eSPF: A Family of Format-Preserving Encryption Algorithms Using MDS Matrices

Donghoon Chang[1], Mohona Ghosh[2], Arpan Jati[1], Abhishek Kumar[1(✉)], and Somitra Kumar Sanadhya[3]

[1] Indraprastha Institute of Information Technology, Delhi, India
{donghoon,arpanj,abhishekk}@iiitd.ac.in
[2] Indian Institute of Information Technology Design and Manufacturing, Jabalpur, India
mohona@iiitdmj.ac.in
[3] Indian Institute of Technology, Ropar, India
somitra@iitrpr.ac.in

Abstract. The construction SPF, presented in Inscrypt-2016 was the first known SPN based format-preserving encryption algorithm. In this work, we significantly improve its performance and flexibility. We term this new construction as eSPF. Unlike SPF, all the basic transformations of eSPF are defined under the field $\mathbb{F}_p$. This allows us to use a MDS matrix instead of the binary matrix used in SPF. The optimal diffusion of MDS matrix leads to an efficient and secure design. However, this change leads to violations in the message format. To mitigate this, we propose a *discarding algorithm* to drop the symbols that are not the elements of the format thus preserving it.

We also present a concrete instantiation of eSPF for digits and its comparison with existing FPE algorithms like FFX and SPF. The performance analysis shows that the proposed design is at least 15 times faster than FFX for most of the practical applications.

Keywords: Format-preserving encryption · MDS matrix · SSN · Cryptanalysis · Substitution-permutation network

1 Introduction

Motivation. Maintaining the confidentiality of messages is one of the main goals of cryptography. Block ciphers are the most popular cryptographic primitives to fulfil this purpose. The conventional block ciphers such as AES [15] and DES [13] handle binary data of specific sizes, for example 128-bit for AES [15]. In many real world applications, it is desirable and essential to have the ciphertext follow the same format as the plaintext. Moreover, ciphertext length expansion is also not allowed in these situations. Encryption of Credit Card Numbers (CCN) or Social Security Numbers (SSN) are examples of such applications. Unfortunately, the conventional block ciphers and their modes such as ECB, CBC, CTR, etc. are not suitable for this purpose.

S.S. Ali et al. (Eds.): SPACE 2017, LNCS 10662, pp. 133–150, 2017.
https://doi.org/10.1007/978-3-319-71501-8_8

Format-Preserving Encryption (FPE) refers to transformation of data that is formatted as a sequence of the symbols in such a way that the encrypted form of the data has the same format and length as the original data. Many financial or e-commerce databases contain CCN or SSN and for both practical and legal reasons, encryption of these values are important. However, these fields that need to be encrypted have fixed formats and a plain use of conventional block cipher will produce ciphertexts violating the specified format.

The problem of encryption over fixed formats was first investigated in the database community by Brightwell and Smith [11]. Schoroeppel and Orman proposed the Hasty Pudding Cipher [32] which first demonstrated an encryption scheme that worked for arbitrary domain. A few years later, Black and Rogaway [9] made the first systematic study of this problem and suggested some approaches to achieve the desired functionality. Being motivated by the real world application, many FPE designs have been proposed such as `FFSEM` [34], `FFX` [3], `BPS` [10], `VFPE` [33], `FEA-1` and `FEA-2` [25]. A special publication of NIST SP800-38G [20] specifies three modes of operation for format-preserving algorithms namely `FF1`, `FF2` and `FF3`. Each of these modes employ an unbalanced Feistel structure and use `AES-128` algorithm as the internal round function. `FF1` and `FF2` invokes `AES-128` algorithm at least 11 times and `FF3` invokes it eight times thus leading to high number of `AES-128` invocations. Moreover, Bellare *et al.* [2] have been shown some message recovery attacks on `FF1` and `FF3`. This was followed by an attack presented in [18] where Durak and Vaudenay presented a practical attack to the `FF3` scheme. Apart from Feistel based FPE schemes another important mechanism of designing FPEs based on card shuffling was adopted in [23,28–30]. In 2016, Chang *et al.* [12] proposed a new FPE algorithm `SPF`, based on a substitution permutation network (SPN) strategy. `SPF` is the first known SPN based FPE algorithm and it has been shown that it is almost 5 times faster than the other known FPE algorithms such as `FF1`, `FF2` and `FF3`. However, the use of binary matrix in the linear layer of `SPF` has two limitations. Firstly, it doesn't allow `SPF` to be applied to formats of all sizes. Secondly, it restricts `SPF` to achieve maximal diffusion and hence the optimal efficiency. In this paper, we aim to address these limitations.

