THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

EAGLESONG: AN ARX HASH WITH FAST DIFFUSION

Alan SZEPIENIEC[1], Tomer ASHUR[2,3]

[1] Nervos Foundation
[2] Cryptomeria
[3] imec-COSIC, KU Leuven
.....
Corresponding author: Alan SZEPIENIEC, E-mail: alan@nervos.org

We propose a hash function based on three design principles: the sponge construction, ARX operations, and the wide trail strategy. While the sponge construction applies generically to any sufficiently strong permutation, the wide trail strategy and the ARX operations are naturally somewhat incompatible. We show that while the ARX operations provide only very weakly nonlinear S-boxes, it *is* possible to build very strong linear diffusion layers with them. As a result, the wide trail argument, which bounds the attacker's success probability in terms of the minimum number of active S-boxes across two rounds, survives. The proposed hash function is one of a very select group of ARX ciphers featuring rigorous bounds against differential and linear cryptanalysis.

Key words: hash function, addition-rotate-xor, sponge, wide trail strategy, diffusion.

1 INTRODUCTION

In the design of symmetric ciphers, the *wide trail strategy* is a common technique offering a strong resilience against differential and linear cryptanalysis [7] and lies at the base of many popular and widely adopted ciphers such as the AES [6]. The technique entails alternating between a confusion layer, in which many highly nonlinear but localized functional blocks operate in parallel on a large state; and a diffusion layer, in which the localized effects of the previous layer are spread out across the entire state. The combined effect of both layers results in an upper bound on the differential and linear probabilities, thereby enabling the designer to estimate the number of rounds required to reach a target security level against the matching attacks.

An alternate design strategy increasingly popular particularly in the context of software-oriented ciphers is the minimalist reliance on only three instructions: Modular Addition, Cyclic Rotation, and Exclusive-or (`xor`), or ARX for short. While these operations do not generally offer the strong non-linearity required by the relatively hard-to-compute functional blocks in the wide trail strategy, ARX ciphers can afford to compensate with a larger number of rounds: compared to the highly

non-linear functional blocks of the wide trail strategy, the addition, rotation, and `xor` functions are relatively fast.

The ARX and wide trail design strategies seem to be fundamentally in opposition to each other. Popular ARX ciphers like Salsa [3], Blake [1], and Speck [2] drop rigorous bounds on the differential and linear probabilities altogether in favor of heuristic arguments. A notable exception to this list but still in support of the opposition of strategies, is the SPARX and LAX family of ciphers [9]. The designers of these families introduce an alternative to the wide trail strategy called the *long trail strategy*, which advocates using large and expensive functional blocks derived from many simple operations (such as ARX operations), coupled with a cheaper and weaker diffusion layer. The long trail strategy admits provable bounds on the linear and differential success probabilities.

Our contribution. We present Eaglesong, a hash function whose design successfully unifies the ARX and wide trail strategies. The key to this fusion is the algebraic interpretation of the addition, rotation, and `xor` operations. This algebraic perspective in turn enables the construction of a strong diffusion layer, a rigorous analysis of its properties, and the derivation of bounds on the linear and differential probabilities in accordance with which the number of rounds is set.

2 SPECIFICATION

Eaglesong is a sponge function. The innovation is in the construction of the permutation F. We revisit the standard construction of a hash function from a sponge function in Section 2.2 for the sake of a self-contained presentation. For other modes of operation derived from the sponge construction we refer the reader to the exposition by Bertoni *et al.* [5].

The Eaglesong permutation F operates on a state of 16 words of 32 bits by applying the round function $N = 43$ times for a 128 bit security level. The round function consists of four steps:

1. *Bit matrix*, with which words are mixed across the entire state. This diffusion is accomplished using an invertible 16×16 matrix over $\mathbb{F}_2$. The action of this matrix is realized with only the `xor` operation.

2. *Circulant multiplication*, in which bits are mixed within each word. This operation is realized with `xor`s and rotations, compatible with multiplication-by-constant in the quotient ring $\mathbb{F}_2[x]/\langle x^{32} + 1\rangle$.

3. *Injection of constants*, in which a predetermined list of random constants are `xor`ed into the state. The constants are different each round.

4. *Nonlinear map*, in which nonlinear operations are applied to the state elements. Here we use integer addition modulo 2^{32}, which is nonlinear with respect to vector spaces over $\mathbb{F}_2$. Modular addition is applied twice, before and after a rotation of the words by 8 bits in opposite directions.

A diagram overview of the round function is presented in Fig. 1.

Bit matrix.

The between-word diffusion layer applies a linear operation to the vector of state elements. Specifically, the operation on the state vector $\mathbf{e}$ is given by a matrix $M \in \mathbb{F}_2^{16\times 16}$ via $\mathbf{e}^\mathsf{T} \mapsto \mathbf{e}^\mathsf{T} M$. As this matrix consists of known elements in the coefficient ring $\mathbb{F}_2$, the multiplications involved in this linear transformation correspond to simply `xor`ing or ignoring the elements of the state vector depending on whether the matching element of the matrix is a 1 or a 0.

The matrix M is determined via the binary BCH code with $m = 5, \delta = 6, n = 31$, extension field $\mathbb{E} = \mathbb{F}_2[x]/\langle x^5 + x^2 + 1\rangle$, and generator polynomial $g(X) = X^{15} + X^{11} + X^{10} + X^9 + X^8 + X^7 +$

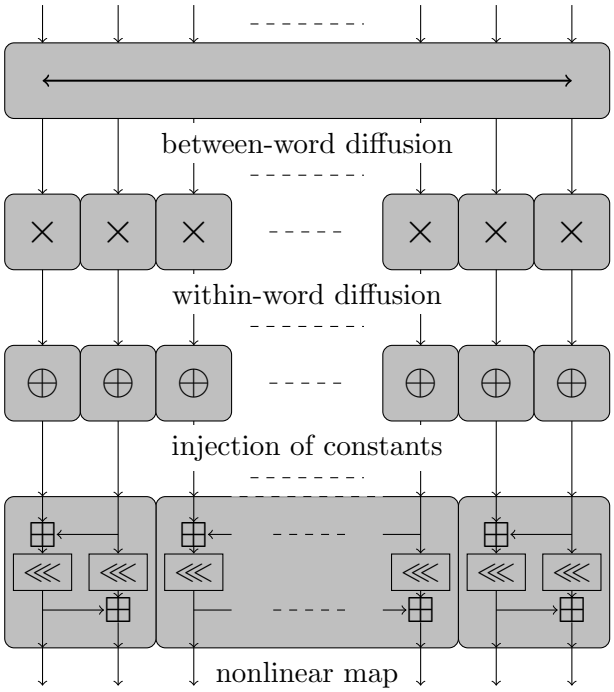


Figure 1: Eaglesong round function F.

