On the Security of Keyed Hashing Based on Public Permutations

Jonathan Fuchs, Yann Rotella, Joan Daemen

On the Security of Keyed Hashing Based on Public Permutations

Jonathan Fuchs[1(✉)][0000−0001−5468−7846], Yann Rotella[2], and Joan Daemen[1][0000−0002−4102−0775]

[1] Radboud University, Nijmegen, The Netherlands
[2] Université Paris-Saclay, UVSQ, CNRS, Versailles, France
jonathan.fuchs@ru.nl, yann.rotella@uvsq.fr, joan.daemen@ru.nl

Abstract. Doubly-extendable cryptographic keyed functions (deck) generalize the concept of message authentication codes (MAC) and stream ciphers in that they support variable-length strings as input and return variable-length strings as output. A prominent example of building deck functions is Farfalle, which consists of a set of public permutations and rolling functions that are used in its compression and expansion layers. By generalizing the compression layer of Farfalle, we prove its universality in terms of the probability of differentials over the public permutation used in it. As the compression layer of Farfalle is inherently parallel, we compare it to a generalization of a serial compression function inspired by Pelican-MAC. The same public permutation may result in different universalities depending on whether the compression is done in parallel or serial. The parallel construction consistently performs better than the serial one, sometimes by a big factor. We demonstrate this effect using Xoodoo[3], which is a round-reduced variant of the public permutation used in the deck function Xoofff.

Keywords: keyed hashing, public permutations, universal hashing, parallel, serial, differential probability

1 Introduction

The doubly-extendable cryptographic keyed (deck) function is a relatively recent cryptographic primitive introduced by Daemen et al. [12]. A deck function generalizes a MAC function and a stream cipher in that it supports variable-length strings as input and returns variable-length strings as output.

Farfalle is a construction for building deck functions from a set of b-bit public permutations and rolling functions and was introduced in 2017 by Bertoni et al. [5]. It consists of a compression phase followed by an expansion phase. The compression phase has two steps, namely, the generation of a variable-length sequence of b-bits secret masks followed by the parallel compression of strings. Each b-bit secret mask is added to a b-bit input string block using a group addition. A public permutation is then applied to each block of the resulting string. The output of all the permutation calls are added together, resulting in a variable called the *accumulator*.

Farfalle is a type of function called *protected hash* (also called hash-then-encrypt). The goal of such a function is to achieve pseudorandom function (PRF) security. This is defined in terms of the advantage of an optimal attacker of distinguishing it from a random oracle, when keyed with a uniformly random key unknown to the attacker. Protected hash functions can be described as follows. For a keyed compression function F_k and a fixed-input length cryptographic function $P_{k'}$, the output Z for a given input m is defined as $Z = P_{k'}(F_k(m))$. We can now isolate the contribution of the compression function F_k to the PRF security of the protected hash function by assuming that $P_{k'}$ is PRP secure. Then the PRF advantage of the protected hash function is upper bound by the sum of the PRF advantage and the success probability, taken over the key space of k of an optimal attacker to generate collisions in the output of F_k: distinct m, m' such that $F_k(m) = F_k(m')$.

This probability is in turn upper bound by the so called ε-universality [25] of F. This is the maximum taken over all distinct message pairs of the probability taken over all keys k, that two distinct messages result in the same output. A typical application of protected hash functions is message authentication code (MAC) computation. Another mainstream approach of building a MAC is the so called Wegman-Carter(-Shoup) (WC(S)) [24] [26], that requires a nonce in the input. Given a fixed-input-length cryptographic function $P_{k'}$, a nonce n, a keyed compression function F_k and an input string m, the output is a tag T that corresponds to the input m and is given by $T = P_{k'}(n) + F(m)$. In the case of Wegman-Carter, the function $P_{k'}$ is assumed to satisfy some level of PRF security. In the case of Wegman-Carter-Shoup, $P_{k'}$ should be hard to distinguish from a random permutation, i.e., it is a pseudo-random permutation (PRP). The security of WC(S) depends on the ability of an attacker to generate forgeries, i.e, creating a valid (m', n, T') tuple whereby the attacker may have obtained one or more valid (m, n, T) tuple from a tag generation oracle. The probability of generating a successful forgery is upper bound by the so called ε-Δuniversality [25] of F. This is an upper bound on the probability taken over all keys k of two distinct input strings having a specific output difference.

Both approaches make use of a keyed compression function which we refer to as *keyed hash functions*. When looking at the proposed keyed hash functions in cryptographic literature, we can divide them into three categories. The first category makes use of strong cryptographic primitives. Notable examples are block-cipher-based modes such as CBC-MAC [2], CMAC [21], PMAC [7], Protected Counter Sums [3] and LightMAC [22]. For these modes PRF advantages can be expressed in terms of the PRP advantage of the underlying block cipher. Other examples are hash-function-based constructions such as NMAC [1] and HMAC [1] that also follow a reductionist security approach. In most of these constructions there is actually no clear separation between the keyed hashing component and PRF/PRP component and both are built using the same primitive and use the same key.

The second category are simple functions built from multiplication and addition, often in a large finite field. The best known examples are GMAC, used in

the authentication encryption mode GCM [23], and poly1305 [4]. The universality of these functions can be derived using simple mathematical arguments. This approach gives functions that are more efficient than those in the first category in many modern CPUs thanks to the presence of dedicated instructions for efficient big integer multiplication of even multiplication in binary fields, e.g., the CLMUL instruction set.

The third category uses fixed-length public permutations. Examples include Farfalle, and the compression phase of Pelican MAC [16], which compresses a variable length input in a serial way using a CBC-MAC like construction, but with a public permutation taking the place of the block cipher. This approach leads also to improved performance. For example per 16-byte input block compression in Pelican MAC requires 4 unkeyed AES rounds while CBC-MAC and PMAC require a full 10-round keyed AES. On platforms where there is no dedicated hardware support for efficient multiplication, like the ARM cortex M3/M4, it is even competitive with functions in the second category. In dedicated hardware this approach is likely to lead to more efficient compression for the same security level as functions making use of multiplication as despite its mathematical simplicity, fast multiplication has a large footprint in hardware.

The problem with this third category is that their security cannot be reduced to the PRP or PRF security of some underlying primitive as in the first category and also the simple mathematical arguments of the second category do not apply. For the serial case there is some analysis by Daemen et al. in [15] and [17] and by Dobraunig et al. in [19] but they assume a random permutation. For the parallel case there is some analysis in [5] and [12] but these are not rigorous and seem to be meant more as design rationale.

1.1 Our Contributions

In this paper we provide a rigorous security analysis of permutation-based keyed hashing without relying on a random permutation. We propose a framework that idealizes the compression phases of Farfalle and Pelican MAC in order to derive upper bounds on their universalities, under the assumption of long independent keys. This framework is meaningful in the same way that cryptanalysis of block ciphers under the assumption of independent round keys is meaningful, or in the same way that proofs on bounds for the distinguishing/differentiating advantage in the random permutation model, like for sponge [6], duplex [14] or Even-Mansour [20] are meaningful.

We express the ε-universality and ε-Δuniversality of the constructions we present as a function of the probability of differentials of the underlying permutations.

Our main contributions are:

– In Section 2 we introduce the notion of *key-then-hash* functions and we generalize the concept of differentials over them. By using this generalization, we are able to define the differential probability of a differential over such keyed hash functions.

– In Sections 3 and 4 we prove that the universality of serial and parallel key-then-hash functions using a public permutation can be expressed in terms of the probability of differentials over the underlying permutation using the new notion of differential probability over keyed hash functions.
  - We show that serial universal hashing using a public permutation f is both $\max_{a,\Delta} \mathsf{DP}_f(a, \Delta)$-universal and $\max_{a,\Delta} \mathsf{DP}_f(a, \Delta)$-$\Delta$universal. This result is given in Theorem 1.
  - We show that the ε-universality of parallel hashing using a public permutation f is $\max_a \sum_\Delta \mathsf{DP}_f^2(a, \Delta)$-universal and as such has the potential to result in better bounds than the ε-universality of serial hashing using the same f. This result is given in Theorem 2 and Theorem 3.
– In Section 5 we apply these results on XOODOO[3], a round-reduced version of the public permutation used in Xoofff [12], i.e, XOODOO[6].

