FOAM: Searching for Hardware-Optimal SPN Structures and Components with a Fair Comparison

Khoongming Khoo[1], Thomas Peyrin[2], Axel Y. Poschmann[3], and Huihui Yap[1]

[1] DSO National Laboratories, 20 Science Park Drive, Singapore 118230.
{kkhoongm,yhuihui}@dso.org.sg
[2] SPMS, Nanyang Technological University, Singapore.
thomas.peyrin@ntu.edu.sg
[3] NXP Semiconductors
axel.poschmann@gmail.com

Abstract. In this article, we propose a new comparison metric, the *figure of adversarial merit* (FOAM), which combines the inherent security provided by cryptographic structures and components with their implementation properties. To the best of our knowledge, this is the first such metric proposed to ensure a fairer comparison of cryptographic designs. We then apply this new metric to meaningful use cases by studying Substitution-Permutation Network permutations that are suited for hardware implementations, and we provide new results on hardware-friendly cryptographic building blocks. For practical reasons, we considered linear and differential attacks and we restricted ourselves to fully serial and round-based implementations. We explore several design strategies, from the geometry of the internal state to the size of the S-box, the field size of the diffusion layer or even the irreducible polynomial defining the finite field. We finally test all possible strategies to provide designers an exhaustive approach in building hardware-friendly cryptographic primitives (according to area or FOAM metrics), also introducing a model for predicting the hardware performance of round-based or serial-based implementations. In particular, we exhibit new diffusion matrices (circulant or serial) that are surprisingly more efficient than the current best known, such as the ones used in `AES`, `LED` and `PHOTON`.

Key words: SPN, lightweight cryptography, figure of adversarial merit, diffusion matrices.

1 Introduction

RFID is a rising technology that is likely to be widely deployed in everyday life, leading to new security challenges. Significant advances in this area have already been obtained. In particular, many lightweight block ciphers [8,10,15,19] have recently been proposed, and designing such ciphers is not an easy task as showed by the numerous candidates that eventually got broken. Moreover, it is interesting to note that in most privacy-preserving RFID protocols proposed [1,16,17] a hash function is required, and since a hash function can be easily built from a block cipher (for example with the Davies-Meyer mode) or a permutation (for example with the sponge construction [7]), a crucial question for the researchers is how to design a hardware efficient permutation.

Hardware efficiency can have very different meanings depending on the utilization scenario targeted by the designer. For example, a classical metric is to estimate the minimum silicon area required by the primitive to perform the cryptographic operations. This, of course, depends on the parameters of the function itself (the area is highly dependent on the amount of memory required) and most lightweight block ciphers have a rather small block size of 64 bits. It is to be noted that the area is usually not directly linked to the security of a primitive, as adding extra rounds will have an impact on the throughput of the implementation, but only a very limited one concerning the area (we assumed that the function has no weakness that is independent of the number of rounds). Area and other metrics such as throughput, latency or power dissipation can be traded-off for one another, making the comparison between different primitives difficult. In the direction of fairer comparisons of hardware implementations of cryptographic primitives, Bogdanov *et al.* [9] introduced the efficiency metric throughput/area in order to take in account these tradeoffs. However, the possibility of trading off throughput for power was not taken in account and Badel *et al.* [2] proposed instead a *figure of merit*, defined as $\mathrm{FOM} = \mathrm{throughput}/\mathrm{area}^2$. However, as of today, no metric takes in account the inherent security of a building block, therefore making it hard to compare for example two diffusion matrices that have different area footprint and different branch number.

The construction of good diffusion matrices has always been an important research topic in cryptography, equally important as the search for good confusion functions. The `AES` [13] for example uses a 4×4 matrix with elements in $GF(2^8)$. This matrix is Maximum Distance Separable (MDS), which means that it has a branch number of 5, optimal for a 4×4 matrix. However, this security feature comes at a cost that computations in $GF(2^8)$ might not be the best choice for some hardware purposes, even though special care has been taken by the designers to choose a circulant matrix instantiated with lightweight coefficients of low Hamming weight.

Recently, Guo *et al.* [14,15] described a new type of diffusion matrix, so-called serial, that trades more clock cycles in the execution for a smaller area. This idea was later extended to the use of linear Feistel-like structures or Linear Feedback Shift Registers (LFSR) to build the diffusion matrix [18,20]. On the opposite side, `PRESENT` [8] uses a simple bit permutation layer, the real diffusion coming in fact directly from the S-box application. The advantage being that a bit permutation layer is basically free in a hardware implementation. Now, one may ask the following question: what is better when the goal is to maximize some hardware metric, a very weak diffusion matrix with a low area footprint, or a strong diffusion matrix but requiring more silicon?

More generally, many different trade-offs exist when building an `AES`-like Substitution-Permutation Network (SPN) primitive, such as the general geometry (number of lines and columns), what size of S-box, what type of matrix, with what branch number, in what finite field, with which irreducible polynomial, etc. When a cryptographer would like to design a permutation with a specific hardware efficiency metric in mind, it is not trivial for him to make the best construction choices directly. Since implementing many different trade-offs is very time consuming, he will have to rely on his own intuition when picking the basic building blocks and choosing the general structure of the primitive, therefore accepting that his final design might not be optimal.

Our contributions. In this article, we study the problem of designing hardware efficient permutations for lightweight symmetric key cryptography purposes, and we propose new promising diffusion matrices as building blocks. We first explain in Section 2 the family of functions that we will study, namely `AES`-like SPN permutations, and we describe a new generalized diffusion layer (i.e. the `ShiftRows` function in `AES`), that allows a provable optimal diffusion even for non-square internal state matrices. Then, we introduce in Section 3 a new metric, the *figure of adversarial merit* (FOAM), that for the first time takes into account the inherent security provided by the primitive. We then explain in Section 4 the various SPN design tradeoffs that we will consider for our comparisons, such as the geometry of the SPN, the S-box size, the type of matrix (circulant or serial), the field size for the diffusion or even the irreducible polynomial. The goal being that the designer only has to input the type of implementation (round and/or serial) and the size of the permutation he would like to build, and he can directly get the SPN structure and its internal components that are the best suited for him. We study in Section 5 the security of the `AES`-like SPN permutations by only taking in account simple linear/differential attacks. In Section 6, we present formulas for estimating various parts of the ASIC implementations. We chose to focus our work on designing permutations only since many cryptographic primitives can be built from them. Therefore, we will not cover other components such as key schedule for a block cipher, or message expansion for a hash function. Moreover, due to the obviously vast amount of implementation trade-offs, we restricted ourselves to the two most important cases: fully serialized and round-based.