$X^5 + X^3 + X^2 + X + 1$. As a linear code, this code is defined by a generator matrix $G \in \mathbb{F}_2^{16\times 31}$. Without loss of generality we may consider the echelon form of G, or in the lingo of coding theory, its systematic form: $G_s = \left(I \,|\, \bar{M}\right)$, where $\bar{M} \in \mathbb{F}_2^{16\times 15}$. Then we find M by adjoining to $\bar{M}$ the unique column vector $\mathbf{k} \in \mathbb{F}_2^{16}$ satisfying $\mathbf{k}^\mathsf{T}\bar{M} = \mathbf{0}$. In particular, M is given by:

$$M = \left(\begin{array}{cccccccccccccccc|c}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1
\end{array}\right) \tag{1}$$

An exhaustive search in the primal code $\{(\mathbf{x}^\mathsf{T}, \mathbf{x}^\mathsf{T}M) \,|\, \mathbf{x} \in \mathbb{F}_2^{16}\}$ and the dual code $\{(\mathbf{x}^\mathsf{T}, \mathbf{x}^\mathsf{T}M^\mathsf{T}) \,|\, \mathbf{x} \in \mathbb{F}_2^{16}\}$ shows that neither have nonzero codewords of Hamming weight less than 8. Moreover, the minimum weight of codewords in the primal and dual code is 5 with respect to the following alternative "pairwise Hamming weight" metric: partition the codeword into adjacent pairs of bits, and then count the number of pairs different from 00.

Circulant multiplication.

The within-word diffusion interprets each word as an element in the ring $\mathbb{F}_2[x]/\langle x^{32}+1\rangle$ and maps it to its product by a trinomial from the same ring. The known coefficient of this multiplication is given by $1+x^{a_i}+x^{b_i}$ where all $a_i \neq 0$ and $b_i \notin \{0, a_i\}$ are determined from SHAKE256 seeded with the ASCII string "*I thought of calling it 'information,' but the word was overly used, so I decided to call it 'uncertainty.' When I discussed it with John von Neumann, he had a better idea.*". We use rejection sampling to ensure that no coefficient is its own inverse, and that the Hamming weight of any product of coefficients is at least 7.

Trinomials are invertible under multiplication in the ring $\mathbb{F}_2[x]/\langle x^{32}+1\rangle$, meaning that this operation maps uniform inputs to uniform outputs. Moreover, given that multiplication by x^{a_i} represents rotation by a_i positions, the multiplication can be computed with two rotations (denoted by $\lll$) and two `xor`s (denoted by $\oplus$):

$$e_i \mapsto e_i \times (1 + x^{a_i} + x^{b_i}) = e_i \oplus (e_i \lll a_i) \oplus (e_i \lll b_i) \ .$$

The powers a_i and b_i for state element e_i are given by the ith elements of the arrays

```
a = [2,13,4,3,27,3,17,3,18,12,4,4,12,7,7,1]
b = [4,22,19,14,31,8,26,12,22,18,7,31,27,17,8,13]
```

This operation has a characterization in the language of coding theory: the set $\{(e_i, e_i \times (1 + x^{a_i} + x^{b_i})) \,|\, e_i \in \mathbb{F}_2[x]/\langle x^{32}+1\rangle\}$ represents a quasi-cyclic code of length $n = 64$ and dimension $k = 32$. Its minimal distance can be verified by way of brute force to be at least four.

Injection of constants.

In this layer, the constant c_i is `xor`ed into state element e_i. The c_i are determined from SHAKE256 seeded with the ASCII string "I have always been on the machines' side.". The are provided in Appendix A, which describes the initialization of the array `injection_constants`, which will be used in the pseudocode.

Nonlinear map.

The nonlinear map consists of three steps: odd-to-even addition, followed by rotation, followed by even-to-odd addition.[1] Specifically: first,

$$e_i \mapsto \begin{cases} e_i \boxplus e_{i+1} & \text{if } i \equiv 0 \bmod 2 \\ e_i & \text{otherwise,} \end{cases}$$

where $\boxplus$ represents addition modulo 2^{32}. Second,

$$e_i \mapsto \begin{cases} e_i \lll 8 & \text{if } i \equiv 0 \bmod 2 \\ e_i \lll 24 & \text{otherwise.} \end{cases}$$

Third,

$$e_i \mapsto \begin{cases} e_i \boxplus e_{i-1} & \text{if } i \equiv 1 \bmod 2 \\ e_i & \text{otherwise.} \end{cases}$$

This map is the only nonlinear component of the Eaglesong round function due to the additions of integers modulo 2^{32}, which are nonlinear for $\mathbb{F}_2[x]/\langle x^{32}+1\rangle$.

[1] Here and elsewhere, indexation starts, as it should, from zero.

This nonlinear map departs from traditional substitution-permutation networks, in which the substitution layer is typically a bricklayer function of S-boxes, applying to each word without mixing them with others. In contrast, the present map mixes pairs of words. Alternatively, the Addition-Rotation-Addition block can be viewed as an S-box applying to pairs of words rather than individual ones.

2.1 The Round Function

The next listing presents pseudocode for the Eaglesong permutation. The argument represents an array of 16 words of 32 bits each. The operators `<<<`, and `^` represent left rotation and `xor`, respectively. The injection constants are stored in the array `injection_constants`. The elements of the matrix M are accessed via the nested array `bitmatrix`.

```
def EaglesongPermutation( state ):
    for round_index in range(0, N):
        # bit matrix
        new = [0 for i in range(0,16)]
        for j in range(0, 16):
            for k in range(0, 16):
                new[j] = new[j] ^ (state[k] * bitmatrix[k][j])
        state = new

        # circulant multiplication
        for i in range(0, 16):
            acc = state[i]
            acc = acc ^ (state[i] <<< a[i]) ^ (state[i] <<< b[i])
            state[i] = acc

        # constants injection
        for i in range(0, 16):
            state[i] = state[i] ^ injection_constants[round_index*16 + i]

        # nonlinear map
        for i in range(0, 8):
            state[2*i] = state[2*i] + state[2*i+1]
            state[2*i] = state[2*i] <<< 8
            state[2*i+1] = state[2*i+1] <<< 24
            state[2*i+1] = state[2*i] + state[2*i+1]

        return state
```

2.2 The Hash Function

The sponge construction consists of two phases, absorbing and squeezing, and is associated with two parameters, the *rate* r and the *capacity* c such that the full state consists of $r + c$ bits. First, the state is initialized to the all zero bitstring. Then a delimiter is appended to the input before the entire input is cut into chunks of r bits, with the last chunk being possibly smaller. Every chunk is `xor`ed into the state, after which the permutation is applied. This describes the absorbing phase. In each iteration of the squeezing phase, r bits are read out from the state and the permutation is applied. To obtain a hash function one truncates the output to whatever output length is specified.