2 Preliminaries

In this section we define the basic notation and definitions required to follow the analysis presented in this paper.

2.1 Notation

Our hash functions operate on strings of elements of an abelian group $\langle \mathrm{G}, + \rangle$ with the neutral element written as 0. We call the elements of G *blocks*, denote the set of ℓ-block strings as G^ℓ and the set of strings of length 1 up to κ as $\mathrm{BS}(\mathrm{G}, \kappa) = \cup_{\ell=1}^{\kappa} \mathrm{G}^\ell$. We denote strings in bold, like $\mathbf{m}$, their blocks by m_i, with indexing starting from 1 and the length of a string $\mathbf{m}$ as $|\mathbf{m}|$. For a string of ℓ zeroes we write 0^ℓ.

In this paper we work with variables $x \in \mathrm{G}$ that have a value that depends on the key $\mathbf{k}$. We denote the probability that a variable x has value X by $\Pr(x = X)$. In words, $\Pr(x = X)$ is the fraction of the keyspace for which variable x has value X. We call two variables independent if $\Pr(x = X, x' = X') = \Pr(x = X) \Pr(x' = X')$ for all $X, X' \in \mathrm{G}$.

The probability mass function (PMF) of a variable x, denoted as g_x, is the array of values $\Pr(x = X)$ over all values X. We have $g_x(X) = \Pr(x = X)$, with the probability taken over the key space. Clearly, $\forall X : 0 \leq g_x(X) \leq 1$ and $\sum_X g_x(X) = 1$. As such, a PMF can be seen as a mapping $g \colon \mathrm{G} \to [0, 1]$.

The convolution of two PMFs g_x, g_y, denoted as $g_x * g_y$, is given by:

$$g_z = g_x * g_y \iff \forall Z : g_z(Z) = \sum_X g_x(X) g_y(Z - X)\,,$$

with $-$ determined by the group operation of G and the summation done over $\mathbb{R}$.

We denote the uniform PMF over G by U: so $\forall X \in \mathrm{G}\ \mathrm{U}(X) = \frac{1}{\#\mathrm{G}}$. When we have two independent variables $x, y \in \mathrm{G}$ then the PMF of their sum (in G) is

the convolution of their PMFs. Moreover, if $z = x + y$ with x and y independent, then $\max_Z g_z(Z) \leq \min(\max_X g_x(X), \max_Y g_y(Y))$. It immediately follows that convolution with an independent uniform variable results in a uniform variable.

2.2 ε-Universality and ε-ΔUniversality

As discussed in Section 1, the security of our keyed hash functions in their relevant use cases is determined by their ε-universality and ε-Δuniversality [25]. We adapt these definitions to our notation in this section.

Definition 1 (ε-universality [25]). *A keyed hash function F is said to be ε-universal if for any distinct strings $\mathbf{m}, \mathbf{m}^*$*

$$\Pr[F_{\mathbf{k}}(\mathbf{m}) = F_{\mathbf{k}}(\mathbf{m}^*)] \leq \varepsilon\,.$$

Definition 2 (ε-Δuniversality [25]). *A keyed hash function F is said to be ε-Δ universal if for any distinct strings $\mathbf{m}, \mathbf{m}^*$ and for all $\Delta \in \mathrm{G}$*

$$\Pr[F_{\mathbf{k}}(\mathbf{m}) - F_{\mathbf{k}}(\mathbf{m}^*) = \Delta] \leq \varepsilon\,.$$

We will prove upper bounds on the universalities of our constructions in Sections 3.2 and 4.2.

2.3 Key-Then-Hash functions

We study keyed hash functions that take as input elements of $\mathrm{BS}(\mathrm{G}, \kappa)$ and return an element of G. The keys are elements of G^κ. When processing an input, the key is first added to the input and then an unkeyed function is applied to the result. We refer to this special case of keyed hash functions as *key-then-hash* functions. A key-then-hash function is denoted as F and is defined as $F\colon \mathrm{G}^\kappa \times \mathrm{BS}(\mathrm{G}, \kappa) \to \mathrm{G}$ with $F_{\mathbf{k}}(\mathbf{m}) = F(\mathbf{k} + \mathbf{m})$. The addition of two strings $\mathbf{m}, \mathbf{m}^*$ with $|\mathbf{m}| \leq |\mathbf{m}^*|$ is defined as $\mathbf{m}' = \mathbf{m} + \mathbf{m}^* = m_1 + m_1^*, m_2 + m_2^*, \ldots, m_{|\mathbf{m}|} + m_{|\mathbf{m}|}^*$, so the sum of two strings is as long as the shortest of the two.

In Sections 3 we show a construction for serial key-then-hash functions whose primitive is a public permutation and in Section 4 we show a parallel construction for key-then-hash functions which also makes use of a public permutation as its underlying primitive.

2.4 Differential Probability over Fixed-length Functions

Definition 3 (Differential probability). *The differential probability of a differential (a, b) over a permutation $f\colon \mathrm{G} \to \mathrm{G}$, denoted as $\mathsf{DP}_f(a, b)$, is:*

$$\mathsf{DP}_f(a, b) = \frac{\#\{x \in \mathrm{G} \mid f(x + a) - f(x) = b\}}{\#\mathrm{G}}\,.$$

We say that input difference a propagates to output difference b with probability $\mathsf{DP}_f(a, b)$.

If $\mathsf{DP}_f(a,b) > 0$, we call input difference a and output difference b *compatible* through f. In our bounds $\max_{a\neq 0,b} \mathsf{DP}_f(a,b)$ plays an important role and we denote it by MDP_f. The differential probabilities of all differentials with a common input difference a form a PMF that we denote as DP_a, so we have $\mathsf{DP}_a(b) = \mathsf{DP}_f(a,b)$.

A useful quantity is the square of the Euclidean norm of DP_a given by $\sum_b \mathsf{DP}_a^2(b)$. We denote it by $\mathsf{NDP}_f(a)$ and denote the maximum of this quantity over all input differences by MNDP_f, hence $\mathsf{MNDP}_f = \max_{a\neq 0} \sum_b \mathsf{DP}_f^2(a,b)$.

2.5 Differentials over Key-Then-Hash Functions and their Differential Probability

Classically, the definition of a differential is defined over fixed-length functions. The introduction of variable-length input makes the definition of a differential non-trivial due to the fact that two strings may differ in value but also in length. In this section, we generalize the concept of differentials to variable-length functions and define the differential probably of such differentials over key-then-hash functions. The core of our proofs in Sections 3.3, 4.3 and 4.4 relies on understanding the relationship between the probability of differentials of the keyed hash function and those of its underlying permutation.

Proposition 1. *For a key-then-hash function, the probability that two messages* $\mathbf{m}, \mathbf{m}^*$ *with* $|\mathbf{m}| \leq |\mathbf{m}^*|$ *result in an output difference* Δ *through* $F_{\mathbf{k}}$ *is given by the following ratio:*

$$\Pr\left[F_{\mathbf{k}}(\mathbf{m}) - F_{\mathbf{k}}(\mathbf{m}^*) = \Delta\right] = \frac{\#\{\mathbf{k} \in \mathrm{G}^\kappa \mid F(\mathbf{a}+\mathbf{k}) - F(0^{|\mathbf{a}|+\lambda}+\mathbf{k}) = \Delta\}}{\#\mathrm{G}^\kappa},$$

where $\lambda = |\mathbf{m}'| - |\mathbf{m}|$ *and* $\mathbf{a}$ *is taken as* $\mathbf{a} \in \mathrm{G}^{|\mathbf{m}|}$ *such that* $\mathbf{m} = \mathbf{a} + \mathbf{m}^*$.

