

A Note on the Security Analysis of Areion

Frank Denis
Fastly Inc.
fde@00f.net

Abstract. Areion is a family of AES-based permutations designed for fast hashing and authenticated encryption of short messages. We identify errors in how its security analysis converts active S-box counts into differential and linear probability bounds, and give corrected calculations.

1 Introduction

Section 5.1 of Isobe et al. [Iso+24] uses active AES S-box counts to estimate resistance to differential and linear attacks. Since the maximum differential probability and squared linear correlation of one box are both 2^{-6} , multiplying this bound over N boxes gives 2^{-6N} . On this basis, the authors conclude that 22 active boxes suffice for 128-bit security.

Several of these boxes share an input. Two branches that compute the same substitution impose only one differential condition, as Taiyama et al. [Tai+23] observed in Section 4.2.1. Counting that substitution once therefore removes the duplication, but does not justify multiplying the remaining probabilities: the Areion-256 example in Section 3.3 still exceeds their product.

The same active-box argument also misses the sum over linear trails. When several trails have the same input and output masks, their correlations add, with signs that depend on the round constants.

Table 1. Selected entries from Table 7 of [Iso+24]. ASD and ASL give lower bounds on differentially and linearly active S-boxes.

Permutation	Rounds	ASD	ASL
Areion-256	2	6	1
Areion-512	2	2	1
Areion-512	3	5	4

2 Round functions and notation

Areion-256 has two 128-bit state words (A, B) , and Areion-512 has four, (A, B, C, D) . Within each word we follow the AES byte layout [Nat23], writing SB, SR and MC for SubBytes, ShiftRows and MixColumns. The AES S-box is S .

With round constant k_r , the four functions in [Iso+24] are

$$\begin{aligned} F_0(x) &= \text{MC}(\text{SR}(\text{SB}(x))), \\ F_1(x) &= \text{SR}(\text{SB}(x)), \\ F_{2,r}(x) &= F_0(F_0(x) \oplus k_r), \\ F_{3,r}(x) &= F_0(F_1(x) \oplus k_r). \end{aligned} \tag{1}$$

After the word shuffle, one round is

$$\begin{aligned} R_{256,r}(A, B) &= (B \oplus F_{2,r}(A), F_1(A)), \\ R_{512,r}(A, B, C, D) &= (B \oplus F_0(A), F_{3,r}(C), D \oplus F_0(C), F_1(A)). \end{aligned} \tag{2}$$

We number rounds and bytes from zero. The words entering round r are A_r, B_r, C_r, D_r , and $e_j(v)$ has byte v at position j and zeros elsewhere. We write byte values and masks in hexadecimal, and counts and indices in decimal. Word strings list bytes in order.

Probabilities and correlations are taken over a uniform input to the stated round segment, with the published constants fixed.

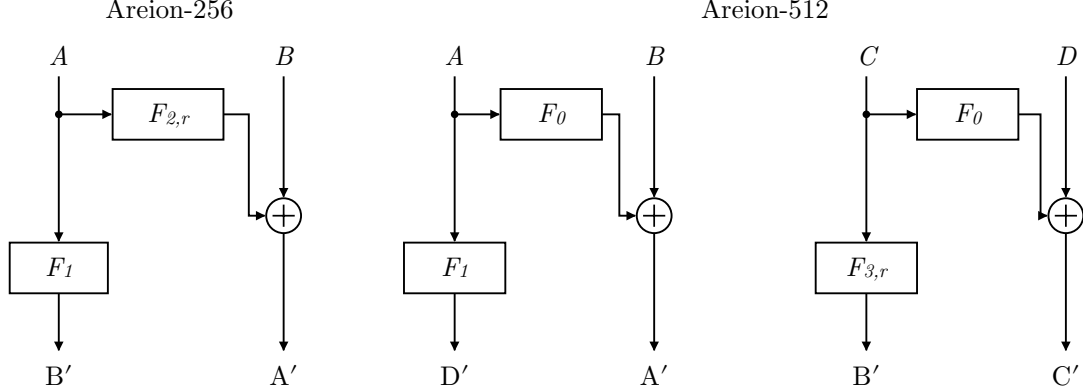


Fig. 1. Round functions of Areion. The primed labels give the output positions after the word shuffle.

