF SECURITY OF SPRF. To show S is a dual PRF, it suffices by Proposition 5.1 to show that S is a PRF. This is the claim of the following theorem.

Theorem 5.2 *Let* $(\mathsf{H}, \mathsf{HC}, \mathsf{R})$ *be a suite of function families. Let* $\mathsf{S} = \mathbf{SPRF}[\mathsf{H}, \mathsf{HC}, \mathsf{R}]$ *be the function family associated to them as above. Let* $\mathcal{A}$ *be an adversary making at most* q *queries to its* FN *oracle. Then the proof constructs adversaries* $\mathcal{A}_{\mathsf{H}}, \mathcal{A}_{\mathsf{HC}}, \mathcal{A}_{\mathsf{R}}$ *such that*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{S}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}_{\mathsf{HC}}) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{R}}(\mathcal{A}_{\mathsf{R}}) + \frac{q(q-1)}{2} \cdot \mathbf{Adv}^{\mathsf{cau}}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}}) \ . \tag{4}$$

The running times of the constructed adversaries are about the same as that of the original.

Proof of Theorem 5.2: Consider games G_0–G_4 of Fig. 4. In the code for games $\mathrm{G}_0, \mathrm{G}_1$, if a line is followed by the name of a game, then that line is included *only* in the named game. Unmarked lines are included in both games. Game G_2 includes the boxed code while game G_3 does not.

We assume wlog that the oracle queries of $\mathcal{A}$ are always all distinct. This means the "If $T[x] = \perp$" test in game $\mathbf{G}^{\mathsf{prf}}_{\mathsf{S}}(\mathcal{A})$ of Fig. 1 will always return true and so we can drop it. The $c = 1$ case of $\mathbf{G}^{\mathsf{prf}}_{\mathsf{S}}(\mathcal{A})$ is thus captured by game G_0. On the other hand, game G_4 captures the $c = 0$ case of game $\mathbf{G}^{\mathsf{prf}}_{\mathsf{S}}(\mathcal{A})$ except that it returns true iff the latter returns false. From Equation (2) we thus have

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{S}}(\mathcal{A}) &= \Pr[\, \mathbf{G}^{\mathsf{prf}}_{\mathsf{S}}(\mathcal{A}) \,|\, c = 1 \,] - \Big(1 - \Pr[\, \mathbf{G}^{\mathsf{prf}}_{\mathsf{S}}(\mathcal{A}) \,|\, c = 0 \,]\Big) \\
&= \Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_4] \\
&= p_0 + p_1 + p_2 + p_3 \ , \qquad (5)
\end{aligned}$$

