KOALA: A Low-Latency Pseudorandom Function

Parisa Amiri Eliasi[1], Yanis Belkheyar[1], Joan Daemen[1], Santosh Ghosh[2], Daniël Kuijsters[1], Alireza Mehrdad[1], Silvia Mella[1], Shahram Rasoolzadeh[1], and Gilles Van Assche[3]

[1] Radboud University, Nijmegen, The Netherlands firstname.lastname@ru.nl
[2] Intel Labs, Hillsboro, USA firstname.lastname@intel.com
[3] STMicroelectronics, Diegem, Belgium firstname.lastname@noekeon.org

Abstract. This paper introduces the KOALA PRF, which maps a variable-length sequence of 64-bit input blocks to a single 257-bit output block. Its design focuses on achieving low latency in its implementation in ASIC. To construct KOALA, we instantiate the recently introduced Kirby construction with the KOALA-P permutation and add an input encoding layer. The KOALA-P permutation is obtained as the 8-fold iteration of a simple round function inspired by that of Subterranean. Based on careful preliminary cryptanalysis, we made a variant of the Subterranean permutation by reordering and modifying it in a way that does not introduce any implementation overhead and enhances the cryptographic resistance of the resulting PRF. Indeed, we demonstrate that KOALA exhibits a high resistance against integral, cube, division property, and higher-order differential attacks. Additionally, we compare the hardware implementation of KOALA with the smallest latency with state-of-the-art low-latency PRF ORTHROS and GLEEOK and the block cipher PRINCE in the same ASIC synthesis setup. Our results show that KOALA outperforms these primitives not only in terms of latency but also with respect to various other performance metrics.

Keywords: PFR · pseudorandom function · cryptographic permutation · Kirby construction · low latency · integral attacks

1 Introduction

The design of cryptographic primitives with minimum evaluation time in hardware implementation, so-called low-latency cryptography, is a relatively young line of research. Modern digital technologies often require a high level of security, but are expected to operate within very short timeframes.

Important examples of such technologies are memory encryption and integrity mechanisms provided by, for example, IBM's SecureBlue, Intel's SGX, and AMD's SEV. Smart cards, like the ones of NXP and STMicroelectronics, perform local memory encryption in an ultra-constrained setting.

Another example is formed by the secure caches in modern CPUs. This application has received significant attention in the last few years, for microarchitectural attacks, e.g., Meltdown and Spectre, have revealed serious security

shortcomings in widely deployed high-end processors. Many hardware-based mitigations for such attacks call for a higher level of encrypted communication inside of CPUs, as well as between CPUs and their surrounding hardware components. To implement new features of this kind in the next generations of mainstream processors, without causing a large performance penalty, low-latency encryption primitives are among the most important building blocks. Suffice it to say that the design of low-latency primitives is an important domain of research.

While various primitives have been developed with a focus on low latency, a significant portion of them are (tweakable) block ciphers. We just mention PRINCE [9,10], MANTIS [5], QARMA [2], SPEEDY [29], BIPBIP [6] and SCARF [12].

Interestingly, in recent times low-latency pseudorandom functions (PRF) have been proposed in the form of ORTHROS [4] and GLEEOK [1], allowing ultra-fast stream encryption or authentication of short messages. Both are based on the sum-of-block-ciphers paradigm: To achieve beyond birthday bound PRF security, in the former the output is the sum of two block cipher invocations and in the latter even three. We investigate a different way to achieve n bits of security, namely with a $2n$-bit permutation and a feedforward, leading to a more efficient implementation in terms of area and latency.

In this paper, we present the design of a PRF with a variable-length input and fixed-length output suitable for a low-latency implementation in hardware as an ASIC. KOALA is an instantiation of the Kirby [30] construction with a new permutation KOALA-P inspired by Subterranean [13,19], and an additional input encoding. Moreover, KOALA can be used as a stream cipher by taking as input the nonce followed by a counter. For this cipher the marginal cost per 256-bit keystream block is one call to KOALA-P. We compare the performance of KOALA with that of ORTHROS and the two instances of GLEEOK, to the best of our knowledge the only PRFs in the literature with the main goal of providing a low-latency ASIC implementation. Our synthesis results, in Table 4, show that KOALA has a lower latency and outperforms ORTHROS and GLEEOK in various other performance measures. We believe that KOALA is a promising new addition to the family of low-latency cryptographic primitives, and we welcome any third-party cryptanalysis.

Contribution. The main contributions of this paper are as follows:

- The design of KOALA, a low-latency PRF that maps a variable-length sequence of 64-bit input blocks to a single 257-bit output block.
- An integral cryptanalysis of KOALA, using bit-based division properties and the open source implementation of all algorithms used.[1]
- An RTL design of KOALA in Verilog, an evaluation of the corresponding ASIC performance, and a comparison with ORTHROS GLEEOK and PRINCE.

Organization of the paper. The paper is organized as follows. We establish some notation and conventions in Section 2. In Section 3, we present the specification of the permutation KOALA-P, the pseudorandom function KOALA, and the

[1] https://github.com/parisaeliasi/KoalaHW

security claim of KOALA. We present a short formalism to describe conditional cube attacks in Section 4 and in Section 5, we use this formalism and bit based integral distinguisher to analyse KOALA. Bounds on the weights of linear and differential trails over KOALA-P are provided in Section 6. In Section 7, we provide the rationale for all components of KOALA. Finally, we present a hardware implementation in Verilog in Section 8, together with area and latency figures for implementation in ASIC. Appendices contains some figures, missing proofs for the interested reader, along with avalanche behaviour and differential, linear and integral distinguishers.

2 Notation and Conventions

We fix the notation and conventions that are used throughout the paper.

We denote the cardinality of a set S by $|S|$. For sets S and T, we write $\mathrm{Funcs}[S,T]$ for the set of all functions from S to T. By the set $\mathbb{N}$ we mean the non-negative integers, i.e, $0 \in \mathbb{N}$. Typically, n and m denote elements of $\mathbb{N}$. A finite sequence $s = (s_0, s_1, \ldots, s_{n-1})$ of elements of a set S is called an n-tuple. In particular, we reserve the word (bit) string for n-tuples over the set $\{0,1\}$. We may also call a bit string a block if it has a fixed length of either 64 or 257 bits. The n-bit string consisting of all ones is denoted as 1^n. When we endow the set $\{0,1\}$ with the structure of a finite field, we write $\mathbb{F}_2$ instead. Indices of tuples are computed modulo n, i.e., these indices are assumed to be elements of $\mathbb{Z}/n\mathbb{Z}$. If s and s' are two bit strings, then we write $s||s'$ for the concatenation of s and s'. For example, $(0,0,1)||(1,0,1)$ is equal to $(0,0,1,1,0,1)$. We often treat n-bit strings as (bit) vectors $u = (u_0, u_1, \ldots, u_{n-1})$ in the n-dimensional vector space $\mathbb{F}_2^n$ over the field $\mathbb{F}_2$. We write e_i for the ith standard basis vector of $\mathbb{F}_2^n$. That is to say, e_i has a 1 in position i and zeros elsewhere. Sometimes, we refer to vectors as points. We make the set $\mathbb{F}_2^n$ into a partially ordered set by defining $u \leq v$ if and only if $u_i \leq v_i$ for all $i \in \mathbb{Z}/n\mathbb{Z}$. The Hamming weight of a vector u is defined as the number of its non-zero coordinates. That is, $\mathrm{HW}(u) = |\{i : i \in \mathbb{Z}/n\mathbb{Z} \wedge u_i \neq 0\}|$. We define an affine subspace of $\mathbb{F}_2^n$ to be any set of the form $a + L$, where $a \in \mathbb{F}_2^n$ is a point and L is a linear subspace of $\mathbb{F}_2^n$. Let S be a subset of $\mathbb{F}_2^n$ and f a function defined on $\mathbb{F}_2^n$. We write $f|_S$ for the restriction of f to S.

3 Specification of KOALA

Our design consists of two layers of abstraction: a permutation called KOALA-P, and a PRF called KOALA, that consists of a prefix-free input encoding function and the instantiation of the Kirby construction with KOALA-P.

First, in Section 3.1, we recall the Kirby construction, introduced in [30]. Second, we specify the KOALA-P permutation in Section 3.2. Third, we present the specification of the KOALA PRF in Section 3.3 and its security claim in Section 3.4.

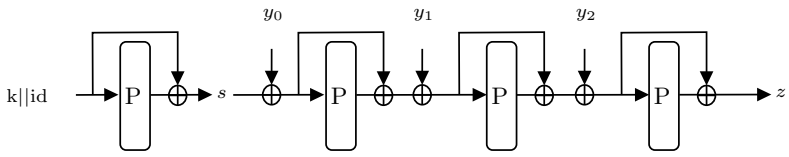

Fig. 1: Illustration of Kirby applied to a 3-tuple of input blocks.

3.1 The Kirby Construction

Kirby is a construction for building a variable-input-length pseudorandom function (VIL-PRF) from a permutation. This construction is specified in Algorithm 1 and illustrated in Figure 1.

