# Basing PRFs on Constant-Query Weak PRFs: Minimizing Assumptions for Efficient Symmetric Cryptography*

Ueli Maurer and Stefano Tessaro

Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
{maurer,tessaros}@inf.ethz.ch

**Abstract.** Although it is well known that all basic private-key cryptographic primitives can be built from one-way functions, finding weak assumptions from which practical implementations of such primitives exist remains a challenging task. Towards this goal, this paper introduces the notion of a *constant-query weak PRF*, a function with a secret key which is computationally indistinguishable from a truly random function when evaluated at a *constant* number $s$ of known random inputs, where $s$ can be as small as two.

We provide iterated constructions of (arbitrary-input-length) PRFs from constant-query weak PRFs that even improve the efficiency of previous constructions based on the stronger assumption of a weak PRF (where polynomially many evaluations are allowed).

One of our constructions directly provides a new practical mode of operation using a constant-query weak PRF for IND-CPA symmetric encryption which is essentially as efficient as conventional PRF-based schemes. Furthermore, our constructions yield efficient modes of operation for keying hash functions (such as MD5 and SHA-1) to obtain iterated PRFs (and hence MACs) which rely solely on the assumption that the underlying compression function is a constant-query weak PRF, which is the weakest assumption ever considered in this context.

## 1 Introduction

### 1.1 Minimizing Assumptions: Constant-Query Weak PRFs

Most cryptographic security proofs are *reductions*: Under the *assumption* that a primitive $P$ exists, the existence of a second primitive $P'$ is shown by means of a concrete construction that uses an implementation of $P$ (usually in a black-box manner) to implement $P'$. For example, $P'$ could be a *pseudorandom function* (PRF), i.e. a function with a secret key which is computationally indistinguishable from a truly random function under arbitrary (adaptive) access. These functions are central primitives as they provide a direct solution to the problems of provably secure symmetric encryption and message authentication.

Ideally, one would like the underlying primitive $P$ to be as *weak* as possible, as in practice it is more likely that an efficient and secure candidate is successfully designed. Also, it is a safe practice to assume that already existing cryptographic functions (such as block ciphers or compression functions of hash functions) only fulfill weaker properties than what they have been originally designed for. Sometimes, however, reductions to weak assumptions turn out to be inefficient and involve large security losses (cf. [14] for a typical example), and hence designers of cryptographic systems are frequently confronted with a *trade-off* between the strength of the underlying assumption and the complexity of the resulting construction.

With the aim of proposing new weak assumptions for the purpose of building symmetric-key primitives, this paper introduces the notion of *constant-query weak pseudorandom functions*:

* This research was partially supported by the Swiss National Science Foundation (SNF), project no. 200020-113700/1. An extended abstract of this paper appears in the proceedings of ASIACRYPT 2008. This is the preliminary full version.

Informally, for some constant $s$, a function $F : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ with $\kappa < s \cdot n$ is an *$s$-query weak PRF* ($s$-WPRF) if $F(K, \cdot)$ (under a secret key $K$) is indistinguishable from a random function when evaluated at $s$ independent *known* random inputs.[1] This notion weakens significantly the regular concept of a *weak pseudorandom function* (WPRF) [19], where indistinguishability for polynomially many random inputs is required. We point out that a WPRF is by itself already much weaker than a PRF, as it possibly exhibits several non-random properties (such as having weak inputs or being commutative, i.e. $F(k, F(k', x)) = F(k', F(k, x))$). On top of this, an $s$-WPRF allows for even more structure: For instance, any $s+1$ distinct inputs $x_1, \ldots, x_{s+1}$ and the corresponding outputs $F(k, x_1), \ldots, F(k, x_{s+1})$ under a secret key $k$ may satisfy an easily verifiable relation with no impact on the pseudorandomness of the function.

In this work, we address the problem of using $s$-WPRFs to construct PRFs. Since $s$-WPRFs imply the existence of one-way functions, a straightforward construction can be obtained using the results of [14, 13]. However, the inefficiency and the security loss of the resulting reduction make this approach unsuitable for any practical use, even if the underlying $s$-WPRF is both highly efficient and secure. For this reason, this paper deals with the question of finding *efficient* constructions of PRFs from $s$-WPRFs: Surprisingly, we are able to provide constructions which are more efficient than existing reductions of PRFs to WPRFs, while only requiring the underlying function to be an $s$-WPRF, for $s$ as low as two. Furthermore, our constructions are iterated and can process inputs of arbitrary input length. This structure makes them well suited to be derived from properly keyed hash functions with very weak compression functions.

The next two sections are devoted to discussing previous work in the contexts of building PRFs from WPRFs and of iterated PRFs and MACs, respectively, and to relating it to our results.

### 1.2 Construction of PRFs from Weak PRFs

The first construction of a PRF from a WPRF is due to Naor and Reingold [19], and a further construction was later proposed by Maurer and Sjödin [17]. Both assume[2] a *length-preserving* underlying function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ (which can be obtained e.g. from a block cipher) and realize a keyed function mapping $\ell$-bit strings to $n$-bit strings (for a fixed input length $\ell$).

The Naor-Reingold Construction [19]. The construction $\mathsf{NR}_\ell$ takes an $\ell$-bit input (with $\ell$ being a power of two) and its secret key consists of $2\ell$ $n$-bit strings $k_{1,0}, k_{1,1}, \ldots, k_{\ell,0}, k_{\ell,1}$. The computation on input $x = (x_1, \ldots, x_\ell)$ proceeds as follows: First, we define $y_i^{(\log \ell + 1)} := k_{i,x_i}$ for all $i = 1, \ldots, \ell$. Then, for all $j = \log \ell, \ldots, 1$ we compute $y_i^{(j)} := F(y_{2i-1}^{(j+1)}, y_{2i}^{(j+1)})$ for all $i = 1, \ldots, 2^{j-1}$ and finally output $y_1^{(1)}$. In other words, the elements of the key corresponding to the individual input bits are chosen as the values of the $\ell$ leaves of a complete binary tree which is evaluated in a bottom-up fashion by computing the value of each inner vertex as $F(y_l, y_r)$, where $y_l$ and $y_r$ are the values of its children, and finally outputting the value of the root. Hence, one evaluation of the construction needs $\frac{\ell}{2} + \frac{\ell}{4} + \cdots + \frac{\ell}{\ell} = \ell - 1$ calls to the underlying function $F$. A more involved construction (which we call $\overline{\mathsf{NR}}_{s,\ell}$) by the same authors uses a key consisting of $s$ $n$-bit values and improves the total number of calls to roughly $\ell / \log s$

[1] The assumption that $s$-WPRFs exist implies the existence of one-way functions, since the mapping $(k, r) \mapsto F(k, r)$ is easily verified to be one-way as long as $\kappa < s \cdot n$. For $\kappa \geq s \cdot n$, such functions can be constructed unconditionally, e.g. using $s$-wise independent functions. (However, optimal unconditional constructions with $\kappa = s \cdot n$ are not known for all parameters $m$.)

[2] The constructions of [19] rely on an intermediate primitive, called a *synthesizer*, but in fact a WPRF $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a synthesizer.

per evaluation, but only accepts $\ell$ and $\log s$ to have the form $2^j + 2$ for some $j \geq 0$. (For both constructions, other input lengths can be achieved through appropriate paddings.)

THE IC-CONSTRUCTION [17]. The construction $\mathsf{IC}_\ell$ takes a $(\kappa + 2n)$-bit key consisting of three values $k_1 \in \{0,1\}^\kappa$ and $r, r' \in \{0,1\}^n$. (The value $r'$ can even be made public.) It first precomputes the values $k_i := F(k_{i-1}, r')$ for all $i = 2, \ldots, \ell$. Furthermore, on an $\ell$-bit input $x = (x_1, \ldots, x_\ell)$, it sets $y_0 := r$, and for all $j = 1, \ldots, \ell$, computes $y_j := F(k_j, y_{j-1})$ if $x_j = 1$, and $y_j := y_{j-1}$ else. Finally, it outputs $y_\ell$. The construction $\mathsf{IC}_\ell$ requires $w(x)$ calls to $F$ when evaluated on input $x$, where $w(x) \leq \ell$ is the hamming weight of $x$. If memory restrictions do not allow storage of the keys $k_2, \ldots, k_\ell$, their values have to be computed at each evaluation and thus the construction requires $(\ell - 1) + w(x)$ calls to $F$ per evaluation, which can be as high as $2\ell - 1$.

A central remark is that in order to prove security of all the aforementioned constructions with respect to adversaries issuing $q$ queries, the underlying WPRF must also be secure when evaluated at (at least) $q$ random inputs.[3] Moreover, in this paper we will focus on iterated constructions of PRFs and MACs where candidates for WPRFs may arise from (keyed) compression functions of hash functions, which have the form $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$ (where e.g. $\kappa = 160$ and $n = 512$ for SHA-1). The above constructions can all be extended in a straightforward way[4] to handle such functions as well, but for the same input length $\ell$ the number of calls would increase considerably if $n > \kappa$ (roughly, by a factor of $\lceil \frac{n}{\kappa} \rceil$ with respect to the case $n = \kappa$, which is e.g. 4 for SHA-1). This holds even if we just want $\kappa$-bit outputs. Hence, this calls for a construction for which the condition $n > \kappa$ does not have a negative impact on the efficiency of the construction.

### 1.3 Assumptions in Iterated MACs and PRFs

Bellare et al. [2] proposed two constructions of *message authentication codes*[5] (MACs), called $\mathsf{HMAC}$ and $\mathsf{NMAC}$, which are obtained by appropriately keying an iterated[6] hash function $H : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^\kappa$ (where the first input is the initialization value) as $\mathsf{HMAC}(k_1 \| k_2, x) := H(IV, k_2 \| H(IV, k_1 \| x))$ (for a fixed known $IV$ and $|k_1|, |k_2|$ both equal to the block length of $H$) and as $\mathsf{NMAC}(k_1 \| k_2, x) := H(k_2, H(k_1, x))$, respectively.[7] (Note that $\mathsf{HMAC}$ only requires black-box usage of $H$.) Even though alternative designs of MACs exist (such as CBC-MAC [5] and UMAC [8] to name a few), these constructions have enjoyed widespread usage due to the large availability of hash function implementations (both in hardware and in software). From the theoretical standpoint, security of $\mathsf{HMAC}/\mathsf{NMAC}$ has been first proved [2] under the assumption that the compression function of $H$ is a PRF (when keyed through the chaining value), and that $H$ is *weakly collision resistant*, i.e. it is hard to find two distinct messages $x, x'$ with $H(K, x) = H(K, x')$ for a secret key $K$ (given oracle access to $H(K, \cdot)$). Bellare [1] subsequently proved $\mathsf{HMAC}/\mathsf{NMAC}$ to be an arbitrary-input-length PRF under the sole assumption of the compression function being a PRF. We point out that the *cascade construction* by Bellare et al. [3] can also be seen as a way to key a hash function with a single key to obtain a PRF

[3] The security statements of [19] are asymptotic, but more concrete statements can be obtained by a closer inspection of the proofs.