where for $i \in \{0, 1, 2, 3\}$ we have let

$$p_i = \Pr[\mathrm{G}_i] - \Pr[\mathrm{G}_{i+1}] \ .$$

Games G_0, G_1	Games $\boxed{\mathrm{G}_2}$, G_3	Game G_4
$hk_0 \leftarrow\!\!\$\ \mathsf{H.Keys}$	$hk_0 \leftarrow\!\!\$\ \mathsf{H.Keys}$	$c' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{FN}}$
$hck_0 \leftarrow\!\!\$\ \mathsf{HC.Keys}$	$hck_0 \leftarrow\!\!\$\ \mathsf{HC.Keys}$	Return $(c' = 1)$
$x_0 \leftarrow\!\!\$\ \mathsf{H.Inp}$	$x_0 \leftarrow\!\!\$\ \mathsf{H.Inp}$	$\underline{\mathrm{FN}((x_1, hk_1, hck_1))}$
$r_0 \leftarrow \mathsf{HC}(hck_0, (hk_0, x_0))$ // G_0	$c' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{FN}}$	$y \leftarrow\!\!\$\ \mathsf{R.Out}$
$r_0 \leftarrow\!\!\$\ \mathsf{HC.Out}$ // G_1	Return $(c' = 1)$	Return y
$c' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{FN}}$	$\underline{\mathrm{FN}((x_1, hk_1, hck_1))}$	
Return $(c' = 1)$	$r_1 \leftarrow \mathsf{HC}(hck_1, (hk_1, x_1))$	
$\underline{\mathrm{FN}((x_1, hk_1, hck_1))}$	$w_0 \leftarrow \mathsf{H}(hk_1, x_0)$	
$r_1 \leftarrow \mathsf{HC}(hck_1, (hk_1, x_1))$	$w_1 \leftarrow \mathsf{H}(hk_0, x_1)$	
$w_0 \leftarrow \mathsf{H}(hk_1, x_0)$	$z_0 \leftarrow (w_0, hk_0, hck_0)$	
$w_1 \leftarrow \mathsf{H}(hk_0, x_1)$	$z_1 \leftarrow (w_1, hk_1, hck_1)$	
$z_0 \leftarrow (w_0, hk_0, hck_0)$	$y_0 \leftarrow\!\!\$\ \mathsf{R.Out}$	
$z_1 \leftarrow (w_1, hk_1, hck_1)$	If $(R[z_1] \neq \perp)$ then	
$y_0 \leftarrow \mathsf{R}(r_0, z_1)$	$\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{y_0 \leftarrow R[z_1]}$	
$y_1 \leftarrow \mathsf{R}(r_1, z_0)$	$R[z_1] \leftarrow y_0$	
$y \leftarrow y_0 * y_1$	$y_1 \leftarrow \mathsf{R}(r_1, z_0)$	
Return y	$y \leftarrow y_0 * y_1$	

Figure 4: **Games for proof of Theorem 5.2.**

We will build adversaries $\mathcal{A}_{\mathsf{H}}, \mathcal{A}_{\mathsf{HC}}, \mathcal{A}_{\mathsf{R}}$ such that

$$p_0 \leq \mathbf{Adv}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}_{\mathsf{HC}}) \tag{6}$$

$$p_1 \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{R}}(\mathcal{A}_{\mathsf{R}}) \tag{7}$$

$$p_2 \leq \frac{q(q-1)}{2} \cdot \mathbf{Adv}^{\mathsf{cau}}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}}) \ . \tag{8}$$

We will also observe that

$$p_3 = p_4 \ . \tag{9}$$

Putting together Equations (5), (6), (7), (8) and (9) we get Equation (4). We now justify the above claims.

In game G_1, the key r_0 for the first application of R is chosen at random rather than obtained as $\mathsf{HC}(hck_0, (hk_0, x_0))$. Consider adversary $\mathcal{A}_{\mathsf{HC}}$ shown in Fig. 5. It is playing game $\mathbf{G}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}_{\mathsf{HC}})$, so it has input hk_0, hck_0, s. It runs $\mathcal{A}$, simulating the latter's FN oracle via a procedure FNSIM that is shown in the code. The key point is that $\mathcal{A}_{\mathsf{HC}}$ invokes its LK oracle to compute w_0. Letting c be the challenge bit in game $\mathbf{G}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}_{\mathsf{HC}})$ we have

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}_{\mathsf{HC}}) \\
&\quad = \ \Pr[\, \mathbf{Adv}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}_{\mathsf{HC}}) \,|\, c = 1 \,] - \Big(1 - \Pr[\, \mathbf{Adv}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}_{\mathsf{HC}}) \,|\, c = 0 \,]\Big) \\
&\quad = \ \Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1] = p_0
\end{aligned}$$

which establishes Equation (6).