To summarize Algorithm 1, Kirby is parameterized by a b-bit permutation P and a key length κ. It operates on a b-bit state that is initialized with a κ-bit secret key k and a $(b-\kappa)$-bit identifier. Then, it alternates between absorption of b-bit input blocks and transformations of the state by means of a call to the permutation P and a feed-forward. The input tuple of strings is assumed to be a codeword in a prefix code [14] (sometimes called a prefix-free code). It returns the final value of the b-bit state as the output. The paper [30] contains a proof of multi-user PRF security in the random permutation model, i.e., security against *generic* attacks.

From now on, we use the term key to refer to the master key k and secret to any intermediate state unknown to the attacker.

Algorithm 1 Definition of construction Kirby[P, κ] copied from [30], where P is a b-bit permutation and κ is a positive integer.

Input
- k — A κ-bit key string.
- id — A $(b-\kappa)$-bit key identifier string.
- y — An n-tuple of b-bit blocks with $n \geq 1$.

Output
- z — A b-bit block.

$s \leftarrow \text{k}||\text{id}$
$s \leftarrow s \oplus \text{P}(s)$
for $i = 0$ to $n-1$ **do**
  $s \leftarrow s \oplus y_i$
  $s \leftarrow s \oplus \text{P}(s)$
end for
$z \leftarrow s$
return z

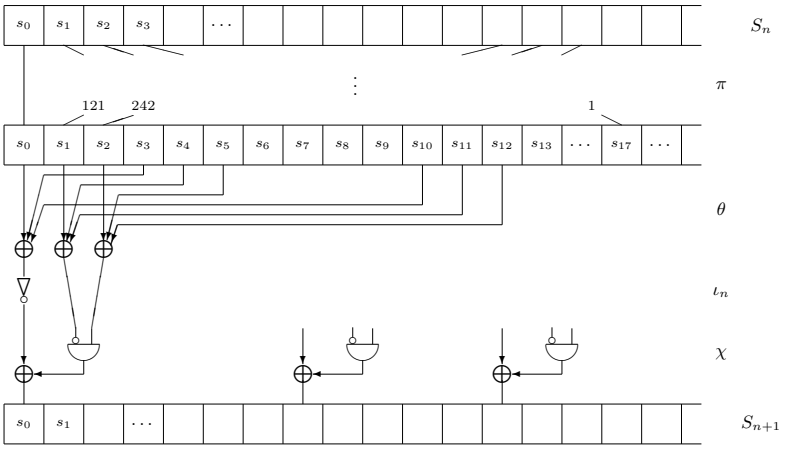

Fig. 2: The KOALA-P round function.

3.2 The KOALA-P Permutation

KOALA-P is a permutation of $\mathbb{F}_2^{257}$ parameterized by the number of rounds $r \leq 8$. It is obtained by the self-composition of a round function, which, in turn, consists of a sequence of step functions.

First, we introduce the step functions: a bit shuffle π, a mixing layer θ, a round constant addition ι_j, and a non-linear layer χ. These functions are defined by how they compute the bit with index $i \in \mathbb{Z}/257\mathbb{Z}$ of a state vector $s \in \mathbb{F}_2^{257}$ according to the following rules:

$$\begin{aligned}
&\pi : s_i \leftarrow s_{121i}\,,\\
&\theta : s_i \leftarrow s_i + s_{i+3} + s_{i+10}\,,\\
&\iota_j : s_i \leftarrow \begin{cases} s_i + 1 & \text{if } i = 0 \text{ and } j \notin \{2,5,6\}\,,\\ s_i & \text{otherwise}\,, \end{cases}\\
&\chi : s_i \leftarrow s_i + s_{i+2} + s_{i+1}s_{i+2}\,.
\end{aligned}$$

Second, we define the round function ρ_j parameterized by the round index j as

$$\rho_j = \chi \circ \iota_j \circ \theta \circ \pi\,.$$

Note that the only difference between the round functions lies in the value of the round constant. Lastly, we denote the composition of r rounds as

$$\text{KOALA-P}[r] = \bigcirc_{j=0}^{r-1} \rho_j\,.$$

3.3 The Koala PRF

The Koala PRF is composed of the following two parts:

- Kirby[Koala-P[8], κ]: the instantiation of Kirby with the permutation Koala-P[8] in which the key length κ is left as a parameter.
- An encoding function EncodePrefixFree defined in Algorithm 3 that maps an n-tuple of 64-bit blocks into a n-tuple of 257-bit blocks.
- An encoding of k and id as k||id||lenght(id) instead of k||id in the original Kirby construction, with length the encoding of the bitlength of id encoded in a single byte.

The ExpandBlock function makes it possible to use 64-bit blocks as input to the Kirby instance. Each 64-bit input block is transformed into a 256-bit string by the ExpandBlock function defined in Algorithm 2. Every 64-bit input block is split into a sequence of 32 2-bit strings. Each of these 2-bit strings naturally encodes an integer value between 0 and 3. This value serves as an index of the single non-zero element in a 4-bit string. The bits of this string are then diffused to different positions of the corresponding 256-bit output string.

The 256-bit strings are each padded with the bit 0, except for the last string, which is padded with the bit 1. This padding is what guarantees that the input tuple y to Algorithm 1 is an element of a prefix code.

The encoding of the k and id is injective, allowing use of different lengths without risking state collision for different keys. Concretely, given a κ-bit key, k, a $(257-\kappa)$-bit key identifier, id, and an n-tuple, d, of 64-bit blocks, we define Koala by

$$\text{Koala}[\kappa](\text{k}, \text{id}, d) = \text{Kirby}[\text{Koala-P}[8], \kappa](\text{k}, \text{id}, \mathsf{EncodePrefixFree}(d)) .$$

3.4 The Koala Security Claim

We present a claim of multi-user PRF security of Koala in the case of μ users. We assume the existence of s identifiers and suppose that μ_i users share the ith identifier. Hence, we have $\mu = \mu_1 + \cdots + \mu_s$.

Claim 1 *We consider an adversary that is restricted to the following resources:*

- *The computational complexity is N and it is equal to the number of evaluations of* Koala-$P[8]$.
- *The data complexity is M and it is equal to the number of distinct input blocks that are processed by* Koala.

The advantage of an adversary in distinguishing an array of μ instances of Koala$[\kappa]$ *loaded with μ independent κ-bit keys, sampled randomly and uniformly, from an array of μ independent random oracles, is upper bounded by*

$$\frac{M(M-1)}{2^{257}} + \frac{2NM}{2^{257}} + \frac{\sum_{i=1}^{s} \mu_i(\mu_i - 1)}{2^{\kappa+1}} + \frac{N \max_i \mu_i}{2^{\kappa}} .$$

This claim follows the proven bound of Kirby in [30] Lemma 1 page 15 against generic attacks using a κ-bit key.

Algorithm 2 Definition of ExpandBlock.

Input
s A 64-bit block.
Output
t A 256-bit string.

for $i = 0$ to 255 **do**

$$t_i \leftarrow \begin{cases} (s_{2i}+1)(s_{2i+1}+1) & \text{if } i \in [0,31]\,, \\ (s_{2i}+1)s_{2i+1} & \text{if } i \in [32,63]\,, \\ s_{2i}(s_{2i+1}+1) & \text{if } i \in [64,95]\,, \\ s_{2i}s_{2i+1} & \text{if } i \in [96,127]\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

where the indices of s are computed modulo 64.

end for
return $(t_0, t_1, \ldots, t_{255})$

Algorithm 3 Definition of EncodePrefixFree

Require: $n \geq 1$
Input
d An n-tuple of 64-bit blocks.
Output
y An n-tuple of 257-bit blocks.

for $i = 0$ to $n-2$ **do**
$y_i \leftarrow \mathsf{ExpandBlock}(d_i)||0$
end for
$y_{n-1} \leftarrow \mathsf{ExpandBlock}(d_{n-1})||1$
return $(y_0, y_1, \ldots, y_{n-1})$

4 Formalism for Integral Cryptanalysis

Together with differential and linear cryptanalysis, integral cryptanalysis form the three most important attack vectors. We use *integral attacks* as an umbrella term for attacks relying on summing the outputs of a function over a well-chosen input set, using different heuristics for constructing the set. To improve the understandability of our explanations of the attacks mounted against KOALA, we first describe the general method used for integral attacks with the minimum mathematical formalism necessary to describe such attacks.

4.1 Framework of Integral Attacks

Integral attacks consist of an offline phase followed by an online phase:

Offline Phase is an analysis step where the adversary accesses the secret dependent polynomial representations of the step functions of a primitive. They apply rewrite rules to these polynomial representations in order to simplify them, e.g., to eliminate variables and lower the degree. Importantly, the rewrite rules determine an affine input space V. Using combinatorial arguments involving the degree or by propagating an initial division property vector [37], the adversary is able to determine the vector of coefficients of some target monomial. To be able to mount a successful attack, this vector should either be a constant that does not depend on the secret at all or depend on the secret in a way that leads to a system of equations that is easy to solve, e.g., linear dependence. The outcome of this step is an affine input space V and a target monomial x^u.