3 Differential analysis

For a function f on n input bits, we write its differential count and probability as

$$D_f(\alpha, \beta) = |\{x : f(x) \oplus f(x \oplus \alpha) = \beta\}|, \quad \text{DP}_f(\alpha, \beta) = 2^{-n} D_f(\alpha, \beta). \quad (3)$$

The counts form the differential distribution table (DDT).

3.1 Shared S-box outputs

The functions $F_0(A)$ and $F_1(A)$ use the same sixteen S-box outputs, since $F_0 = \text{MC} \circ F_1$. Their output differences are therefore linked by MC.

An input difference $(0, e_0(01), 0, 0)$ in round 0 of Areion-512 becomes $(e_0(01), 0, 0, 0)$ after that round. The following round reaches output difference

$$(\text{MC}(e_0(1f)), 0, 0, e_0(1f)) \quad (4)$$

exactly when

$$S((A_1)_0) \oplus S((A_1)_0 \oplus 01) = 1f. \quad (5)$$

The four solutions, **00**, **01**, **bc** and **bd**, give probability $\frac{4}{256} = 2^{-6}$, since all other bytes are free. All intermediate differences are fixed, so this is a single characteristic. Yet the two active S-boxes counted in Table 7 of [Iso+24] would instead give 2^{-12} under that paper's S-box-product rule.

3.2 Two substitutions per byte in Areion-512

In F_3 , both substitutions act byte by byte before the only MixColumns layer. Writing

$$T_c(x) = S(S(x) \oplus c), \quad c_{r,j} = (\text{SR}^{-1}(k_r))_j, \quad (6)$$

and $T^{(r)}(x)_j = T_{c_{r,j}}(x_j)$, we can express the function as

$$F_{3,r} = \text{MC} \circ \text{SR}^2 \circ T^{(r)}. \quad (7)$$

The $F_0(C)$ branch also uses $S(x)$, so we keep both byte outputs in

$$J_c(x) = (S(x), T_c(x)). \quad (8)$$

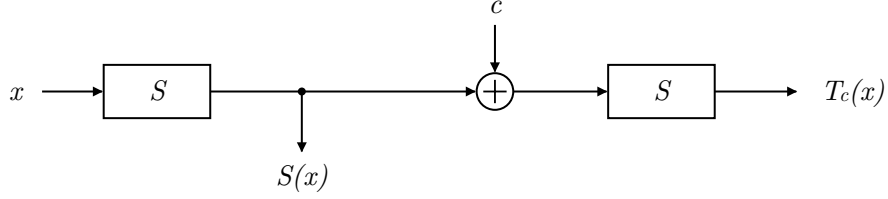


Fig. 2. The joint byte function J_c .

When $c_{0,0} = 44$, input difference **01** gives output differences **ff** and **99** only for inputs **82** and **83**. The individual S-box transitions **01**→**ff** and **ff**→**99** each have probability 2^{-7} . Their product is 2^{-14} , whereas the joint probability is 2^{-7} .

Every possible active transition of the joint function has exactly two solutions among the 256 byte inputs, so its probability is 2^{-7} under uniform input. This holds for every constant byte and nonzero input difference; Appendix A gives the exhaustive check.

3.3 A two-round differential for Areion-256

In Areion-256, MixColumns sits between the two substitutions of F_2 , coupling four bytes in each column. Over rounds 0 and 1, input difference $(0, e_0(01))$ can reach output difference

$$(\text{MC}(\text{SR}(\gamma)), e_0(1f)), \quad \gamma = \text{cfbcfb97000000000000000000000000}. \quad (9)$$

The first output word is **85cfcf4a9797a235fb16edfbdf63bcbc**.

Round 0 leaves difference $(e_0(01), 0)$. With $u = F_1(A_1)$ and $v = \text{MC}(u) \oplus k_1$, the second output requires $\Delta u = e_0(1f)$ and $u_0 \in \{63, 65, 7a, 7c\}$, giving $\Delta v = \text{3e1f1f210000000000000000000000}$.