Proof. We start by looking at the probability of any pair of strings $\mathbf{m}, \mathbf{m}^*$ leading to an output difference Δ:

$$\Pr\left[F_{\mathbf{k}}(\mathbf{m}) - F_{\mathbf{k}}(\mathbf{m}^*) = \Delta\right] = \frac{\#\{\mathbf{k} \in \mathrm{G}^\kappa \mid F(\mathbf{m}+\mathbf{k}) - F(\mathbf{m}^*+\mathbf{k})\}}{\#\mathrm{G}^\kappa}.$$

Now we prove that given an offset $\mathbf{o} \in \mathrm{G}^\kappa$ the following holds:

$$\Pr\left[F(\mathbf{m}+\mathbf{k}) - F(\mathbf{m}^*+\mathbf{k}) = \Delta\right] = \Pr\left[F(\mathbf{m}+\mathbf{o}+\mathbf{k}) - F(\mathbf{m}^*+\mathbf{o}+\mathbf{k}) = \Delta\right].$$

We denote $\mathcal{S} = \{\mathbf{k} \in \mathrm{G}^\kappa \mid F(\mathbf{m}+\mathbf{k}) - F(\mathbf{m}^*+\mathbf{k}) = \Delta\} \subseteq \mathrm{G}^\kappa$ and we note the following property. Since G^κ is an abelian group, adding an offset $\mathbf{o}$ to every element of $\mathcal{S}$ does not change its size. Furthermore, since string addition is associative, offsetting $\mathbf{m}, \mathbf{m}^*$ by $\mathbf{o}$ is equivalent to adding $\mathbf{o}$ to every element of $\mathcal{S}$. Therefore, by choosing $\mathbf{o}$ such that $\mathbf{m}^* + \mathbf{o} = 0^{|\mathbf{m}^*|}$, we get the expression in Proposition 1.

The tuple $(\mathbf{a}, \lambda)$ can be seen as the string equivalent of an input difference. This leads us to the following definition of a difference.

Definition 4 (Difference between two strings). *We define the difference between two strings* $\mathbf{m}, \mathbf{m}^*$ *with* $|\mathbf{m}| \leq |\mathbf{m}^*|$ *as the pair* $(\mathbf{a}, \lambda) \in \mathrm{G}^{|\mathbf{m}|} \times \mathbb{Z}_{\geq 0}$ *where* $\mathbf{a} = m_1 - m_1^*, m_2 - m_2^*, \ldots, m_{|\mathbf{m}|} - m_{|\mathbf{m}|}^*$ *and* $\lambda = |\mathbf{m}^*| - |\mathbf{m}|$.

When $\lambda = 0$ we say a difference is *equal-length* and otherwise we say it is *unequal-length*. Now, we define the probability of differentials over F.

Definition 5 (Generalized differentials and their DP). *Given an input difference* $(\mathbf{a}, \lambda)$ *and an output difference* Δ*, the differential probability of the differential* $(\mathbf{a}, \lambda, \Delta)$ *over* F*, denoted as* $\mathsf{DP}_F(\mathbf{a}, \lambda, \Delta)$ *is given by:*

$$\mathsf{DP}_F(\mathbf{a}, \lambda, \Delta) = \frac{\#\{\mathbf{k} \in \mathrm{G}^\kappa \mid F(\mathbf{a} + \mathbf{k}) - F(0^{|\mathbf{a}|+\lambda} + \mathbf{k}) = \Delta\}}{\#\mathrm{G}^\kappa}.$$

In order to simplify notation, when $\lambda = 0$ we omit it from the differential. Note that if we take $\lambda = 0$ and $\mathbf{a} = a \in \mathrm{G}$ we get the classical definition of differential probability.

As for fixed-length functions, the differential probabilities of all differentials with a common input difference $(\mathbf{a}, \lambda)$ form a PMF that we denote as $\mathsf{DP}_{F(\mathbf{a},\lambda)}$, so we have $\mathsf{DP}_{F(\mathbf{a},\lambda)}(\Delta) = \mathsf{DP}_F(\mathbf{a}, \lambda, \Delta)$. From Definitions 1 and 2, we can say that a keyed hash function F is $\max_{\mathbf{a},\lambda} \mathsf{DP}_F(a, \lambda, 0)$-universal and $\max_{\mathbf{a},\lambda,\Delta} \mathsf{DP}_F(a, \lambda, \Delta)$ -Δuniversal. We focus for the rest of the paper on proving upper bounds on the differential probability of our constructions.

3 Serial Key-Then-Hash Construction

We will consider the universality of serial key-then-hash functions based on an unkeyed permutation f. The construction is described in Section 3.1. The main theorem is provided in Section 3.2. We show that the universality of such construction is equal to the maximum differential probability over all non-trivial differentials of the underlying permutation. We prove Theorem 1 in Section 3.3.

3.1 Construction

We define the serialization of a public permutation in Algorithm 1 and depict it in Figure 1. The construction takes as parameters a public permutation $f \colon \mathrm{G} \to \mathrm{G}$ and a maximum string length κ. Its inputs are a key $\mathbf{k} \in \mathrm{G}^\kappa$ and a message $\mathbf{m} \in \mathrm{BS}(\mathrm{G}, \kappa)$ and it returns a digest $h \in \mathrm{G}$.

3.2 Security of the Serial Construction

In this section we express the universality of the serialization of a public permutation f in terms of the probability of differentials over f. Furthermore, we prove that Serial$[f]$ is ε-universal and ε-Δuniversal for the same value of ε.

Theorem 1 (Universality of Serial$[f]$**).** *The serialization of a public permutation* f*,* Serial$[f]$*, is* MDP_f*-universal and* MDP_f*-*Δ*universal.*

Algorithm 1: The serialization Serial$[f]$

Parameters: A public permutation $f\colon \mathrm{G} \to \mathrm{G}$ and a maximum length κ
Inputs : A key $\mathbf{k} \in \mathrm{G}^{\kappa}$ and a message $\mathbf{m} \in \mathrm{BS}(\mathcal{A}, \kappa)$
Output : A digest $h \in \mathrm{G}$

Processing :
$\mathbf{x} \leftarrow \mathbf{m} + \mathbf{k}$
$h \leftarrow 0$
for $i \leftarrow 1$ **to** $|\mathbf{m}|$ **do**
  $h \leftarrow f(x_i + h)$
end
return h

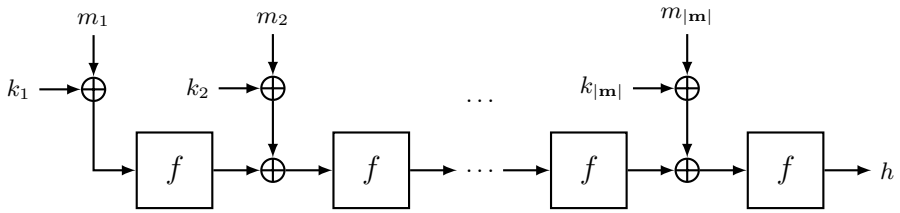

Fig. 1: The serialization Serial$[f]$

3.3 Proof of Theorem 1

In order to determine the probability of differentials over Serial$[f]$, denoted as DP_{S}, we have to consider both equal-length and unequal-length differences. We show in Lemma 1 that differentials with unequal-length input differences all have $\mathsf{DP}_{\mathrm{S}} = \frac{1}{\#\mathrm{G}}$ regardless of $\mathbf{a}$ and Δ. In Lemma 2 we prove a recursive expression for DP_{S} for equal-length differences with $|\mathbf{a}| > 1$. By combining these two lemmas we are able to construct a proof for Theorem 1.