The next listing presents pseudocode for the general sponge and the hash function derived from it. Note that Eaglesong uses the delimiter byte `0x06`, but in principle any nonzero byte will do. Likewise, Eaglesong is defined to have an output length of 32 bytes or 256 bits, but in principle

any output length is possible. However, the rate and capacity are fixed to $c = r = 256$ bits. In this code we use `//` to denote integer division, *i.e.*, with the fractional part being disregarded.

```
def EaglesongSponge( input_bytes, num_output_bytes, delimiter ):
    # parameters
    capacity = 256
    rate = 256

    state = [0 for i in range(0, 16)]

    # absorbing
    for i in range(0, ((len(input_bytes)+1)*8+rate-1) // rate):
        for j in range(0, rate//32):
            integer = 0
            for k in range(0, 4):
                if i*rate//8 + j*4 + k < len(input_bytes):
                    integer = (integer << 8) ^ input_bytes[j*4 + k]
                elif i*rate//8 + j*4 + k == len(input_bytes):
                    integer = (integer << 8) ^ delimiter
            state[j] = state[j] ^ integer

        state = EaglesongPermutation(state)

    # squeezing
    output_bytes = [0] * num_output_bytes
    for i in range(0, num_output_bytes//(rate//8)):
        for j in range(0, rate//32):
            for k in range(0, 4):
                output_bytes[i*rate//8 + j*4 + k] = (state[j] >> (8*k)) &
  0xff

        state = EaglesongPermutation(state)

    return output_bytes

def EaglesongHash( input_bytes ):
    # run the sponge in hashing mode (delimiter: 0x06) and truncate to 32
  bytes == 256 bits
    return EaglesongSponge(input_bytes, 32, 0x06)
```

3 SECURITY ANALYSIS

The sponge construction is secure when instantiated with a random permutation [4]. Any attack on the resulting hash function that is more efficient than a generic attack, can be transformed into a distinguisher capable of telling the given permutation apart from the uniform distribution on permutations. We assume for our own security analysis that the attacker has a better interface to this permutation than he has in practice, namely that he can choose the entire input and see the entire output, or vice versa, rather than just *rate*-many bits. However, this simplification concentrates the space of possible concrete permutations down to a single sample, which is easy to distinguish from the uniform distribution. We therefore extend the definition of the permutation by including a key k of *capacity*-many bits which is `xor`ed into the state in every round together with the injection constants. This transforms the permutation $P(x)$ into a block cipher $E(k, x)$, and allows us to consider any attack that might win the PRP game: distinguishing $E(k, x)$ for

random k, from uniformly random permutations. Nevertheless, the sponge function is only ever instantiated with the concrete permutation $P(x) = E(0, x)$.

We assess the security of Eaglesong (as a block cipher in the PRP game) from the perspective of three branches of statistical cryptanalysis: differential cryptanalysis, linear cryptanalysis, and rotational cryptanalysis; and provide bounds on the distinguishing advantage for one query — or conversely, on the requisite number of queries for a reasonably successful distinguisher — as a function of the number of rounds N. Of these branches, linear cryptanalysis yields the weakest bound, and the number of rounds is set accordingly. For the sake of brevity we omit an exhaustive list of alternative statistical cryptanalyses and rely on the overwhelmingly accurate right-hand rule that differential and linear cryptanalysis are the best performers anyway. Algebraic forms of cryptanalysis are omitted because modular addition is not algebraically compatible with rotation and `xor`; as a result, any polynomial representation of even one round will explode in size.

3.1 Differential Cryptanalysis.

Differential cryptanalysis studies the propagation of differences through various stages of the cipher. Specifically, let $F : \{0,1\}^n \to \{0,1\}^n$ be a function, then for fixed *input difference* $\Delta x \in \{0,1\}^n$ and *output difference* $\Delta y \in \{0,1\}^n$ we define the *differential probability* as

$$\mathrm{DP}_F(\Delta x, \Delta y) \stackrel{\triangle}{=} \Pr_X[F(X) \oplus F(X \oplus \Delta x) = \Delta y] \ . \tag{2}$$

When the cipher is parameterized by an unknown key K, we prefer the *expected differential probability* instead:

$$\mathrm{EDP}_F(\Delta x, \Delta y) \stackrel{\triangle}{=} \mathop{\mathrm{E}}_K \left[\Pr_X[F(X;K) \oplus F(X \oplus \Delta x;K) = \Delta y] \right] \ . \tag{3}$$

The utility of these notions stems from their capacity to be outliers. If the attacker knows a suitable differential pair $(\Delta x, \Delta y)$ for which $\mathrm{EDP}_F(\Delta x, \Delta y)$ is much larger than it would be if F were random (see [8] for the expected behaviour of random permutations), then by computing this value from a large enough number of plaintext and ciphertext pairs, the attacker can determine which function he is interfacing with — the cipher, or a random permutation. From the designer's standpoint, it is imperative to make the EDP indistinguishable from that of a random permutation, for all differential pairs. From a security analysis perspective, a common method is to bound the *maximum expected differential probability (MEDP)*:

$$\mathrm{MEDP}_F \stackrel{\triangle}{=} \max_{\Delta x, \Delta y \in \{0,1\}^n \setminus \{0\}} \mathrm{EDP}_F(\Delta x, \Delta y) \ . \tag{4}$$