Finally, the results obtained by our analysis are given in Section 7, with the best diffusion matrices and SPN parameters we could find for many different scenarios. Notably, we show that the diffusion matrices of ciphers such as `AES`, `LED` or `PHOTON` are not the best possible choices. For example, in the case of `AES` encryption, a circulant matrix with coefficients (0x01,0x01,0x04,0x8d) would have been, surprisingly, a better choice in terms of implementation while keeping the same MDS security.

2 Generic SPN with generalized optimal diffusion

In this section, we describe the family of `AES`-like SPN functions. Our scope is classical, but we propose a new generalized diffusion layer that allows an optimal diffusion even for non-square internal state matrices.

2.1 Extended `AES`-like permutations

An n-bit `AES`-like SPN permutation transforms an $r \times c$ array of s-bit cells ($n = r \times c \times s$). During one round, each cell is first transformed by an s-bit S-box (similar to the `AES` `SubBytes` operation). Then each r-cell column is transformed by an $r \times r$ diffusion matrix (similar to the `AES` `MixColumns` operation), followed by an optimal diffusion[4] which permutes the c cells of each row to provide further mixing (similar to the `AES` `ShiftRows` operation). Finally, an $(r \times c)$-cell constant is xored to complete a round transformation (in block-cipher design, this phase is a subkey addition, but we will not consider key-schedules here). In `AES`, we have a square array $r = c = 4$ and cell size $s =$ 8-bit. The diffusion matrix is usually defined over the finite field $GF(2^s)$ because of the s-bit cell size. Sometimes, we might actually use a smaller subfield

[4] Note that here, without loss of generality, we apply the permutation operations from right-to-left, i.e. `SC` (SubCells) is first applied, followed by `MC` (MixColumn) and then the optimal diffusion.

of size $GF(2^i)$, i divides s, in order to define the diffusion matrix. This framework captures many known ciphers such as `AES`, `PRESENT`, `LED`, etc.

A cell is called differentially (resp. linearly) active if its value (resp. mask value) is non-zero in a differential (resp. linear) attack. The differential branch number of a diffusion matrix is the minimum number of differentially active input and output cells (among all non-zero inputs). The notion of linear branch number is similar, except that we consider the transpose of the diffusion matrix instead. From this point onwards, we will not distinguish between differential and linear branch number unless necessary. That is, when we say a matrix has branch number B, both its differential and linear branch numbers are equal to B. The maximum branch number for an r by r diffusion matrix is $r+1$, and a matrix which achieves this optimal branch number is called MDS. If the diffusion matrix has branch number r, then it is called almost-MDS.

2.2 The generalized optimal diffusion

In this section, we generalize the concept of optimal diffusion [13] for non-square state array. This has been done already when $r < c$ with a security bound equivalent to the case where $r = c$ (square array) [13]. When $r > c$ and c divides r, a simple generalization has been proposed in [11] where a 4-round security bound is proven when the diffusion matrix is MDS. In this section, we propose a generalized optimal diffusion for the case $r > c$ where c may not divide r and the diffusion matrix may not be MDS, i.e. for all branch number $B \leq r+1$.

An example of optimal diffusion is the `ShiftRows` operation of `AES` which helps to diffuse the effect of the `AES SubBytes` and `MixColumns` operation over 32-bit to the whole 128-bit block. The `AES ShiftRows` transforms a 4×4 byte-array by rotating row r to the left by r bytes, for $r = 0, 1, 2, 3$. Due to `ShiftRows`, each byte of an input column is mapped to a different output column. This is captured by the concept of optimal diffusion (another example is `SQUARE` cipher [12]'s `ArrayTranspose` map).

Definition 1. *For an r-by-r cell-array, the optimal diffusion map is a cell-permutation that maps each cell of an input column to a different output column.*

However, the optimal diffusion only applies for $r \times c$ cell array where $r \leq c$. When $r > c$, there are not enough output columns c to map each of the r cells of an input column. Thus, we extend a new concept from [11] called Generalized Optimal Diffusion (`GOD`) for $r \times c$ cell-array when

$r > c$, which we describe below[5]. Our strategy is to distribute the cells of an input column as uniformly as possible to each output column.

Definition 2. *For an $r \times c$ cell-array, a generalized optimal diffusion is a cell-permutation such that looking at any r-cell column:*

1. *$\lceil r/c \rceil$ input cells are mapped to each of $(r \mod c)$ output columns.*
2. *$\lfloor r/c \rfloor$ input cells are mapped to each of $c - (r \mod c)$ output columns.*

Example 1. Consider $r = 5$, $c = 3$. For each input column of 5 cells, $\lceil 5/3 \rceil = 2$ input cells are mapped to each of $(5 \mod 3) = 2$ columns. $\lfloor 5/3 \rfloor = 1$ input cell is mapped to $3 - (5 \mod 3) = 1$ column. One example is given by the transform of the following arrays:

$$\begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \\ a_5 & b_5 & c_5 \end{pmatrix} \text{ maps to } \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ c_3 & a_3 & b_3 \\ c_4 & a_4 & b_4 \\ b_5 & c_5 & a_5 \end{pmatrix}$$

Theorem 1. *Consider a 4-round `AES`-like SPN as follows (omitting the constant addition since it has no effect on our reasoning):*

$$\texttt{GOD} \circ \texttt{MC} \circ \texttt{SC} \circ \texttt{GOD} \circ \texttt{MC} \circ \texttt{SC} \circ \texttt{GOD} \circ \texttt{MC} \circ \texttt{SC} \circ \texttt{GOD} \circ \texttt{MC} \circ \texttt{SC},$$

where

1. `SubCells` *is a nonlinear substitution layer with $r \times c$ s-bit S-boxes acting in parallel.*
2. `MixColumns` *is a layer of c parallel* `MixColumn` *transforms each mapping r cells to r cells with branch number B, i.e.* $\texttt{MixColumns}(\texttt{x}_1, \ldots, \texttt{x}_\texttt{c}) = (\texttt{MixColumn}(\texttt{x}_1), \ldots, \texttt{MixColumn}(\texttt{x}_\texttt{c}))$*, each x_i corresponding to a column of r cells.*
3. `GOD` *(generalized optimal diffusion) is as defined above which distributes the r cells of an input column almost uniformly to c output columns.*

Then the number of active S-boxes over 1 and 2 rounds are at least 1 and B respectively. For 4 rounds it is at least $B \times B'$ where $B' = \max\{2; x+y\}$ and:

$$\begin{cases} y = \min\{2 \times (r \bmod c)); \lfloor B/\lceil r/c \rceil \rfloor\} \\ x = \qquad \lceil (B - \lceil r/c \rceil \times y)/\lfloor r/c \rfloor \rceil \end{cases}$$

[5] The Generalized Optimal Diffusion (`GOD`) defined in [11] applies only when r is a multiple c. Here, we define `GOD` for any $r > c$.