Lemma 1 (DP_{S} of unequal-length differences). *For any differential* $(\mathbf{a}, \lambda, \Delta)$ *with* $\lambda > 0$, $\mathsf{DP}_{\mathrm{S}} = \frac{1}{\#\mathrm{G}}$.

Proof. Let $\mathbf{m}, \mathbf{m}^*$ be two strings with length $|\mathbf{m}| < |\mathbf{m}^*|$ and difference $(\mathbf{a}, \lambda, b)$. The probability that Serial$[f](\mathbf{m} + \mathbf{k}) = x$ is $\frac{1}{\#\mathrm{G}}$ since it is the result of $f(\mathsf{cv} + k_{|\mathbf{m}|} + m_{|\mathbf{m}|})$, where cv is the intermediate value accumulating the first $|\mathbf{m}| - 1$ blocks. Since we take the probability over all keys, and therefore over all values of $k_{|\mathbf{m}|}$, the probability distribution of the permutation input is uniform and hence also its output. Similarly, the probability that Serial$[f](\mathbf{m}^* + \mathbf{k}) = y$ is $\frac{1}{\#\mathrm{G}}$ since it is the result of $f(\mathsf{cv}^* + k_{|\mathbf{m}^*|} + m_{|\mathbf{m}^*|})$, where cv^* is the intermediate value accumulating the first $|\mathbf{m}^*| - 1$ blocks. The value of y is independent of the value of x since they result from f being computed under the addition of $k_{|\mathbf{m}|}$ and $k_{|\mathbf{m}^*|}$ respectively and they are two secret key blocks chosen independently of each other from a uniform distribution. We get that the output difference is

Δ if $x = \Delta + y$. Hence, we can partition the sample space. We use the following condition. Serial$[f](\mathbf{m}+\mathbf{k}) = \Delta + y$ given the event Serial$[f](\mathbf{m}^* + \mathbf{k}) = y$. Each partition has probability $\frac{1}{\#\mathrm{G}^2}$. By applying the law of total probability we obtain the expression in Lemma 1.

Lemma 2 (DP_S of an extra message block with $\lambda = 0$). *Let* $(\mathbf{a}, a)$ *be the concatenation of* $\mathbf{a} \in \mathrm{BS}(\mathrm{G}, \kappa)$ *and* $a \in \mathrm{G}$. *The differential probability of the differential* $((\mathbf{a}, a), \Delta)$ *over* Serial$[f]$ *is given by:*

$$\mathsf{DP}_\mathrm{S}((\mathbf{a}, a), \Delta) = \sum_{t \in \mathrm{G}} \mathsf{DP}_\mathrm{S}(\mathbf{a}, t)\mathsf{DP}_f(a + t, \Delta)\,.$$

Proof. We prove this using the law of total probability. We start by looking at the conditional probability that $a + t$ propagates to Δ through f given that $\mathbf{a}$ propagates to t through Serial$[f]$ for any value $t \in \mathrm{G}$. Since the key blocks are chosen independently and at random from a uniform distribution these two events are independent from each other. Therefore, it happens with probability $\mathsf{DP}_\mathrm{S}(\mathbf{a}, t)\mathsf{DP}_f(a + t, \Delta)$. By applying the law of total probability we get the expression in Lemma 2.

Proof (Theorem 1). Using Lemma 2, we first show that the DP of an equal-length differential $((\mathbf{a}, a), \Delta)$ is upper bounded by $\max_{\Delta'} \mathsf{DP}_\mathrm{S}(\mathbf{a}, \Delta')$:

$$\begin{aligned}
\mathsf{DP}_\mathrm{S}((\mathbf{a}, a), \Delta) &= \sum_{t \in \mathrm{G}} \mathsf{DP}_\mathrm{S}(\mathbf{a}, t)\mathsf{DP}_f(a + t, \Delta) \\
&\leq \sum_{t \in \mathrm{G}} (\max_t \mathsf{DP}_\mathrm{S}(\mathbf{a}, t))\mathsf{DP}_f(a + t, \Delta) \\
&= \max_{\Delta'} \mathsf{DP}_\mathrm{S}(\mathbf{a}, \Delta') \sum_t \mathsf{DP}_f(t, \Delta)\,.
\end{aligned}$$

Since f is a permutation, we have $\sum_t \mathsf{DP}_f(t, \Delta) = \sum_t \mathsf{DP}_{f^{-1}}(\Delta, t) = 1$ and we obtain:

$$\mathsf{DP}_\mathrm{S}((\mathbf{a}, a), \Delta) = \max_{\Delta} \mathsf{DP}_\mathrm{S}(\mathbf{a}, \Delta)\,. \tag{1}$$

As (1) holds for any input difference $(\mathbf{a}, a)$ and output difference Δ we have:

$$\max_{(\mathbf{a}, a), \Delta} \mathsf{DP}_\mathrm{S}((\mathbf{a}, a), \Delta) \leq \max_{\mathbf{a}, \Delta} \mathsf{DP}_\mathrm{S}(\mathbf{a}, \Delta)\,.$$

It follows that the maximum DP over all equal-length differentials with an input difference of length ℓ is upper bounded by the maximum DP over all equal-length differentials with an input difference of length $\ell - 1$. This can be applied recursively until we reach $\ell = 1$, yielding $\max_{a_1, \Delta} \mathsf{DP}_\mathrm{S}(a_1, \Delta) = \max_{a_1, \Delta} \mathsf{DP}_f(a_1, \Delta) = \mathsf{MDP}_f$. So the maximum DP over all equal-length differentials is MDP_f.

For unequal-length differentials Lemma 1 states that the DP equals $\frac{1}{\#\mathrm{G}}$. As $\mathsf{MDP}_f > \frac{1}{\#\mathrm{G}}$, this finishes the proof.

4 Parallel Key-Then-Hash Construction

Similarly to Section 3, we consider the universality of a construction based on an unkeyed permutation f. However, in this construction, strings are compressed in a parallel way. In Section 4.1 we define the construction. The main theorems of this section are provided in Section 4.2. In Sections 4.3 and 4.4 we prove Theorems 2 and 3 respectively.

4.1 Construction

We describe the parallelization of a public permutation in Algorithm 2 and depict it in Figure 2. The construction takes as parameters a public permutation $f\colon \mathrm{G} \to \mathrm{G}$ and a maximum string length κ. The inputs are a key $\mathbf{k} \in \mathrm{G}^{\kappa}$ and a string $\mathbf{m} \in \mathrm{BS}(\mathrm{G}, \kappa)$ and it returns a digest $h \in \mathrm{G}$.

Algorithm 2: The parallelization Parallel$[f]$

Parameters: A public permutation $f\colon \mathrm{G} \to \mathrm{G}$ and a maximum length κ
Inputs : A key $\mathbf{k} \in \mathrm{G}^{\kappa}$ and a message $\mathbf{m} \in \mathrm{BS}(\mathrm{G}, \kappa)$
Output : A digest $h \in \mathrm{G}$

$\mathbf{x} \leftarrow \mathbf{m} + \mathbf{k}$
$h \leftarrow 0$
for $i \leftarrow 1$ **to** $|\mathbf{m}|$ **do**
  $h \leftarrow h + f(x_i)$
end
return h

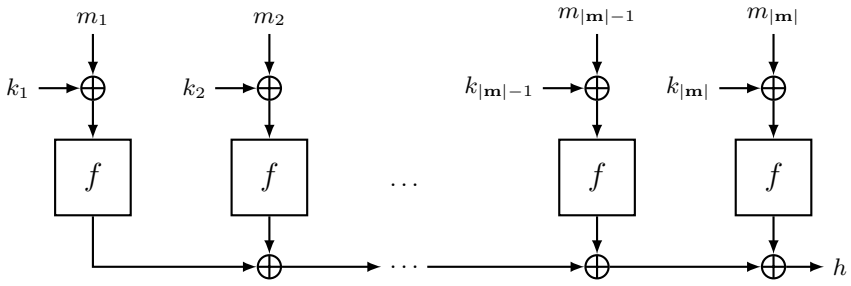

Fig. 2: The parallelization Parallel$[f]$

4.2 Security of Parallel Construction

In this section we describe the universality of the parallelization of a public permutation f in terms of the differential probability of f. Unlike the serialization of a public function, the ε-universality and ε-Δ universality of Parallel[f] are in general different.