If the function F decomposes as a sequence of stages, *i.e.*, $F = f_0 \circ f_1 \circ \cdots \circ f_{r-1}$, then one can consider a vector of differences called a *differential characteristic* $\boldsymbol{\Delta x} = (\Delta x_0, \Delta x_1, \ldots, \Delta x_r)$ and the associated *(expected) differential characteristic probability* ((E)DCP):

$$\mathrm{DCP}_F(\boldsymbol{\Delta x}) \stackrel{\triangle}{=} \prod_{i=0}^{r-1} \mathrm{DP}_{f_i}(\Delta x_i, \Delta x_{i+1}) \ , \tag{5}$$

$$\mathrm{EDCP}_F(\boldsymbol{\Delta x}) \stackrel{\triangle}{=} \mathop{\mathrm{E}}_K \left[\prod_{i=0}^{r-1} \Pr_X[f_i(X;K) \oplus f_i(X \oplus \Delta x_i;K) = \Delta x_{i+1}] \right] \ . \tag{6}$$

We hope to bound the maximum of this value, the *maximum expected differential characteristic probability (MEDCP)*:

$$\mathrm{MEDCP}_F \stackrel{\triangle}{=} \max_{\boldsymbol{\Delta x} \in (\{0,1\}^n)^{r+1}} \mathrm{EDCP}_F(\boldsymbol{\Delta x}) \ . \tag{7}$$

We say a bit in a differential characteristic is *active* if it is set, and components in the cipher's circuit are *active* for the characteristic if it has at least one active input bit.

We apply two heuristics. First, we ignore the positive reinforcement of characteristic probabilities due to different characteristics that have the same first and last value, *i.e.* $\boldsymbol{\Delta x} \neq \boldsymbol{\Delta y}$ but $\Delta x_0 = \Delta y_0$ and $\Delta x_{r-1} = \Delta y_{r-1}$. Second, we assume that the differential probabilities are independent in each stage. Although these assumptions are strictly speaking false, they are common and have proved to be "good enough" over the years.

We decompose the Eaglesong permutation F as a sequence of N rounds f_i, $i \in \{0, \ldots, N-1\}$. With the above heuristics we can then identify MEDCP_F with the Nth power of MEDCP_{f_i}. Furthermore, after decomposing a single round function into its four layers, one observes that the bit matrix, circulant multiplication, and injection of constants (and key), do not decrease the EDP because they are all affine. All EDP decrease must therefore come from the nonlinear map, and more specifically from the modular additions contained therein.

Modular addition takes two outputs X and Y, and produces one output, $Z = X + Y \mod 2^{32}$. With respect to nonzero input differences, we distinguish two cases. First, $\Delta x, \Delta y \in \{0, 2^{31}\}$ giving rise to a $\Delta z \in \{0, 2^{31}\}$ with probability 1. Second, and more interestingly, Δx or Δy or both take values from outside the set $\{0, 2^{31}\}$, and whatever value the output difference Δz takes, it takes it with probability at most one half.

The Addition-Rotation-Addition (ARA) block is constructed such that there is always at least one addition incurring an MEDP degradation of 0.5 if the block is active. In fact, the event of 0.5 MEDP degradation is rare. We count only four difference characteristics with this probability:

$$(2^{31}, 0) \xrightarrow[\text{DP=1}]{\boxplus_1} (2^{31}, 0) \xrightarrow[\text{DP=1}]{\lll} (2^7, 0) \xrightarrow[\text{DP=0.5}]{\boxplus_2} (2^7, 2^7) \tag{8}$$

$$(2^{31}, 2^{31}) \xrightarrow[\text{DP=1}]{\boxplus_1} (0, 2^{31}) \xrightarrow[\text{DP=1}]{\lll} (0, 2^{23}) \xrightarrow[\text{DP=0.5}]{\boxplus_2} (0, 2^{23}) \tag{9}$$

$$(2^{23}, 0) \xrightarrow[\text{DP=0.5}]{\boxplus_1} (2^{23}, 0) \xrightarrow[\text{DP=1}]{\lll} (2^{31}, 0) \xrightarrow[\text{DP=1}]{\boxplus_2} (2^{31}, 2^{31}) \tag{10}$$

$$(2^7, 2^7) \xrightarrow[\text{DP=0.5}]{\boxplus_1} (0, 2^7) \xrightarrow[\text{DP=1}]{\lll} (0, 2^{31}) \xrightarrow[\text{DP=1}]{\boxplus_2} (0, 2^{31}) \ . \tag{11}$$

Aside from the difference $(0, 0)$, which propagates with probability 1, any other characteristic occurs with probability at most 0.25.

The distinction between differential probability 0.5, and 0.25 or less, motivates a distinction between weakly active (0.5), and strongly active (0.25 or less), Addition-Rotation-Addition blocks. In particular, it enables an argument lower-bounding the minimum number of strongly active Addition-Rotation-Addition blocks across any two rounds. We know that the bitmatrix induces a code of minimum distance 5 under the pairwise Hamming weight metric. An exhaustive computer enumeration of all possible output patterns of 1, 2, 3, or 4 weakly active Addition-Rotation-Addition blocks (and no strongly active ones), indicates that in all cases 5 or more blocks are strongly active in the next layer. The same observation holds in reverse. The worst possible scenario not captured by our enumeration is still five active blocks across two rounds, but with at least two strongly active ones. Consequently, the EDCP of any characteristic spanning two rounds is at most $0.5^3 \cdot 0.25^2 = 2^{-7}$. Finally, the MEDCP of the full N rounds is this number raised to the power $N/2$:

$$\mathrm{MEDCP}_F \leq 2^{-7N/2} \ . \tag{12}$$

3.2 Linear Cryptanalysis

Where differential cryptanalysis studies the propagation of differences in pairs of inputs and outputs, linear cryptanalysis studies the propagation of masks and the inputs and outputs that they satisfy. Specifically, a mask $a, b \in \{0,1\}^n$ on an input-output pair (X, Y) with $X \in \{0,1\}^n$ and $Y = F(X)$

for some function F is *satisfying* if the sum of inner products is zero, *i.e.*, $a \cdot X \oplus b \cdot Y = 0$. For a function F, we associate the *linear potential* (LP) to this event; the expectation of that potential, in the case of a keyed primitive; and the maximum of that expectation:

$$\mathrm{LP}_F(a,b) \stackrel{\triangle}{=} \left(2 \cdot \Pr_X[a \cdot X \oplus b \cdot F(X) = 0] - 1\right)^2 \ ; \tag{13}$$

$$\mathrm{ELP}_F(a,b) \stackrel{\triangle}{=} \mathop{\mathrm{E}}_K \left[\mathrm{LP}_{F(\cdot;K)}(a,b)\right] \ ; \tag{14}$$

$$\mathrm{MELP}_F \stackrel{\triangle}{=} \max_{a,b \in \{0,1\}^n \setminus \{0\}} \mathrm{ELP}_F(a,b) \ . \tag{15}$$