1.1 Our Contribution

We present a new substitution permutation design approach to construct efficient format-preserving encryption family. This is realized by using MDS matrix in the diffusion layer unlike `SPF` construction where binary matrix is used. As a result, higher diffusion is achieved in lesser number of rounds. Moreover, SPF construction doesn't work for format sizes which are multiples of 3 due to its design of the binary matrix. Our proposed construction does not have this limitation and works for any domain size. The other notable advantage of our proposed construction is that one instantiation may work for many formats.

We define the basic transformations for the proposed construction and a concrete instance for digits. The construction uses an iterated block cipher as the

underlying building block. It consists of two algorithms - a non format preserving encryption scheme to generate the *keystream* and a *discarding algorithm* to ensure that the format is preserved.

The domain size of real-world applications of FPE motivates us to incorporate tweak in the proposed design. We propose a new *key scheduling and tweak scheduling algorithm* to realize our design goal. Further, we estimate a lower bound on the number of active S-boxes for different number of rounds for the proposed construction. The security of our design is then analyzed against differential, linear, square, related tweak and key scheduling attacks. Finally, we compare the efficiency of a concrete instance of our proposed construction for the most popular and widely used format - 'digit', with FFX and BPS and show that the proposed design is almost 15 times faster than FFX.

The rest of the paper is organized as follows. In Sect. 2, the important preliminaries are described. The proposed eSPF construction is presented in Sect. 3. The concrete instance of eSPF for digits is presented in Sect. 4. We analyze the security of the proposed scheme against the standard attacks in Sect. 5. This is followed by performance analysis of the same in Sect. 6. Finally, we conclude our work in Sect. 7.

2 Preliminaries

Let $\Sigma = \{0, 1, 2, \ldots, N-1\}$ be the alphabet set, where $N \geq 2$. The size N of the set Σ is referred to as the 'format size' and the elements of Σ are referred to as 'symbols', for example, for digits, $N = 10$. Σ^* denotes the set of strings with elements from Σ. We assume that the plaintext contains symbols only from Σ. If this is not the case, suitable encoding and decoding functions could be used and then one can apply the "rank-then-encipher" approach [3] to use the methods described in this work.

2.1 The Notations Used in the Paper

The following notations have been used throughout the paper.

$\lvert\Sigma\rvert$	:	The number of elements in the set Σ.
$\boxplus_N$	:	Symbol wise addition modulo N.
$\lceil x \rceil$	:	Smallest integer just greater than x.
$S[i]$	:	i^{th} symbol of the string S from the left.
$S \mid\mid T$	:	Concatenation of two strings S and T.
$\lvert S\rvert_N$	:	Length of the string S in base N.
$\mathbb{F}_p$	:	Galois Field $\mathsf{GF}(p)$ where p is prime.

2.2 Specification

An instance $\mathtt{eSPF}_r^N$ denotes a member of eSPF family that has format size N and consists of r-rounds. The input/output of each intermediate round is denoted as

state [15]. Each *state* consists of $n = 16$ symbols. For ease of representation and discussion, we represent each *state* as a 4×4 two-dimensional array of symbols.