We provide the proof of this theorem in the full version of this paper. We note that it is tight in the sense that it naturally provides a 4 round path that corresponds to a "luckiest" scenario for the attacker, which involves the minimum number of active Super-Sboxes (the $(c \times s)$-bit S-boxes composed of two `SubCells` layers surrounding one `MixColumns`).

Let us look at an application example of Theorem 1 to derive the number of active S-boxes of an `AES`-like SPN structure, which cannot be deduced by the known results of [11,13]. Consider an SPN structure with state size 24-cell, the diffusion matrices being an 8×8 matrix with branch number 7, i.e. $r = 8$, $c = 3$ and $B = 7$. By Theorem 1, we have $y = 2$ and $x = 1$, therefore $B' = \max\{2; x+y\} = 3$ and there are $B \times B' = 7 \times 3 = 21$ active S-boxes guaranteed over 4 rounds of this 24-cell SPN structure.

3 FOAM: Figure Of Adversarial Merit

As explained in the introduction, the various trade-offs inherent in any design of a cryptographic primitive make a fair and consistent comparison of software and hardware implementations thereof a challenging task. For hardware implementations exist a few metrics, like the *Area-Time (AT) product*, which multiplies the area in *Gate Equivalents* (GE) occupied by the design with the number of clock cycles required (the smaller the number, the more efficient is the design). Closely related is the *hardware efficiency* [9], which divides the throughput at a given frequency by the area (hence the greater the number, the better the design). In order to also address the area-power trade-off, [2] proposed a new *Figure of Merit (FOM)*: throughput divided by the area squared. The latter two metrics are frequency dependent, which can complicate comparisons.

We propose a new metric called Figure of Adversarial Merit (FOAM) in order to resolve the aforementioned shortcomings. It is defined as

$$FOAM(x) = \frac{1}{S(x) \times A^2}$$

where $S(x)$ and A are basically equivalent to special definitions of speed and area, respectively. More precisely, $S(x)$ denotes the speed of the cipher based on the number of rounds required to achieve a certain security x against some set of attacks (in this article, we will later restrict ourselves to simple differential/linear attacks). For a round-based permutation, it is defined as $S(x) = p(x) \times t$ where $p(x)$ represents the number of rounds required to achieve security x, and t the number of clock cycles to perform one round. Moreover, for SPN-based primitives, we decompose the area requirements A into six parts: the intermediate state memory cost C_{mem}, the S-boxes implementation cost C_{sbox}, the diffusion matrix implementation cost C_{diff}, the constant addition C_{cst}, the control logic cost C_{log}, and the IO logic cost C_{io}:

$$FOAM(x) = \frac{1}{S(x) \times A^2} = \frac{1}{p(x) \times t \times (C_{mem} + C_{sbox} + C_{diff} + C_{cst} + C_{log} + C_{io})^2}$$

This FOAM metric will be useful to compare different design strategies, different building blocks (such as diffusion matrices) with a simple value computation. Even better, we would like to roughly compare all these possible design trade-offs without having the hassle to implement all of them: in Section 6 we present formulas to estimate these six subparts of the area cost and the number t of clock cycles required to perform one round. The value $p(x)$ can be deduced by the number of active S-boxes proven in Theorem 1 and the S-box cryptographic properties (see Section 5). Note that in the rest of the paper, we consider that the security aimed by the designer is equal to the permutation size, i.e. we are aiming at a security of 2^n computations (thus $p(x) = p(2^n)$).

4 Trade-offs considered

We explain all the various trade-offs we consider when building an `AES`-like SPN permutation. The goal being that a designer specifies a permutation bitsize n, the metric he would like to maximize (area, FOAM), the degree up to which serial or round-based implementations are important, and he directly obtains the best parameters to build his permutation.

The S-box. One of the first choice of the designer is the size of the S-box, and we will consider two possible trade-offs: $s = 4$ and $s = 8$. Note that, for simplicity, we will consider that the S-box chosen has perfect differential and linear properties relative to its size (one could further extend the trade-offs to non-optimal but smaller S-boxes, but the search space being very broad we leave this as an open problem).

The geometry of the internal state. When building an `AES`-like SPN permutation, one can consider several internal state geometries (the values r and c). The classical case is a square state, like for `AES`. However, depending on the diffusion matrices available, it might be worth considering more line-shaped or column-shaped designs.

Diffusion matrix field size. The designer can choose the field size 2^i in which the matrix computations will take place. The classical case, like in `AES`, being that the field size for the diffusion matrix is the same as the S-box. However, depending on the diffusion matrices available, it might be worth considering designs with thinner diffusion layers but repeated several times. For example, in the case of `AES`, instead of the `MixColumns` matrix one could use a 4×4 diffusion matrix on $GF(2^4)$ applied two times (one time on the 4 MSB and one time on the 4 LSB of the 8-bit cells in the `AES` column). Overall, we will cover a scope from binary matrices (in GF(2)) up to matrices on the same field size as the S-box (in $GF(2^s)$).

Irreducible polynomial for the diffusion matrix field. Once the field size 2^i is fixed, the designer can choose the irreducible polynomial defining the field. For $i = 1$ and $i = 2$ only a single polynomial exists, while for $i = 4$ at most 3 choices are possible ($\alpha^4 + \alpha + 1$, $\alpha^4 + \alpha^3 + 1$ and $\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$). For the $i = 8$ case, many polynomials are possible (this was already observed by [3]), thus in order to focus the search space we will only consider the irreducible polynomial used in `AES` ($\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1$) and in `WHIRLPOOL` hash function [5] ($\alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1$).