[4] One can simply base the above constructions on the function $F' : (k_1 \| \ldots \| k_c, r) \mapsto F(k_1, r) \| \ldots \| F(k_c, r)$ (possibly chopping some bits) where $c = \lceil n/\kappa \rceil$ (the function $F'$ can be shown to be a WPRF). Note that more involved range-extension techniques (such as those from [11, 17, 20]) do not work here, as they require a length-preserving function beforehand.

[5] Informally, a MAC is a cryptographic function with a secret key which is "unpredictable" (also see below for a formal definition). While a PRF is a MAC, the latter notion is strictly weaker than the former.

[6] i.e. based on the Merkle-Damgård construction [10, 18], cf. also Section 2

[7] Practical implementations usually consider single-keyed versions which, for simplicity, are not discussed here.

under the same assumption, at the expense of using a prefix-free encoding of the inputs. More recently, Fischlin [12] presented security proofs for HMAC/NMAC (when used as a MAC rather than as a PRF) relying on non-malleability properties of the underlying compression function. A further recent line of research [15, 22] has been concerned with increasing the efficiency of the HMAC/NMAC constructions by imposing slightly stronger requirements on the underlying compression function (i.e. pseudorandomness under mild types of related-key attacks).

The bottom line is that in order to deploy one of these constructions in practice, it is relevant to assess the level of confidence one is willing to put in the given compression function, but in view of continuous cryptanalytic achievements this is far from being a simple task. This issue motivates us to take steps in the opposite direction: We raise the question of constructing iterated MACs (and PRFs) with *very low* requirements on the given compression function, while guaranteeing limited impact on the performance when compared with constructions with stronger underlying security assumptions. In particular, we consider constructions which only require the underlying compression function to be an $s$-WPRF (for $s$ as small as two).

### 1.4 Contributions and Outline of this Paper

This paper initiates the study of constant-query WPRFs, and in particular investigates the problem of constructing efficient PRFs from these primitives.

- In Section 3, we present our first construction (called the RC-construction) of an arbitrary-input-length PRF from any $s$-WPRF $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$ (for some constant $s \geq 2$). As a special case of our construction, one obtains a fixed-input-length PRF which, for input length $\ell$, requires $\approx \frac{\ell}{\log s}$ calls to $F$ per evaluation, hence improving on earlier constructions despite the weaker underlying assumption of an $s$-WPRF.
- Careful instantiation of the RC-construction yields an efficient symmetric encryption scheme relying on the sole assumption of an $s$-WPRF (for some $s \geq 2$), while requiring (on average) only $1 + \frac{1}{s-1}$ calls to $F$ per $\kappa$-bit block of encrypted data and minimal storage overhead. Furthermore, the RC-construction directly yields constructions of efficient PRGs from $s$-WPRFs.
- Section 4 presents a further construction, called the *nested* RC-*construction*, which improves the throughput of the RC-construction for long messages making a novel use of pairwise independence, while still solely relying on the underlying function being an $s$-WPRF.
- Finally, Section 5 addresses the problem of deriving our constructions by keying iterated hash functions (such as SHA-1 or MD5) whose compression function is an $s$-WPRF: If minimal (and natural) regularity properties are additionally guaranteed by the compression function, the keying can be done in an entirely black-box way. Furthermore, this is the weakest assumption on the compression function for which modes of operations leading to secure PRFs and MACs have ever been considered.

The basic tools needed in the rest of the paper are reviewed in Section 2, where the notion of a constant-query pseudorandom function is also formally defined.

## 2 Preliminaries

### 2.1 Notational Conventions

Throughout this paper, for a set $\mathcal{U}$, we denote as $\mathcal{U}^n$, $\mathcal{U}^*$, and $\mathcal{U}^+$ the sets of *sequences* $s = (u_1, u_2, \ldots, u_{|s|})$ of elements of $\mathcal{U}$ of length $|s| = n$, of arbitrary length with the empty sequence $\epsilon$, and of arbitrary length $|s|$ without the empty sequence $\epsilon$, respectively. (For the case $\mathcal{U} = \{0,1\}$ we usually talk of *strings*.) The notation $s \| s'$ stands for the concatenation of sequences $s$ and $s'$,

and $u^r$ is the sequence $(u, u, \ldots, u)$ consisting of $r$ repetitions of the symbol $u \in \mathcal{U}$. Given a two-argument function $F : \mathcal{U} \times \mathcal{V} \to \mathcal{Y}$ we denote by $F(u, \cdot)$ the function $\mathcal{V} \to \mathcal{Y}$ obtained by fixing the first input to $u$. Finally, $A^{\mathcal{O}}(r)$ denotes the (oracle) algorithm $A$ which runs on input $r$ with access to the oracle $\mathcal{O}$. Algorithms are in general randomized, and throughout this paper we fix some RAM model of computation for these algorithms. In particular, an algorithm $A$ is said to have *running time* $t$ if the sum of its description length and the worst-case number of steps it takes (counting oracle queries as single steps), taken over all randomness values, all inputs, and all compatible oracles, is at most $t$.

### 2.2 Cryptographic Functions

Pseudorandom Functions (PRFs). For some set $\mathcal{X}$ (generally $\mathcal{X} = \{0,1\}^\ell$ or $\mathcal{X} = \{0,1\}^*$) we consider *keyed* functions of the form $F : \{0,1\}^\kappa \times \{0,1\}^\rho \times \mathcal{X} \to \{0,1\}^n$, where the first and the second parameters are called the *public* and the *private* part of the *key*,[8] respectively. The third parameter is the *input* of $F$. We define the *PRF advantage* of $D$ in distinguishing $F$ from random as the quantity

$$\mathbf{Adv}_F^{\mathsf{PRF}}(D) := \left| \mathsf{P}\left[D^{F(K,R,\cdot)}(R) = 1\right] - \mathsf{P}\left[D^{\mathbf{R}_{\mathcal{X},n}}(R) = 1\right] \right|,$$

where $K$ and $R$ are independent and uniformly chosen from $\{0,1\}^\kappa$ and $\{0,1\}^\rho$, respectively, whereas $\mathbf{R}_{\mathcal{X},n}$ is a *random function* mapping elements of $\mathcal{X}$ to $n$-bit strings, i.e. an oracle which associates with each $x \in \mathcal{X}$ a uniformly-distributed independent $n$-bit string. (Whenever $\mathcal{X}$ is finite, this is equivalent to a randomly chosen function $\mathcal{X} \to \{0,1\}^n$.) For notational convenience we introduce the shorthand $\mathbf{Adv}_F^{\mathsf{PRF}}(t, q)$ to indicate the best advantage taken over all distinguishers with running time $t$ and making at most $q$ queries. Informally, $F$ is a PRF if $\mathbf{Adv}_F^{\mathsf{PRF}}(t, q)$ is "negligible" for all $t$ and $q$ polynomial in some (understood) security parameter.[9] We often consider the case $\mathcal{X} = \{0,1\}^*$: Such a PRF is called an *arbitrary-input-length PRF* (AIL-PRF), and for this case we define $\mathbf{Adv}_F^{\mathsf{PRF}}(t, q, \ell)$ as the maximal advantage taken over all distinguishers with running time $t$ making at most $q$ queries each of length at most $\ell$.

Weak Pseudorandom Functions (WPRFs). This notion weakens a PRF to only withstand attacks where the function is queried on independent random *known* inputs. (Sometimes, this is called a *known-plaintext attack* (KPA) in the literature.) Formally, for some function $g$, we let $\mathcal{S}^g$ be the oracle that returns an ordered pair $(r, g(r))$ for a fresh random $r$ each time it is invoked. Then, for a keyed function $F : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ we define the *WPRF advantage* of the distinguisher $D$ in distinguishing $F$ from random as

$$\mathbf{Adv}_F^{\mathsf{WPRF}}(D) := \left| \mathsf{P}[D^{\mathcal{S}^{F(K,\cdot)}} = 1] - \mathsf{P}[D^{\mathcal{S}^{\mathbf{R}_{m,n}}} = 1] \right|,$$

where $\mathbf{R}_{m,n}$ is a random function mapping $m$-bit strings to $n$-bit strings and $K$ is a random $\kappa$-bit secret key.[10] Additionally $\mathbf{Adv}_F^{\mathsf{WPRF}}(t, q)$ stands for the best advantage taken over all distinguishers with running time $t$ making at most $q$ queries. For a constant $s$, we call a function $F : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ with $\kappa < s \cdot n$ an *$s$-weak pseudorandom function* ($s$-WPRF) if $\mathbf{Adv}_F^{\mathsf{WPRF}}(t, s)$ is negligible for all polynomial running times $t$, and we simply call it a *weak pseudorandom function* (WPRF) if $\mathbf{Adv}_F^{\mathsf{WPRF}}(t, q)$ is negligible for all polynomially bounded $t$ and $q$.

[8] We take this unconventional point of view as the constructions of this paper will allow part of the key to be publicly revealed with no harm to their security, and there are settings where this is a useful feature.

[9] If one considers both parts of the key as a single secret key, this implies that $F$ is a PRF according to the usual definition considered in the literature.

[10] In contrast to the definitions of PRFs and MACs (below), here we only consider a fully-secret key.

MESSAGE AUTHENTICATION CODES (MACS). A keyed function $F : \{0,1\}^\kappa \times \{0,1\}^\rho \times \{0,1\}^* \to \{0,1\}^n$ is a MAC if it is "unpredictable" under a secret key. Formally, for an adversary $A$, we define its *MAC advantage* as

$$\mathbf{Adv}_F^{\mathsf{MAC}}(A) := \mathsf{P}[A^{F(K,R,\cdot)}(R) = (x,y) \wedge F(K,R,x) = y \wedge x \text{ new}],$$

where $K$ and $R$ are random independent $\kappa$- and $\rho$-bit strings, respectively, and "$x$ new" means that $x$ was not queried by $A$ to the given oracle. We define $\mathbf{Adv}_F^{\mathsf{MAC}}(t,q,\ell)$ to be the best advantage of an adversary with running time $t$ issuing at most $q-1$ queries to $F(K,R,\cdot)$, each of length at most $\ell$ (and the message $x$ output has also length at most $\ell$). It is a well-known fact that a secure AIL-PRF is also a good MAC, namely $\mathbf{Adv}_F^{\mathsf{MAC}}(t,q,\ell) \leq \mathbf{Adv}_F^{\mathsf{PRF}}(t',q,\ell) + \frac{1}{2^n}$, where $t \approx t'$.