The first output also requires $\Delta \text{SB}(v) = \gamma$, so the four active bytes of v lie in $\{d4, ea\}$, $\{0f, 10\}$, $\{2d, 32\}$ and $\{8f, ae\}$, respectively.

Of the sixteen possible columns, only **d41032ae** and **ea0f2d8f** meet the condition on u_0 after applying $\text{MC}^{-1}(v \oplus k_1)$. Every value of the first column of v is equally likely, and the other columns can have any value, so the exact probability is $\frac{2}{2^{32}} = 2^{-31}$.

By comparison, the six active S-box occurrences match Table 7's two-round count and give 2^{-36} under the published estimate. Even after counting the shared substitution only once, the product is $2^{-6}(2^{-7})^4 = 2^{-34}$. The exact probability depends on all five conditions holding together in the same column.

4 Linear analysis

For masks a, b and a function f on n input bits, we define its Walsh coefficient and correlation by

$$W_f(a, b) = \sum_x (-1)^{a \cdot x \oplus b \cdot f(x)}, \quad C_f(a, b) = 2^{-n} W_f(a, b), \quad (10)$$

where $u \cdot v$ is the binary dot product. The linear probabilities in [Iso+24] are squared correlations.

4.1 Summing the linear trails

A *linear hull* contains all trails with given input and output masks. For J_c with input mask a and output masks b, g , its correlation is the signed sum over intermediate masks m :

$$C_{J_c}(a; b, g) = \sum_{m \in \{0,1\}^8} (-1)^{m \cdot c} C_S(a, b \oplus m) C_S(m, g). \quad (11)$$

Setting $b = 0$ gives the correlation of T_c .

4.2 An approximation over the last two rounds

Since $F_0 = \text{MC} \circ F_1$, the outputs satisfy

$$\begin{aligned} A_{r+1} \oplus \text{MC}(D_{r+1}) &= B_r, \\ A_{r+2} \oplus \text{MC}(D_{r+2}) &= F_{3,r}(C_r). \end{aligned} \quad (12)$$

For a binary linear map P , its adjoint P^* is defined by $u \cdot P(v) = P^*(u) \cdot v$. Applying this to Equation 12, a mask λ on A_{r+2} together with $\text{MC}^*\lambda$ on D_{r+2} gives the parity $\lambda \cdot F_{3,r}(C_r)$.

Over the last two rounds, 13 and 14, choose input mask $e_9(40)$ on C_{13} and the following output masks λ and δ . All other word masks are zero.

A15: eb59fc60000000000000000000000000
D15: 002e0000000000000000000000000000

They give $\text{MC}^*\lambda = \delta$ and $(\text{MC} \circ \text{SR}^2)^*\lambda = e_9(2e)$. With $c_{13,9} = 25$, the parity is

$$40 \cdot (C_{13})_9 \oplus 2e \cdot T_{25}((C_{13})_9). \quad (13)$$

There are 84 zeros and 172 ones among the 256 input bytes, giving an exact correlation of $\frac{84-172}{256} = -\frac{11}{32}$.

Its magnitude is 22 times the bound 2^{-6} on each nonzero term in Equation 11, since each term contains two active AES S-boxes. Setting the constant to zero changes the correlation to $-\frac{1}{32}$.

4.3 An exact three-round approximation

For the three-round example, we use the following nonzero word masks in rounds 6, 7 and 8:

input C: 00006f00000000000000000000000000
input D: 00000000000000000000ae6d24eb00000000
output A: 1f6a074300000000021a5b5d6000000000
output D: 006100500000000003d00da0000000000

Calling these masks $\alpha, \mu, \lambda, \delta$, we have $\text{MC}^*\lambda = \delta$, $(\text{MC} \circ \text{SR})^*\mu = e_2(0c)$, and $(\text{MC} \circ \text{SR}^2)^*\lambda = 00000000000000000003d61da5000000000$.