Instead of differential characteristics, we have *linear trails* that analogously represent sequences of masks applied to the inputs and outputs of a function that decomposes into various stages $F = f_0 \circ \cdots \circ f_{r-1}$. We associate the *((maximum) expected) linear trail potential* (((M)E)LTP) accordingly.

$$\mathrm{LTP}_F(\boldsymbol{a},\boldsymbol{b}) \stackrel{\triangle}{=} \prod_{i=0}^{r-1} \mathrm{LP}_{f_i}(a_i,b_i) \ ; \tag{16}$$

$$\mathrm{ELTP}_F(\boldsymbol{a},\boldsymbol{b}) \stackrel{\triangle}{=} \mathop{\mathrm{E}}_K \left[\prod_{i=0}^{r-1} \mathrm{LP}_{f_i(\cdot;K)}(a_i,b_i)\right] \ ; \tag{17}$$

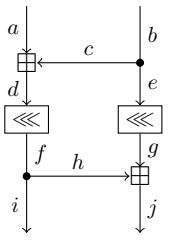
$$\mathrm{MELTP}_F \stackrel{\triangle}{=} \max_{\boldsymbol{a},\boldsymbol{b}} \mathrm{ELTP}_F(\boldsymbol{a},\boldsymbol{b}) \ . \tag{18}$$

Like their differential counterparts, the utility of these notions stems from their capacity to be outliers. An attacker who computes a sufficiently close approximation to the actual linear probability can distinguish the cipher from a random permutation. The task of the designer is then to push the linear potential sufficiently close to zero so that it cannot be detected using the amount of data given.

Also like in differential cryptanalysis, the affine operations do not affect the ELP; all ELP decrease must come from the modular additions in the nonlinear layer. The key challenge is therefore to determine $\mathrm{MELP}_{\mathsf{ARA}}$.

First, we determine the MELP of modular addition, where (a,b) is the mask applied to the input and c is applied to the output. Observe that the mask $a = b = c = 1$ is always satisfied, reflecting the fact that the least significant bit of the output is always the exclusive-or of the least significant bits of the inputs. In all other cases, $\mathrm{LP}_{\mathsf{ModAdd}}((a,b),c) \leq \frac{1}{4}$.

For the ARA block, we observe four trails that are satisfied with potential 1, associated with all combinations of masks (0,0,0) or (1,1,1) for the two adders. The first and last elements of these trails should be considered inactive masks. The trails are as follows, with elements ordered according to the picture.



(	a,	b,	c,	d,	e,	f,	g,	h,	i,	j	)
(	0,	0,	0,	0,	0,	0,	0,	0,	0,	0	)
(	1,	1,	1,	1,	0,	256,	0,	0,	256,	0	)
(	0,	256,	0,	0,	256,	0,	1,	1,	1,	1	)
(	1,	257,	1,	1,	256,	256,	1,	1,	257,	1	)

(19)

All other trails activate at least one adder and are hence satisfied with potential at most 2^{-2}. This observation motivates an exhaustive enumeration of all masks for the full state in the input to the

nonlinear layer in round $i+1$, in order to compute the matching activity pattern at the output of the nonlinear layer in round i, as well as the other way around. Our computer enumeration indicates that in all nonzero cases at least 3 pairs of words are active in the other round. Supplementing this exhaustive enumeration with a symbolic treatment of one active pair of words (with all the others being inactive) shows that on the other side of the linear layers at least two pairs of words are active. These enumerations establish that 3 is a lower-bound on the minimum number of active S-boxes in any trail spanning two rounds, and so

$$\text{MELTP}_F \leq \text{MELP}^{N/2}_{f_i \circ f_{i+1}} \leq \text{MELP}^{3\times N/2}_{\mathsf{ModAdd}} = 2^{-3N}. \tag{20}$$

3.3 Rotational Cryptanalysis

Rotational cryptanalysis [10] studies the propagation of rotational pairs, specifically in the context of ARX ciphers. A rotational pair is a pair of values $(X, \overrightarrow{X})$ where $\overrightarrow{X} = X \lll r$ for some fixed rotation amount r. Rotational pairs propagate deterministically through `xor`s and rotations:

$$\overrightarrow{X \oplus Y} = \overrightarrow{X} \oplus \overrightarrow{Y} \ ; \tag{21}$$

$$\overrightarrow{X \lll s} = \overrightarrow{X} \lll s \ . \tag{22}$$

However, rotational pairs only propagate through modular additions with a probability that depends on the integers' width and on the rotation amount. This probability approaches 0.375 for $r = 1$ and for nonzero rotations and large integers. For integers of 32 bits this number is a good enough approximation.

$$P = \Pr_{X,Y}\left[\overrightarrow{X + Y \bmod 2^{32}} = \overrightarrow{X} + \overrightarrow{Y} \bmod 2^{32}\right] \leq 0.375 \ . \tag{23}$$

A random permutation $F : \{0,1\}^n \to \{0,1\}^n$ maps a random rotational pair[2] $(I, \overrightarrow{I})$ to a rotational pair $(O, \overrightarrow{O})$ with probability 2^{-n}. By contrast, assuming the additions in an ARX cipher are independent events, such a cipher maps a rotational pair to a rotational pair with probability at most P^q, where q is the number of additions. When the latter probability is larger, it can be computed from a sufficiently large set of samples and used to distinguish the cipher from a random permutation.

The Eaglesong function F has 2×8 additions per round, or $16N$ in total. The probability of mapping a random rotational pair to another rotational pair is therefore 0.375^{16N}, which is less than 2^{-256} as soon as $N \geq 12$. We note that other ARX ciphers argue security against rotational cryptanalysis based on the injection of constants; in contrast, our argument is independent of the addition of these constants.

References

[1] Aumasson, J., Meier, W., Phan, R. C., and Henzen, L. *The Hash Function BLAKE.* Information Security and Cryptography. Springer, 2014.

[2] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptology ePrint Archive 2013* (2013), 404.

[3] Bernstein, D. J. The Salsa20 family of stream ciphers. In *New Stream Cipher Designs - The eSTREAM Finalists*. 2008, pp. 84–97.

[2] When I consists of multiple integers, one obtains $\overrightarrow{I}$ by rotating each integer separately by r positions.