CASCADE AND ITERATED HASH FUNCTIONS. For $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$, it is convenient to define its *cascade* $F^* : \{0,1\}^\kappa \times (\{0,1\}^n)^+ \to \{0,1\}^\kappa$ as the function which, on input $k \in \{0,1\}^\kappa$ and $(x_1,\ldots,x_\lambda) \in (\{0,1\}^n)^+$ (with $x_1,\ldots,x_\lambda \in \{0,1\}^n$) first computes $y_0 := k$ and $y_i = F(y_{i-1}, m_i)$ for all $i = 1,\ldots,\lambda$, and subsequently outputs $y_\lambda$. In this work we also consider *iterated hash functions* [18, 10] $H : \{0,1\}^* \to \{0,1\}^\kappa$ with underlying *compression function* $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$ ($n$ is generally called the *block length*) and *initialization value* $IV \in \{0,1\}^\kappa$ which are defined such that every input $x \in \{0,1\}^*$ is first padded as $(x_1,\ldots,x_\lambda) \in (\{0,1\}^n)^+$ and subsequently the value $F^*(IV,(x_1,\ldots,x_\lambda))$ is output. In general, the last block $x_\lambda$ contains some padding bits as well as the length of the message (the so-called *MD-strengthening*) to preserve collision resistance of the compression function. Examples of such functions are those from the MD and the SHA families.

UNIVERSAL HASHING. Let $H : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^n$, and let $\delta : \mathbb{N} \to \mathbb{R}^+$. We say that $H$ is *$\delta$-almost universal* ($\delta$-AU) if

$$\mathsf{p}_H^{\mathsf{COLL}}(x,x') := \mathsf{P}[H(K,x) = H(K,x')] \leq \delta(\max\{|x|,|x'|\})$$

for all distinct $x,x' \in \{0,1\}^*$, where $K$ is a randomly chosen $\kappa$-bit key. We stress that we extend the standard notion [9, 21] to deal with arbitrary input lengths by letting $\delta$ be a function of the message length.[11] The following lemma extends to the arbitrary-input-length case the well-known fact that $\delta$-AU hash functions can be used to extend the domain of PRFs. (We omit its proof which follows the lines of the fixed-input-length case.)

**Lemma 1.** *Let $H : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^m$ be $\delta$-AU, and let $F : \{0,1\}^\kappa \times \{0,1\}^\rho \times \{0,1\}^m \to \{0,1\}^n$ be a keyed function. Define $HF : \{0,1\}^{\kappa+\kappa'} \times \{0,1\}^\rho \times \{0,1\}^* \to \{0,1\}^n$ such that $HF(k\|k',r,x) := F(k',r,H(k,x))$. Then, for all distinguishers $D$ with running time $t$ making $q$ queries, each of length at most $\ell$, there exists a distinguisher $D'$ making $q$ queries such that*

$$\mathbf{Adv}_{HF}^{\mathsf{PRF}}(D) \leq \mathbf{Adv}_F^{\mathsf{PRF}}(D') + \tfrac{1}{2} \cdot q^2 \cdot \delta(\ell),$$

*and with running time $t' = t + q \cdot t_H(\ell)$, where $t_H(\ell)$ is the time needed to evaluate $H$ on inputs of length at most $\ell$.*

## 3 The Randomized Cascade Construction

### 3.1 Description and Security of the Construction

In this section, we present the first iterated construction of this paper. It is reminiscent of the cascade construction of Bellare et al. [3], but only requires the underlying function $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$ to be an $s$-WPRF with $s \geq 2$ being a parameter of the construction. As in [3], the construction relies on the concept of a prefix-free encoding, which we briefly introduce.

[11] Alternatively, one could use the notion of a cAU-hash function from [1], but this will not be necessary here.

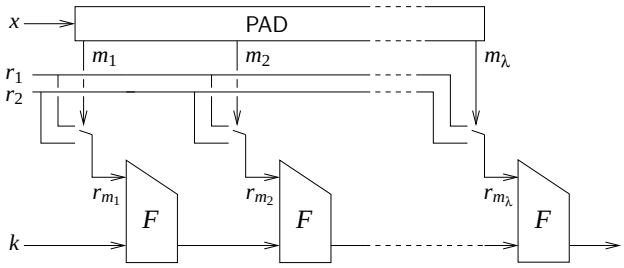


**Fig. 1.** The construction $\mathsf{RC}^F_{2,\mathsf{ENC}}$

PREFIX-FREE ENCODINGS. For a set $\mathcal{X}$, the efficiently computable function $\mathsf{ENC} : \mathcal{X} \to \{1,\ldots,s\}^+$ (i.e. outputting a non-empty sequence of elements of $\{1,\ldots,s\}$) is a *prefix-free encoding scheme* if for all distinct $x, x' \in \mathcal{X}$ the sequence $\mathsf{ENC}(x)$ is not a prefix of the sequence $\mathsf{ENC}(x')$. (In particular, $\mathsf{ENC}$ must be injective.) If $\mathcal{X} = \{0,1\}^*$, a prefix-free encoding scheme is e.g. obtained by encoding canonically the input as a sequence in $\{1,\ldots,s-1\}^*$, and then appending the symbol $s$ to the sequence. Other variants exist, but it is generally desirable that $\mathsf{ENC}$ operates *on-line*, i.e. the encoding is progressively output while the input bits are provided, without the need to know the entire input before starting the encoding process. If $\mathcal{X} = \{0,1\}^\ell$ for some fixed $\ell$, then prefix-freeness is achieved "for free" by encoding all inputs as sequences in $\{1,\ldots,s\}^*$ of equal length $\lceil \frac{\ell}{\log_2 s} \rceil$.

CONSTRUCTION. The *randomized cascade construction* with parameter $s$ and input set $\mathcal{X}$ (where usually either $\mathcal{X} = \{0,1\}^*$ or $\mathcal{X} = \{0,1\}^\ell$ for a fixed $\ell$) for the function $F$ and prefix-free encoding scheme $\mathsf{ENC}$, denoted $\mathsf{RC}^F_{s,\mathcal{X},\mathsf{ENC}}$, is a mapping $\{0,1\}^\kappa \times \{0,1\}^{sn} \times \mathcal{X} \to \{0,1\}^\kappa$: It takes a key consisting of a $\kappa$-bit private part $k$ and an $sn$-bit long public part, which is interpreted as the concatenation of $s$ $n$-bit strings $r_1,\ldots,r_s$. On input $x \in \mathcal{X}$, the $\kappa$-bit output is computed through the following two steps:

1. Compute $\mathsf{ENC}(x) = (m_1,\ldots,m_\lambda) \in \{1,\ldots,s\}^+$;
2. Output $F^*(k,(r_{m_1},\ldots,r_{m_\lambda}))$.

As an example, the construction is depicted in Figure 1 for the special case $s = 2$. For notational convenience, we use the shorthands $\mathsf{RC}^F_{s,\mathsf{ENC}}$ for $\mathcal{X} = \{0,1\}^*$ (and omit the prefix-free encoding when it is generally understood from the context), as well as $\mathsf{RC}^F_{s,\ell}$ for $\mathcal{X} = \{0,1\}^\ell$ (where the canonical encoding described above is used). We also generically refer to the construction as the $\mathsf{RC}$-construction.

EFFICIENCY COMPARISONS. A fair comparison between the $\mathsf{RC}$-construction and previous results can be undertaken for the fixed-input-length construction $\mathsf{RC}_{s,\ell}$ only. In the length-preserving case ($\kappa = n$), the construction $\mathsf{RC}_{\ell,s}$ is comparable to (for the case $s = 2$) the $\mathsf{NR}$- and the $\mathsf{IC}$-constructions in terms of calls to $F$, and outperforms them for $s > 2$. Furthermore, we obtain the same space-time trade-off of the $\overline{\mathsf{NR}}_{s,\ell}$-construction, but we allow for all possible values of $s$. Our construction also limits the effects of possibly very long input paddings in the $\mathsf{NR}$- and $\overline{\mathsf{NR}}$-constructions. The efficiency improvement of our construction is however more evident in the case where $n > \kappa$, as even if $s = 2$, the number of calls to $F$ of (the extended versions of) all other constructions is larger at least by a factor $\lceil \frac{n}{\kappa} \rceil$ (the factor is e.g. 4 when instantiating $F$ with the compression function of SHA-1). Finally, because of the iterated structure, efficient sequential evaluation of $\mathsf{RC}_{s,\ell}$ requires (beside sufficient storage for the key material) $\kappa$ bits only to store the "chaining value".

SECURITY. In order to give precise security bounds for the RC-construction, it is convenient to think of the prefix-free encoding ENC in terms of a (possibly infinite) directed tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ consisting of all sequences $(m_1, \ldots, m_j)$ which are a prefix of $\mathsf{ENC}(x)$ for some input $x$ (in particular, including the encodings themselves and the empty sequence $\epsilon$). Furthermore, for each $(m_1, \ldots, m_j) \in \mathcal{V}$ there exists a directed edge to $(m_1, \ldots, m_j, m_{j+1})$ for all $m_{j+1} \in \{1, \ldots, s\}$ such that $(m_1, \ldots, m_{j+1}) \in \mathcal{V}$. Hence, it is easy to see that $\epsilon$ is the root of the directed tree and its leaves are exactly the encodings of the inputs. We provide two examples of such trees in Figure 2.

Every sequence of queries to the RC-construction defines a subtree of $\mathcal{T}$ consisting of the paths from the root to the encodings of the queries: For notational convenience, we define the shorthand $L(x_1, \ldots, x_q)$, for $q$ inputs $x_1, \ldots, x_q$, to be the amount of inner vertices (i.e. vertices which are not leaves) of the sub-tree induced by the evaluations of $x_1, \ldots, x_q$. It is easy to verify that for $\mathsf{RC}_{s,\ell}$ we have $L(x_1, \ldots, x_q) \leq 1 + q(\lceil \frac{\ell}{\log s} \rceil - 1)$. Also, for the case where the inputs are strings with arbitrary length, we define (always with respect to the understood encoding) $L(q, \ell) := \max_{x_1,\ldots,x_q : |x_i| \leq \ell} L(x_1, \ldots, x_q)$.

Consequently, one can see an interaction with the RC-construction as a process where the tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ defined by ENC is traversed and $\kappa$-bit values are assigned to all visited vertices: While the root $\epsilon$ is assigned a random $\kappa$-bit value, the value of each visited vertex $(m_1, \ldots, m_j)$ is set to $F(z, r_{m_j})$, with $z$ being the value of the parent vertex $(m_1, \ldots, m_{j-1})$. A query with input $x$ is answered with the value at the corresponding leaf $\mathsf{ENC}(x)$. By the definition of an $s$-WPRF, it is easy to see that evaluating $F$ under some given (pseudo-)random secret key at $s$ independent random inputs produces $s$ pseudorandom outputs,[12] and hence intuitively the above process sets the values of all visited vertices to pseudorandom values (and in particular this holds for the leaves). However, to formalize this intuition, we have to show that it is indeed possible to recycle the same values $r_1, \ldots, r_s$ for each invocation of $F$.