Adversary $\mathcal{A}^{\text{Lk}}_{\mathsf{HC}}(hk_0, hck_0, s)$	Adversary $\mathcal{A}^{\text{Fn}}_{\mathsf{R}}$	Adversary $\mathcal{A}_{\mathsf{H}}$
$r_0 \leftarrow s$	$hk_0 \leftarrow\!\!\$\ \mathsf{H.Keys}$	$i \leftarrow 0$
$c' \leftarrow\!\!\$\ \mathcal{A}^{\text{FnSim}}$	$hck_0 \leftarrow\!\!\$\ \mathsf{HC.Keys}$	$c' \leftarrow\!\!\$\ \mathcal{A}^{\text{FnSim}}$
Return c'	$x_0 \leftarrow\!\!\$\ \mathsf{H.Inp}$	$j_1 \leftarrow\!\!\$\ \{2, \ldots, i\}$
$\underline{\text{FnSim}((x_1, hk_1, hck_1))}$	$c' \leftarrow\!\!\$\ \mathcal{A}^{\text{FnSim}}$	$j_2 \leftarrow\!\!\$\ \{1, \ldots, j_1 - 1\}$
$r_1 \leftarrow \mathsf{HC}(hck_1, (hk_1, x_1))$	Return c'	Return (v_{j_1}, v_{j_2})
$w_0 \leftarrow \text{Lk}(hk_1)$	$\underline{\text{FnSim}((x_1, hk_1, hck_1))}$	$\underline{\text{FnSim}((x_1, hk_1, hck_1))}$
$w_1 \leftarrow \mathsf{H}(hk_0, x_1)$	$r_1 \leftarrow \mathsf{HC}(hck_0, (hk_0, x_0))$	$i \leftarrow i + 1$
$z_0 \leftarrow (w_0, hk_0, hck_0)$	$w_0 \leftarrow \mathsf{H}(hk_1, x_0)$	$v_i \leftarrow x_1$
$z_1 \leftarrow (w_1, hk_1, hck_1)$	$w_1 \leftarrow \mathsf{H}(hk_0, x_1)$	$y \leftarrow\!\!\$\ \mathsf{R.Out}$
$y_0 \leftarrow \mathsf{R}(r_0, z_1)$	$z_0 \leftarrow (w_0, hk_0, hck_0)$	Return y
$y_1 \leftarrow \mathsf{R}(r_1, z_0)$	$z_1 \leftarrow (w_1, hk_1, hck_1)$	
$y \leftarrow y_0 * y_1$	$y_0 \leftarrow \text{Fn}(z_1)$	
Return y	$y_1 \leftarrow \mathsf{R}(r_1, z_0)$	
	$y \leftarrow y_0 * y_1$	
	Return y	

Figure 5: **Adversaries for proof of Theorem 5.2.**

Game G_2 maintains a table $R[\cdot]$ that is initially everywhere $\perp$. It optimistically picks y_0 at random and sets $R[z_1]$ to this value. However, in between these two steps, it first checks whether $R[z_1]$ was already defined, and if so, sets the flag bad to true. This means that the setting of $R[z_1]$ to the newly-chosen y_0 was wrong. Accordingly (via the boxed code which is included in game G_2) a correction is made, resetting y_0 back to $R[z_1]$, so that in this game, $R[z_1]$ is the result of a random function on z_1. Now consider adversary $\mathcal{A}_{\mathsf{R}}$ shown in Fig. 5. It has an Fn oracle, and runs $\mathcal{A}$. In the simulation of $\mathcal{A}$'s oracle, it applies Fn to z_1 to get y_0. With c the challenge bit in game $\mathbf{G}^{\mathsf{prf}}_{\mathsf{R}}(\mathcal{A}_{\mathsf{R}})$, we have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{R}}(\mathcal{A}_{\mathsf{R}}) &= \Pr[\,\mathbf{G}^{\mathsf{prf}}_{\mathsf{R}}(\mathcal{A}_{\mathsf{R}}) \,|\, c = 1\,] - \Big(1 - \Pr[\,\mathbf{G}^{\mathsf{prf}}_{\mathsf{R}}(\mathcal{A}_{\mathsf{R}}) \,|\, c = 0\,]\Big) \\ &= \Pr[\text{G}_1] - \Pr[\text{G}_2] = p_1\end{aligned}$$

which establishes Equation (7).