Online Phase is an execution step where the adversary accesses a cryptographic oracle for a fixed master key. They recover the vector of coefficients of the target monomial x^u by summing over the affine input space V that was obtained during the offline phase. The vector of coefficients is then used as a distinguisher or to set up a system of equations in the secret bits that may lead to the recovery of the master key.

We restrict ourselves to input sets that form an affine space. Within this restriction, examples of integral attacks include higher-order differential cryptanalysis [28], square attacks [18], and (conditional) cube attacks [21, 27]. We present a unified mathematical foundation upon which these attacks are built.

This section is organized as follows. In Section 4.2, we make explicit the link between functions defined on an affine space and their representation on this space as a multivariate polynomial, called the algebraic normal form. In Section 4.3, we introduce an notion of the derivative of a function and show how it can be computed by means of the summation of outputs of the function.

4.2 Algebraic Normal Form

To understand how to find input spaces for an integral attack, we need to explain how to represent the restriction of a vectorial Boolean function to some affine space as a tuple of multivariate polynomials: the algebraic normal form (ANF). We present the necessary tools and results from computational commutative algebra and make the relation between the algebraic normal form and substitutions, which determine the input sets, explicit. We also illustrate in Example 1 the notation and terminology used. For an accessible introduction to computational commutative algebra, we refer to the book by Cox et al. [15].

A monomial in the variables $x_0, \ldots, x_{n-1}$ is a product of the form $x_0^{u_0} \cdots x_{n-1}^{u_{n-1}}$ with $u \in \mathbb{N}^n$. To abbreviate, we write this as x^u. The degree of the monomial x^u is defined as $u_0 + \cdots + u_{n-1}$. Polynomials are finite linear combinations of monomials with coefficients in $\mathbb{F}_2$. The degree of a polynomial is the largest of the degrees of its monomials. The zero polynomial has degree $-\infty$. We denote the set of polynomials in the variables $x_0, \ldots, x_{n-1}$ and with coefficients in

$\mathbb{F}_2$ by $R_n = \mathbb{F}_2[x_0, \ldots, x_{n-1}]$. These variables correspond to the bits which are controlled by an adversary, e.g., the input bits.

Let $p_0, \ldots, p_{n-1}$ be polynomials of the form $p_i = x_i$ or $p_i = c_i + \sum_{j=i+1}^{n-1} a_{ij} x_j$ for constants $c_i \in \mathbb{F}_2$ and coefficients $a_{ij} \in \mathbb{F}_2$. During cryptanalysis, we make use of a set of rewrite rules of the form $x_i \to p_i$, i.e., we *substitute* x_i with the polynomial p_i. Rules of the form $x_i \to x_i$ are said to be *trivial* in the sense that no substitution is performed. A set of rewrite rules defines a set of polynomials of the form $x_i - p_i$, which is completely specified by a tuple (A, c), where $A = (a_{ij})$ is an $n \times n$ matrix over $\mathbb{F}_2$ and $c = (c_0, \ldots, c_{n-1})$ is a vector in $\mathbb{F}_2^n$. The matrix A is in row echelon form, up to a permutation of its rows, which implies that the order in which the corresponding rewrite rules are applied does not matter. The tuple (A, c) defines the affine space $V = \{v \in \mathbb{F}_2^n : Av = c\}$ of points that satisfy the equation $Av = c$.

We have seen that a rewrite rule of the form $x_i \to p_i$ gives us a relation of the form $x_i = p_i$. Moreover, we have relations of the form $x_i^2 = x_i$ due to the fact that the square on $\mathbb{F}_2$ is the identity map. We can introduce these relations by working with polynomials modulo the ideal I generated by the set

$$G = \{x_0^2 - x_0, \ldots, x_{n-1}^2 - x_{n-1}, x_0 - p_0, \ldots, x_{n-1} - p_{n-1}\}\,.$$

For our purposes, the central algebraic object is the quotient ring R_n/I.

Polynomials in R_n give rise to elements of $\mathrm{Funcs}[V, \mathbb{F}_2]$. Indeed, for any point $a \in V$, there is a unique ring homomorphism $\varepsilon_a \colon R_n \to \mathbb{F}_2$ with $\varepsilon_a(x_i) = a_i$ given by substituting x_i by a_i. This leads to a map $\phi \colon R_n \to \mathbb{F}_2^V$ that is defined by $\phi(p) = f$ with $f(a) = \varepsilon_a(p)$ for all $a \in V$. The kernel of ϕ is equal to I. By the first isomorphism theorem for rings [15, p. 247], there is an isomorphism $\overline{\phi}$ between $\mathbb{F}_2^V$ and R_n/I.

The set G forms *a* Gröbner basis [15, p. 78] for I with respect to the lexicographic order. Define $W = \{u \in \mathbb{F}_2^n : u_i = 0 \text{ if } x_i \neq p_i\}$ as the set of vectors for which the ith component is zero if x_i is eliminated by a substitution. The remainder of any polynomial $p \in R_n$ on division by G, denoted $\overline{p}^G$, is unique and of the form

$$\overline{p}^G = \sum_{u \in W} \alpha_u x^u\,,$$

for certain constant bits $\alpha_u \in \mathbb{F}_2$ [15, p. 83]. Therefore, the set of all possible remainders after division by G, which we denote as R_G, forms a complete set of coset representatives of I in R_n. Indeed, let $\psi \colon R_n \to R_n$ be defined by $\psi(p) = \overline{p}^G$ for all $p \in R_n$. The kernel of ψ is equal to I. By the first isomorphism theorem for rings, there is an isomorphism $\overline{\psi}$ between R_n/I and R_G.

To conclude, we have an isomorphism $\mathcal{N} = \overline{\psi} \circ \overline{\phi}$ between the set of Boolean functions defined on V and the set of remainders R_G. We are now able to make precise how a function is represented on V.

Definition 1. *Let $f \colon V \to \mathbb{F}_2$ be a Boolean function defined on V. The representation of f as a multivariate polynomial, called the algebraic normal form (ANF) of f, is defined as the unique remainder $\mathcal{N}(f)$ upon division by G.*

The degree of a remainder $p \in R_G$ with $p \neq 0$ is defined as $\deg(p) = \max\{\mathrm{HW}(u) : u \in W \text{ and } \alpha_u \neq 0\}$.

Definition 2. *Let $f\colon V \to \mathbb{F}_2$ be a Boolean function defined on V. The algebraic degree of f, denoted by $\deg(f)$, is defined as the degree of its ANF.*

If f depends on a secret vector $s \in \mathbb{F}_2^\kappa$, e.g., a secret key or state, then the coefficients α_u of $\mathcal{N}(f)$ are Boolean functions of the secret bits, i.e., α_u maps the secret s to some bit $\alpha_u(s) \in \mathbb{F}_2$. In this case, we can rewrite the definition of the degree as $\deg(f) = \max\{\mathrm{HW}(u) : u \in W \text{ and there exists an } s \in \mathbb{F}_2^\kappa \text{ with } \alpha_u(s) \neq 0\}$. Note that our definitions match with the usual definitions of ANF and algebraic degree in the case that both A and c are zero.

There is a straightforward generalization of these notions to vectorial Boolean functions defined on V.

Definition 3. *The algebraic normal form of $F = (f_0, \ldots, f_{m-1})\colon V \to \mathbb{F}_2^m$ is defined as $\mathcal{N}(F) = (\mathcal{N}(f_0), \ldots, \mathcal{N}(f_{m-1})) \in R_n^m$. Its algebraic degree is defined as $\deg(F) = \max\{\deg(f_0), \ldots, \deg(f_{m-1})\}$.*

We illustrate how to apply rewrite rules to $\mathcal{N}(f)$, where f is some Boolean function, in order to change its properties, such as the presence of certain monomials. The resulting polynomial is the ANF of the restriction of f to the affine space determined by the rewrite rules.

Example 1. The function $f\colon \mathbb{F}_2^3 \to \mathbb{F}_2$ is defined by the truth table in Table 1. It follows that $\mathcal{N}(f)(x_0, x_1, x_2) = x_0 + x_2 + x_1x_2$. Therefore, the algebraic degree of f is 2. Now we make the isomorphism $\mathcal{N}$ implicit.

Table 1: Truth table of f.

x	000	001	010	011	100	101	110	111
$f(x)$	0	1	0	0	1	0	1	1

We apply the rewrite rule $x_1 \to x_2$. This rule, together with the trivial rules, defines the matrix

$$A = \begin{pmatrix} 0\,0\,0 \\ 0\,1\,1 \\ 0\,0\,0 \end{pmatrix}.$$

and the constant $c = (0, 0, 0)$. Clearly, A is in row echelon form, up to a permutation of its rows. Moreover, $V = \{v \in \mathbb{F}_2^3 : Av = 0\} = \{v \in \mathbb{F}_2^3 : v_1 = v_2\}$. When we restrict f to V, i.e., when we consider $f|_V\colon V \to \mathbb{F}_2$, we find that its ANF is equal to x_0. The restriction has algebraic degree 1 and it depends on a single variable.