Type of diffusion matrix. The designer can choose what type of matrix he will implement, the two main hardware-friendly types being circulant or serial. In the circulant case, the designer picks r coefficients $Z = (Z_0, \ldots, Z_{r-1})$ and the matrix Z is defined as

$$\begin{pmatrix} Z_0 & Z_1 & Z_2 & . & . & . & Z_{r-2} & Z_{r-1} \\ Z_{r-1} & Z_0 & Z_1 & . & . & . & Z_{r-3} & Z_{r-2} \\ Z_{r-2} & Z_{r-1} & Z_0 & . & . & . & Z_{r-4} & Z_{r-3} \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ Z_1 & Z_2 & Z_3 & . & . & . & Z_{r-1} & Z_0 \end{pmatrix}$$

In the serial case, the designer picks r coefficients $Z = (Z_0, \ldots, Z_{r-1})$ and the matrix Z is defined as

$$\begin{pmatrix} 0 & 1 & 0 & 0 & . & . & 0 & 0 \\ 0 & 0 & 1 & 0 & . & . & 0 & 0 \\ 0 & 0 & 0 & 1 & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & . & 0 & 1 \\ Z_0 & Z_1 & Z_2 & . & . & . & Z_{r-2} & Z_{r-1} \end{pmatrix}^r$$

The matrix therefore takes r operations to be computed.

Branch number of the diffusion matrix. In general, implementing a matrix with very good diffusion property will cost more area and/or cycles than a weak one. For example, the `AES` matrix has ideal MDS diffusion property, but certainly requires more area to implement than a simple binary matrix with weaker properties. Since the former is bigger but stronger and the latter is smaller and weaker, it is not clear which alternative will lead to the best FOAM. Therefore, the designer can choose between a wide range of possibilities concerning the branch number B of the diffusion matrix, from $B = 3$ to $B = r + 1$ (MDS).

5 Security assessment of `AES`-like primitives

The FOAM metric takes into account the security of the permutation with regards to simple differential/linear attacks. We would like to evaluate this security for the `AES`-like SPN permutations we are considering. Theorem 1 gives us the minimal number of active S-boxes for a given number of rounds[6], and knowing the S-box cryptographic properties we can compute the maximum differential and linear characteristic probabilities of our generic SPN ciphers easily. In other words, we can easily compute the number of rounds $p(x) = p(2^n)$ required to achieve the aimed security 2^n.

As stated before, for simplicity, in the rest of this article we will consider that the S-boxes have perfect differential and linear properties: for a 4-bit S-box the maximum differential and linear characteristic probabilities are 2^{-2} (e.g. PRESENT S-box), while for a 8-bit S-box the maximum differential and linear characteristic probabilities are 2^{-6} (e.g. AES S-box). One can extend the trade-off by considering other S-boxes, that might require a smaller area, but will have worse security properties.

Reusing the example from Section 2.2, from Theorem 1, there are at least 21 active S-boxes over 4 rounds of this SPN permutation. Suppose that 8-bit S-boxes are of maximum differential and linear probabilities 2^{-6}. Then the maximum differential and linear characteristic probabilities over four rounds are upper-bounded by $(2^{-6})^{21} = 2^{-126}$.

We are aware that other attacks rather than simple differential/linear might exist. However, our goal here is not to fully specify a permutation, but to compare many trade-offs and design strategies that will lead to

[6] We note that the number of active S-boxes given by Theorem 1 is tight if the number of rounds is not equal to 3 modulo 4 (even in that case the theorem gives a very close estimation). This does not mean that the maximum differential and linear characteristic probabilities computed are tight, since it is unknown how many active S-boxes can use the maximum differential and linear characteristic probabilities at the same time (this remains an open problem).

good hardware performances. Therefore, we emphasize that the number of rounds $p(x)$ is not the number of rounds that should be chosen by a designer. This number should be carefully chosen after thorough crypt-analysis work on the entire primitive. Yet, we believe that this simple differential/linear criterion is a quite accurate way to compare the security of `AES`-like SPN permutations.

6 Implementations in ASIC

In this section, we introduce some notation before we present formulas to estimate serialized and round-based implementations (we restricted ourselves to these two important practical cases due to the obviously vast amount of implementation trade-offs). Please note that all estimates have to be seen as *lower bounds*, as we use scan flip-flops, and consider neither reset nor I/O requirements, which can significantly impact the area count in practice. We argue that those requirements –though very important in practice– are highly application specific, and will be the same for any permutation for a given target application scenario. Thus for a fair comparison of permutation constructions we will not consider them. In practice, a higher throughout can be achieved by using pipelining techniques to reduce the critical path at the cost of additional area. As this design goal is, again, highly application specific and FOAM is designed to be frequency independent, we have not considered it in our analysis.

We have estimated all serial architectures with the single optimization goal of minimal area in mind. In practice, some design decisions will most likely use another trade-off point more in favor of smaller time and larger area. To reflect this, we have estimated all round-based architectures optimized for maximum FOAM.

The table below provides an overview over the hardware building blocks we used, their notation and typical area requirements for a UMC 180 nm technology. [7]

Notation	Description	GE	Notation	Description	GE
DFF	1-input flip-flop	4.67	XOR	2-input exclusive Or	2.67
SFF	2-input flip-flop	6	$SB4$	4 x 4 S-box (PRESENT)	22
MUX	2-input multiplexer	2.33	$SB8$	8 x 8 S-box (AES)	233

[7] This is just *one example* for a technology and the area of the building blocks can be easily adapted for other technologies.

We give in Table 1 the estimates for the various parts of the ASIC implementations. The details on how these formulas were obtained will be provided in the full version of this paper.

Table 1: Estimates for various parts of the ASIC implementations. (i denotes the exponent for the field $GF(2^i)$; a_r, a_c and a_p denote the counters for rows, columns and rounds respectively; cg and oc denote clock gating and other combinational logic respectively; b denotes the area requirement for the finite state machine.)

	Serial architectures	Round-based architectures
C_{mem}	$s \cdot (r - \lfloor \frac{i}{s} \rfloor) \cdot SFF + \lfloor \frac{i}{s} \rfloor s \cdot DFF$, $c = 1$ $2 \cdot s \cdot r \cdot SFF + s \cdot r \cdot (c-2) \cdot DFF$, $c \geq 2$	$s \cdot r \cdot c \cdot SFF$
C_{sbox}	$SB4$, $s = 4$ $SB8$, $s = 8$	$r \cdot c \cdot SB4$, $s = 4$ $r \cdot c \cdot SB8$, $s = 8$
C_{diff}	$A \cdot XOR$, for serial mat. $A \cdot XOR + (s \cdot r - i) \cdot DFF + i \cdot MUX$, for circulant mat.	$A \cdot r \cdot c \cdot \frac{s}{i} \cdot XOR$
C_{cst}	$s \cdot XOR$	$s \cdot r \cdot c \cdot XOR$
C_{log}	$a_r + a_p + SFF \cdot 2 + oc$, $c = 1$ $a_r + a_c + a_p + b + cg + oc$, $c \geq 2$	$a_p + b$
C_{io}	$s \cdot MUX$	0
t	$r \cdot c + (c-1) + (\frac{s}{i} \cdot r + 1 - \lfloor \frac{i}{s} \rfloor) \cdot c$, $c \geq 2$ serial mat. $r \cdot c + (c-1) + (2 \cdot \frac{s}{i} \cdot r) \cdot c$, $c \geq 2$ circulant mat. $r \cdot c + \frac{s}{i} \cdot r$, $c = 1$ serial mat. $r \cdot c + (2 \cdot \frac{s}{i} \cdot r - 1)$, $c = 1$ circulant mat.	1