The following theorem formally captures the main security statement for the RC-construction (for a general input set $\mathcal{X}$). Its proof can be found in Appendix A.2.

**Theorem 1.** *Let $s \geq 2$, let $\mathcal{X}$ be a set, and let $\mathsf{ENC} : \mathcal{X} \to \{1, \ldots, s\}^+$ be a prefix-free encoding scheme. Furthermore, let $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$. For all $L$ and all distinguishers $D$ with running time $t$ and with $L(x_1, x_2, \ldots) \leq L$ for all possible query sequences $x_1, x_2, \ldots \in \mathcal{X}$, there exists a distinguisher $D' = D'(D)$ such that*

$$\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{RC}^F_{s,\mathcal{X},\mathsf{ENC}}}(D) \leq L \cdot \left[ \mathbf{Adv}^{\mathsf{WPRF}}_F(D') + s^2 \cdot 2^{-(n+1)} \right],$$

*where $D'$ makes exactly $s$ queries and has running time $t' = t + \mathcal{O}(L \cdot t_F)$, with $t_F$ being the time needed to evaluate $F$.*

We remark that the term $s^2 2^{-(n+1)}$ is negligible, as $s$ is assumed to be constant. Combined with the above observations on $L$, the theorem directly yields the following security bounds for the specialized variants of the RC-construction:

$$\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{RC}^F_s}(t, q, \ell) \leq L(q, \ell) \cdot \left[ \mathbf{Adv}^{\mathsf{WPRF}}_F(t', s) + s^2 \cdot 2^{-(n+1)} \right],$$
$$\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{RC}^F_{s,\ell}}(t, q) \leq \left[ 1 + q \left( \left\lceil \frac{\ell}{\log s} \right\rceil - 1 \right) \right] \cdot \left[ \mathbf{Adv}^{\mathsf{WPRF}}_F(t'', s) + s^2 \cdot 2^{-(n+1)} \right],$$

with $t' = t + \mathcal{O}(L(q, \ell) \cdot t_F)$ and $t'' = t + \mathcal{O}((1 + q\,(\lceil \ell / \log s \rceil - 1)) \cdot t_F)$.

The most important observation is that all variants of the RC-construction require $F$ to be only an $s$-WPRF. A minor positive aspect of the randomized cascade construction (if compared

[12] Except in the case where two of the random inputs $r_1, \ldots, r_s$ collide, which happens with small probability only.

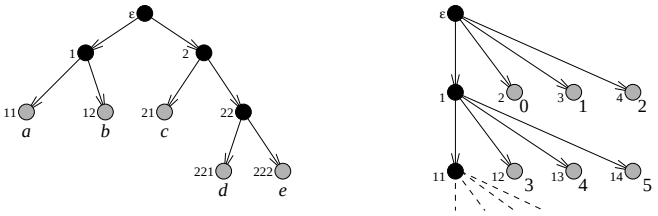


**Fig. 2.** Example trees associated with prefix-free encodings. Left: Encoding mapping inputs $a$, $b$, $c$, $d$, and $e$ to sequences $(1,1)$, $(1,2)$, $(2,1)$, $(2,2,1)$, and $(2,2,2)$, respectively. Right: Encoding $\mathsf{CTRENC}$ used for efficient counter-mode evaluation.

with other constructions) is the absence of any $q$-dependent birthday-like term in the above inequalities. Furthermore, if we assume that $F$ is indeed secure against $q$ queries, the security of the $\mathsf{RC}_{s,\ell}$-construction is comparable to the one of the $\mathsf{IC}_\ell$-construction if we assume (in fact, very optimistically) that the best WPRF-distinguishing advantage grows linearly in the number of queries, i.e. $\mathbf{Adv}_F^{\mathsf{WPRF}}(t,q) = \Theta(q \cdot \mathbf{Adv}_F^{\mathsf{WPRF}}(t,s))$.

Larger Output Sizes. It is easy to increase the output size of the $\mathsf{RC}$-construction (if needed) with the addition of a minor number of invocations of $F$ per evaluation, which is independent of the input length: To obtain a construction $\overline{\mathsf{RC}}^F : \{0,1\}^\kappa \times \{0,1\}^{ns} \times \mathcal{X} \to \{0,1\}^{\phi\kappa}$ with output size $\phi \cdot \kappa$, we fix $\phi$ distinct strings $a_1, \ldots, a_\phi \in \mathcal{X}$ such that $L(a_1, \ldots, a_\phi)$ is minimal. Then, given key with private part $k$ and public part $r_1, \ldots, r_s$, on input $x \in \mathcal{X}$, to compute $\overline{\mathsf{RC}}^F(k, r_1 \| \ldots \| r_s, x)$ we first compute $k' := \mathsf{RC}^F(k, r_1 \| \ldots \| r_s, x)$ and finally output $\mathsf{RC}^F(k', r_1 \| \ldots \| r_s, a_1) \| \ldots \| \mathsf{RC}^F(k', r_1 \| \ldots \| r_s, a_\phi)$. Security of this construction can be inferred by the fact that evaluating it at input $x$ accounts to evaluating at inputs $(x, a_1), \ldots, (x, a_\phi)$ a variant of the $\mathsf{RC}$-construction with input set $\mathcal{X} \times \{a_1, \ldots, a_\phi\}$ and the prefix-free encoding $\mathsf{ENC}'(x, a) := \mathsf{ENC}(x) \| \mathsf{ENC}(a)$.

### 3.2 Efficient Encryption and PRGs from the RC-Construction

This section addresses two important applications of the $\mathsf{RC}$-construction. Firstly, we show that careful instantiation of the used prefix-free encoding yields a symmetric-encryption scheme from an $s$-WPRF which is (already for minimal values of $s$) nearly as efficient as PRF-based encryption schemes (such as counter-mode and CBC-encryption). This result shows the feasibility of very efficient symmetric cryptography from very weak assumptions. Furthermore, we show that the $\mathsf{RC}$-construction yields PRG constructions from constant-query weak PRFs which improve on previous results. We omit the proofs of the technical claims (which are mostly corollaries of Theorem 1 or are based on standard techniques).

Symmetric Encryption from the RC-Construction. Given a PRF $F : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ (in practice usually realized by a block cipher) one obtains an efficient stateful IND-CPA[13] encryption scheme for arbitrary-length messages by using $F$ in so-called *counter-mode*, i.e. given a secret key $k$, we keep a counter $\mathsf{ctr}$ (initially 0), and the encryption of a plaintext $x$ (which we assume for simplicity to have length $|x|$ multiple of $n$) is the ciphertext $[\mathsf{ctr}, x \oplus (F(k,\mathsf{ctr}) \| F(k,\mathsf{ctr}+1) \| \ldots \| F(k,\mathsf{ctr}+|x|/n-1))]$ (and $\mathsf{ctr}$ is increased by $|x|/n$), where integers are canonically mapped to $m$-bit strings. Note in particular that we need one call to $F$

[13] Informally, a (stateful or randomized) encryption scheme $(E, D)$ is *IND-CPA secure* [4, 16] if for a secret key $K$ no polynomial-time adversary can distinguish the encryptions $E(K, x_0)$ and $E(K, x_1)$ for any two equally long messages $x_0, x_1$ of its choice even if it can obtain adaptively chosen encryptions $E(K, x)$ for arbitrary $x$'s.

for each $n$-bit block of encrypted data. Variants of *randomized stateless* counter-mode encryption (where one chooses a fresh random counter at every encryption instead of keeping a state) based on any WPRF $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ were presented in [11, 17]. As with a full PRF, these schemes only require one call per $n$-bit block of encrypted data, but the underlying WPRF must be secure against as many queries as the amount of encrypted message blocks.

One can substantially weaken the assumption to an $s$-WPRF by using the $\mathsf{RC}$-construction in stateful counter mode (with any encoding scheme). However, a dramatic increase of efficiency is achieved using a prefix-free encoding scheme $\mathsf{CTRENC} : \mathbb{N} \to \{1, \ldots, s\}^+$ tailored at this mode of operation, defined as

$$\mathsf{CTRENC}(i) := 1^{i \operatorname{div} s-1} \| (2 + (i \bmod s - 1)).$$

The tree arising from this encoding scheme is illustrated in Figure 2: In particular, it is clear that the sequence of values $\mathsf{RC}^F_{s,\mathsf{CTRENC}}(0), \mathsf{RC}^F_{s,\mathsf{CTRENC}}(1), \ldots$ can be computed very efficiently in an iterated way using only $\kappa + sn$ bits of memory and needing approximately $1 + \frac{1}{s-1}$ calls to $F$ per $\kappa$-bit block of encrypted data. Furthermore, the values $r_1, \ldots, r_s$ can be chosen publicly by one communicating party (provided an authenticated channel is available), hence reducing the cost of key establishment to the generation of the $\kappa$-bit private part of the key. Security against (adaptive) chosen-ciphertext attacks based on any $s$-WPRF can be then obtained by standard techniques appending a MAC of the ciphertext [7] (e.g. using any of the PRF constructions presented in this paper).

Pseudorandom Generators from $s$-WPRFs. Recall that a *pseudorandom generator* (PRG) is a length-expanding function $G : \{0,1\}^\kappa \to \{0,1\}^m$ such that $G(K)$ is computationally indistinguishable from a random $m$-bit string under a random $K$. Surprisingly, constructing a good PRG from a WPRF (or an $s$-WPRF) turns out not to be a straightforward task: In contrast to PRFs, a WPRF $F$ does not generally allow to find few "good" inputs $x_1, \ldots, x_t$ such that the mapping $k \mapsto F(k, x_1) \| \ldots \| F(k, x_t)$ is a PRG. However, one can use this approach employing the $\mathsf{RC}$-construction as the underlying PRF: For any $t$ *fixed* inputs $x_1, \ldots, x_t$ ($t > 2$) the mapping $\mathsf{G}^F : \{0,1\}^{sn+\kappa} \to \{0,1\}^{sn+t\kappa}$ such that $\mathsf{G}^F(r_1, \ldots, r_s, k)$ equals

$$r_1 \| \cdots \| r_s \| \mathsf{RC}^F_s(k, r_1 \| \ldots \| r_s, x_1) \| \cdots \| \mathsf{RC}^F_s(k, r_1 \| \ldots \| r_s, x_t)$$

is a PRG if $F$ is an $s$-WPRF. (The order of the strings in the concatenation is irrelevant.) Note that an important advantage is that the strings $r_1, \ldots, r_s$ can be output as well. For example, given a 2-WPRF $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, the mapping $\overline{\mathsf{G}}^F : \{0,1\}^{3n} \to \{0,1\}^{6n}$ such that $\overline{\mathsf{G}}^F(k, r_0, r_1)$ is set to

$$r_0 \| F(F(k, r_0), r_0) \| F(F(k, r_0), r_1) \| F(F(k, r_1), r_0) \| F(F(k, r_1), r_1) \| r_1 \tag{1}$$

is a length-doubling PRG which requires 6 calls to $F$. In particular, 3 calls are necessary in order to input only one both halves of the output. This improves a construction given in [17], which needed 3 and 4 calls, respectively.