An alternative way of wording this is that we compose f with the map $L\colon \mathbb{F}_2^2 \to V$ given by $(x_0, x_1) \mapsto (x_0, x_1, x_1)$ and that the algebraic normal form of $f \circ L$ is equal to x_0.

Like in the example, we make implicit in the nexts section the correspondence between Boolean functions and their representation as a tuple of remainders.

4.3 Properties of Derivatives

The integral attacks that we consider in this section, rely on practically computable properties of the derivative of a Boolean function. All definitions and results are extended to the case of vectorial Boolean functions by applying them to each coordinate Boolean function.

Definition 4. *For vectors $u, v \in \mathbb{F}_2^n$, define the derivative of the monomial x^v with respect to u by*

$$\partial_u x^v = \begin{cases} x^{v-u} & \text{if } u \leq v\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

and extend linearly to functions $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$. We call $\partial_u f$ the derivative of f with respect to u.

Note that this definition coincides with that of the usual partial derivative.

Example 2. Let $f\colon \mathbb{F}_2^3 \to \mathbb{F}_2$ be given by $f(x) = x_0 + x_2 + x_1x_2$. Its derivatives are equal to

$$\begin{aligned} \partial_{(0,0,0)} f(x) &= x_0 + x_2 + x_1x_2 & \partial_{(1,0,0)} f(x) &= 1 \\ \partial_{(0,0,1)} f(x) &= x_1 + 1 & \partial_{(1,0,1)} f(x) &= 0 \\ \partial_{(0,1,0)} f(x) &= x_2 & \partial_{(1,1,0)} f(x) &= 0 \\ \partial_{(0,1,1)} f(x) &= 1 & \partial_{(1,1,1)} f(x) &= 0 \end{aligned}$$

The first important property of the derivative is the duality between the derivatives of f and outputs of f on an affine space by means of integral.

Proposition 1. *Let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ and $a, u \in \mathbb{F}_2^n$. We have*

$$\begin{aligned} f(x+a) &= \sum_{0 \leq u \leq a} \partial_u f(x)\,, \text{ and} \\ \partial_u f(x) &= \sum_{0 \leq a \leq u} f(x+a)\,. \end{aligned}$$

See in Appendix A, Proposition 6 for the proof.

The following corollary shows how to compute the coefficient α_u of x^u in f by summing over the outputs of f corresponding to inputs for which u takes on all possible values.

Corollary 1. *Let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ and $a, u \in \mathbb{F}_2^n$. We have*

$$\alpha_u = \sum_{0 \leq a \leq u} f(a)\,.$$

Proof. This follows from the second equality in Proposition 1 and the fact that $\partial_u f(0) = \alpha_u$, by definition. □

The second important property of the derivative concerns its degree.

Proposition 2. *The degree of the derivative of f with respect to u satisfies*

$$\deg(\partial_u f) \leq \deg(f) - \mathrm{HW}(u).$$

Proof. By definition, we have $\partial_u f = \sum_{u \leq v} \alpha_v x^{v-u}$. Let w be such that $\alpha_w \neq 0$ and $\deg(\partial_u f) = \mathrm{HW}(w-u)$. Using that $u \leq w$ and that x^w is a monomial in f, we find that $\deg(\partial_u f) = \mathrm{HW}(w-u) = \mathrm{HW}(w) - \mathrm{HW}(u) \leq \deg(f) - \mathrm{HW}(u)$. □

The coefficient of any monomial x^u with the Hamming weight of u exceeding the degree of the function is 0.

Proposition 3. *If $\mathrm{HW}(u) \geq \deg(f(x))$, then $\partial_u f(x)$ is the coefficient α_u of x^u in f. In particular, if $\mathrm{HW}(u) > \deg(f(x))$, then this coefficient α_u is 0.*

Proof. If $\mathrm{HW}(u) \geq \deg(f(x))$, then $\deg(\partial_u f(x)) \leq 0$. This implies that $\partial_u f(x)$ is a constant, i.e., $\partial_u f(x) = \partial_u f(a)$ for any $a \in \mathbb{F}_2^n$. In particular, this is true for a equal to 0. By definition, it follows that $\partial_u f(0) = \alpha_u$. If $\mathrm{HW}(u) > \deg(f(x))$, then $\deg(\partial_u f(x)) < 0$, which implies that α_u is 0. □

5 Integral Attacks Applied to Koala

In this section we focus on the class of integral attacks. They forms an important attack vector to consider in the analysis of Koala, due to the fact that the Koala-P round function has degree 2. In particular, we restrict ourselves to analyzing the substructure $\mathcal{E}_r$ of Koala in which only a single block is processed and consider a round-reduced version of Koala-P.

Definition 5. *Define an expansion function $\gamma\colon \mathbb{F}_2^{257} \times \mathbb{F}_2^{64} \to \mathbb{F}_2^{257}$ by*

$$\gamma(s, x) = (\mathsf{ExpandBlock}(x)||1) + s.$$

The substructure $\mathcal{E}_r\colon \mathbb{F}_2^{257} \times \mathbb{F}_2^{64} \to \mathbb{F}_2^{257}$ is given by

$$\mathcal{E}_r(s, x) = \gamma(s, x) + \text{Koala-P}[r](\gamma(s, x)).$$

The summary of the following is that we believe that $\mathcal{E}_r$ with the number of rounds $r \geq 6$ is secure against integral attacks.

We first investigate distinguishing bit-based division properties. The division property was introduced in [36] as a generalization of integral distinguishers. Based on previous works [20, 25, 26, 35, 37, 38], we created different tools to search for two-subset and two types of three-subset division properties within round-reduce version of Koala. Then, we look at cube and conditional cube distinguishers, exploiting the inner structure of the ExpandBlock function and

the round function to search for integral distinguishers with smaller input size. Based on the results found, we conjecture on the feasibility of key recovery attacks using those distinguishers. In both cases, the goal of the attack is to find an affine subspace V of $\mathbb{F}_2^{64}$, the domain of the 64-bit input string x, such that the ANF of $\mathcal{E}_r$ on this subspace has a coefficient that is independent of or linear in key variables, for some monomial in input variable.

5.1 Bit-based Division Property Analysis

We divided our work into two steps, first using the two-subset division property and then using different types of three-subset division property. For further explanation on the division property we refere to [36] for basic concepts and the two subset division property, and to [26] for the three subset division property.

Using the algorithm from [35], and the model from [20] for the two subset division property, we check whether distinguishers exist within the round-reduced version of KOALA. The model from [20] was very powerful to model the large state of KOALA, and combined with the algorithm from Sun et al., we managed to model the propagation of division trails and to compute the existence of distinguisher up to 6 rounds. This technique, consisting in modeling the propagation of division trail using linear constraint, can lead to false positive results due to the lossy modeling of the constraint. However, from a designer's point of view, finding no distinguisher is enough, as this model captures all valid two-subset division trails. We found some distinguishers for up to 5 rounds but none for 6, showing the absence of exploitable two-subset division property for 6 rounds.

Then, we look at the three-subset-division property. We also used the model from [20] to model the propagation of division trail combined with the algorithm from [38]. For a specific set of input, we could compute the coefficient of the monomial containing all input variables after a certain number of rounds.

The result obtained was the monomial's presence, absence, or unknown status for each output coordinate, meaning for the latter that either the tool did not find the result or that such input is unlikely to result in an exploitable distinguisher. Due to the degree 2 round function used and the result found with the two-subset division property, we assume that there are distinguishers for up to 5 rounds. Therefore, we investigate 5 and 6 rounds distinguishers with our three-subset division property tools. As for five rounds, the expected maximum degree is 64 (2^5 for the round function time 2 for the ExpandBlock). This mean that for all secrets s for each output bit coordinates $\bigoplus_{x \in \mathbb{F}_2^{64}} \mathcal{E}_5(s, x) = 0$, leading to a 5-round integral distinguisher. Therefore, we investigated for 6 rounds, and we found that with the same input set, there is no unexploitable distinguisher for each output coordinate. We assume that this result came from the presence of monomials of the form $\prod_{i=0}^{63} x_i \prod_{j \in J_p} s_j$ for each coordinate p after 6 rounds. Therefore, we search if some quadratic secret dependency, meaning $|J_p| = 2$, could lead to an exploitable distinguisher. For all pairs of secret-bit tested, we did not found any distinguisher

for 6 rounds. We provide in https://github.com/parisaeliasi/KoalaHW all code used to compute those results, and we give in Appendix C some of the affine space leading to distinguisher for reduced round version.

5.2 Conditional Cube Attack

To push the analysis further, we consider what happens when we restrict our view of $\mathcal{E}_r$ to non-trivial affine subspaces of $\mathbb{F}_2^{64}$. These affine subspaces are obtained by applying substitutions that limit the interaction of variables through the rounds. In other words, we looked at a subspace of the input vector space that can decrease the degree of the ANF of specific output coordinates.

A variable x_i is said to interact with a variable x_j in F if x_j appears in $\partial_{e_i} F$. When it does not interact with any other variable, we call it isolated.