7 Results and new diffusion matrices

In this section we provide the results of our framework, as well as new diffusion matrices that are very interesting for hardware implementations. As explained in Section 4, the designer's input is the permutation bitsize n, the metric he would like to maximize (area or FOAM), and the degree up to which serial or round-based implementations are important. To illustrate our method, we focused on the case where the designer would like to build a 64-bit permutation (which is a typical state size for a lightweight block cipher). For the implementation types, we focused on three scenarios: only serial implementation is important, only round-based implementation is important, serial and round-based implementations are equally important for the designer. Further, we only considered encryption.

Before describing our results, we first explain how we found good diffusion matrices (circulant and serial), which outperform known ones from the AES, LED ciphers and the PHOTON hash function.

7.1 Lightweight coefficients

Consider the AES matrix, a circulant matrix with coefficients (0x01, 0x01, 0x02, 0x03) over $GF(2^8)$ defined by the irreducible polynomial $\alpha^8+\alpha^4+\alpha^3+\alpha+1$. The matrix appears to be very lightweight due to the low Hamming weight of its entries. But surprisingly, we found an even lighter circulant matrix over the same field with coefficients[8] (0x01,0x01,0x04,0x8d). We now explain why this is so.

We first illustrate how to compute the number of XORs required to implement a multiplication by a finite field element x, by using $GF(2^8)$ defined by $\alpha^8+\alpha^4+\alpha^3+\alpha+1$ as an example. Let $x = x_7\cdot\alpha^7 + x_6\cdot\alpha^6 + \cdots x_1\cdot\alpha + x_0 = (x_7, x_6, \cdots, x_1, x_0)$. For ease of explanation, we employ hexadecimal encoding: $(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$ can be encoded as a tuple of hexadecimal numbers (0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01). Then, the multiplication of 0x04 is represented as:

$$\begin{aligned}\text{0x04}\cdot x &= (x_5, x_4, x_3+x_7, x_2+x_6+x_7, x_1+x_6, x_0+x_7, x_6+x_7, x_6)\\ &= (\text{0x20, 0x10, 0x88, 0xc4, 0x42, 0x81, 0xc0, 0x40}).\end{aligned}$$

We see that the number of XORs required for the multiplication of 0x04 by x is 6. Now we can compute

$$\begin{aligned}\text{0x8}d\cdot x &= (\alpha^7+\alpha^3+\alpha^2+1)\cdot x\\ &= \big(\text{0xb1, 0x58, 0x2c, 0x96, 0xfa, 0x4c, 0xa6, 0x62}\big)\oplus\big(\text{0x10, 0x88, 0xc4, 0x62, 0xa1, 0xc0, 0x60, 0x20}\big)\\ &\quad\oplus\big(\text{0x20, 0x10, 0x88, 0xc4, 0x42, 0x81, 0xc0, 0x40}\big)\oplus\big(\text{0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01}\big)\\ &= (\text{0x01, 0x80, 0x40, 0x20, 0x11, 0x09, 0x04, 0x03})\\ &= (x_0, x_7, x_6, x_5, x_0+x_4, x_0+x_3, x_2, x_0+x_1)\end{aligned}$$

Due to the '*cancellation of XORs*', we see that multiplication of x by 0x8d requires only 3 XORs. In a similar fashion, the multiplication of x by 0x02 and 0x03 requires 3 and 11 XORs respectively.

Hence we are able to come up with the XOR count table for any finite field. Table 2 of Appendix A shows the XOR count for $GF(2^4)$ defined by $\alpha^4+\alpha+1$. The tables for $GF(2^4)$ and $GF(2^8)$ defined by different irreducible polynomials are provided in the full version of this paper.

Now we explain how to use the tables to calculate A the number of XORs required to implement a row of a matrix. Denote a given row of an $r\times r$ matrix by $(x_1, x_2, \cdots x_r)$ over a finite field $GF(2^i)$. Let γ_j be the XOR count(e.g. Table 2 of Appendix A for $i=4$) corresponding to the field element x_j. Then A is equal to $(\gamma_1+\cdots+\gamma_r)+(z-1)\cdot i$, where z is the number of non-zero elements in the row. We give some examples: row

[8] We use the binary representation to represent finite field elements. E.g., 0x8d is 10001101 in binary, which corresponds to the finite field element $\alpha^7+\alpha^3+\alpha^2+1$ in $GF(2^8)$.

(0x1,0x1,0x4,0x9) uses $(0+0+2+1)+3\times 4=15$ XORs to implement over $GF(2^4)$; the AES matrix uses $(0+0+3+11)+3\times 8=38$ XORs to implement per row over $GF(2^8)$. Similarly, the circulant matrix with coefficients (0x01,0x01,0x04,0x8d) uses 33 XORs to implement per row over $GF(2^8)$, and is thus lighter than the AES matrix.

7.2 Subfield construction

In this section, we describe the subfield construction[9] which allows us to outperform the AES matrix even more than the optimal matrix found in Section 7.1. As computed in the previous subsection, the MDS circulant matrix $circ(0x1, 0x1, 0x4, 0x9)$ over $GF(2^4)$ defined by $\alpha^4+\alpha+1$ requires 15 XORs to implement per row. Using the method of [11, Section 3.3], we can form a circulant MDS matrix over $GF(2^8)$ by using two parallel copies of $Q = circ(0x1, 0x1, 0x4, 0x9)$ over $GF(2^4)$. The matrix is formed by writing each byte q_j as a concatenation of two nibbles $q_j = (q_j^L||q_j^R)$. Then the MDS multiplication is computed on each half $(u_1^L, u_2^L, u_3^L, u_4^L) = Q\cdot(q_1^L, q_2^L, q_3^L, q_4^L)$ and $(u_1^R, u_2^R, u_3^R, u_4^R) = Q\cdot(q_1^R, q_2^R, q_3^R, q_4^R)$ over $GF(2^4)$. The result is concatenated to form four output bytes (u_1, u_2, u_3, u_4) where $u_j = (u_j^L||u_j^R)$. This matrix needs just $15\times 2=30$ XORs to implement per row. In comparison, the lightest MDS circulant matrix $circ$(0x01,0x01,0x04,0x8d) over $GF(2^8)$ defined by $\alpha^8+\alpha^4+\alpha^3+\alpha+1$ requires more XORs (33 XORs per row).