[4] Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. On the indifferentiability of the sponge construction. In *EUROCRYPT 2008* (2008), N. P. Smart, Ed., vol. 4965 of *Lecture Notes in Computer Science*, Springer, pp. 181–197.

[5] Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. Cryptographic sponge functions. *SHA-3 competition (round 3)* (2011).

[6] Daemen, J., and Rijmen, V. Rijndael for AES. In *AES Candidate Conference* (2000), pp. 343–348.

[7] Daemen, J., and Rijmen, V. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[8] Daemen, J., and Rijmen, V. Probability distributions of correlation and differentials in block ciphers. *J. Mathematical Cryptology 1*, 3 (2007), 221–242.

[9] Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Grossschädl, J., and Biryukov, A. Design strategies for ARX with provable bounds: Sparx and LAX. In *ASIACRYPT 2016 Part I* (2016), J. H. Cheon and T. Takagi, Eds., vol. 10031 of *Lecture Notes in Computer Science*, pp. 484–513.

[10] Khovratovich, D., and Nikolic, I. Rotational cryptanalysis of ARX. In *FSE 2010* (2010), S. Hong and T. Iwata, Eds., vol. 6147 of *Lecture Notes in Computer Science*, Springer, pp. 333–346.

A Constants

constants_listing.py

```
injection_constants = [ 0x0067f2ba , 0x144d4881 , 0xc2664523 , 0x66ad5fc5 , 0x1eef1bb1 , 0xdb681b7c , 0x4c769f30 , 0x08a43f86 ,
0x59373142 , 0x2b3c3837 , 0x52de4037 , 0x9da45857 , 0xf4a000dd , 0xb99954c0 , 0x98d27e29 , 0xcb4ba929 ,
0x96dc1253 , 0x40d04933 , 0x04fae66e , 0x86cc8a13 , 0x57c67712 , 0x39fe36ca , 0x95275370 , 0x25f08c24 ,
0x59601ded , 0x91cd8f8b , 0x5dd8bca0 , 0xebd1dabf , 0xbbc6a43d , 0xd4dd8533 , 0x10c2a70f , 0x8e126e04 ,
0x298d5d32 , 0x4a83905c , 0x0a83a41d , 0xa3380990 , 0x79aff2de , 0xecbb19ff , 0xe8ada3cb , 0xba05617e ,
0xbc22b19d , 0x87190fe1 , 0xdd0f372e , 0x080f5f63 , 0x748d3325 , 0x5f362e82 , 0x879d3bad , 0xf4bd54fb ,
0x96a1e9f5 , 0x1e4807f5 , 0x505f3e7f , 0xb4a59dde , 0x7617b425 , 0x8978d93a , 0x5ee7ab8d , 0xd2919598 ,
0x40c1e3a7 , 0x910fef70 , 0x02795f33 , 0xb18fbe0e , 0x8c726d1f , 0xcb676367 , 0x797a5686 , 0x66a635bf ,
0x33e1a9bc , 0xb66a4af8 , 0x35e53258 , 0x3c242245 , 0x03729204 , 0xea8a8163 , 0x7624e6db , 0x5c9656b5 ,
0x54505158 , 0x41269e67 , 0x8ba0c1e5 , 0x9bf7f594 , 0x54ab3d91 , 0x742514d1 , 0x8a30b1b7 , 0x4ddff441 ,
0xe2382d56 , 0x080b535e , 0xcff52810 , 0xfbe17da4 , 0x270542bb , 0x709d4cee , 0xeb53dc56 , 0x327036b5 ,
0x1106284e , 0x0a3a6af7 , 0x34c3c363 , 0x2a448599 , 0x1c85c433 , 0x6a50bb5c , 0x4e7b5dbf , 0x430ef00b ,
0xb15b23ef , 0xeea7a4fa , 0x78e3595f , 0xa31c6677 , 0x5f4b0ec0 , 0x23e06218 , 0x737777f3 , 0xe2c26d3d ,
0xb6d3e954 , 0x95df747c , 0x8e46e5a6 , 0xabe201b9 , 0x295d3f66 , 0x2815374b , 0xdada17b0 , 0xa6276c65 ,
```

```
15 0x1323863d , 0x2326b83f , 0x06de5548 , 0x06ded5fe , 0x4d06e4a8 , 0x698e5532
     , 0xfdf8e39f , 0xd90baaca ,
16 0x433f0a37 , 0xf13f824e , 0x1e81ec14 , 0x8557d175 , 0x3ef83a87 , 0xaf5179bf
     , 0xa6f70b47 , 0x3210b313 ,
17 0x693a73c8 , 0xa0e86cd3 , 0xc1ed15c9 , 0x7857f7af , 0xcbf83762 , 0x7d6d9fad
     , 0xab950994 , 0x2ca8991b ,
18 0x6a357f8c , 0x362fc760 , 0x4f3c2f2d , 0x7c332cde , 0x7d862e7d , 0x3cd1063b
     , 0x6b5744d3 , 0x11e2bfa4 ,
19 0x3574f020 , 0xabb4f72e , 0xf0008f74 , 0x8127b198 , 0x64530f90 , 0xaccba020
     , 0x1f24bbdc , 0x96bc0975 ,
20 0x09cdbb86 , 0xfe1bf47c , 0xdc820173 , 0xb8bc05f7 , 0xc248b84d , 0xda6c59b2
     , 0xe4cb0de6 , 0xa18f594b ,
21 0xff089509 , 0x4f1181c2 , 0x10c22171 , 0xdd0d4493 , 0x0054b1bf , 0x0e2f85bb
     , 0xa29a4565 , 0xafd205cb ,
22 0x14fc2da6 , 0x97cc4fb1 , 0x80f668bf , 0x4d39af51 , 0xcdf9c7a3 , 0xdeed39b4
     , 0xa2148d07 , 0xcbc51d4f ,