In game G_3, we may set bad, but, since the boxed code is absent, y_0 is always a fresh, random value. Games G_2, G_3 are identical until bad (differ only in code following the setting of bad to true) so by the Fundamental Lemma of Game Playing [7],

$$p_2 = \Pr[\text{G}_2] - \Pr[\text{G}_3] \le \Pr[\text{G}_3 \text{ sets } \mathsf{bad}]\ . \tag{10}$$

Now consider adversary $\mathcal{A}_{\mathsf{H}}$ in Fig. 5. We claim that

$$\Pr[\text{G}_3 \text{ sets } \mathsf{bad}] \le \frac{q(q-1)}{2} \cdot \mathbf{Adv}^{\mathsf{cau}}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}})\ . \tag{11}$$

The reason is that for game G_3 to set bad, a z_1 value must repeat across queries. By assumption the queries are distinct, so the only way this could happen is if there were queries $j_1 < j_2$ such that the w_1, hk_1, hck_1 values in these queries were the same but the x_1 values were different. This would be a collision for $\mathsf{H}(hk_0, \cdot)$. Now we have to argue that such a collision can be found by a cau adversary $\mathcal{A}_{\mathsf{H}}$. This adversary does not know hk_0, so how can it simulate $\mathcal{A}$? In game G_3, the

point y_0 is always random. Since $\mathsf{R.Out}$ is a group, y is also random. So $\mathcal{A}_{\mathsf{H}}$ can simulate $\mathcal{A}$'s oracle by just returning random values. It does this, collecting all the x_1 values in the queries. In the end it picks at random two of these values and returns them. This justifies Equation (11), which, combined with Equation (10), justifies Equation (8).

As we have just said, in game G_3, the point y_0 is always random and independent of anything else. Since $\mathsf{R.Out}$ is a group, y is also random. This justifies Equation (9) and completes the proof. ▌

6 Instantiations

We instantiate our SPRF construction to get symmetric and dual PRFs under specific assumptions.

6.1 Construction from CR hash functions

We give a construction from any keyless collision-resistant hash function. It itself will play the role of H. The following lemma says that for suitable choices of parameters, an extractor —see Section 2 for background— will provide a leakage hardcore function.

Lemma 6.1 *Let* $\mathsf{H}\colon \{\varepsilon\} \times \{0,1\}^n \to \{0,1\}^r$ *be a keyless function family. Let* $\mathsf{Ext}\colon \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ *be a function family that is universal. Let* $\mathsf{HC}\colon \{0,1\}^s \times (\{\varepsilon\} \times \{0,1\}^n) \to \{0,1\}^m$ *be defined by* $\mathsf{HC}(hck, (\varepsilon, x)) = \mathsf{Ext}(hck, x)$*. Let* $\mathcal{A}$ *be a lhc-adversary. Then*

$$\mathbf{Adv}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}) \leq 2^{-(n+2-m-r)/2}\ . \tag{12}$$

The result is information-theoretic, meaning is true regardless of the running time of $\mathcal{A}$.

Proof of of Lemma 6.1: Let random variable X be uniformly distributed over $\{0,1\}^n$. Let U_s, U_m be random variables distributed uniformly over $\{0,1\}^s$ and $\{0,1\}^m$, respectively, and let $Y = \mathsf{H}(\varepsilon, X)$. The following chain of inequalities, which establishes the lemma, is justified below:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A}) &\leq \mathbf{SD}((U_s, \mathsf{Ext}(U_s, X), Y), (U_s, U_m, Y)) && (13)\\
&\leq \frac{1}{2}\sqrt{2^{m-\mathbf{H}_\infty(X|Y)}} && (14)\\
&\leq 2^{-(n+2-m-r)/2}\ . && (15)
\end{aligned}$$

Let X and U_s represent, respectively, the randomly chosen x_0 and hck in game $\mathbf{G}^{\mathsf{lhc}}_{\mathsf{H},\mathsf{HC}}(\mathcal{A})$ of Fig. 2. Then $\mathsf{Ext}(U_s, X)$ represents s_1 while U_m represents s_0. Since H is keyless, the only information $\mathcal{A}$ can get from its Lk oracle is $Y = \mathsf{H}(\varepsilon, X)$. The statistical distance of Equation (13) then represents the maximum possible advantage that $\mathcal{A}$ can obtain. The three random variables (X, Y), U_s, U_m are independent so we can apply Lemma 2.1 to get Equation (14). Since $|Y| = r$ we have $\mathbf{H}_\infty(X|Y) \geq n - r$, which, together with some simplification, yields Equation (15). ▌