Theorem 2 (Δ-Universality of Parallel[f]**).** *The parallelization of a public permutation f,* Parallel[f]*, is MDP_f-Δuniversal.*

Theorem 3 (Universality of Parallel[f]**).** *The parallelization of a public permutation f,* Parallel[f]*, is MNDP_f-universal.*

4.3 Proof of Theorem 2

In Lemma 3 we show that the PMF of an input difference to Parallel[f], denoted as $\mathsf{DP}_{\mathrm{P}(\mathbf{a},\lambda)}$, can be obtained by convolution of the PMFs DP_{a_i} and the uniform distribution U.

Lemma 3 (DP of differentials over Parallel[f] **).** *The PMF of an input difference $(\mathbf{a}, \lambda)$ to* Parallel[f] *is given by:*

$$\mathsf{DP}_{\mathrm{P}}(\mathbf{a}, \lambda) = \mathsf{DP}_{a_1} * \mathsf{DP}_{a_2} * \ldots * \mathsf{DP}_{a_\ell} * \mathrm{U}\,.$$

Proof. Assume we process from left to right. We will express the PMF of the chaining value after processing a new block with difference a as a function of the PMF of the chaining value before processing that new block. We denote the difference in the partial message by $\mathbf{a}$. Per definition we have for the PMF of that chaining value $\mathsf{DP}_{\mathrm{P}(\mathbf{a})}$. The PMF of the difference in the new block is DP_a. The new value of the chaining value is the sum of these two variables. If the PMFs are independent, then the resulting PMF is given by the convolution of the two. These PMFs are indeed independent, as the two PMFs are governed by non-overlapping key blocks and their distribution is over all possible keys. This can be applied recursively and we obtain for any two partial messages of equal length that $\mathsf{DP}_{\mathrm{P}(\mathbf{a})} = \mathsf{DP}_{a_1}\mathsf{DP}_{a_2}\ldots\mathsf{DP}_{a_{|\mathbf{a}|}}$.

Now we will absorb a new block that is only present in one of the two messages. The difference between the two messages is now simply the value of the output of the permutation for the input block in the longest message. The PMF of this value is the U distribution as its input due to the presence of a key block. The new value of the chaining value is the sum of these two variables. These PMFs are independent, as the two PMFs are here also governed by non-overlapping key blocks. Therefore the PMF of the sum is the convolution of the PMFs. Convolution with the uniform distribution gives again the uniform distribution.

Combining the two results proves the lemma.

Proof (Theorem 2). Similarly to the proof of Theorem 1, we will prove an upper bound on $\max_{\mathbf{a},\lambda,\Delta} \mathsf{DP}_{\mathrm{P}}(\mathbf{a}, \lambda, \Delta)$.

By applying Lemma 3, we get the following upper bounds:

$$\begin{aligned}\max_{\mathbf{a},\lambda,\Delta} \mathsf{DP}_{\mathrm{P}}(\mathbf{a},\lambda,\Delta) &\leq \max\left\{\max_{\mathbf{a},\Delta\in\mathrm{G}} \mathsf{DP}_{\mathrm{P}}(\mathbf{a},\Delta), \max_{\mathbf{a},\lambda,\Delta} \mathsf{DP}_{\mathrm{P}}(\mathbf{a},\lambda,\Delta)\right\}\\ &= \max\left\{\max_{a,\Delta\in\mathrm{G}} \mathsf{DP}_f(a,\Delta), \max_{x\in\mathrm{G}} \mathrm{U}(x)\right\}\\ &= \max_{a,\Delta\in\mathrm{G}} \mathsf{DP}_f(a,\Delta).\end{aligned}$$

4.4 Proof of Theorem 3

In this section we will prove Theorem 3 by using the same technique used in the proof of Theorem 2 but on two-block equal-length differentials.

Proof (Theorem 3). Since f is a permutation, it is impossible to achieve a 0 output difference with single block equal-length differences. From the proof of Theorem 2, we know that the following holds:

$$\max_{\mathbf{a},\lambda} \mathsf{DP}_{\mathrm{P}}(\mathbf{a},\lambda,0) \leq \max_{a_1,a_2,\Delta\in\mathrm{G}} \mathsf{DP}_{\mathrm{P}}((a_1,a_2),\Delta) \tag{2}$$

$$= \max_{a_1,a_2,\Delta\in\mathrm{G}} \sum_{t\in\mathrm{G}} \mathsf{DP}_f(a_1,t)\mathsf{DP}_f(a_2,\Delta-t)\,. \tag{3}$$

The right-hand part of (3) can be seen as a scalar product of vectors with components indexed by t. The scalar product of two vectors is upper bound by the square of the maximum of the norm of the two vectors where the norm of a vector is square root of the sum of squares of its coordinates, i.e, the Euclidian norm. Hence, we have the following upper-bound.

$$\begin{aligned}\max_{\mathbf{a},\lambda} \mathsf{DP}_{\mathrm{P}}(\mathbf{a},\lambda,0) &\leq \max\left\{\sum_{t\in\mathrm{G}} \mathsf{DP}_f^2(a_1,t), \sum_{t\in\mathrm{G}} \mathsf{DP}_f^2(a_2,\Delta-t)\right\}\\ &= \max\left\{\sum_{t\in\mathrm{G}} \mathsf{DP}_f^2(a_1,t), \sum_{t\in\mathrm{G}} \mathsf{DP}_f^2(a_2,t)\right\}\,.\end{aligned}$$

This is true for any a_1 or a_2, hence we have:

$$\max_{\mathbf{a},\lambda} \mathsf{DP}_{\mathrm{P}}(\mathbf{a},\lambda,0) \leq \max_{a\in\mathrm{G}} \sum_{t\in\mathrm{G}} \mathsf{DP}_f^2(a,t)\,.$$

Equality is achieved by taking $a_2 = -a_1$.

We summerize the results of the Theorems 1, 2 and 3 in Table 1.

5 Application to Xoodoo

In this section we determine, bound and estimate the quantities of Xoodoo that determine the universalities in our keyed hash constructions: MDP_f and MNDP_f.

	Serial[f]	Parallel[f]
Δuniversality	MDP_f	MDP_f
universality	MDP_f	MNDP_f

Table 1: ϵ-Δuniversality and ϵ-universality of Serial[f] and Parallel[f] based on Theorems 1, 2 and 3.

The specification for the round function of Xoodoo can be found in the Appendix section 6.

In Sections 5.1 and 5.2 we provide some the background knowledge required to understand this section. In Sections 5.3 and 5.4 we discuss MDP_f and MNDP_f of Xoodoo[3] and Xoodoo[4] respectively.

By Theorem 1 it immediately implies the MDP_f-universality and MDP_f-Δuniversality of Serial[Xoodoo[3]] and Serial[Xoodoo[4]], and by Theorems 2 and 3 the MDP_f-Δuniversality and MNDP_f-universality of Parallel[Xoodoo[3]] and Parallel[Xoodoo[4]].

5.1 Differential Propagation Basics

As we will only discuss differential probabilities over f, we will just write DP for DP_f. Determining the DP of differentials over an iterated permutation passes via *differential trails*: a chaining of differentials over a sequence of successive rounds.

Definition 6 (Differential trail). *An r-round differential trail, denoted as Q, is a sequence of $r+1$ differences: an input difference, $r-1$ intermediate differences and an output difference, where the round differentials (q_{i-1}, q_i) have non-zero DP, namely,*

$$Q = (q_0, q_1, q_2, \ldots, q_{r-1}, q_r) \text{ with } \mathsf{DP}(q_{i-1}, q_i) > 0 \text{ for all } i\,.$$

The differential probability of a trail, denoted $\mathsf{DP}(Q)$, is the probability that a random pair with input difference q_0 propagates via intermediate differences $q_1, q_2, \ldots$ to output difference q_r.