Further, we can serialize the above multiplication to do the left half followed by the right half, in which case only 15 XORs are needed to implement one row of the MDS matrix over $GF(2^8)$. Another advantage of subfield construction is exemplified by the SPN-Hash construction [11]. Instead of finding an 8×8 serial MDS matrix over $GF(2^8)$ exhaustively, two parallel copies of the PHOTON 8×8 serial MDS matrix over $GF(2^4)$ were concatenated to form the 8×8 serial MDS matrix over $GF(2^8)$ for SPN-Hash.

We can generalize this method to form a diffusion matrix with branch number B over $GF(2^s)$ from s/i copies of a diffusion matrix of the same branch number over a subfield $GF(2^i)$, where i divides s.

7.3 Good matrices

We search for optimal low-weight $r\times r$ circulant and serial matrices of different branch numbers (3 to $r+1$) over the finite fields $GF(2)$, $GF(2^2)$,

[9] This idea of subfield construction was used in the SHA3 submission ECHO [6] and later in WHIRLWIND [4] and SPN-Hash [11].

$GF(2^4)$ and $GF(2^8)$, and list them in Table 3 of Appendix A. Using the construction of Section 7.2, we can form diffusion matrices to transform nibbles and bytes from these subfields.

The optimal matrices are found by exhaustively checking the branch number of all matrices and choosing the one with the least number of XORs according to the method explained in Section 7.1. To check the branch number of matrix Q, we concatenate it with the identity matrix I_r to form $(I_r|Q)$, the generating matrix of the corresponding linear code, and use the MAGMA software to find the distance[10]. For branch number B, we check that both Q and its transpose Q^t has branch number B.

The matrices are optimal in the sense that they need minimal number of XORs to implement. In the events of a tie between two matrices, possibly over different finite field representations, we just list one of them. For example, the circulant matrices $circ$(0x01,0x01,0x04,0x8d) over $GF(2^8)$ defined by $\alpha^8+\alpha^4+\alpha^3+\alpha+1$ and $circ$(0x01,0x01,0x04,0x8e) over $GF(2^8)$ defined by $\alpha^8+\alpha^4+\alpha^3+\alpha^2+1$ both outperforms the `AES` matrix by using 33 XORs to implement one row, so we just list the latter. We use "-" when no circulant matrix with branch number B exists (verified by exhaustive search or coding theory bounds). For example, it can be verified that 8×8 circulant MDS matrix does not exist in the finite field $GF(2^4)$. However, we could not find the optimal 8×8 circulant MDS matrix over $GF(2^8)$. Because the search space is too big to exhaust, we just list the `WHIRLPOOL` matrix which is MDS and low weight.

We use "*" to denote that we have not found the serial matrix with branch number B at this point of time due to the huge search space. For instance, as the search space is too big to exhaust, we could not find a 8×8 serial MDS matrix over $GF(2^8)$. In this case, we can employ the method of subfield construction (described in Section 7.2), i.e. use two parallel copies of the 8×8 MDS serial matrix with last row (0x2,0xd,0x2,0x4,0x3,0xd,0x5,0x2) (refer to second row of 8×8 subtable of Table 3) over $GF(2^4)$ to obtain the desired matrix over $GF(2^8)$.

7.4 Application: FOAM Comparison for 64-bit SPN Structures

In this section, we compare the FOAM metric for 64-bit SPN Structures. Table 4 in Appendix A gives the results for a SPN structure based on

[10] We are aware that better techniques than naive exhaustive search might be used here. However, such improvements are not the goal of this article and we leave them as potential future work.

4-bit `PRESENT` S-box with circulant matrices or serial matrices. Due to space constraints, we will provide the results for a SPN structure based on 8-bit `AES` S-box with circulant matrices or serial matrices in the full version of this paper. The diffusion matrices are based on the optimal matrices found in Section 7.3. To compute $p(2^{64})$, the number of rounds to achieve differential/linear probability $\leq 2^{-64}$, we use the fact that the differential/linear probability of the `PRESENT` S-box is 2^{-2} and that of the `AES` S-box is 2^{-6}. Then we lower bound the number of active S-boxes by concatenating 4-round bounds with $B \times B'$ active S-boxes from Theorem 1, 2-round bounds with B active S-boxes and 1-round bound which involves only 1 active S-box. We also write down t, the time to compute one round for serialized implementation (the time t for round based implementation is the constant 1, so it is not presented).

We compute the FOAM for round-based and serialized implementation based on the formula found in Section 6. We also present the FOAM for half-half implementation, where we take the average, i.e. equal weighting, of the round-based and serialized FOAM. This corresponds to implementations which are good for both scenarios. However, this represents just one example, as the weighting of the scenarios is clearly a designer's choice. The structure with the best area and FOAMs are in bold.

We see that for designing 64-bit SPN:

1. For minimal area the geometry is the most important criterion, while the choice of the field of the MDS matrix is of less importance. The geometry should be chosen, such that c is maximized, and consequently, many internal columns can be realized with 1-input flip-flops. A serial matrix is favorable over a circulant matrix and in general smaller fields allow to save a few GE, but come at a high timing overhead.
2. `PRESENT` **S-box**
   - When Circulant Matrices are used with `PRESENT` S-box in Table 4 from Appendix A, the 4×4 almost-MDS circulant matrix $circ$(0x1, 0x1, 0x1, 0x0) over $GF(2^4)$ gives the best FOAM for round-based, serial and half-half implementations.
   - When Serial Matrices are used with `PRESENT` S-box in Table 4 from Appendix A, the 4×4 almost-MDS serial matrix with last row (0x1, 0x0, 0x2, 0x1) over $GF(2^4)$ defined by $\alpha^4+\alpha+1$ gives the best FOAM for round-based, serial and half-half implementations.
3. `AES` **S-box**
   - From our results for `AES` S-box (provided in the full version of the paper), when Circulant Matrices are used with `AES` S-box, two parallel copies of the 4×4 MDS matrix $circ$(0x1, 0x1, 0x4,