Our symmetric and dual PRF S is parameterized by integers m, r. Given the latter, we select n so that $2^{-(n+2-m-r)/2}$ is negligible. Then we select a function family $\mathsf{Ext}\colon \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ that is universal. Next we select a keyless, collision-resistant function family $\mathsf{H}\colon \{\varepsilon\} \times \{0,1\}^n \to \{0,1\}^r$. (The definition is recalled in Section 2.) Since it is collision resistant, it is certainly cau. Finally we select a PRF $\mathsf{R}\colon \{0,1\}^m \times \mathsf{R.Inp} \to \mathsf{R.Out}$ such that $\{0,1\}^r \times \{\varepsilon\} \times \{0,1\}^s \subseteq \mathsf{F.Inp}$ and $\mathsf{R.Out}$ is a commutative group, for simplicity $\{0,1\}^l$ for some l with the group operation being bitwise xor. This is not an extra assumption because CR functions imply OWFs which imply PRFs.

We let HC be defined as in Lemma 6.1. We now have a suite $(\mathsf{H}, \mathsf{HC}, \mathsf{R})$ and can apply our **SPRF** transform. The resulting symmetric and dual PRF is $\mathsf{S} : (\{0,1\}^n \times \{\varepsilon\} \times \{0,1\}^s) \times (\{0,1\}^n \times \{\varepsilon\} \times \{0,1\}^s) \to \{0,1\}^l$ defined by

$$\begin{array}{l}
\underline{\text{Function family } \mathsf{S}(((x_0, \varepsilon, sk_0), (x_1, \varepsilon, sk_1))} \\
r_0 \leftarrow \mathsf{Ext}(sk_0, x_0) \ ; \ r_1 \leftarrow \mathsf{Ext}(sk_1, x_1) \\
w_0 \leftarrow \mathsf{H}(\varepsilon, x_0) \ ; \ w_1 \leftarrow \mathsf{H}(\varepsilon, x_1) \\
z_0 \leftarrow (w_0, \varepsilon, sk_0) \ ; \ z_1 \leftarrow (w_1, \varepsilon, sk_1) \\
y_0 \leftarrow \mathsf{R}(r_0, z_1) \ ; \ y_1 \leftarrow \mathsf{R}(r_1, z_0) \\
y \leftarrow y_0 \oplus y_1 \\
\text{Return } y
\end{array}$$

6.2 Construction from any OWP

We show that the existence of one-way permutations (OWPs) implies the existence of dual PRFs. We do this by instantiating our SPRF construction using an iterated OWP for H and a leakage hardcore function obtained via the BMY PRG [8, 21].

Let $\mathsf{F} : \{\varepsilon\} \times X \to X$ be a keyless one-way family of permutations with domain and range a set X. (The standard definition of a OWP is indeed keyless.) For $i \geq 1$ let $\mathsf{F}^{(i)} : \{\varepsilon\} \times X \to X$ be the i-th iterate of F, defined inductively by

$$\mathsf{F}^{(0)}(\varepsilon, x) = x \quad \text{and} \quad \mathsf{F}^{(i)}(\varepsilon, x) = \mathsf{F}(\varepsilon, \mathsf{F}^{(i-1)}(\varepsilon, x)) \text{ for } i \geq 1 \ .$$