An alternative approach to building a PRF from an $s$-WPRF $F$ would consist of first constructing a length-doubling PRG $G$ from $F$, and subsequently using the well-known $\mathsf{GGM}$-construction [13] to build a PRF with a $\kappa$-bit key and $\ell$-bit inputs by outputting, on input $x = (x_1, \ldots, x_{\ell-1}, x_\ell) \in \{0,1\}^\ell$ and key $k$, the $\kappa$-bit value $G_{x_\ell}(G_{x_{\ell-1}}(\cdots G_{x_1}(k) \cdots))$, where $G_i(k)$ for $i = 0, 1$ gives the first and the second half of the output of $G$, respectively. However, it is not hard to see that all constructions following this approach turn out to be less efficient than using the $\mathsf{RC}$-construction directly (e.g. using the PRG of Equation 1 one needs 3 calls of $F$ per input bit).

## 4 The Nested Randomized Cascade Construction

Even though the RC-construction can be practically efficient in special instantiation scenarios discussed earlier, its throughput is a major bottleneck in the case where the construction is used as a PRF (or a MAC) which is invoked at arbitrary inputs with variable lengths. Furthermore, the prefix-free encoding can be a limiting factor in the arbitrary-input-length case. This section presents a construction with better efficiency for long messages (i.e. longer than $\kappa$ bits) and with no prefix-freeness requirements. Its core ingredient is a novel use of pairwise independence.

Pairwise-Independent Mappings. Recall that a mapping[14] $M : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ is *pairwise independent* if the values $M(K,x)$ and $M(K,x')$ are independent and uniformly distributed for all distinct $x, x' \in \{0,1\}^m$ under a random $\kappa$-bit key $K$. Most pairwise-independent mappings satisfy the following property, which will be central in our construction.

**Definition 1.** *A pairwise-independent mapping $M : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ is* key programmable *if there exists a (possibly randomized) algorithm* SAMPLE *which on input $(x, x', y, y')$ (where possibly $x = x'$, $y = y'$) returns a uniformly chosen element from the set $\{k \mid M(k,x) = y, M(k,x') = y'\}$.*

If $M$ is key programmable, the following two random experiments are equivalent to sampling a random $\kappa$-bit key $K$: (i) For some $m$-bit string $x$, sample $Y$ as a uniform random $n$-bit string and $K := \mathsf{SAMPLE}(x, x, Y, Y)$; and (ii) For $n$-bit strings $x \neq x'$, sample $Y, Y'$ as independent random $n$-bit strings and $K := \mathsf{SAMPLE}(x, x', Y, Y')$. Both the last two sampling strategies are used to ensure that $M(K,x) = Y$ (and possibly $M(K,x') = Y'$) for values $Y, Y' \in \{0,1\}^n$ which, although uniform and independent, are provided externally.

We provide two examples of key-programmable pairwise-independent mappings.

*Example 1.* Let $M$ be such that given $k_1, k_2 \in \{0,1\}^n$ and the input $x \in \{0,1\}^n$, the output $M(k_1 \| k_2, x)$ equals $k_1 \oplus (k_2 \odot x)$, where $\oplus$ and $\odot$ are addition and multiplication of $n$-bit strings interpreted as elements of the extension field $GF(2^n)$. The unique $k_1 \| k_2$ such that $M(k_1 \| k_2, x) = y$ and $M(k_1 \| k_2, x') = y'$ (with $x \neq x'$) can efficiently be found solving the corresponding system of two equalities. Is only a single constraint $M(k_1 \| k_2, x) = y$ given, one chooses a random $n$-bit string $k_2$ and sets $k_1 := (k_2 \odot x) \oplus y$.

*Example 2.* An alternative is the mapping $M'$ whose $(nm+n)$-bit key consists of an $(m \times n)$-binary matrix $\mathbf{A}$ and of a $n$-dimensional binary column vector $\mathbf{b}$, and on input $x$ the output is $\mathbf{A}x + \mathbf{b}$, where $x$ is interpreted as an $m$-dimensional column vector, and addition and multiplications are modulo 2. The function $M'$ needs a larger key than $M$ described above, but avoids finite-field multiplications.

Construction. The main idea of the nested RC-construction (called NRC, for short) is to combine an iterated phase where blocks are processed at a higher rate (but which satisfies a property weaker than pseudorandomness) with a second phase where the $\mathsf{RC}_{s,\kappa}$-construction (for fixed input length $\kappa$ and a parameter $s$) is invoked on the output of the first phase (with independent key material).

More precisely, let $M : \{0,1\}^{\kappa'} \times \{0,1\}^m \to \{0,1\}^n$ be a key-programmable pairwise-independent mapping and let $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$ be the given compression function. The construction $\mathsf{PI}_M^F : \{0,1\}^{\kappa+\kappa'} \times \{0,1\}^* \to \{0,1\}^\kappa$ takes a key $k \| k'$, where $k \in \{0,1\}^\kappa$ and $k' \in \{0,1\}^{\kappa'}$. On input $x \in \{0,1\}^*$, it pads[15] $x$ as $(x_1, \ldots, x_\lambda)$, where $x_1, \ldots, x_\lambda \in \{0,1\}^m$, and outputs $F^*(k, (M(k', x_1), \ldots, M(k', x_\lambda)))$.

[14] We use the word mapping, rather than hash function, to stress the fact that $m = n$ may also hold.

[15] According to the canonical padding which pads a string $x$ to have length being a multiple of $m$ by appending a 1 and sufficiently many 0's: The resulting padded string consists hence of $\left\lceil \frac{|x|+1}{m} \right\rceil$ $m$-bit blocks.

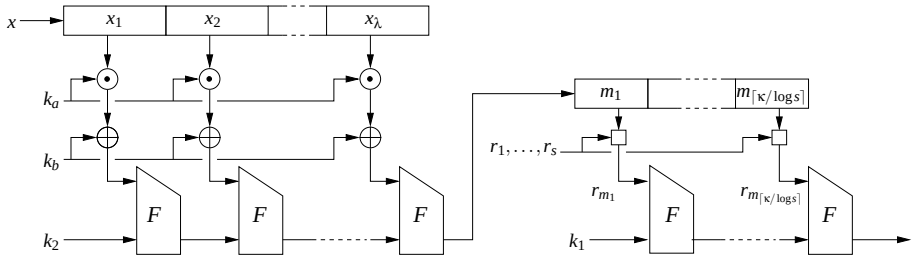


**Fig. 3.** The construction $\mathsf{NRC}^F_{M,s}$ for the special case $M(k_a\|k_b, x) = (k_a \odot x) \oplus k_b$.

Moreover, given the additional parameter $s$, we define the nested construction $\mathsf{NRC}^F_{M,s} : \{0,1\}^{2\kappa+\kappa'} \times \{0,1\}^{sn} \times \{0,1\}^* \to \{0,1\}^\kappa$ such that

$$\mathsf{NRC}^F_{M,s}(k_1\|k_2\|k', r_1\| \ldots \|r_s, x) := \mathsf{RC}^F_{s,\kappa}(k_1, r_1\| \ldots \|r_s, \mathsf{PI}^F_M(k_2\|k', x)).$$

It is easy to verify that in order to process a message $x$, the construction needs totally $\left\lceil \frac{|x|+1}{m} \right\rceil + \lceil \frac{\kappa}{\log s} \rceil$ calls to the underlying function $F$.

It is tempting to increase the throughput of the construction by choosing a mapping $M$ with $m$ much larger than $n$. However, all known constructions of pairwise-independent hash functions (in particular key-programmable ones) require keys twice as long as the *input* (rather than the output), and hence such an approach would entail a much longer key. In fact, we believe the length-preserving mapping $M$ presented above to be a viable practically efficient solution: This special case of the construction is depicted in Figure 3.

Security. The following theorem precisely quantifies the security of the NRC-construction.

**Theorem 2.** *Let $M : \{0,1\}^{\kappa'} \times \{0,1\}^m \to \{0,1\}^n$ be a key-programmable pairwise-independent mapping, and $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$. For all $s \geq 2$ and for all distinguishers $D$ with running time $t$ making $q$ queries, each of length at most $\ell$, there exist distinguishers $D'$ and $D''$ such that*

$$\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{NRC}^F_{M,s}}(D) \leq \left(1 + q\left(\lceil \tfrac{\kappa}{\log s} \rceil - 1\right)\right) \cdot \left(\mathbf{Adv}^{\mathsf{WPRF}}_F(D') + s^2 \cdot 2^{-(n+1)}\right) + \lceil \tfrac{\ell+1}{m} \rceil \cdot q^2 \cdot \left(\mathbf{Adv}^{\mathsf{WPRF}}_F(D'') + 2^{-n}\right) + q^2 \cdot 2^{-(\kappa+1)},$$

*where $D'$ makes $s$ queries and has running time $t' = t + \mathcal{O}(q(\frac{\ell}{m} + \frac{\kappa}{\log s}) \cdot t_F)$, whereas $D''$ makes two queries and has running time $t'' = \mathcal{O}\left(\frac{2\ell}{m} \cdot t_F\right)$. (Here $t_F$ denotes the time needed for an evaluation of $F$.)*

The core of the proof consists of showing that whenever $F$ is a WPRF for two-query adversaries, the PI-construction is $\delta$-AU for a suitable function $\delta$ to be computed below. In the following, given two inputs $x, x'$ with corresponding padded strings $(x_1, \ldots, x_\lambda)$ and $(x'_1, \ldots, x'_{\lambda'})$ (where without loss of generality $\lambda < \lambda'$), let $\lambda^*$ be maximal with the property that $x_1 = x'_1, \ldots, x_{\lambda^*} = x'_{\lambda^*}$ (in particular, $\lambda^* := 0$ if $x_1 \neq x'_1$), and define the quantity $\Lambda(x, x')$ as $\lambda + \lambda' - \lambda^* - 1$ if $(x_1, \ldots, x_\lambda)$ is not a prefix of $(x'_1, \ldots, x'_{\lambda'})$, and as $\lambda + 1$ otherwise. In particular, note that $\Lambda(x, x') \leq \lambda + \lambda' \leq 2\max\{\lambda, \lambda'\} \leq 2\lceil \frac{\ell+1}{m} \rceil$ if $|x|, |x'| \leq \ell$.

The following lemma provides a precise upper bound on the collision probability of the PI-construction in terms of the WPRF distinguishing advantage of a distinguisher $D_{x,x'}$ (which in particular only depends on $x$ and $x'$) for $F$. Its proof is deferred to Appendix A.3.