23 0x463036fa , 0x11fbdbfd , 0xd13c67aa , 0x58d508bc , 0x6fc88549 , 0x9e857d0a
     , 0xe067fdbc , 0x8895b7f1 ,
24 0xfaaabbbe , 0x3b2b65dd , 0x975a2fb7 , 0x181de9e7 , 0xaf21a6b6 , 0xb51cdf18
     , 0xf5d41d2d , 0x00ceafd0 ,
25 0x2d90c0c4 , 0xe2e44fb7 , 0x7ce0520d , 0x3cbfafcd , 0x700da77e , 0x05dedb60
     , 0x12a32947 , 0x0bc30c1f ,
26 0xda7b69ec , 0xee80c223 , 0xeefba2f3 , 0x3232b02f , 0x7becd954 , 0x06a596e0
     , 0x597e011c , 0x828cc456 ,
27 0xf3225aa1 , 0x6307fb6a , 0x4c43af24 , 0xcce2abb8 , 0x73ccb8dd , 0x63460bf0
     , 0xba5e5548 , 0x8d6d5364 ,
28 0xe7b5ebde , 0xe5e85b36 , 0xde7df680 , 0x0bafe47f , 0x69e0c759 , 0x58abe207
     , 0xb9c5f5b4 , 0x32695542 ,
29 0x8f2f7acd , 0x191000f7 , 0x176d6f65 , 0x32167b3b , 0xed4a3340 , 0x2dc33eef
     , 0x0813a1ac , 0x141e223e ,
30 0x913f6a6b , 0xe7eff439 , 0x9aac426c , 0x2915414a , 0x01c7ddfb , 0x3603eaf5
     , 0xd9a408f8 , 0xd167db80 ,
31 0x5a6f0000 , 0x7d06633b , 0x9b33c1ec , 0xb367ff40 , 0x2a6d551a , 0xa02dcf37
     , 0xb07f3c85 , 0x40dee8ec ,
32 0xb4759bdb , 0x3a7385c2 , 0x49e28080 , 0xf53a139d , 0x11cf2086 , 0xaa5485d9
     , 0x89bc7b88 , 0x90a5b40b ,
33 0xbcfac927 , 0x1397f8af , 0xcecb4b83 , 0x3f4ef87f , 0xd36532ba , 0xc63ee448
     , 0x124788dc , 0xf97ab8ae ,
34 0x0ac629a2 , 0x688f262b , 0x7775b9a7 , 0xe68df1ec , 0xbd0daa15 , 0xce0d8732
     , 0x3c71e01e , 0xaa1281a5 ,
35 0xa501cdee , 0x4843016a , 0x73e8244d , 0xd62088c3 , 0xc097a06e , 0xe46b1df9
     , 0xead9bfec , 0x3a839e86 ,
36 0x634ba7e2 , 0x4a15c89c , 0x45ee25d1 , 0x66ca230c , 0xe5d90546 , 0x974568fc
     , 0x50eceed6 , 0x013445b5 ,
37 0x0d62d4c5 , 0x74c19ce2 , 0x13b2137a , 0x1d207fe4 , 0x25665312 , 0x2e55cf3e
     , 0x5874b698 , 0xaa41c229 ,
38 0xb6eeceb9 , 0x92a4b58e , 0x4fda333f , 0x996062ef , 0x66875469 , 0x876a8c62
     , 0x2ee82dea , 0x95d480fc ,
39 0x18c341b7 , 0xe02d8f43 , 0xb8e1edd6 , 0xd9456784 , 0x9c227f0f , 0x9ddc5416
     , 0x8216ca20 , 0x46774f4b ,
40 0xc89c494e , 0x46399419 , 0x52265251 , 0x0e0f8323 , 0x3deb9e5c , 0x7ae0bddd
     , 0x0c05e472 , 0x9d8f81d0 ,
41 0x129acd92 , 0x0a58587a , 0x195606a7 , 0x9141a830 , 0xbcb59c33 , 0x82ed9dc0
     , 0x5234c1b2 , 0x4e7bfd00 ,
42 0x8f946f62 , 0x1ac61234 , 0xe1327cec , 0xd60e7859 , 0xce2d6888 , 0xd7ee4c9a
     , 0x3e194ce9 , 0x4f973964 ,
43 0x771dbbd1 , 0x78fe2d16 , 0x1e0d53a0 , 0x1e88344c , 0xa32814ab , 0xdd961c4e
     , 0x2eb3b9a7 , 0xa7a05616 ,
44 0x1d93bfb5 , 0x80f49c17 , 0x8e854c79 , 0x9d6a60dc , 0xa6284a52 , 0x69148d25
     , 0x58ed3557 , 0x98a1e5bb ,
45 0xde1e4aa7 , 0x6aec1d24 , 0x17db262b , 0xb2e3faae , 0xa7725794 , 0x7d4bcd05
     , 0x2d03a71d , 0xa56c5f3c ,
```

```
46 0x9a10583d , 0x2ffaafd5 , 0x418bdb30 , 0x89b36a08 , 0x3b0f90a6 , 0xea7318a6
     , 0xdfc2cbcb , 0x84dd1f99 ,
47 0x565082af , 0xe4811929 , 0xf1aa8463 , 0xc6846bc9 , 0xeebe115e , 0x49ddec59
     , 0x1c3ab3a4 , 0x7ead7007 ,
48 0x1efc059e , 0xefe0cfc7 , 0x492004e2 , 0x5145b626 , 0xd22224cd , 0x0da3bcc1
     , 0xdaaf670a , 0xe8895409 ,
49 0x56c5e982 , 0x008a3039 , 0xba617f54 , 0xc100cf18 , 0x763a06c2 , 0x22a4aaed
     , 0x433f7d29 , 0x19a98e53 ,
50 0x0871ad12 , 0x7e0e95cd , 0x44fb3328 , 0xc5681e39 , 0x06b65d00 , 0x73bd3891
     , 0xa487951c , 0x4495f631 ,
51 0x257ad2d7 , 0xe5e64dde , 0xa7b4a8dc , 0x4a054749 , 0x8c53b180 , 0x94288b1f
     , 0x36e03580 , 0xfa6e4aa8 ,
52 0x4b4f6496 , 0xa281c06a , 0x4a371d12 , 0xc81e81a5 , 0x80717c78 , 0x85488861
     , 0xaa696cb3 , 0xda997ef2 ,
53 0x7ead05fd , 0xb9bc8987 , 0x26434df6 , 0x23d1f162 , 0x8e341be9 , 0x15abf661
     , 0xafd29d55 , 0xb04d378a ,
54 0xae554f94 , 0x3b9d32f8 , 0x25fc532e , 0xa2aa5998 , 0xf56c1c33 , 0x30b2101c
     , 0x5b2e393c , 0x0b4a60b5 ,