A useful concept when studying differential propagation is the *restriction weight*.

Definition 7 (Restriction weight of a differential [9]). *The restriction weight of a differential $\mathsf{DP}(a, b) > 0$ is defined as $w(a, b) = -\log_2 \mathsf{DP}(a, b)$.*

Definition 8 (Restriction weight of a differential trail [9]). *The restriction weight of a differential trail $Q = (q_0, q_1, \ldots, q_{r-1})$ is defined as $w(Q) = \sum_i w(q_{i-1}, q_i)$, hence the sum of the restriction weights if its round differentials.*

In the following we will omit the qualification "restriction" and simply speak of weight. We use the term *weight profile* of a trail for the sequence of weights of its round differentials. The weight of any given round differential is in general

easy to compute and hence so is the weight of any given trail. Often $2^{-w(Q)}$ is a good approximation for $\mathsf{DP}(Q)$. If the propagation through the round differentials of a trail are independent, we call it a *Markov trail* and it satisfies $\mathsf{DP}(Q) = 2^{-w(Q)}$. This is not the case in general and an attention point that must be verified. Trails are linked to differentials: the DP of an r-round differential is the sum of the DPs of all trails connecting input difference and output difference:

$$\mathsf{DP}(a, b) = \sum_{Q \text{ with } q_0=a, q_r=b} \mathsf{DP}(Q)\,.$$

Trails with common input and output difference contribute to the same differential and are said to *cluster*. We call a trail that is the only one in its differential a *lone trail*.

Papers on trail search often report on large sets of trails with common features rather than individual trails. These sets are called *trail cores*.

Definition 9 (Differential trail core [18]). *An r-round differential trail core, denoted as $\widetilde{Q}$, is a set of differential trails over r rounds with a shared core of intermediate differences $(q_1, q_2, \ldots, q_{r-1})$ with $\mathsf{DP}(q_i, q_{i+1}) > 0$ for all $1 \leq i < r-1$.*

Given an r-round trail core $\widetilde{Q}$ and an r-round differential (a, b), the trail core will contribute to $\mathsf{DP}(a, b)$ if a is compatible with q_1 and q_{r-1} with b.

Determining $\mathsf{NDP}_f(a)$ requires computing of $\mathsf{DP}(a, b)$ for all output differences b and in typical iterated permutations this is infeasible. However, it is reasonable to assume that for a given input difference a all output differences b with $\mathsf{DP}(a, b) > T$ are known. Then we can use the following lemma to upper bound $\mathsf{NDP}_f(a)$.

Lemma 4. *For any limit T, we have:*

$$\mathsf{NDP}_f(a) \leq \sum_{b \text{ with } \mathsf{DP}(a,b)>T} \mathsf{DP}^2(a, b) + T.$$

Proof. Partitioning the differentials gives:

$$\sum_b \mathsf{DP}^2(a, b) = \sum_{\substack{b \text{ with} \\ \mathsf{DP}(a,b)>T}} \mathsf{DP}^2(a, b) + \sum_{\substack{b \text{ with} \\ \mathsf{DP}(a,b)\leq T}} \mathsf{DP}^2(a, b)\,.$$

The second sum in the right-hand side is at most T, namely, using

$$\sum_b \mathsf{DP}(a, b) = 1$$

gives

$$\sum_{\substack{b \text{ with} \\ \mathsf{DP}(a,b)\leq T}} \mathsf{DP}^2(a, b) = \sum_{\substack{b \text{ with} \\ \mathsf{DP}(a,b)\leq T}} \mathsf{DP}(a, b)\mathsf{DP}(a, b) \leq T \sum_{\substack{b \text{ with} \\ \mathsf{DP}(a,b)\leq T}} \mathsf{DP}(a, b) \leq T\,.$$

This proves the lemma.

5.2 Differential Propagation in Xoodoo

Xoodoo is a family of 384-bit permutations with a classical iterated structure: it iteratively applies a round function to a state. It is parameterized by its number of rounds: Xoodoo with r rounds is denoted Xoodoo$[r]$. The round function consist of a linear layer that we will call λ followed by a non-linear layer called χ. The non-linear layer has algebraic degree two. A consequence of this is that in round differentials the value of $\mathsf{DP}(a, b)$ is fully determined by the input difference a and hence the same for all compatible output differences b [12]. Moreover, the inverse of χ also has algebraic degree 2 and therefore in round differentials the value of $\mathsf{DP}(a, b)$ is also fully determined by the output difference b and hence the same for all compatible input differences a. The consequence of these two properties is that all trails in a trail core have the same weight and we can speak about $w(\widetilde{Q})$ without ambiguity, with $w(\widetilde{Q})$ the weight of any trail in the core.

Thanks to the shift-invariance of the Xoodoo round function, trails (and trail cores) occur in classes with members that are equivalent under horizontal shifts. These classes have size 2^d with d ranging from 0 to 7. The vast majority of trail cores are in classes of size $2^7 = 128$.

The non-linear layer χ operates in parallel and independently on 3-bit parts of the state, the so-called *columns*: it is a layer of invertible non-linear S-boxes. This has its implications for clustering. A trail core $\widetilde{Q}$ contributes to a differential (a, b) if a and q_1 are compatible. The input difference a fully determines the difference at the input of χ of the first round: it is $\lambda(a)$. So $\lambda(a)$ and q_1 must be compatible over χ.

A non-zero difference in a column at the input of χ can only propagate to a non-zero difference at its output and a zero difference in a column at its input can only propagate to a zero difference at its output. So $\lambda(a)$ must be active in exactly the same columns as q_1. We say that $\lambda(a)$ and q_1 must have the same *column activity pattern*. Similarly, $\lambda(q_{r-1})$ and b must have the same column activity pattern.

From this follows that a trail in trail cores $\widetilde{Q}$ can only cluster with a trail in trail core $\widetilde{Q}'$ if q_1 and q'_1 have the same column activity pattern and if q_{r-1} and q'_{r-1} have the same column activity pattern.

5.3 MDP_f and MNDP_f of Xoodoo[3]

In the Xoodoo GitHub repository [11] there is a list available of all 3-round trail core classes of Xoodoo[3] with weight up to 52. The list has 201 entries. In [8] it is reported that all 3-round trails with weight up to 50 are lone Markov trails. The lowest weight, 36, is attained by 4 trail core classes, hence under reasonable assumptions, we have $\mathsf{MDP}_f = 2^{-36}$. The assumptions are that there are no trail cores with weight above 50 clustering to form a differential with $\mathsf{DP} > 2^{-36}$. This would require the clustering of at least $2^{54-36} = 2^{18}$ Markov trails. The fact that there are only 201 trail core classes with weight below 52 that contain only lone Markov trails makes this extremely unlikely.

For estimating MNDP_f it is interesting to make use of Lemma 4. For its application, we need to make a reasonable assumption on the limit T. If trails remain to be lone Markov trails up to some weight, e.g., 70, the DP of differentials coincides with the values predicted by the weights of trails and we can take $T = 2^{-54}$. Clearly, as the weight of trails increases, the likelihood of clustering and dependence of round differentials does increase. Still, as discussed in [8], it is unlikely these effects are noticeable for trails with relatively low weight in permutations with round functions in which no superboxes can be identified. Xoodoo is such a permutation and any trails leading to the best collision attacks would have weight well below the permutation width that is 384.