**Lemma 2.** *For all distinct inputs $x, x' \in \{0,1\}^*$, there exists a two-query distinguisher $D_{x,x'}$ such that*

$$\mathsf{p}^{\mathsf{COLL}}_{\mathsf{PI}^F_M}(x,x') \leq \Lambda(x,x') \cdot \left(\mathbf{Adv}^{\mathsf{WPRF}}_F(D_{x,x'}) + 2^{-n}\right) + 2^{-\kappa},$$

*where $D_{x,x'}$ has running time $\mathcal{O}\left(\Lambda(x,x') \cdot t_F\right)$.*

In particular, given some $\ell$, let $x, x'$ be strings with $|x|, |x'| \leq \ell$ maximizing $\mathbf{Adv}^{\mathsf{WPRF}}_F(D_{x,x'})$, and set $D'' := D_{x,x'}$. Then $D''$ has running time $t'' = \mathcal{O}\left(\frac{2\ell}{m} \cdot t_F\right)$. We define $\delta(\ell) := 2\left\lceil\frac{\ell+1}{m}\right\rceil \cdot (\mathbf{Adv}^{\mathsf{WPRF}}_F(D'') + 2^{-n}) + 2^{-\kappa}$. The function $\mathsf{PI}^F_M$ is $\delta$-universal by Lemma 2, and this implies Theorem 2 using Lemma 1 and Theorem 1.

## 5 Black-Box Keying of Iterated Hash Functions

The iterated structure of the RC- and the NRC-constructions makes compression functions ideal candidates for instantiating the underlying $s$-WPRF. In general, however, we may be constrained to only have black-box access to an implementation of an iterated hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$ (cf. Section 2) with direct access neither to the initialization value $IV$ nor to the underlying compression function $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$. To overcome this obstacle, we encode (as in HMAC) an $n$-bit key as the first block of the input to the hash function $H$. More precisely, given the prefix-free encoding scheme $\mathsf{ENC} : \{0,1\}^* \to \{1,\ldots,s\}^+$, we consider the construction $\mathsf{HRC}^F_{s,\mathsf{ENC}}$ which takes a key with private part $k \in \{0,1\}^n$ and public part $r_1, \ldots, r_s \in \{0,1\}^n$, and on input $x$ with $\mathsf{ENC}(x) = (m_1, \ldots, m_\lambda)$ outputs the value

$$\mathsf{HRC}^H_{s,\mathsf{ENC}}(k, r_1\| \ldots \|r_s, x) := H(k\|r_{m_1}\| \ldots \|r_{m_\lambda}),$$

and analogously we define $\mathsf{HRC}_{s,\ell}$ for inputs of fixed-length $\ell$ (using the canonical encoding to the base $s$). Furthermore, with $M : \{0,1\}^{\kappa'} \times \{0,1\}^m \to \{0,1\}^n$ being a key-programmable pairwise-independent mapping, we consider the construction $\mathsf{HNRC}^H_{M,s}$ which takes a key with private part $k_1, k_2 \in \{0,1\}^n$, $k' \in \{0,1\}^{\kappa'}$ and public parts $r_1, \ldots, r_s$. On input input $x$ (padded as $(x_1, \ldots, x_\lambda)$) it outputs

$$\mathsf{HNRC}^H_{M,s}(k_1\|k_2\|k', r_1\| \ldots \|r_s, x) := \\ \mathsf{HRC}^H_{s,\kappa}(k_1, r_1\| \ldots \|r_s, H(k_2\|M(k', x_1)\| \ldots \|M(k', x_\lambda))).$$

In order to lift the security statements of the RC- and the NRC-constructions to both the HRC- and HNRC-constructions, the assumption that $F$ is an $s$-WPRF is not sufficient: First, it is necessary that the $\kappa$-bit output $F(IV, K)$ is computationally indistinguishable from a uniformly-distributed random string of length $\kappa$ (under a secret random $K$); This guarantees that the chaining value obtained after the first evaluation of $F$ is pseudorandom and can be used as the "key" for the RC- or the PI-construction. A further problem is due to the fact that we generally cannot enforce the last $n$-bit block processed by $F$ to be random because of the padding introduced by $H$, and this issue should not destroy the pseudorandomness of the outputs. To our rescue, however, comes the fact that each such block is processed keying $F$ with a *fresh* pseudorandom value: It is hence enough to additionally guarantee that for an arbitrary *fixed* $n$-bit string $x$ and a random secret $\kappa$-bit string $K$, the string $F(K, x)$ is computationally indistinguishable from a random $\kappa$-bit string.

We stress that both these extra properties are very weak requirements: In fact, a good compression function should satisfy them even unconditionally. It is sufficient, for example, that $F(IV, \cdot)$ and $F(\cdot, x)$ (for all $x \in \{0,1\}^n$) are all (nearly-)regular functions. (We refer the reader to [6] for a discussion on regularity-properties of hash functions.). With these two additional assumptions on the compression function $F$ of $H$, the security bounds of the RC and the NRC-construction can be lifted to their black-box counterparts. We omit the proofs (which are very similar to the ones of the original constructions) from the current version of this paper.

## 6 Conclusions and Open Problems

We have shown that efficient arbitrary-input-length PRFs (and consequently MACs and encryption schemes) can be constructed under very weak assumptions, i.e. weak PRFs where security holds only for a limited number of queries. Our results provide new insights into the property of weak pseudorandomness and show the feasibility of basing efficient symmetric cryptography on very weak assumptions.

A natural open question is whether there exist constructions of PRFs from WPRFs which take explicit advantage of more secure WPRFs (i.e. tolerating many queries) to achieve more efficient constructions than what we propose and what was considered in the literature (e.g. processing linearly-many bits per invocation even for short inputs). We conjecture, however, that this is not possible. A further direction arising from our work consists of finding further examples of cryptographic primitives where restricting adversaries in terms of queries leads to interesting phenomena such as those observed in this paper for weak pseudorandomness.

## References


1. M. Bellare, "New proofs for NMAC and HMAC: Security without collision-resistance," in *CRYPTO 2006*, vol. 4117 of *LNCS*, pp. 602–619, 2006.
2. M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *CRYPTO '96*, vol. 1109 of *LNCS*, pp. 1–15, 1996.
3. M. Bellare, R. Canetti, and H. Krawczyk, "Pseudorandom functions revisited: The cascade construction and its concrete security," in *FOCS '96*, pp. 514–523, 1996.
4. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," in *FOCS '97*, pp. 394–403, 1997.
5. M. Bellare, J. Kilian, and P. Rogaway, "The security of the cipher block chaining message authentication code," *Journal of Computer and System Sciences*, vol. 61, no. 3, pp. 362–399, 2000.
6. M. Bellare and T. Kohno, "Hash function balance and its impact on birthday attacks," in *EUROCRYPT 2004*, vol. 3027 of *LNCS*, pp. 401–418, 2004.
7. M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *ASIACRYPT 2000*, vol. 1976 of *LNCS*, pp. 531–545, 2000.
8. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast and secure message authentication," in *CRYPTO '99*, vol. 1666 of *LNCS*, pp. 216–233, 1999.
9. J. L. Carter and M. N. Wegman, "Universal classes of hash functions," *Journal of Computer and System Sciences*, vol. 18, no. 2, pp. 143–154, 1979.
10. I. B. Damgård, "A design principle for hash functions," in *CRYPTO '89*, vol. 435 of *LNCS*, pp. 416–427, 1989.
11. I. B. Damgård and J. B. Nielsen, "Expanding pseudorandom functions; or: From known-plaintext security to chosen-plaintext security," in *CRYPTO 2002*, vol. 2442 of *LNCS*, pp. 449–464, 2002.
12. M. Fischlin, "Security of NMAC and HMAC based on non-malleability," in *CT-RSA 2008*, vol. 4964 of *LNCS*, pp. 138–154, 2008.
13. O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," in *FOCS '84*, pp. 464–479, 1984.
14. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1364–1396, 1999.
15. S. Hirose, J. H. Park, and A. Yun, "A simple variant of the Merkle-Damgård scheme with a permutation," in *ASIACRYPT 2007*, vol. 4833 of *LNCS*, pp. 113–129, 2007.
16. J. Katz and M. Yung, "Complete characterization of security notions for probabilistic private-key encryption," in *STOC 2000*, pp. 245–254, 2000.
17. U. Maurer and J. Sjödin, "A fast and key-efficient reduction of chosen-ciphertext to known-plaintext security," in *EUROCRYPT 2007*, vol. 4515 of *LNCS*, pp. 498–516, 2007.
18. R. C. Merkle, "A certified digital signature," in *CRYPTO '89*, vol. 435 of *LNCS*, pp. 218–238, 1989.
19. M. Naor and O. Reingold, "Synthesizers and their application to the parallel construction of pseudo-random functions," *Journal of Computer and System Sciences*, vol. 58, no. 2, pp. 336–375, 1999.
20. K. Pietrzak and J. Sjödin, "Range extension for weak PRFs; the good, the bad, and the ugly," in *EUROCRYPT 2007*, vol. 4515 of *LNCS*, pp. 517–533, 2007.
21. D. R. Stinson, "Universal hashing and authentication codes," in *CRYPTO '91*, vol. 576 of *LNCS*, pp. 74–85, 1991.
22. K. Yasuda, "Boosting Merkle-Damgård hashing for message authentication," in *ASIACRYPT 2007*, vol. 4833 of *LNCS*, pp. 216–231, 2007.

# A Security Proofs

## A.1 Random Beacons

In our security proofs, we make repeated use of a special oracle, called an *$\kappa$-bit beacon* $\mathbf{B}_\kappa$, which on each invocation ignores its input and returns a fresh random $\kappa$-bit string. We are going to make repeated use of the following lemma, which relates the WPRF-advantage of a distinguisher $D$ to its advantage when distinguishing $F(K,\cdot)$ (with random secret $K$) from a $\kappa$-bit beacon under $\mathcal{S}$ (which we denote from now on as $\mathbf{Adv}_F^{\mathsf{WPRB}}(D)$).

**Lemma 3.** *For $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\kappa$, a $\kappa$-bit beacon $\mathbf{B}_\kappa$, a $\kappa$-bit random key $K$, and all distinguishers $D$ issuing $s$ queries, we have*

$$\mathbf{Adv}_F^{\mathsf{WPRB}}(D) \leq \mathbf{Adv}_F^{\mathsf{WPRF}}(D) + \tbinom{s}{2} \cdot 2^{-n}.$$

The proof of the lemma follows by an application of the triangle inequality and the fact that a beacon and a random function behave identically as long as the same input $r$ is not returned twice.