55 0xf1697b81 , 0x2279842d , 0x3d0fae97 , 0xdb27019d , 0x9b6e991f , 0x9a4f4623
     , 0x749c523f , 0x1e9eba0f ,
56 0x1359831c , 0x003b529a , 0x2a18ec2f , 0x2e334d7b , 0xa7bebfc9 , 0x7731da0f
     , 0x5f474760 , 0xc5ebb9c3 ,
57 0x2b2dab6a , 0x966f5cfc , 0x0ec5846c , 0xef93c8fd , 0xf6114e4a , 0x6cb39d68
     , 0x82138464 , 0xe2468a02 ,
58 0x782ed783 , 0x3cddebcd , 0x8ef571ae , 0x78f89b5f , 0x289ba653 , 0xc654535f
     , 0x3f98f3c3 , 0xdcbf9357 ,
59 0xd4fae200 , 0xc9de4349 , 0xf2128b62 , 0xc3051c9a , 0xc8ad5701 , 0xf5a5a851
     , 0xd5a6778d , 0xc062d691 ,
60 0x0ead5941 , 0x16b97151 , 0x700dae99 , 0xf294d819 , 0xb3948b14 , 0x92cfdb8d
     , 0xa8d12849 , 0xabdbfeae ,
61 0x8dd229dc , 0x2023c206 , 0x34c0754d , 0x562db512 , 0x4393bce4 , 0x536c8500
     , 0x7a8a88b8 , 0xbd4f80fc ,
62 0x588e0a05 , 0x866b5000 , 0xcf9db63d , 0x9802d083 , 0x2bbcfbae , 0x327fb9be
     , 0x94004a66 , 0x11d94771 ,
63 0x8bfe96fd , 0x4569a189 , 0xe81c1c27 , 0xcf1ce084 , 0xd45bbae0 , 0xd9949390
     , 0xfd84f0c7 , 0xb669d73e ,
64 0xdb364d0b , 0xf8f2cdde , 0xbc0386dc , 0x034f3408 , 0xb5188c54 , 0x0627035f
     , 0xbb22774f , 0x197d1f63 ,
65 0xbac7a456 , 0x17f7ec88 , 0x12fbb595 , 0x056af49e , 0xa39f34f7 , 0x15f54804
     , 0x04b5a3b9 , 0x800302b6 ,
66 0xb51f5f5d , 0x3178655d , 0x0ea072a0 , 0x8fc08df3 , 0xbbdc3ecc , 0x3be053e3
     , 0x8d89fff0 , 0x184bb2ad ,
67 0xbc5afc93 , 0x33cb0c21 , 0x12fb8f40 , 0xa534a9cf , 0xc9ce7416 , 0x92c9cec3
     , 0x84b78375 , 0x258c1ec3 ,
68 0xd232a922 , 0xdfe2e45d , 0xb7268e03 , 0x2cc1cc01 , 0xf753a5dc , 0x42915c0d
     , 0xf2fcaa23 , 0xe26251a6 ,
69 0x8772b8f2 , 0x0006df06 , 0x7f6c381e , 0x749ece9b , 0x46d419e9 , 0x8bcf7df1
     , 0xf7fa10e2 , 0xc55d0150 ,
70 0x529789c0 , 0x679e5256 , 0xc8f8b104 , 0x8cd7d294 , 0xd88f2489 , 0x0efa445d
     , 0x8b30982e , 0xe4a355bf ,
71 0xb3683ef7 , 0x95188867 , 0x4e3312b6 , 0x1db01078 , 0xc439e0e9 , 0xe005c964
     , 0x0c103d73 , 0x927957a0 ,
72 0xf595ce3c , 0xb5f36a8d , 0xba393c2e , 0xc0b9a923 , 0xcf9f820b , 0x170db6ae
     , 0x8b4656ac , 0xcd465476 ,
73 0x3213fd40 , 0x7a8e5553 , 0x78dbed52 , 0x3e8c5dc7 , 0x290e735b , 0x38e8c7c2
     , 0xbd8fdfec , 0x3b087538 ,
74 0xc451220c , 0x23d54ae6 , 0x23aa7512 , 0x6f5f0ec2 , 0x7c041b7f , 0x39f67880
     , 0x14c66832 , 0x9b8b50be ,
75 0x95a10214 , 0xcfd59851 , 0x131cf800 , 0xb5a302cf , 0xec4971fa , 0x08747ece
     , 0x6d19faad , 0xb0619453 ,
76 0x3b665799 , 0x3355d0f4 , 0x01da2765 , 0xf4c358e3 , 0xf62210ef , 0xede2fbf8
     , 0x3c386a85 , 0x2e2be2c5 ,
```

```
77 0x00da4098 , 0x62151fce , 0x8988f581 , 0x3c52074b , 0xad334a50 , 0xe2787d29
     , 0x7ccb2f7c , 0xc1cf024e ,
78 0x309f6a7f , 0x09ea24c1 , 0x69812825 , 0x710edb3f , 0x2a4c140d , 0x87b0f697
     , 0x6cbd5952 , 0x77d3d743 ,
79 0x1621d024 , 0x3b781a30 , 0x05e4a0f5 , 0xeca327c3 , 0xbfb12825 , 0x51a4f2dc
     , 0x9ca8fd74 , 0xc1dc81fa ,
80 0xd424333d , 0x3cd3a051 , 0xed6a5e0c , 0xe1cb66aa , 0x87d09c55 , 0x0ee2783e
     , 0x39649710 , 0x98dabbc3 ,
81 0x08d9cf63 , 0x15069380 , 0x682b1514 , 0x1cedb4b3 , 0xb5fd41a4 , 0x384b060d
     , 0x9c7f0236 , 0x8879d8e2 ,
82 0x001aa8b3 , 0x3b897edd , 0x854149ba , 0x908886d8 , 0x3cfb7f86 , 0x91959420
     , 0xdbc61401 , 0x27adccf2 ,
83 0xc4c23620 , 0xf9768e4c , 0x7b71c0c0 , 0x69d7b70b , 0xf56afb84 , 0x01e65f6c
     , 0xc33f0df7 , 0x36918f32 ,
84 0xb41ada04 , 0x70dd5e75 , 0x73031084 , 0x688eedba , 0x0f06829c , 0x893435e8
     , 0x799eba27 , 0xb6341572 ,
85 0xbac5f1a1 , 0x5c26fcc4 , 0xadd03e2e , 0xa57ca6ff , 0xd9a3675e , 0x5f9bee94
     , 0x9e7ddac3 , 0xb8b2a558 ,
86 0x6e19c32f , 0x9429b014 , 0xf9070869 , 0x4b72e424 , 0xe675d4c1 , 0xdf61fb28
     , 0xac235d28 , 0xfc651d91 ]
```