Definition 6. *Let $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function and let $i \in \mathbb{Z}/n\mathbb{Z}$. We call the variable x_i* isolated *in F if $\deg(\partial_{e_i} F) \leq 0$, i.e., if the derivative is a constant. We call F* linear *with respect to a set of variables $x_{i_1}, \ldots, x_{i_l}$ if these variables are isolated in F. By linearization of F, we mean the application of substitutions, after which F is linear with respect to the remaining variables.*

Linearizing γ: The following proposition shows that linearization of γ with respect to the variables $x_{i_1}, \ldots, x_{i_l}$, by applying suitable substitutions that lead to an affine space V, causes the absence of the monomial $x_{i_1} \cdots x_{i_l}$ in $\mathcal{E}_r|_V$. Intuitively, this is a consequence of the function having a much lower algebraic degree when restricted to particular affine subspaces than it has on the entire vector space. For a proof, we refer to Appendix A.

Proposition 4. *Let $r \geq 0$, $l \geq 2^r + 1$, and $\{i_1, \ldots, i_l\} \subseteq \mathbb{Z}/64\mathbb{Z}$ be a subset of indices of size l. If V is an affine subspace of $\mathbb{F}_2^{64}$ such that $x_{i_1}, \ldots, x_{i_l}$ are isolated in $\gamma|_V$, then $\partial_{e_{i_1}+\cdots+e_{i_l}} \mathcal{E}_r|_V = 0$.*

Each monomial of degree 2 in γ is of the form $x_i x_{i+1}$ for some index $i \in \mathbb{Z}/64\mathbb{Z}$. To linearize such a monomial, i.e., to have it depend on only a single variable, we can restrict ourselves to substitutions of the form $x_i \to x_{i+1}$ and $x_i \to a$ for a constant $a \in \mathbb{F}_2$.

In other terms, linearization of γ fixes 32 of the 64 input variables x_i. Therefore, using Proposition 4 and choosing r equal to 5, we find a distinguisher over $\mathcal{E}_5$.

Linearizing γ and ρ_0: The following proposition shows how to decrease the number of variables that are involved in the target monomial, i.e., to decrease the size of the input set over which we need to sum to obtain the coefficient of this target monomial. For a proof, we refer to Proposition 8 in Appendix A.

Proposition 5. *Let $r \geq 1$, $l \geq 2^r$, and $\{i_1, \ldots, i_l\} \subseteq \mathbb{Z}/64\mathbb{Z}$ be a subset of indices of size l. If V is an affine subspace of $\mathbb{F}_2^{64}$ such that x_{i_1} is isolated in $\rho_0 \circ \gamma|_V$ and $x_{i_2}, \ldots, x_{i_l}$ are isolated in $\gamma|_V$, then $\partial_{e_{i_1}+\cdots+e_{i_l}} \mathcal{E}_r|_V = 0$.*

With Proposition 5, we sketch how to use a conditional cube attack [27] to recover particular bits of the secret. Let V_g be an affine subspace of $\mathbb{F}_2^{64}$ that depends on a guess $g \in \mathbb{F}_2^m$ for some subset of bits of the secret s. We call P_i the property that x_{i_1} is isolated in $\rho_0 \circ \gamma|_{V_g}$. The attack consits in finding such V_g for wich a correct secret guess will make P_i true and false for an incorrect guess. We obtain V_g by applying substitutions that depend on g. For example, $\mathcal{E}_r$ adds x_i to s_i, so if we apply the rewrite rule $x_i \to g_j$, where g_j is a guess for s_i, then a correct guess for s_i effectively removes the effect of s_i in any further processing. To verify our guess, we recover the coefficient of the monomial $x_{i_1} \cdots x_{i_l}$ in $\mathcal{E}_r|_{V_g}$ by means of summation in the online phase. We write $u \in_R S$ if u is randomly and uniformly selected from the set S, and for $s \in_R \mathbb{F}_2^{257}$, we have

$$g = (s_{i_1}, \ldots, s_{i_m}) \Longrightarrow \partial_{e_{i_1}+\cdots+e_{i_l}} \mathcal{E}_r|_{V_g}(s, \cdot) = 0\,,$$
$$g \neq (s_{i_1}, \ldots, s_{i_m}) \Longrightarrow \Pr\left(\partial_{e_{i_1}+\cdots+e_{i_l}} \mathcal{E}_r|_{V_g}(s, \cdot) \neq 0\right) \approx 1\,.$$

We used that technique first to analyze a simpler version of our scheme: an Even-Mansour construction [22] in which the permutation is a round-reduced variant of the Subterranean permutation. As both elements are already well known, this was the starting point of our design. We found a key recovery attack for 6 rounds, using 32 isolated variables. The attack led to the recovery of key bits 0 and 2, requiring three days of computation on a desktop computer and it is possible to use this attack to recover each pair of bits; each can be performed in parallel. Consequently, a theoretical attack on 7 and 8 rounds exist using respectively 64 and 128 isolated variables. Those attacks would work the same with the KOALA-P permutation instead of the Subterranean permutation as the components of the round function are very similar. However, together with the ExpandBlock function, we did not manage to attack the same number of rounds. The restriction from 257 to 64 bits for the input reduces the number of possible input sets for the attacker. With the degree 2 ExpandBlock function, it also reduces the number of rounds required to reach the maximum degree term in the ANF. Based on our observation, by using linearization, we can obtain the degree estimation as shown in Table 10 in the Appendix C.

From the three-subset division property, we saw that the degree after 5 rounds reaches 64, following the upper bound. Attempts at attacking more than 5 rounds, the trivial input set containing all 64 input variables can be used as a distinguisher, or the input set with 32 variables chosen carefully to linearize the ExpandBlock function. However, none of those methods can attack 6 rounds as the input is too small. Linearizing the input injection and the first round would mean that after 6 rounds, it could be possible to find the output coordinate for which the maximum degree would be 32. To linearize one variable for the input injection and the first round, we need to set 10 variables to constant, meaning that, at most, 6 variables can be linearized for these two rounds. As for the conditional cube attack above, we investigate a combination of variables linearized for the ExpandBlock function and linearized for the ExpandBlock function and the first round. So, let's assume we linearize one variable for the input injection and the first round. Then, we have 54 variables left, and to linearize those for the

input injection, we reduce the space to 27. This leads us to think it is impossible to attack 6 rounds using this technique.

6 Trail Bounds of Koala-P

In this section, we present bounds on the weights of differential and linear trails over the Koala-P permutation. We support this analysis with the best linear and differential trails for up to 3 rounds in Appendix D. Since we introduced a new permutation Koala-P, we decided to investigate first the permutation alone without considering the input injection. However we believe that with the result provided and the restriction on the input to 64-bit due to the input ExpandBlock function, it is very unlikely that a 7/8-round differential with high enough probability or a 7/8-round linear approximation with high enough correlation could be found that would be useful in an attack on Koala.

6.1 Bounds on Differential Trails

To investigate the differential propagation properties of Koala-P, we used the differential trail search approach introduced in [32]. For more details, we refer the reader to [31,32]. The general idea of this approach is to generate all 2-round *trail cores* with high differential probability (DP) and extend them iteratively to longer trail cores. A trail core represents a set of trails that are equal in all intermediate differences and only their input and the output difference are different. The *restriction weight* $\mathrm{w_r}$ of a trail core is the minimum among the restriction weights of all trails in it. For a differential trail Q, we use this restriction weight to approximate the differential probability $\mathrm{DP}(Q)$. Hence, if $\mathrm{w_r} \ll b$ (the permutation width), then $\mathrm{DP}(Q) \approx 2^{-\mathrm{w_r}(Q)}$.

We report on the lower bounds on the restriction weights of trails for different numbers of rounds of Koala-P and also Subterranean in Table 2. The bounds for Koala-P are tight for up to 3 rounds since we scanned the space up to restriction weight 26.

For 4-round trails, we scanned the space up to restriction weight 51 and found there is no trail up to this weight. During our search, we found a 4-round trail with restriction weight 60, implying that the best 4-round trail should weigh between 52 and 60. This means that 4-round trail for Koala-P are likely to have a lower bound close the one for Subterranean. For 5, 6, and 7 rounds, we found no trails, but the space was scanned up to the limits listed in Table 2. Moreover, in the case of 8 rounds, since each 8-round trail can be divided into two 4-round trails and since all 4-round trails have weight at least 52, each 8-round trail has weight at least $2 \times 52 = 104$.

6.2 Bounds on Linear Trails

For the linear trail search we could not build further on a similar work for Subterranean as there are no results known. Instead, we adapted works on Simon

Table 2: Lower bounds on the restriction weight of differential trails in KOALA-P.

number of rounds	1	2	3	4	5	6	7	8
KOALA-P	2	8	26	$[52, 60]$	≥ 54	≥ 60	≥ 78	≥ 104
Subterranean	2	8	25	58	≥ 62	≥ 78	≥ 80	≥ 116

Table 3: Lower bounds on the correlation weight of linear trails in KOALA-P.

number of rounds	1	2	3	4
correlation weight	2	8	26	$[38, 54]$

and SPECK in [23, 33] to create a mixed integer linear programming (MILP) model for the propagation of linear masks. Our model provides lower bounds on the correlation weight of linear trails for a low number of rounds, where the correlation weight of a trail is the binary log of its correlation squared [16]. We use the Gurobi optimizer [24] to solve the model and find linear trails with the minimum correlation weight over 1,2,3 and 4 rounds.