Lemma 5. *Assuming that all differentials with* $\mathsf{DP}(a, b) > T$ *for* $T > 2^{-54}$ *correspond to lone Markov trails, we can upper bound* MNDP_f *as*

$$\mathsf{MNDP}_f \leq T + \max_a \sum_{\substack{\widetilde{Q} \text{ with } w(\widetilde{Q})<54 \\ \text{and } \mathsf{DP}(a,q_1)>0}} 2^{w(q_{r-1})-2w(\widetilde{Q})}.$$

Proof. The contribution of a trail core $\widetilde{Q}$ to $\mathsf{NDP}_f(a)$ is only non-zero if q_1 is compatible with a, and in that case it is $2^{w(q_{r-1})-2w(\widetilde{Q})}$. Namely, in a trail core with q_1 compatible with a given input difference a there are $2^{w(q_{r-1})}$ trails, that each have DP equal to $2^{-w(\widetilde{Q})}$.

For two trail cores to contribute to $\mathsf{NDP}_f(a)$ for some value of a, they must have equal column activity patterns in q_1. Or in other words, two trail cores that have different column activity patterns in q_1 cannot both contribute to $\mathsf{NDP}_f(a)$ for some a. We can partition the trail cores in the 201 trail core classes per their activity pattern in q_1: we call these *activity classes*. Then for each partition we just add the contributions of the trail cores as in Lemma 5.

Over all 3-round trail cores in [11] the ones that have highest contribution to $\mathsf{MNDP}_f(a)$ for some input difference a have weight profile $(4, 4, 28)$ and they contribute $2^{28-2\times36} = 2^{-44}$. There are three of them and they are described in [12] and called *single-orbital fans*. For each single-orbital fan, there are 4 other trail cores with the same column activity pattern in q_1, that contribute to $\mathsf{MNDP}_f(a)$ respectively $2^{-54}, 2^{-54}, 2^{-56}$ and 2^{-58}, resulting in $2^{-44} + 2^{-53} + 2^{-56} + 2^{-58} + T = (1.00226) \times 2^{-44} + T$. All other classes of trail core classes result in lower values for $\mathsf{NDP}_f(a)$. We see that the value of $\mathsf{NDP}_f(a)$ is dominated by its "lightest" trail core and that additional trail cores make it go up only slightly. We think it is reasonable to expect that this is also the case for the (unknown) trail cores with weight above 52. Still, for MNDP_f to deviate significantly from 2^{-44}, T would have to go up to 2^{-45} or so, which would imply considerable clustering and/or non-Markov trails.

So we conclude that $\mathsf{MNDP}_f \approx 2^{-44}$ and this value reflects the contribution of the single-orbital fan as dominant trail core. We see that for Xoodoo[3], MDP_f is a factor 2^8 larger than MNDP_f.

5.4 MDP_f and MNDP_f of Xoodoo[4]

The trail cores of Xoodoo[4] are far less documented than those of Xoodoo[3]. Still, [13] reports that the 4-round trails with lowest weight have weight 80 and documents these trail core classes, only 2 of them.

It is not known whether the trails in these trail cores classes are Markov trails. Moreover, they have a high degree of symmetry and the trail cores in the classes cluster two-by-two. Assuming Markov trails and no more clustering occurs this would yield $\mathsf{MDP}_f = 2^{-79}$. Based on arguments in [8] we do not expect non-Markov behaviour and/or additional clustering trails to affect this value significantly.

For MNDP_f we need again to look at the weight profiles of the trail cores. Those of two trail core classes are respectively $(32, 24, 16, 8)$ and $(8, 16, 24, 32)$.

If we approximate MNDP_f by the contribution of single trail cores, the former would give $\mathsf{NDP}_f(a) \approx 2^{-160+32} = 2^{-128}$ and the latter $\mathsf{NDP}_f(a) \approx 2^{-160+8} = 2^{-152}$. However, as said, the trail cores cluster two-by-two: the clustering trail cores have equal differences q_1 and different differences q_3, but with equal activity patterns. When fixing an input difference a, in each trail core there are 2^{32} trails to different output differences b, all with weight 80. Among those 2^{33} output differences, there are exactly 2^{16} where two trails arrive and therefore they have $\mathsf{DP}(a, b) = 2^{-79}$. So assuming all these trails are Markov trails, their contribution to $\mathsf{NDP}_f(a)$ is $(2^{33} - 2^{16})2^{-160} + 2^{16}2^{-158} = (2^{33} + 2^{16}3)2^{-160} = 1.00005 \times 2^{-127}$.

Due to the limited knowledge about 4-round trails we cannot tell whether there are no trail cores that lead to a lower value of MNDP_f. Still, an interesting observation is that the preliminary value of MDP_f is a factor 2^{48} higher than that of MNDP_f.

We summerize the results on Section 5 in Table 2.

	MDP_f	MNDP_f
Xoodoo[3]	2^{-36}	2^{-44}
Xoodoo[4]	2^{-79}	1.00005×2^{-127}

Table 2: Initial results on MDP_f and MNDP_f of Xoodoo[3] and Xoodoo[4].

6 Conclusion

By assuming long independent keys, we are able to idealize the compression phase of Farfalle. The assumption allowed us to study a class of keyed hash functions that first add a key to the input string and then do unkeyed processing. We study it by first generalizing the notion of differentials over said class of keyed hash functions by also taking into account the difference in length between two input strings. We then show that it is possible to express the universality of our constructions in terms of the probability of differentials over the underlying

public permutation. In the case of serial key-then-hash functions, we show that the universality and Δuniversality is given by MDP_f of the public permutation. These upper bounds are tight and are achieved by equal-length message pairs with a message difference of length 1. For parallel key-then-hash functions, we show that the universality is given by MNDP_f and the Δuniversality is given by MDP_f. These upper bounds are once again tight and are achieved by equal-length message pairs with a message difference of length 2 and 1 respectively. While MDP_f is a very well known property of public permutations, MNDP_f is still not well studied. For many public permutations, MNDP_f is significantly smaller than MDP_f thus making it a very compelling case to use them in parallel key-then-hash instead of serial in the proteced hash setting.

Acknowledgments

The authors would like to thank Bart Mennink for his valuable inputs during the finalization of this paper. Joan Daemen and Jonathan Fuchs are supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 1–15. Springer (1996), https://doi.org/10.1007/3-540-68697-5_1
2. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. In: Desmedt, Y. (ed.) Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings. Lecture Notes in Computer Science, vol. 839, pp. 341–358. Springer (1994), https://doi.org/10.1007/3-540-48658-5_32
3. Bernstein, D.J.: How to Stretch Random Functions: The Security of Protected Counter Sums. J. Cryptology 12(3), 185–192 (1999), http://dx.doi.org/10.1007/s001459900051, https://cr.yp.to/papers.html#stretch
4. Bernstein, D.J.: The Poly1305-AES Message-Authentication Code. In: Gilbert, H., Handschuh, H. (eds.) Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers. Lecture Notes in Computer Science, vol. 3557, pp. 32–49. Springer (2005), https://doi.org/10.1007/11502760_3
5. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Farfalle: parallel permutation-based cryptography. IACR Trans. Symmetric Cryptol. 2017(4), 1–38 (2017), https://tosc.iacr.org/index.php/ToSC/article/view/801
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Indifferentiability of the Sponge Construction. In: Smart, N.P. (ed.) Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008.