0x9) over $GF(2^4)$ defined by $\alpha^4 + \alpha + 1$ gives the best FOAM for round-based implementation. The 4×4 MDS matrix $circ$(0x01, 0x01, 0x04, 0x8e) over $GF(2^8)$ defined by $\alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1$ gives the best FOAM for serial and half-half implementations.
   - When Serial Matrices are used with AES S-box, two parallel copies of the 4×4 MDS serial matrix with last row (0x2, 0x1, 0x1, 0x4) over $GF(2^4)$ defined by $\alpha^4+\alpha+1$ gives the best FOAM for round-based implementation. The 8×8 serial matrix (having branch number 6) with last row (0x01, 0x01, 0x00, 0x00, 0x01, 0x01, 0x02, 0x00) over $GF(2^8)$ defined by $\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1$ gives the best FOAM for serial and half-half implementations and is also very competitive for round-based FOAMs. It is thus a very interesting choice for many different applications.
4. Structures based on PRESENT S-box have higher FOAM for round-based and half-half implementations than those based on AES S-box. On the other hand, structures based on AES S-box have higher FOAM for serial implementation than PRESENT S-box, because they need significantly less rounds.
5. For structures using both types of S-boxes, 4×4 matrices have higher FOAM than 2×2 and 8×8 matrices.
6. Based on the above observations, we do not always go for the matrix with the best branch number: for PRESENT S-box in Table 4 from Appendix A, we use almost-MDS 4×4 matrix which gives better trade-offs and a higher FOAM than MDS matrix. Moreover, we found that when AES S-box is used with 8×8 matrices, we go for the one with branch number 6 instead of the optimal 9.

References

1. Avoine, G., Oechslin, P.: A Scalable and Provably Secure Hash-Based RFID Protocol. In: PerCom Workshops, IEEE Computer Society (2005) 110–114
2. Badel, S., Dagtekin, N., Nakahara, J., Ouafi, K., Reffé, N., Sepehrdad, P., Susil, P., Vaudenay, S.: ARMADILLO: A Multi-purpose Cryptographic Primitive Dedicated to Hardware. In: CHES. (2010) 398–412
3. Barkan, E., Biham, E.: In How Many Ways Can You Write Rijndael? In: ASIACRYPT. (2002) 160–175
4. Barreto, P.S.L.M., Nikov, V., Nikova, S., Rijmen, V., Tischhauser, E.: Whirlwind: a new cryptographic hash function. Des. Codes Cryptography **56**(2-3) (2010) 141–162
5. Barreto, P.S.L.M., Rijmen, V.: Whirlpool. In: Encyclopedia of Cryptography and Security (2nd Ed.). (2011) 1384–1385
6. Benadjila, R., Billet, O., Gilbert, H., Macario-Rat, G., Peyrin, T., Robshaw, M., Seurin, Y.: SHA-3 Proposal: ECHO. Submission to NIST (2008)

7. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Sponge functions. Ecrypt Hash Workshop 2007 (May 2007)
8. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In Paillier, P., Verbauwhede, I., eds.: CHES. Volume 4727 of LNCS., Springer (2007) 450–466 http://lightweightcrypto.org/present/.
9. Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash Functions and RFID Tags: Mind the Gap. In: CHES. (2008) 283–299
10. Cannière, C.D., Dunkelman, O., Knezevic, M.: KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: CHES. (2009) 272–288
11. Choy, J., Yap, H., Khoo, K., Guo, J., Peyrin, T., Poschmann, A., Tan, C.H.: SPN-Hash: Improving the Provable Resistance against Differential Collision Attacks. In: AFRICACRYPT. (2012) 270–286
12. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher Square. In: FSE. (1997) 149–165
13. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
14. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: CRYPTO. (2011) 222–239
15. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: CHES. (2011) 326–341
16. Henrici, D., Götze, J., Müller, P.: A Hash-based Pseudonymization Infrastructure for RFID Systems. In: SecPerU. (2006) 22–27
17. Lee, S.M., Hwang, Y.J., Lee, D.H., Lim, J.I.: Efficient Authentication for Low-Cost RFID Systems. In: ICCSA (1). (2005) 619–627
18. Sajadieh, M., Dakhilalian, M., Mala, H., Sepehrdad, P.: Recursive Diffusion Layers for Block Ciphers and Hash Functions. In: FSE. (2012) 385–401
19. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In: CHES. (2011) 342–357
20. Wu, S., Wang, M., Wu, W.: Recursive Diffusion Layers for (Lightweight) Block Ciphers and Hash Functions. In Knudsen, L., Wu, H., eds.: Selected Areas in Cryptography. Volume 7707 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 355–371

A Tables

Table 2: XORs required to implement a multiplication by x over $GF(2^4)$.

x (hexadecimal representation)	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\alpha^4 + \alpha + 1$	0	0	1	5	2	6	5	9	3	1	8	6	5	3	8	6

Table 3: Good Circulant Matrices of Size 2×2, 4×4 and 8×8
B denotes the branch number; The "First Row" and the "Last Row" column (in hexadecimal) represents the first row of the circulant matrix and the last row of the serial matrix (as described in Section 4) respectively; A denotes the number of XOR gates needed to implement one row of the circulant matrix and the last row of the serial matrix respectively.

2×2					
		Circulant matrices		Serial matrices	
Finite Field	B	First Row	A	Last Row	A
$GF(2^8)$, $\alpha^8+\alpha^4+\alpha^3+\alpha^2+1$	3	1,2	11	1,2	11
$GF(2^4)$, $\alpha^4+\alpha+1$	3	1,2	5	1,2	5
$GF(2^2)$, $\alpha^2+\alpha+1$	3	1,2	3	1,2	3
$GF(2)$	3	-	-	-	-

4×4					
		Circulant matrices		Serial matrices	
Finite Field	B	First Row	A	Last Row	A
$GF(2^8)$, $\alpha^8+\alpha^4+\alpha^3+\alpha^2+1$	5	1,1,4,8e	33	1,2,1,4	33
	4	1,1,1,0	16	1,0,2,1	19
	3	1,0,0,2	11	1,0,0,1	8
$GF(2^4)$, $\alpha^4+\alpha+1$	5	1,1,4,9	15	2,1,1,4	15
	4	1,1,1,0	8	1,0,2,1	9
	3	1,0,0,2	5	1,0,0,1	4
$GF(2^2)$, $\alpha^2+\alpha+1$	5	-	-	-	-
	4	1,1,1,0	4	1,0,2,1	5
	3	1,0,0,2	3	1,0,0,1	2
$GF(2)$	5	-	-	-	-
	4	1,1,1,0	2	-	-
	3	-	-	1,0,0,1	1