During our trail search, we scanned the space up to correlation weight 26 and found a tight bound on the correlation weight of up to 3 rounds. For 4 rounds, we found a trail with the weight 54. Since the search is top-down, we only know that the minimum weight for a trail of 4 rounds is between 38 and 54. Table 3 represents the lower bounds on the correlation weight of up to 4 rounds.

6.3 Clustering

Trails may cluster, differential trails that have the same input and output differences contribute to the same differential. Similarly, linear trails that have the same input and output masks contribute to the same linear approximation. Even if each contribution is small, the sum of all the contributions might not be. Still, as studied in [8], in permutations such as KOALA-P, the maximum DP of differentials and the maximum correlation of linear approximations is typically very close to that of a single dominant trail. We decided to leave the study of clustering as future work.

7 Design Rationale of KOALA

This section presents the rationale behind the design of KOALA. The starting point of the design was the Even-Mansour construction, instantiated with 8 iterations of the Subterranean round function for use in counter mode. We selected the round function for its short critical path, consisting of one 2-bit `NAND` gate and three 2-bit `XOR` gates, together with an `INV` gate. As we alluded to in Section 5, the evolution from this initial design to KOALA has been driven by the goal of resistance against integral attacks. Indeed, when we allow an adversary

to inject 257-bit blocks, a practical attack exists on 6 rounds and a theoretical attack on 7 rounds. Hence, 8 rounds would not be sufficient. Instead of changing the number of rounds, we started looking for changes in the design preferably with no or small implementation overhead.

The first step was to consider the round function itself. We changed ι to remove symmetry between the rounds that could possibly be exploited in cryptanalysis, e.g., slide attacks [7]. We changed θ and π to increase the number of variables that appear in the derivatives of the first few rounds for any variable. Finally, we reversed the order of the step functions. In particular, we moved χ to the end of the round to increase the diffusion of input before the non-linear layer of the first round. Also, at the permutation output, any linear layer after the non-linear does not contribute to its cryptographic strength. The result of these changes is the Koala-P permutation. However, those modifications alone were not enough to prevent 6 and 7-round attacks working on Subterranean. The next step was to allow only injecting a single 64-bit input block into the state instead of a 257-bit block. This ExpandBlock function is tailored to the Koala-P permutation in the sense that they have been designed together to resist integral attacks as described in 5. The ExpandBlock function essentially cuts the dimension of any affine space that an adversary can inject into half, and its implementation cost in terms of additional gates and gate delay is small compared to that of an extra round. Due to the input restriction to 64-bit, we adopted the Kirby construction instead of Even-Mansour, as it allows for inputs consisting of an arbitrary number of 64-bit blocks. It is very suitable for low-latency applications, has a tight security bound in multi-user settings, and we handle the requirement of prefix-free input to Koala with a padding at the output of the ExpandBlock function.

8 Performance

We discuss a hardware architecture aimed for ASICs and report the synthesis results. The corresponding Verilog code and a software reference code for generating test vectors can be found at https://github.com/parisaeliasi/KoalaHW.

8.1 Hardware Architecture of Koala

The block diagram for Koala is illustrated in Figure 3. It has one 257-bit state register S, a combinational circuit for computing $h(s, \text{sqz}) := \mathsf{ExpandBlock}(s)||\,\text{sqz}$, a circuit for computing Koala-P, and control logic for absorbing and squeezing driven by two control signals: init and sqz.

- init $= 1$ the state is initialized with the image of key and id.
- init $= 0$ the operation is driven by sqz.
- sqz $= 0$ a non-final block absorbed, S updated and no output,
- sqz $= 1$ a final block absorbed, S not updated and output generated.

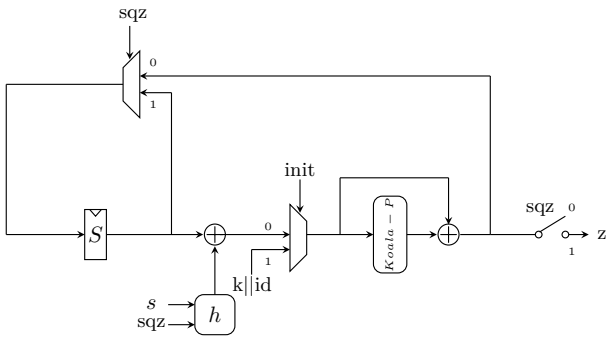

Fig. 3: Block diagram of the KOALA circuit.

The circuit guarantees that the input to KOALA is a prefix code by adding sqz in the input block, effectively indicating a final block. In stream cipher operation one first initializes the state, absorbs the blocks of the nonce with sqz = 0 and squeezes the keystream blocks by absorbing successive counter value blocks with sqz = 1. Four 2-bit `NOR` gates and two `INV` gates can encode the 2-bit input word (s_{2i} and s_{2i+1}) to a 4-bit output word, as explained in Algorithm 2. KOALA-P is implemented with a fully unrolled circuit, where the logic of the 8 rounds is replicated and chained. Unrolling is the natural strategy to achieve low-latency, since it allows the evaluation of the whole permutation in one clock cycle.

8.2 Hardware Results and Comparison

We compare KOALA with two other low-latency PRFs: ORTHROS [4] and GLEEOK [1]. Both provide 128-bit output blocks. For additional comparison, we also consider the 64-bit block cipher PRINCE [9] and an instantiation of KOALA where KOALA-P is replaced by 8 rounds of the Subterranean permutation denoted by KIRBY+sub.

For ORTHROS, GLEEOK, and PRINCE, we used the RTL code publicly available [3,11]. Note that these circuits are completely combinational. In fact, they do not need any flip-flop to store the intermediate cipher state. On the contrary, KOALA's circuit has the storage element S to support variable-length inputs. Nevertheless, KOALA has smaller area than ORTHROS and GLEEOK.

The RTL codes were synthesized with Cadence Genus version 21.15 using the standard cell library Nangate 15nm. We ran the synthesis flow multiple times for each cipher with different timing constraints, until the clock period is just above the critical path of the circuit. In Table 4, we report the best results in terms of maximum throughput/area^2 for each cipher. The maximum throughput (MaxTp) is intended here as the maximum number of bits that a circuit can output per second, and it is computed as output width divided by latency. More results, including the minimum latency reached by each cipher, are given in Table 13.

We can observe that Koala and Kirby+sub achieve the lowest latency and highest throughput among all ciphers and have similar area. This confirms that the modifications we made to Subterranean round function do not introduce significant implementation overhead while improving the security as shown in Section 5.2. Koala takes twice the area of Prince, but compensates this with a 257-bit output, 4 times longer than that of Prince. Gleeok-128 occupies twice the area of Koala and its output is 128 bits, only half that of Koala. While Gleeok-256 has block width similar to that of Koala, but its area is more than 5 times bigger. With respect to the metric MaxTp/Area^2, Prince achieves the best tradeoff, thanks to its very compact circuit. However, when we consider the lowest latency in Table 13, Koala outperforms Prince, Orthros and Gleeok thanks to its small area and larger output width.

Table 4: Synthesis results for the Nangate 15nm library.

Cipher	Output width	Area		Latency	MaxTp	MaxTp/Area
	[bits]	[μm^2]	[GE]	[ps]	[Gbits/s]	[Mbits/(s $\times$ μm^2)]
Koala	257	4175	21236	395	651	156
Kirby+sub	257	4167	21196	399	644	155
Prince	64	1696	8627	482	133	78.4
Orthros	128	5993	30482	400	320	53.4
Gleeok-128	128	9887	50291	400	320	32.4
Gleeok-256	256	26043	132462	550	465	17.8

9 Conclusion

With the design of Koala, we provide an open-source implementation of a new PRF for low-latency that performs much better than Orthrosand Gleeokon several metrics. The security analysis performed and supported with all open-source tools used, shows that using 8 rounds of Koala-P with the ExpandBlock should be secure against known attacks.

Acknowledgements

This work was partially funded by Intel through the Crypto Frontiers Research Center. Alireza Mehrdad, Joan Daemen, and Daniël Kuijsters are supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. Shahram Rasoolzadeh is supported by the Netherlands Organisation for Scientific Research (NWO) under TOP grant TOP1.18.002 SCALAR. Parisa Amiri Eliasi is supported by the Cryptography Research Center of the Technology Innovation Institute (TII), Abu Dhabi (UAE), under the TII-Radboud project with the title Evaluation and Implementation of Lightweight Cryptographic Primitives and Protocols. Silvia Mella was funded by the Dutch Research Council (NWO) through the PROACT project (NWA.1215.18.014).