## A.2 Proof of Theorem 1

Throughout the proof, we consider $L+1$ hybrid experiments $H_0, H_1, \ldots, H_L$ where $D$ is given random inputs $r_1, \ldots, r_s$ and interacts which a (randomized) oracle $\mathcal{X} \to \{0,1\}^\kappa$ that keeps track of all vertices of the subtree of $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ induced by the queries of $D$. In particular, the oracle assigns to all *internal* vertices $v$ of the subtree increasing integer values $\mathsf{label}(v)$ when they are visited for the first time, with $\mathsf{label}(\epsilon) = 0$. Note that by our assumption on $D$ we have $\mathsf{label}(v) < L$. Furthermore, it associates $\kappa$-bit values $\mathsf{value}(v)$ with all visited vertices: Initially $\mathsf{value}(\epsilon)$ is set to a random value, whereas all remaining values are *undefined*. In $H_i$, an oracle query $x \in \mathcal{X}$ (with $\mathsf{PAD}(x)$) by $D$ is answered by looking for the highest $\lambda^*$ such that $\mathsf{value}(m_1, \ldots, m_{\lambda^*})$ is defined (we abuse notation setting $(m_1, m_0) = \epsilon$), and for all $j = \lambda^* + 1, \ldots, \lambda$ defining $\mathsf{value}(m_1, \ldots, m_j)$ to a random value if the label of $(m_1, \ldots, m_{j-1})$ is smaller than $i$, and to $F(\mathsf{value}(m_1, \ldots, m_{j-1}), r_{m_j})$ otherwise. Finally, the value of $(m_1, \ldots, m_\lambda)$ is returned to $D$ as the oracle's output.

A formal pseudo-code description of the $i$'th hybrid experiment $H_i$ is given in Figure 4. It is clear that in $H_0$ all queries are answered with the exact same distribution as when $D$ interacts with $\mathsf{RC}_s^F(K, R_1, \ldots, R_s, \cdot)$ for random $\kappa$-bit $K$ and random $n$-bit strings $R_1, \ldots, R_s$, whereas the oracle provides independent random answers to distinct queries in $H_L$. Denoting as $D(H_i)$ the binary random variable giving the output of $D$ in the experiment $H_i$, we can hence rewrite $\mathbf{Adv}_{\mathsf{RC}_s^F}^{\mathsf{PRF}}(D) = |\mathsf{P}[D(H_0) = 1] - \mathsf{P}[D(H_L) = 1]|$.

For all $i = 0, \ldots, L-1$, we construct a distinguisher $D_i$ with access to the oracle $\mathcal{S}^g$ for some function $g : \{0,1\}^n \to \{0,1\}^\kappa$. It simulates the experiment $H_i$ with two substantial modifications: (i) It initially performs $s$ queries to the given oracle obtaining pairs $(r_1, y_1), \ldots, (r_s, y_s)$, and simulates the execution of $D^{(\cdot)}(r_1, \ldots, r_s)$; (ii) In the process of answering an oracle query $x \in \mathcal{X}$ to the simulated $D$, where $\mathsf{label}(m_1, \ldots, m_{j-1})$ is exactly $i$, it sets $\mathsf{value}(m_1, \ldots, m_j)$ equal $y_{m_j}$ instead of (as before) $F(\mathsf{value}(m_1, \ldots, m_{j-1}), r_{m_j})$. (The cases $< i$ and $> i$ are handled as above.) Otherwise, oracle queries are answered as in $H_i$, and $D_i$ finally outputs the output of the simulated $D$.

The following can easily be shown about the distinguisher $D_i$.

**Lemma 4.** *Let $\mathbf{B}_\kappa$ be a $\kappa$-bit beacon and $K$ a uniform $\kappa$-bit key. Then*

$$\mathsf{P}[D_i^{\mathcal{S}^{F(K,\cdot)}} = 1] = \mathsf{P}[D(H_i) = 1], \text{ and}$$
$$\mathsf{P}[D_i^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1] = \mathsf{P}[D(H_{i+1}) = 1]$$

**Initialization:**

$\mathsf{value}(\epsilon) \xleftarrow{R} \{0,1\}^\kappa$; $r_1, \ldots, r_t \xleftarrow{R} \{0,1\}^n$; $\mathsf{cnt} := 0$

run distinguisher $D^{(\cdot)}(r_1, \ldots, r_s)$

**upon** receiving an oracle query $x \in \mathcal{X}$ from $D$ **do**

$(m_1, \ldots, m_\lambda) := \mathsf{PAD}(x)$

$\lambda^* :=$ maximal value such that $\mathsf{value}(m_1, \ldots, m_{\lambda^*})$ is defined

**for all** $j = \lambda^* + 1, \ldots, \lambda$ **do**

**if** $\mathsf{label}(m_1, \ldots, m_{j-1})$ is undefined **then** $\mathsf{label}(m_1, \ldots, m_{j-1}) := \mathsf{cnt}$; $\mathsf{cnt} := \mathsf{cnt} + 1$

**if** $\mathsf{label}(m_1, \ldots, m_{j-1}) < i$ **then** $\mathsf{value}(m_1, \ldots, m_j) \xleftarrow{R} \{0,1\}^\kappa$

**else** $\mathsf{value}(m_1, \ldots, m_j) := F(\mathsf{value}(m_1, \ldots, m_{j-1}), r_{m_j})$

**end for**

**return** $\mathsf{value}(m_1, \ldots, m_\lambda)$ to $D$

**Fig. 4.** Pseudo-code description of the experiment $H_i$. The notation $\xleftarrow{R}$ indicates assignment of a uniformly-random value from the set on the right-hand side to the variable on the left-hand side.

*for all* $i = 0, \ldots, L-1$.

*Proof.* Clearly, the behavior of $H_i$ is unchanged if we first generate $k$ uniformly at random, as well as the values $(r_1, F(k, r_1)), \ldots, (r_s, F(k, r_s))$ (for independent random $n$-bit strings $r_1, \ldots, r_s$), and only when needed we set the value of the unique inner vertex $(m_1, \ldots, m_{j-1})$ with label $i$ to $k$ and the values of the children vertices $(m_1, \ldots, m_j)$ (if needed) to $F(k, r_{m_j})$. The experiment stays unchanged even if we set the value of $(m_1, \ldots, m_{j-1})$ with value $i$ to a fresh random value independent of $k$, as this value is never output and does not influence the distribution of any further value. In particular, in this case $k$ can be secret and the values $(r_1, F(k, r_1)), \ldots, (r_s, F(k, r_s))$ are obtained externally: This is exactly the experiment where $D_i$ interacts with $\mathcal{S}^{F(k,\cdot)}$ for a random $k$. The second equality follows from the obvious fact that $y_1, \ldots, y_s$ returned by a beacon are $s$ fresh random values. □

To conclude, we define the distinguisher $D'$ that picks an index $i \in \{0, \ldots, L-1\}$ uniformly at random and runs $D_i$. We obtain for a uniformly-distributed $\kappa$-bit key $K$

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{WPRB}}_F(D') &= \left|\mathsf{P}[D'^{\mathcal{S}^{F(K,\cdot)}} = 1] - \mathsf{P}[D'^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1]\right| \\
&= \frac{1}{L}\left|\sum_{i=0}^{L-1} \mathsf{P}[D_i^{\mathcal{S}^{F(K,\cdot)}} = 1] - \mathsf{P}[D_i^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1]\right| \\
&\overset{\text{L. 4}}{=} \frac{1}{L}\left|\sum_{i=0}^{L-1} \mathsf{P}[D(H_i) = 1] - \mathsf{P}[D(H_{i+1}) = 1]\right| = \frac{1}{L}\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{RC}^F_s}(D),
\end{aligned}$$

and the bound in the theorem follows by Lemma 3. It also easy to verify that the distinguisher $D'$ can be implemented with the given running time. (We assume that the running time of $\mathsf{PAD}$ is minimal and can hence be neglected.)

### A.3 Proof of Lemma 2

Throughout this proof, we fix two inputs $x, x'$ and we let $(x_1, \ldots, x_\lambda)$ and $(x'_1, \ldots, x'_{\lambda'})$ be their corresponding paddings. Without loss of generality, assume that $\lambda \leq \lambda'$, and let $\Lambda := \Lambda(x, x')$ and $\lambda^*$ be defined as above. The proof considers two distinct cases: (i) $(x_1, \ldots, x_\lambda)$ is *not* a prefix of $(x'_1, \ldots, x'_{\lambda'})$, and (ii) $(x_1, \ldots, x_\lambda)$ is a prefix of $(x'_1, \ldots, x'_{\lambda'})$. For the proof, we also neglect the time-complexity of $\mathsf{SAMPLE}$ (as it is minor with respect to evaluating $F$). For convenience, in the following we denote $\mathsf{SAMPLE}(x, x, y, y)$ as $\mathsf{SAMPLE}(x, y)$.

*Case 1.* Similarly to Theorem 1, one proves that both outputs are pseudorandom, and hence can only collide with low probability, as otherwise one would be able to efficiently distinguish then from random inputs.

We assign $\lambda + \lambda - \lambda^* - 1 = \Lambda$ integer labels to all prefixes of the paddings of $x$ and $x'$ such that $\mathsf{label}(\epsilon) := 0$, $\mathsf{label}(x_1, \ldots, x_i) := i$ for all $i = 1, \ldots, \lambda - 1$, and $\mathsf{label}(x'_1, \ldots, x'_{\lambda^*+i}) := \lambda + i - 1$ for all $i = 1, \ldots, \lambda' - \lambda^* - 1$. Moreover, we define for all $i \in \{0, \ldots, \Lambda\}$ the hybrid experiment $H_i$ in which we first set $\mathsf{value}(\epsilon)$ to a uniform $\kappa$-bit string and $k'$ to a random $\kappa'$-bit string, and subsequently we set $\mathsf{value}(x''_1, \ldots, x''_j)$ iteratively for all (non-empty) prefixes $(x''_1, \ldots, x''_j)$ of $(x_1, \ldots, x_\lambda)$ or $(x'_1, \ldots, x'_{\lambda'})$ to a fresh random $\kappa$-bit strings if $\mathsf{label}(x''_1, \ldots, x''_{j-1}) < i$, and to $F(\mathsf{value}(x''_1, \ldots, x''_{j-1}), M(k', x''_j))$ otherwise. Additionally, let $p_i$ be the probability that $\mathsf{value}(x_1, \ldots, x_\lambda)$ equals $\mathsf{value}(x'_1, \ldots, x'_{\lambda'})$ in $H_i$: By inspection, it is easy to verify that $p_0 = \mathsf{p}^{\mathsf{COLL}}_{\mathsf{PI}^F_M}(x, x')$ and $p_L = 2^{-\kappa}$ (as in $H_\Lambda$ both values are random and independent).