The output masks cancel all terms except $\lambda \cdot F_{3,7}(C_7)$. Since D_6 is uniform and independent of C_6 , the word $C_7 = D_6 \oplus F_0(C_6)$ is also uniform and independent of C_6 . Replacing D_6 with $C_7 \oplus F_0(C_6)$ in the input parity then gives the exact product

$$C_{F_0}(\alpha, \mu) C_{F_{3,7}}(\mu, \lambda). \quad (14)$$

The first factor has one active byte, while the second has four independent bytes with the Walsh coefficients in Table 2. Their product is

$$\frac{32 \times (-60) \times (-60) \times (-64) \times 72}{256^5} = -\frac{2025}{2^{22}}. \quad (15)$$

Table 2. Walsh coefficients for the three-round approximation. Round 8 contributes the cancellation in Equation 12.

Function	Byte	c	Input mask	Output mask	W
S in round 6	2		6f	0c	32
T_c in round 7	8	66	ae	3d	-60
T_c in round 7	9	0d	6d	61	-60
T_c in round 7	10	74	24	da	-64
T_c in round 7	11	80	eb	50	72

Table 7 lists $\text{ASL} = 1$ and $\text{ASL} = 4$ for two and three rounds. Using these counts to bound the correlations gives $|C| \leq \frac{1}{8}$ and $|C| \leq 2^{-12}$, respectively, both too small for the examples above. This is because the rule bounds a single trail, while each T_c correlation here sums over the 255 nonzero intermediate masks.

A Conventions and the joint-count check

Byte $4j + i$ is in row i , column j of an AES word, and $(\text{SR}(x))_{4j+i} = x_{4((j+i) \bmod 4)+i}$.

The published constants are read in little-endian byte order. The full word needed in Section 3.3 is

$$k_1 = 896c4eec98fa2e08d0319f29223809a4. \quad (16)$$

Unlisted constant bytes either cancel in the differences or lie on a path with zero difference or zero mask.

We obtain the count in Section 3.2 by grouping byte inputs with the same pair of output differences:

```

CheckJointCounts():
  for c in 0..255:
    t(x) = S(S(x) XOR c)
    for a in 1..255:
      counts = empty map, with default value 0
      for x in 0..255:
        b = S(x) XOR S(x XOR a)
        g = t(x) XOR t(x XOR a)
        counts[b,g] += 1
      assert every stored count equals 2

```

B Notes on the zero-sum derivation

Inverse formula. On printed page 19 of [Iso+24], $G_0 = F_1$ and $G_1(x) = \text{MC}(F_1(\text{MC}(x) \oplus k_r))$, so $F_{2,r} = G_1 \circ G_0$. The printed inverse instead uses $G_0 \circ G_1$ in the second expression. The corrected formula is

$$x_0 = G_0^{-1}(y_1), \quad x_1 = y_0 \oplus G_1(G_0(x_0)). \quad (17)$$

The printed order does not recover the all-zero input with constant k_1 .

Remark on mixing layers. On printed page 20, the term $F_1(F_{3,r}(x) \oplus C_2)$ is described as 2.5 AES rounds, although its exact form

$$\text{SR}(\text{SB}(\text{MC}(\text{SR}^2(T^{(r)}(x))) \oplus C_2)) \quad (18)$$

contains only one MixColumns layer.

With four input bytes free and the others fixed, each output byte either stays constant or takes every value equally often. The values appear 2^{32} or 2^{24} times, both even, so the XOR sum is still zero.

References

- [Iso+24] Isobe, T., Ito, R., Liu, F., Minematsu, K., Nakahashi, M., Sakamoto, K., and Shiba, R. Areion: Highly-Efficient Permutations and Its Applications (Extended Version). (2024). Cryptology ePrint Archive, Report 2023/794; revision dated 14 November 2024. <https://eprint.iacr.org/2023/794>
- [Nat23] National Institute of Standards and Technology. Advanced Encryption Standard (AES). (2023). FIPS 197-upd1. <https://doi.org/10.6028/NIST.FIPS.197-upd1>
- [Tai+23] Taiyama, K., Utusmi, S., Shiba, R., Sakamoto, K., and Isobe, T. Evaluation of Differential Characteristics Probability and Collision Attack Evaluation for Large Block Permutations Areion and Simpira v2. In: Computer Security Symposium 2023. pp. 463–470 (2023). In Japanese; English author spelling and title as printed. <https://ipsj.ixsq.nii.ac.jp/records/228677>