References

1. Anand, R., Banik, S., Caforio, A., Ishikawa, T., Isobe, T., Liu, F., Minematsu, K., Rahman, M., Sakamoto, K.: Gleeok: A family of low-latency prfs and its applications to authenticated encryption. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(2), 545–587 (2024). https://doi.org/10.46586/TCHES.V2024.I2.545-587
2. Avanzi, R.: The QARMA block cipher family. almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Trans. Symmetric Cryptol. **2017**(1), 4–44 (2017)
3. Banik, S.: Orthros (2021), https://github.com/subhadeep-banik/orthros
4. Banik, S., Isobe, T., Liu, F., Minematsu, K., Sakamoto, K.: Orthros: A low-latency PRF. IACR Trans. Symmetric Cryptol. **2021**(1), 37–77 (2021)
5. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016)
6. Belkheyar, Y., Daemen, J., Dobraunig, C., Ghosh, S., Rasoolzadeh, S.: BipBip: A low-latency tweakable block cipher with small dimensions. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(1), 326–368 (2023)
7. Biryukov, A., Wagner, D.: Slide attacks. In: Knudsen, L. (ed.) Fast Software Encryption. pp. 245–259. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
8. Bordes, N., Daemen, J., Kuijsters, D., Assche, G.V.: 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)
9. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7658, pp. 208–225. Springer (2012)
10. Bozilov, D., Eichlseder, M., Knezevic, M., Lambin, B., Leander, G., Moos, T., Nikov, V., Rasoolzadeh, S., Todo, Y., Wiemer, F.: PRINCEv2 - more security for (almost) no overhead. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12804, pp. 483–511. Springer (2020)
11. Caforio, A.: Gleeok (2023), https://github.com/qantik/gleeok
12. Canale, F., Güneysu, T., Leander, G., Thoma, J.P., Todo, Y., Ueno, R.: SCARF - A low-latency block cipher for secure cache-randomization. In: Calandrino, J.A., Troncoso, C. (eds.) 32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023. USENIX Association (2023)
13. Claesen, L.J.M., Daemen, J., Genoe, M., Peeters, G.: Subterranean: A 600 mbit/sec cryptographic VLSI chip. In: Proceedings 1993 International Conference on Com-

puter Design: VLSI in Computers & Processors, ICCD '93, Cambridge, MA, USA, October 3-6, 1993. pp. 610–613. IEEE Computer Society (1993)
14. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. The MIT Press, 2nd edn. (2001)
15. Cox, D.A., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms. Undergraduate Texts in Mathematics, Springer, fourth edn. (2015)
16. Daemen, J.: Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis. K.U.Leuven (1995), http://jda.noekeon.org/
17. Daemen, J., Hoffert, S., Assche, G.V., Keer, R.V.: The design of Xoodoo and Xoofff. IACR Trans. Symmetric Cryptol. **2018**(4), 1–38 (2018)
18. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1267, pp. 149–165. Springer (1997)
19. Daemen, J., Massolino, P.M.C., Rotella, Y.: The Subterranean 2.0 cipher suite, 2019 (2019)
20. Derbez, P., Lambin, B.: Fast MILP models for division property. IACR Trans. Symmetric Cryptol. **2022**(2), 289–321 (2022). https://doi.org/10.46586/TOSC.V2022.I2.289-321
21. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. IACR Cryptol. ePrint Arch. p. 385 (2008)
22. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. J. Cryptol. **10**(3), 151–162 (1997)
23. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: Milp-based automatic search algorithms for differential and linear trails for Speck. 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. 268–288. Springer (2016)
24. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023)
25. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset - improved cube attacks against Trivium and Grain-128AEAD. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 466–495. Springer (2020). https://doi.org/10.1007/978-3-030-45721-1_17
26. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset. J. Cryptol. **34**(3), 22 (2021)
27. Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional cube attack on reduced-round Keccak sponge function. IACR Cryptol. ePrint Arch. p. 790 (2016)
28. Lai, X.: Higher order derivatives and differential cryptanalysis (01 1994)
29. Leander, G., Moos, T., Moradi, A., Rasoolzadeh, S.: The SPEEDY family of block ciphers engineering an ultra low-latency cipher from gate level for secure processor architectures. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(4), 510–545 (2021)
30. Lefevre, C., Belkheyar, Y., Daemen, J.: Kirby: A robust permutation-based prf construction. Cryptology ePrint Archive, Paper 2023/1520 (2023), https://eprint.iacr.org/2023/1520
31. Mehrdad, A., Mella, S., Grassi, L., Daemen, J.: Differential trail search in cryptographic primitives with big-circle chi: Application to Subterranean. IACR Trans. Symmetric Cryptol. **2022**(2), 253–288 (2022)

32. Mella, S., Daemen, J., Assche, G.V.: New techniques for trail bounds and application to differential trails in Keccak. IACR Trans. Symmetric Cryptol. **2017**(1), 329–357 (2017)
33. Shi, D., Hu, L., Sun, S., Song, L., Qiao, K., Ma, X.: Improved linear (hull) cryptanalysis of round-reduced versions of SIMON. Sci. China Inf. Sci. **60**(3), 39101:1–39101:3 (2017)
34. Stanley, R.P.: Enumerative Combinatorics, Cambridge Studies in Advanced Mathematics, vol. 1. Cambridge University Press, Cambridge, NY (2012)
35. Sun, L., Wang, W., Wang, M.: MILP-aided bit-based division property for primitives with non-bit-permutation linear layers. IET Inf. Secur. **14**(1), 12–20 (2020). https://doi.org/10.1049/IET-IFS.2018.5283
36. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 287–314. Springer (2015)
37. Todo, Y., Morii, M.: Bit-based division property and application to Simon family. IACR Cryptol. ePrint Arch. p. 285 (2016)
38. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: MILP method of searching integral distinguishers based on division property using three subsets. IACR Cryptol. ePrint Arch. p. 1186 (2018), https://eprint.iacr.org/2018/1186

Appendix

A Missing Proofs

Proposition 6. *Let* $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ *and* $a, u \in \mathbb{F}_2^n$. *We have*

$$f(x+a) = \sum_{0 \le u \le a} \partial_u f(x)\,, \text{ and}$$
$$\partial_u f(x) = \sum_{0 \le a \le u} f(x+a)\,.$$

Proof. The first equality can be seen as follows. Using the ANF of f, we find that

$$\begin{aligned} f(x+a) &= \sum_{0 \le w} \alpha_w (x+a)^w && = \sum_{0 \le w} \alpha_w \left(\sum_{0 \le u \le w} x^{w-u} a^u \right) \\ &= \sum_{0 \le w} \left(\sum_{0 \le u \le w} \alpha_w x^{w-u} a^u \right) && = \sum_{0 \le u} \left(\sum_{u \le w} \alpha_w x^{w-u} a^u \right) \\ &= \sum_{0 \le u} \left(\sum_{u \le w} \alpha_w x^{w-u} \right) a^u && = \sum_{0 \le u \le a} \left(\sum_{u \le w} \alpha_w x^{w-u} \right) \\ &= \sum_{0 \le u \le a} \partial_u f(x)\,, \end{aligned}$$

where we have applied the definition of the derivative and used the fact that $a^u = 1$ if and only if $0 \leq u \leq a$. The second equality follows from the Möbius inversion formula [34, p. 264] applied to the first. □

Proposition 7. *Let $r \geq 0$, $l \geq 2^r + 1$, and $\{i_1, \ldots, i_l\} \subseteq [0, 63]$ be a subset of indices of size l. If V is an affine subspace of $\mathbb{F}_2^{64}$ such that $x_{i_1}, \ldots, x_{i_l}$ are isolated in $\gamma|_V$, then*

$$\partial_{e_{i_1}+\cdots+e_{i_l}} \mathcal{E}_r|_V = 0\,.$$

Proof. This proof is in the same style as Theorem 2 from [27]. Let V be an affine subspace such that each x_{i_j} is isolated in $\gamma|_V$. By linearity, it suffices to prove both $\partial_{e_{i_1}+\cdots+e_{i_l}} \gamma|_V = 0$ and $\partial_{e_{i_1}+\cdots+e_{i_l}} \textsc{Koala-P}[r] \circ \gamma|_V = 0$. The first equality is trivial, so we only prove the second. To that end, let $f_1, \ldots, f_t$ be the monomials containing $x_{i_1}, \ldots, x_{i_l}$ in the output of $\gamma|_V$. By definition, the degree of each f_j is at most one with respect to $x_{i_1}, \ldots, x_{i_l}$. Now, any monomial T in $\textsc{Koala-P}[r] \circ \gamma|_V$ of maximum degree with respect to $x_{i_1} x_{i_2} \cdots x_{i_l}$ is of the form

$$T = f_1 f_2 \cdots f_h \quad \text{for some } h \in \mathbb{Z}_{\geq 0} \text{ with } h \leq 2^r\,,$$

because the algebraic degree of each ρ_j is 2. It follows that T contains at most h different $x_{i_1}, \ldots, x_{i_l}$. Suppose now that

$$\partial_{e_{i_1}+\cdots+e_{i_l}} T \neq 0\,.$$

Then $x_{i_1} \cdot x_{i_2} \cdots x_{i_l}$ divides T, which implies that $h \geq l$. Therefore,

$$h \geq l \geq 2^r + 1 > 2^r$$

which is a contradiction. Since T does not appear in the derivative, any lower degree monomials do not appear either and the result follows. □