Proceedings. Lecture Notes in Computer Science, vol. 4965, pp. 181–197. Springer (2008), https://doi.org/10.1007/978-3-540-78967-3_11
7. Black, J., Rogaway, P.: A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In: Knudsen, L.R. (ed.) Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2332, pp. 384–397. Springer (2002), https://doi.org/10.1007/3-540-46035-7_25
8. Bordes, N., Daemen, J., Kuijsters, D., Van Assche, G.: Thinking Outside the Superbox. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 337–367. Springer (2021), https://doi.org/10.1007/978-3-030-84252-9_12
9. Daemen, J.: Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis. K.U.Leuven (1995), http://jda.noekeon.org/
10. Daemen, J., Hoffert, S., Peeters, M., Assche, G.V., Keer, R.V.: Xoodoo cookbook. Cryptology ePrint Archive, Paper 2018/767 (2018), https://eprint.iacr.org/2018/767, https://eprint.iacr.org/2018/767
11. Daemen, J., Hoffert, S., Van Assche, G., Van Keer, R.: DC-Xoodoo-3r.txt. https://github.com/KeccakTeam/Xoodoo/blob/master/XooTools/Trails/DC-Xoodoo-3r.txt/ (2018)
12. Daemen, J., Hoffert, S., Van Assche, G., Van Keer, R.: The design of Xoodoo and Xoofff. IACR Trans. Symmetric Cryptol. 2018(4), 1–38 (2018), https://doi.org/10.13154/tosc.v2018.i4.1-38
13. Daemen, J., Mella, S., Van Assche, G.: Tighter trail bounds for Xoodoo. Cryptology ePrint Archive, Paper 2022/1088 (2022), https://eprint.iacr.org/2022/1088, https://eprint.iacr.org/2022/1088
14. Daemen, J., Mennink, B., Van Assche, G.: Full-State Keyed Duplex With Built-In Multi-User Support. IACR Cryptol. ePrint Arch. p. 498 (2017), http://eprint.iacr.org/2017/498
15. Daemen, J., Rijmen, V.: A New MAC Construction ALRED and a Specific Instance ALPHA-MAC. In: Gilbert, H., Handschuh, H. (eds.) Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers. Lecture Notes in Computer Science, vol. 3557, pp. 1–17. Springer (2005), https://doi.org/10.1007/11502760_1
16. Daemen, J., Rijmen, V.: The Pelican MAC Function. IACR Cryptol. ePrint Arch. 2005, 88 (2005), http://eprint.iacr.org/2005/088
17. Daemen, J., Rijmen, V.: Refinements of the ALRED construction and MAC security claims. IET Inf. Secur. 4(3), 149–157 (2010), https://doi.org/10.1049/iet-ifs.2010.0015
18. Daemen, J., Van Assche, G.: Differential Propagation Analysis of Keccak. In: Canteaut, A. (ed.) Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7549, pp. 422–441. Springer (2012), https://doi.org/10.1007/978-3-642-34047-5_24
19. Dobraunig, C., Mennink, B.: Security of the Suffix Keyed Sponge. IACR Trans. Symmetric Cryptol. 2019(4), 223–248 (2019), https://doi.org/10.13154/tosc.v2019.i4.223-248

20. Even, S., Mansour, Y.: A Construction of a Cipher from a Single Pseudorandom Permutation. J. Cryptol. 10(3), 151–162 (1997), https://doi.org/10.1007/s001459900025
21. Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC. In: Johansson, T. (ed.) Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers. Lecture Notes in Computer Science, vol. 2887, pp. 129–153. Springer (2003), https://doi.org/10.1007/978-3-540-39887-5_11
22. Luykx, A., Preneel, B., Tischhauser, E., Yasuda, K.: A MAC Mode for Lightweight Block Ciphers. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 43–59. Springer (2016), https://doi.org/10.1007/978-3-662-52993-5_3
23. McGrew, D.A., Viega, J.: The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH. RFC 4543, 1–14 (2006), https://doi.org/10.17487/RFC4543
24. Shoup, V.: On Fast and Provably Secure Message Authentication Based on Universal Hashing. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 313–328. Springer (1996), https://doi.org/10.1007/3-540-68697-5_24
25. Stinson, D.R.: On the Connections Between Universal Hashing, Combinatorial Designs and Error-Correcting Codes. Electron. Colloquium Comput. Complex. 2(52) (1995), http://eccc.hpi-web.de/eccc-reports/1995/TR95-052/index.html
26. Wegman, M.N., Carter, L.: New Hash Functions and Their Use in Authentication and Set Equality. J. Comput. Syst. Sci. 22(3), 265–279 (1981), https://doi.org/10.1016/0022-0000(81)90033-7

A Xoodoo Specification

We quote the specification of the Xoodoo round function, taken verbatim from the Xoodoo Cookbook [10].

> Xoodoo is a family of permutations parameterized by its number of rounds r and denoted Xoodoo$[r]$.
> Xoodoo has a classical iterated structure: It iteratively applies a round function to a state. The state consists of 3 equally sized horizontal *planes*, each one consisting of 4 parallel 32-bit *lanes*. Similarly, the state can be seen as a set of 128 *columns* of 3 bits, arranged in a 4×32 array. The planes are indexed by y, with plane $y = 0$ at the bottom and plane $y = 2$ at the top. Within a lane, we index bits with z. The lanes within a plane are indexed by x, so the position of a lane in the state is determined by the two coordinates (x, y). The bits of the state are indexed by (x, y, z) and the columns by (x, z). *Sheets* are the arrays of three lanes on top of each other and they are indexed by x. The Xoodoo state is illustrated in Figure 3.

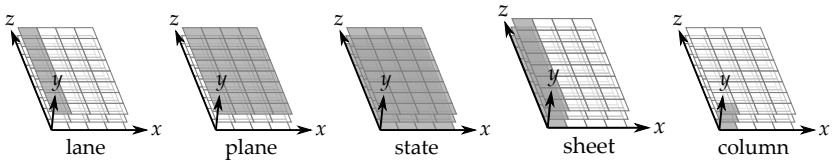

Fig. 3: Toy version of the Xoodoo state, with lanes reduced to 8 bits, and different parts of the state highlighted. [10]

The permutation consists of the iteration of a round function $\mathrm{R_i}$ that has 5 steps: a mixing layer θ, a plane shifting ρ_{west}, the addition of round constants ι, a non-linear layer χ and another plane shifting ρ_{east}.
We specify Xoodoo in Algorithm 3, completely in terms of operations on planes and use thereby the notational conventions we specify in Table 3.

A_y	Plane y of state A
$A_y \lll (t, v)$	Cyclic shift of A_y moving bit in (x, z) to position $(x + t, z + v)$
$\overline{A_y}$	Bitwise complement of plane A_y
$A_y + A_{y'}$	Bitwise sum (XOR) of planes A_y and $A_{y'}$
$A_y \cdot A_{y'}$	Bitwise product (AND) of planes A_y and $A_{y'}$

Table 3: Notational conventions [10]

Algorithm 3: Definition of Xoodoo$[r]$ with r the number of rounds [10]

Parameters: Number of rounds r
for Round index i from $1 - r$ to 0 **do**
  $A = \mathrm{R_i}(A)$
end for

Here $\mathrm{R_i}$ is specified by the following sequence of steps:

θ :
  $P \leftarrow A_0 + A_1 + A_2$
  $E \leftarrow P \lll (1, 5) + P \lll (1, 14)$
  $A_y \leftarrow A_y + E$ for $y \in \{0, 1, 2\}$
ρ_{west} :
  $A_1 \leftarrow A_1 \lll (1, 0)$
  $A_2 \leftarrow A_2 \lll (0, 11)$
ι :
  $A_0 \leftarrow A_0 + \mathrm{C}_i$
χ :
  $B_0 \leftarrow \overline{A_1} \cdot A_2$
  $B_1 \leftarrow \overline{A_2} \cdot A_0$
  $B_2 \leftarrow \overline{A_0} \cdot A_1$
  $A_y \leftarrow A_y + B_y$ for $y \in \{0, 1, 2\}$
ρ_{east} :
  $A_1 \leftarrow A_1 \lll (0, 1)$
  $A_2 \leftarrow A_2 \lll (2, 8)$

i	c_i	i	c_i	i	c_i	i	c_i
−11	0x00000058	−8	0x000000D0	−5	0x00000060	−2	0x000000F0
−10	0x00000038	−7	0x00000120	−4	0x0000002C	−1	0x000001A0
−9	0x000003C0	−6	0x00000014	−3	0x00000380	0	0x00000012

Table 4: The round constants c_i with $-11 \leq i \leq 0$, in hexadecimal notation (the least significant bit is at $z = 0$) [10].