For all $i = 0, \ldots, \Lambda - 1$, we construct a distinguisher $D_i$ which is given access to the oracle $\mathcal{S}^g$ for some $g : \{0,1\}^n \to \{0,1\}^\kappa$. We distinguish two cases for the initialization of $D_i$. If $i \neq \lambda^*$, $D_i$ issues a single query to the given oracle, obtaining a pair $(r, y)$. Furthermore, with $(x''_1, \ldots, x''_j)$ being the *unique* sequence such that $\mathsf{label}(x''_1, \ldots, x''_{j-1}) = i$, the distinguisher $D_i$ samples $k' := \mathsf{SAMPLE}(x''_j, r)$ and sets $y(x''_j) := y$. In the case where $i = \lambda^*$, the distinguisher $D_{\lambda^*}$ performs *two* queries, which deliver pairs $(r, y)$ and $(r', y')$, samples $k' := \mathsf{SAMPLE}(x_{\lambda^*+1}, x'_{\lambda^*+1}, r, r')$, and sets $y(x_{\lambda^*+1}) := y$ and $y(x'_{\lambda^*+1}) = y'$.

Subsequently, in both cases the distinguisher $D_i$ simulates $H_i$ using the sampled $k'$, with the exception that if $\mathsf{label}(x''_1, \ldots, x''_{j-1}) = i$ holds, it sets $\mathsf{value}(x''_1, \ldots, x''_j) := y(x''_j)$ (the cases $< i$ and $> i$ are unchanged). Finally, it outputs 1 if $\mathsf{value}(x_1, \ldots, x_\lambda)$ equals $\mathsf{value}(x'_1, \ldots, x'_{\lambda'})$, and 0 otherwise.

The following lemma characterizes the probabilities of $D_1, \ldots, D_\Lambda$ outputting 1 in terms of the probabilities $p_0, \ldots, p_\Lambda$.

**Lemma 5.** *For all $i = 0, \ldots, \Lambda - 1$ and a random $\kappa$-bit key $K$, we have $\mathsf{P}[D_i^{\mathcal{S}^{F(K,\cdot)}} = 1] = p_i$ and $\mathsf{P}[D_i^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1] = p_{i+1}$.*

*Proof.* We only look at the case $i = \lambda^*$ (the other cases are similar and easier). We obtain an experiment equivalent to $H^*_\lambda$ by generating $k \in \{0,1\}^\kappa$, $r, r' \in \{0,1\}^n$ independently and uniformly at random, and sampling $k' = \mathsf{SAMPLE}(x_{\lambda^*+1}, x'_{\lambda^*+1}, r, r')$. Furthermore, we set $y(x_{\lambda^*}) = F(k, r)$ and $y(x'_{\lambda^*+1}) = F(k, r')$. Also, we set $\mathsf{value}(x_1, \ldots, x_j)$ to a fresh random value for all $j < \lambda^*$, $\mathsf{value}(x_1, \ldots, x_{\lambda^*}) := k$, $\mathsf{value}(x_1, \ldots, x_{\lambda^*+1}) = y(x_{j+1})$, $\mathsf{value}(x'_1, \ldots, x'_{\lambda^*+1}) = y(x'_{j+1})$. The rest is generated as in $H_i$ (with the sampled $k'$). Note that we can obtain a further equivalent experiment by letting $\mathsf{value}(x_1, \ldots, x_{\lambda^*})$ to a fresh random value as well, and hence $k$ can be kept secret, and the pairs $(r, F(k, r))$ and $(r', F(k, r'))$ are generated externally. But this is exactly the experiment where $D$ interacts with $\mathcal{S}^{F(k,\cdot)}$ for a random $k$. The second equality follows by the fact that the values returned by the oracle are uniformly-random.

We define $D_{x,x'}$ as the distinguisher running $D_i$ for a uniformly chosen $i \in \{0, \ldots, \Lambda - 1\}$. This yields (for a uniform random $\kappa$-bit string $K$)

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{WPRB}}_F(D_{x,x'}) &= \left|\mathsf{P}[D_{x,x'}^{\mathcal{S}^{F(K,\cdot)}} = 1] - \mathsf{P}[D_{x,x'}^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1]\right| \\
&= \frac{1}{\Lambda}\left|\sum_{i=0}^{\Lambda-1} \mathsf{P}[D_i^{\mathcal{S}^{F(K,\cdot)}} = 1] - \mathsf{P}[D_i^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1]\right| \\
&= \frac{1}{\Lambda}\left|\sum_{i=0}^{\Lambda-1} p_i - p_{i+1}\right| \geq \frac{1}{\Lambda}\left[\mathsf{p}^{\mathsf{COLL}}_{\mathsf{PI}^F_M}(x, x') - 2^{-\kappa}\right],
\end{aligned}$$

and the desired bound follows using Lemma 3.

*Case 2.* To upper bound the collision probability in this case, one first shows that the output of the $\mathsf{PI}$-construction on input $x$ is pseudorandom. Then, one shows that if the evaluation on input $x'$ one replaces the value corresponding to the output on input $x$ by a random value, the fact that the values collide with good probability yields a two-query distinguisher breaking the 2-WPRF property for $F$. Here, we make these two steps at once to provide a single distinguisher $D_{x,x'}$ as promised in the lemma.

As a first step, we define for all $i \in \{0,\ldots,\lambda+1\}$ a random experiment $H_i$ where we first sample a uniform random $\kappa'$-bit string $k'$ and subsequently set $\mathsf{value}(x'_1,\ldots,x'_j)$ to a fresh random value for all $j = 0,\ldots,i$ (with the convention $(x'_1,\ldots,x'_0) = \epsilon$) and (if $i+1 \leq \lambda'$) $\mathsf{value}(x'_1,\ldots,x'_j) := F(\mathsf{value}(x'_1,\ldots,x'_{j-1}), M(k',x'_j))$ for all $j = i+1,\ldots,\lambda'$. Furthermore, we denote as $q_i$ the probability that $\mathsf{value}(x'_1,\ldots,x'_\lambda)$ and $\mathsf{value}(x'_1,\ldots,x'_{\lambda'})$ collide in $H_i$. We remark that $q_0 = \mathsf{p}^{\mathsf{COLL}}_{\mathsf{PI}^F_M}(x,x')$ and $q_{\lambda+1} = 2^{-\kappa}$ hold. In particular, the latter is true since $\mathsf{value}(x'_1,\ldots,x'_\lambda)$ is random and independent of $\mathsf{value}(x'_1,\ldots,x'_{\lambda'})$.

We now construct $\lambda+1$ distinguishers $D_1,\ldots,D_{\lambda+1}$ which expect an oracle $\mathcal{S}^g$ for some $g : \{0,1\}^m \to \{0,1\}^n$. These distinguishers have two different forms:

- The distinguisher $D_i$ (for $i \in \{1,\ldots,\lambda\}$) issues a single oracle query, which delivers a pair $(r,y)$, and samples $k' := \mathsf{SAMPLE}(x'_i, r)$. Subsequently, it sets $\mathsf{value}(x'_1,\ldots,x'_i) := y$ and iteratively computes $\mathsf{value}(x'_1,\ldots,x'_j) := F(\mathsf{value}(x'_1,\ldots,x'_{j-1}), M(k',x'_j))$ for all $j = i+1,\ldots,\lambda'$. Finally, it outputs 1 if and only if $\mathsf{value}(x'_1,\ldots,x'_\lambda)$ and $\mathsf{value}(x'_1,\ldots,x'_{\lambda'})$ collide, and 0 otherwise.
- The distinguisher $D_{\lambda+1}$ makes two queries to the given oracle, obtaining two pairs $(r,y)$, $(r',y')$, and samples $k' := \mathsf{SAMPLE}(x'_{\lambda+1}, r)$. Furthermore, it sets $\mathsf{value}(x'_1,\ldots,x'_{\lambda+1}) := y$ and (if $\lambda' > \lambda+1$) computes the values $\mathsf{value}(x'_1,\ldots,x'_j) := F(\mathsf{value}(x'_1,\ldots,x'_{j-1}), M(k',x'_j))$ for all $j = \lambda+2,\ldots,\lambda'$. Finally, it outputs 1 is $F(\mathsf{value}(x'_1,\ldots,x'_{\lambda'}), r') = y'$, and 0 otherwise.

The following lemma relates the success probabilities of the distinguishers $D_1,\ldots,D_{\lambda+1}$ to the collision probabilities $q_0,\ldots,q_{\lambda+1}$.

**Lemma 6.** *Let $\mathbf{B}_\kappa$ be a $\kappa$-beacon, $K$ a random $\kappa$-bit key, and $D_1,\ldots,D_{\lambda+1}$ as above. Then, the following two properties hold:*

*(i)* $\mathsf{P}[D_i^{\mathcal{S}^{F(K,\cdot)}} = 1] = q_{i-1}$ *and* $\mathsf{P}[D_i^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1] = q_i$ *for all* $i \in \{1,\ldots,\lambda\}$*;*
*(ii)* $\mathsf{P}[D_{\lambda+1}^{\mathcal{S}^{F(K,\cdot)}} = 1] \geq q_\lambda$ *and* $\mathsf{P}[D_{\lambda+1}^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1] = q_{\lambda+1}$.

*Proof.* Note that (i) is analogous to Lemma 5. For (ii), note that the collision probability $q_\lambda$ corresponds (for $D_{\lambda+1}$) to the probability that $\mathsf{value}(x'_1,\ldots,x'_{\lambda'})$ equals $k$. In this case the distinguisher always outputs 1, but $F(\mathsf{value}(x'_1,\ldots,x'_{\lambda'}), r') = y'$ may hold even if $\mathsf{value}(x'_1,\ldots,x'_{\lambda'}) \neq k$. The second equality follows from the obvious fact that in the case where $D_{\lambda+1}$ queries $\mathcal{S}^{\mathbf{B}_\kappa}$ the value $y'$ is random and independent from $F(\mathsf{value}(x'_1,\ldots,x'_{\lambda'}), r')$.

The final distinguisher $D_{x,x'}$ picks a random $i \in \{1,\ldots,\lambda+1\}$ and runs $D_i$. Then, with $\Lambda = \lambda+1$, Lemma 6 yields (for a uniform random $\kappa$-bit key $K$)

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{WPRB}}_F(D_{x,x'}) &= \left|\mathsf{P}[D_{x,x'}^{\mathcal{S}^{F(K,\cdot)}} = 1] - \mathsf{P}[D_{x,x'}^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1]\right| \\
&\geq \frac{1}{\Lambda}\sum_{i=1}^{\Lambda} \mathsf{P}[D_i^{\mathcal{S}^{F(K,\cdot)}} = 1] - \mathsf{P}[D_i^{\mathcal{S}^{\mathbf{B}_\kappa}} = 1] \\
&\geq \frac{1}{\Lambda}\sum_{i=0}^{\Lambda-1} q_i - q_{i+1} = \frac{1}{\Lambda}\left[\mathsf{p}^{\mathsf{COLL}}_{\mathsf{PI}^F_M}(x,x') - 2^{-\kappa}\right].
\end{aligned}$$

Once again, the upper bound of the lemma is obtained by applying Lemma 3.