Proposition 8. *Let $r \geq 1$, $l \geq 2^r$, and $\{i_1, \ldots, i_l\} \subseteq [0, 63]$ be a subset of indices of size l. If V is an affine subspace of $\mathbb{F}_2^{64}$ such that x_{i_1} is isolated in $\rho_0 \circ \gamma|_V$ and $x_{i_2}, \ldots, x_{i_l}$ are isolated in $\gamma|_V$, then*

$$\partial_{e_{i_1}+\cdots+e_{i_l}} \mathcal{E}_r|_V = 0\,.$$

Proof. This proof has been adapted from Theorem 2 from [27]. Let V be an affine subspace of $\mathbb{F}_2^{64}$ such that x_{i_1} is isolated in $\rho_0 \circ \gamma|_V$ and $x_{i_2}, \ldots, x_{i_l}$ are isolated in $\gamma|_V$. By linearity, it suffices to prove both $\partial_{e_{i_1}+\cdots+e_{i_l}} \gamma|_V = 0$ and $\partial_{e_{i_1}+\cdots+e_{i_l}} \textsc{Koala-P}[r] \circ \gamma|_V = 0$. The first equality is trivial, so we only prove the second. To that end, let $f_1, \ldots, f_s$ be the monomials containing x_{i_1} in $\rho_0 \circ \gamma|_V$. By definition, the degree of each f_i is exactly one with respect to x_{i_1}. Similarly, let $g_1, \ldots, g_t$ be the monomials containing $x_{i_2}, \ldots, x_{i_l}$ in $\rho_0 \circ \gamma|_V$. By definition the degree of each g_j is at most two with respect to $x_{i_2}, \ldots, x_{i_m}$. Moreover, x_{i_1} does not divide any g_j, because that would contradict the assumption that it is

isolated. Now, any monomial T in KOALA-P$[r] \circ \gamma|_V$ of maximum degree with respect to $x_{i_1}x_{i_2}\cdots x_{i_l}$ is of the form

$$T = f_1 f_2 \cdots f_h g_1 g_2 \cdots g_{h'} \quad \text{for some } h, h' \in \mathbb{Z}_{\geq 0} \text{ with } h + h' \leq 2^{r-1}\,,$$

because the algebraic degree of G is 2. It follows that T contains at most $2h'$ different $x_{i_2}, \ldots, x_{i_l}$ and at most one x_{i_1}. Suppose now that

$$\partial_{e_{i_1}+\cdots+e_{i_l}} T \neq 0\,.$$

Then $x_{i_1} \cdot x_{i_2} \cdots x_{i_l}$ divides T, which implies that $2h' \geq l$ and $h \geq 1$. Therefore,

$$h + h' \geq 1 + \frac{l}{2} = 1 + \frac{2^r}{2} = 1 + 2^{r-1} > 2^{r-1}\,,$$

which is a contradiction. Since T does not appear in the derivative, any lower degree monomials do not appear either and the result follows. □

B Diffusion Test

In Table 5 we use the definition of [17] for the avalanche dependency weight and entropy. We report on the avalanche behaviour for the Subterranean permutation, the KOALA-P permutation and the KOALA-P permutation with the input injection. We provide at https://github.com/parisaeliasi/KoalaHW the C code use to produce those result.

Table 5: Diffusion test for the Subterranean permutation, the KOALA-P permutation and the KOALA-P with the ExpandBlock function. For each we compute the dependency (D), the weight (W) and the entropy (E).

	Subterranean			Koala-P			Koala-P + ExpandBlock		
number of round	D	W	E	D	W	E	D	W	E
1	9	6.00	5.99	9	6.002	5.99	36	12.18	30.90
2	81	36.00	65.20	81	35.99	65.20	167	55.43	140.72
3	255	109.20	236.54	251	108.75	230.97	257	122.95	254.62
4	257	128.36	256.99	257	128.39	256.99	257	128.50	256.99

C Integral Distinguishers

We give in the Table 6, Table 8 and Table 9 integral distinguishers found with our tool for reduced-round versions of KOALA. We represent the input affine space used with the list of input variable indexes, and the output bit coordinate correspond to the output bit index after r round where this affine space leads to a

integral distinguisher. We also provide at https://github.com/parisaeliasi/KoalaHW all code to reproduce our result and to search for integral distinguisher.

Due to the input injection we can consider terms in monomials as a product of two input variable. Therefore, we know in advance that if x_{2i} appears than it will always be together with x_{2i+1}. This strongly reduces the search space for monomials.

D Differential and Linear Trails

In this section we provide the trail with least weight for 1, 2 and 3 round as found by our tools. We represent trails with a list of pair of indices, one for the round number and one for the index position within the state.

E Additional Hardware Results

In Table 13, we present more synthesis results and highlight the best result for each metric.

Table 6: 1 Round integral distinguisher

Affine Space	Output Bit Coordinate
[24, 25, 28, 29]	6, 233
[32, 33, 60, 61]	14, 44, 74
[0, 1, 36, 37]	48, 78
[16, 17, 52, 53]	184, 214
[2, 3, 38, 39]	65, 95
[8, 9, 12, 13]	97, 127
[10, 11, 14, 15]	114, 144, 174
[22, 23, 50, 51]	216, 246

Table 7: 2 Rounds integral distinguisher

Affine Space	Output Bit Coordinate
[24, 25, 28, 29, 32, 33, 60, 61]	157, 161, 164, 166, 168
[0, 1, 36, 37, 16, 17, 52, 53]	29, 33, 36, 39, 43
[2, 3, 38, 39, 22, 23, 50, 51]	62, 66, 69, 72, 73, 76

Table 8: 3 Rounds integral distinguisher

Affine Space	Output Bit Coordinate
[24, 25, 28, 29, 32, 33, 60, 61, 0, 1, 36, 37, 16, 17, 52, 53]	87, 94, 97, 155, 165, 206, 213, 216
[2, 3, 38, 39, 8, 9, 12, 13, 10, 11, 14, 15, 22, 23, 50, 51]	38, 200

Table 9: 4 Rounds integral distinguisher

Affine Space	Output Bit Coordinate
[24, 25, 28, 29, 32, 33, 60, 61, 0, 1, 36, 37, 16, 17, 52, 53, 2, 3, 38, 39, 8, 9, 12, 13, 10, 11, 14, 15, 22, 23, 50, 51]	15, 49, 54, 84, 91, 94, 239

Table 10: Upper bound on the degree growth based on the type of linearization used.

	number of rounds					
type of linearization	1	2	3	4	5	6
0 round	4	8	16	32	64	128
ExpandBlock function	2	4	8	16	32	64
ExpandBlock function + first round	1	2	4	8	16	32

Table 11: Differential trail for Koala

Round	Weight	Indexes
1	2	[1, 0]
2	6	[2, 0], [2, 247], [2, 254]
3	18	[3, 0], [3, 65], [3, 72], [3, 75], [3, 141], [3, 148], [3, 151], [3, 247], [3, 254]

Table 12: Linear trail for Koala

Round	Weight	Indexes
1	2	[1, 0]
2	6	[2, 0], [2, 106], [2, 182]
3	18	[3, 0], [3, 26], [3, 82], [3, 102], [3, 106], [3, 158], [3, 177], [3, 182], [3, 233]

Table 13: Extended synthesis results on Nangate 15nm.

Cipher	Output width [bits]	Area [μm^2]	Area [GE]	Latency [ps]	MaxTp [Gbits/s]	MaxTp/Area [Mbits/(s $\times$ μm^2)]	MaxTp/Area^2 [Gbits/(s $\times$ mm^4)]
Koala	257	4079.67	20750	472	544	133.34	32.715
		4175.07	21236	395	651	155.92	37.326
		5639.80	28686	300	857	151.95	26.933
		6621.41	33678	290	886	133.80	20.213
Kirby+sub	257	4167.30	21196	399	644	154.53	37.089
		5203.97	26469	300	857	164.68	31.633
		6035.42	30698	290	886	146.80	24.329
Prince	64	1696.19	8627	482	133	78.41	46.152
		1935.95	9847	450	142	73.38	37.947
		2957.03	15040	410	156	52.75	17.852
Orthros	128	5898.98	30004	499	257	43.57	07.372
		5978.75	30410	449	285	47.67	07.975
		5993.05	30482	400	320	53.39	08.910
		7295.73	37108	370	346	47.42	06.499
		8556.58	43521	360	356	41.60	04.856
Gleeok-128	128	9726.98	49474	436	294	30.22	03.103
		9887.61	50291	400	320	32.36	03.273
		13270.55	67498	370	346	26.07	01.964
Gleeok-256	256	25986.71	132175	600	427	16.43	00.632
		26043.19	132462	550	465	17.85	00.686
		29288.50	148969	520	492	16.79	00.574
		31468.54	160057	510	502	15.95	00.507