Our symmetric and dual PRF S is parameterized by an integer m. Let $\mathsf{R} : \{0,1\}^m \times \mathsf{R.Inp} \to \mathsf{R.Out}$ be a PRF such that $X \times \{\varepsilon\} \times \{\varepsilon\} \subseteq \mathsf{F.Inp}$ and $\mathsf{R.Out}$ is a commutative group, for simplicity $\{0,1\}^l$ for some l with the group operation being bitwise xor. This is not an extra assumption because OWPs imply PRFs. Let $\mathsf{H} = \mathsf{F}^{(m)}$ be the m-fold iterate of F. Let $\mathsf{HC}_1 : \{\varepsilon\} \times (\{\varepsilon\} \times \mathsf{H.Inp}) \to \{0,1\}$ be a hardcore predicate for F. We assume it is keyless, since we can find such hardcore predicates for any OWP. Let $\mathsf{HC} : \{\varepsilon\} \times (\{\varepsilon\} \times \mathsf{H.Inp}) \to \{0,1\}^m$ be defined by

$$\begin{array}{l}
\underline{\text{Function family } \mathsf{HC}(\varepsilon, (\varepsilon, x))} \\
\text{For } i = 0, \ldots, m \text{ do} \\
\quad b_i \leftarrow \mathsf{HC}_1(\varepsilon, (\varepsilon, x)); \ x \leftarrow \mathsf{F}(\varepsilon, x) \\
\text{Return } b_1 b_2 \ldots b_m
\end{array}$$

Then from [8, 21] we know that there is a choice of HC_1 such that HC is a hardcore function for $\mathsf{H} = \mathsf{F}^{(m)}$ assuming only one-wayness of F. Now we have two observations. First, since F, and hence H, is keyless, and we know that HC is a hardcore function for H, Lemma 4.1 implies that it is also a *leakage* hardcore function for H. Second, H is trivially cau, because it is a permutation family, so there simply do not exist collisions. We can thus apply our **SPRF** transform to the suite $(\mathsf{H}, \mathsf{HC}, \mathsf{R})$ to get a symmetric function family S that, by Theorem 5.2, is a PRF.

6.3 Construction from any OWF?

A construction from any OWF eludes us. Let us discuss why some obvious approaches fail. We know that UOWHFs, as defined by [19], exist given any OWF, as shown in [20, 16, 14]. A UOWHF is certainly cau so can play the role of H for our transform. However it is not clear to us how to find a leakage hardcore function for a UOWHF H. The difficulty is that H is non-trivially keyed, so the leakage oracle $\textsc{Lk}$ provides potentially useful information to the adversary.

7 Acknowledgments

We thank Stefano Tessaro for helpful comments on a previous draft.

References

[1] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, Apr. 2012. 4

[2] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, Aug. 2001. 3

[3] M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Heidelberg, Aug. 2006. 3, 4, 5, 7

[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15. Springer, Heidelberg, Aug. 1996. 3

[5] M. Bellare and D. Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, Heidelberg, Aug. 2010. 8

[6] M. Bellare, R. Dowsley, B. Waters, and S. Yilek. Standard security does not imply security against selective-opening. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 645–662. Springer, Heidelberg, Apr. 2012. 3

[7] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 6, 13

[8] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984. 5, 15

[9] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008. 7

[10] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, Jan. 2005. 4, 8

[11] P. Gaži, K. Pietrzak, and M. Rybár. The exact PRF-security of NMAC and HMAC. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 113–130. Springer, Heidelberg, Aug. 2014. 3

[12] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, Oct. 1986. 3, 4, 8

[13] S. Hada. Zero-knowledge and code obfuscation. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 443–457. Springer, Heidelberg, Dec. 2000. 3

[14] I. Haitner, T. Holenstein, O. Reingold, S. P. Vadhan, and H. Wee. Universal one-way hash functions via inaccessible entropy. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 616–637. Springer, Heidelberg, May 2010. 15

[15] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 4, 7

[16] J. Katz and C.-Y. Koo. On constructing universal one-way hash functions from arbitrary one-way functions. Cryptology ePrint Archive, Report 2005/328, 2005. `http://eprint.iacr.org/2005/328`. 15

[17] A. B. Lewko and B. Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *ACM CCS 09*, pages 112–120. ACM Press, Nov. 2009. 4

[18] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51(2):231–262, 2004. 4, 8, 10

[19] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989. 15

[20] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990. 15

[21] A. C.-C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, Nov. 1982. 5, 15