8×8					
		Circulant matrices		Serial matrices	
Finite Field	B	First Row	A	Last Row	A
$GF(2^8)$, $\alpha^8+\alpha^4+\alpha^3+\alpha^2+1$	9	1,1,4,1,8,5,2,9	105	*	*
	8	1,0,1,1,2,2,1,8e	57	1,1,2,0,1,8d,2,1	57
	7	1,0,0,1,1,1,2,8e	46	1,1,2,1,0,0,1,8d	46
	6	1,0,0,0,1,1,1,2	35	1,1,0,0,1,1,2,0	35
	5	1,0,0,0,0,1,1,2	27	1,0,0,1,1,1,0,0	24
	4	1,0,0,0,0,0,1,1	16	1,0,0,0,0,1,1,0	16
	3	1,0,0,0,0,0,0,2	11	1,0,0,0,0,0,1,0	8
$GF(2^4)$, $\alpha^4+\alpha+1$	9	-	-	2,d,2,4,3,d,5,2	50
	8	1,0,1,1,2,9,2,1	27	*	*
	7	1,0,0,1,1,1,2,9	22	1,0,2,1,1,1,2,0	22
	6	1,0,0,0,1,1,1,2	17	1,1,0,0,1,1,2,0	17
	5	1,0,0,0,0,1,1,2	13	1,0,0,1,1,1,0,0	12
	4	1,0,0,0,0,0,1,1	8	1,0,0,0,0,1,1,0	8
	3	1,0,0,0,0,0,0,2	5	1,0,0,0,0,0,1,0	4
$GF(2^2)$, $\alpha^2+\alpha+1$	9 - 8	-	-	-	-
	7	-	-	2,1,0,3,1,2,0,1	13
	6	1,0,0,0,1,1,1,2	9	1,0,0,1,1,1,0,2	9
	5	1,0,0,0,0,1,1,2	7	1,0,0,1,1,1,0,0	6
	4	1,0,0,0,0,0,1,1	4	1,0,0,0,0,1,1,0	4
	3	1,0,0,0,0,0,0,2	3	1,0,0,0,0,0,1,0	2
$GF(2)$	9 - 6	-	-	-	-
	5	-	-	1,0,0,1,1,1,0,0	3
	4	1,0,0,0,0,0,1,1	2	1,0,0,0,0,1,1,0	2
	3	-	-	1,0,0,0,0,0,1,0	1

Table 4: FOAM for 64-bit SPN based on 4-bit `PRESENT` S-box and Circulant Matrices or Serial Matrices

Circulant Matrices										
Finite Field	r	c	B	$p(2^{64})$	t	Area (GE) rd based	Area (GE) serial	FOAM $\times 10^{-9}$ rd based	FOAM $\times 10^{-9}$ serial	FOAM $\times 10^{-9}$ half-half
$GF(2^4)$	2	8	3	16	55	1156	541	46.76	3.88	25.32
$GF(2^2)$	2	8	3	16	87	1199	**540**	43.48	2.46	22.97
$GF(2^4)$	4	4	5	8	51	1579	652	50.16	5.77	27.96
	4	4	4	8		1280	633	**76.34**	**6.12**	**41.23**
	4	4	3	16		1156	630	46.76	3.09	24.92
$GF(2^2)$	4	4	4	8	83	1280	627	76.34	3.83	40.08
	4	4	3	16		1199	629	43.48	1.90	22.69
$GF(2)$	4	4	4	8	147	1280	624	76.34	2.18	39.26
$GF(2^4)$	8	2	8	8	49	2091	873	28.58	3.35	15.96
	8	2	7	10		1882	864	28.22	2.73	15.48
	8	2	6	12		1669	851	29.92	2.35	16.14
	8	2	5	14		1498	840	31.83	2.07	16.95
	8	2	4	16		1284	827	37.89	1.87	19.88
	8	2	3	22		1161	823	33.73	1.37	17.55
$GF(2^2)$	8	2	6	12	81	1712	834	28.45	1.48	14.96
	8	2	5	14		1541	829	30.09	1.28	15.69
	8	2	4	16		1284	821	37.89	1.15	19.52
	8	2	3	22		1204	823	31.38	0.83	16.10
$GF(2)$	8	2	4	16	145	1284	818	37.89	0.64	19.27

Serial Matrices										
Finite Field	r	c	B	$p(2^{64})$	t	Area (GE) rd based	Area (GE) serial	FOAM $\times 10^{-9}$ rd based	FOAM $\times 10^{-9}$ serial	FOAM $\times 10^{-9}$ half-half
$GF(2^4)$	2	8	3	16	39	1156	513	46.76	6.09	26.42
$GF(2^2)$	2	8	3	16	63	1199	**508**	43.48	3.85	23.67
$GF(2^4)$	4	4	5	8	35	1579	586	50.16	10.39	30.27
	4	4	4	8		1322	570	**71.48**	**10.99**	**41.23**
	4	4	3	16		1113	561	50.41	5.66	28.04
$GF(2^2)$	4	4	4	8	55	1365	559	67.08	7.26	37.17
	4	4	3	16		1113	556	50.41	3.67	27.04
$GF(2)$	4	4	3	16	87	1113	553	50.41	2.35	26.38
$GF(2^4)$	8	2	9	6	33	3074	794	17.64	8.01	12.82
	8	2	7	10		1882	724	28.22	5.78	17.00
	8	2	6	12		1669	711	29.92	5.00	17.46
	8	2	5	14		1455	697	33.73	4.45	19.09
	8	2	4	16		1284	687	37.89	4.02	20.95
	8	2	3	22		1118	681	36.36	2.97	19.67
$GF(2^2)$	8	2	7	10	51	2053	700	23.72	4.00	13.86
	8	2	6	12		1712	689	28.45	3.44	15.94
	8	2	5	14		1455	681	33.73	3.02	18.37
	8	2	4	16		1284	676	37.89	2.68	20.29
	8	2	3	22		1118	675	36.36	1.95	19.16
$GF(2)$	8	2	5	14	83	1455	673	33.73	1.90	17.81
	8	2	4	16		1284	671	37.89	1.67	19.78
	8	2	3	22		1118	673	36.36	1.21	18.78