The transformation of an input string of length n over symbol set Σ to *state* is described by the function $\mathsf{STATE}(X)$ (Algorithm 1); while the inverse transformation of a *state* to produce a string over Σ^n is described by the function $\mathsf{STRING}(state)$ (Algorithm 2).

Algorithm 1. $\mathsf{STATE}(X)$

input : string X
output: *state*

for $i \leftarrow 0$ **to** $(n-1)$ **do**
  $j \leftarrow i \mod 4$;
  $k \leftarrow \lfloor i/4 \rfloor$;
  $state[j,k] \leftarrow X[i]$;
return *state*

Algorithm 2. $\mathsf{STRING}(state)$

input : *state*
output: string X

for $i \leftarrow 0$ **to** 3 **do**
  for $j \leftarrow 0$ **to** 3 **do**
    $n \leftarrow (i + j \times 4)$;
    $X[n] \leftarrow state[i,j]$;
return X

3 The eSPF Construction

eSPF contains two components: a non format preserving encryption E_k and a *Discarding Algorithm (DA)*, followed by modular addition. To achieve diffusion in our encryption scheme, we use MDS matrix. MDS matrices have tremendous applications not only in the coding theory but also in the design of symmetric cryptographic primitives, for example, AES [15], Camellia [1], SQUARE [14] etc. owing to their highest possible branch number. This make them a natural choice for the diffusion layer since higher branch number ensures higher diffusion rate as well as lesser number of rounds, finally leading to a secure and efficient primitive construction.

Having an MDS matrix for diffusion functionality, requires the operations to be done over a finite field. This stringent requirement, limits the possible format size N to the cardinality of $\mathbb{F}_{p^b}$, where p is a prime and b is an integer. A suitable S-box and MDS matrix over the finite field $\mathbb{F}_{p^b}$ is then used to realize the substitution and permutation layer of our scheme.

As the operations are performed in $\mathbb{F}_{p^b}$, we need to perform a process to discard symbols which are not in format. This discarding process is equivalent of cycle-walking or using modular operation to ensure non-violation of the format. We show that the rate of discarding symbols in our case is low for practical scenarios and it does not affect the efficiency of our scheme significantly. Another important advantage of this construction is that, given one instance, the construction may be used for other formats as well, if their format size is smaller than or equal to p^b which is not possible in the SPF family.

3.1 The Round Transformations

Each round of eSPF consists of the following five following transformations which updates the internal state:

$$\mathsf{SB} \circ \mathsf{SR} \circ \mathsf{MC} \circ \mathsf{KA} \circ \mathsf{TA}$$

SubBytes (SB): A SubBytes transformation $S : \mathbb{F}_{p^b} \rightarrow \mathbb{F}_{p^b}$ is used to create confusion in the cipher. It is a permutation consisting of a bijective mapping to each element of the state. Typically an S-box is the multiplicative inverse function in the field $\mathbb{F}_{p^b}$, i.e.,

$$S : x \rightarrow x^{-1}$$

This mapping is very popular and believed to be a good choice for designing differentially and linearly resistant cipher. However, another popular approach is to do a brute force search and choosing substitution layer based on analyzing the differential and linear properties along with the implementation cost. Many lighweight ciphers adopt the second approach.

ShiftRows (SR): This transformation shifts the rows cyclically over different offsets. Similar to AES, there is no shift over the first row, whereas symbols of second, third and fourth row are shifted left by one, two and three positions respectively.

MixColumns (MC): Permutation layer is used to introduce diffusion in the cipher to make sure that any local differences of an internal state before permutation layer propagates to the larger area of the state after this layer. In many modern ciphers, the linear diffusion layer is realized by using a $r \times r$ matrix that operates on the *state* column by column.

KeyAddition (KA): The key addition transformation modifies the *state* by adding round key symbol wise using modulo addition p^b.

TweakAddition (TA): Similar to KA step, given a sub tweak Tw_i and the current *state* S_i the tweak addition is a symbol wise addition modulo p^b.

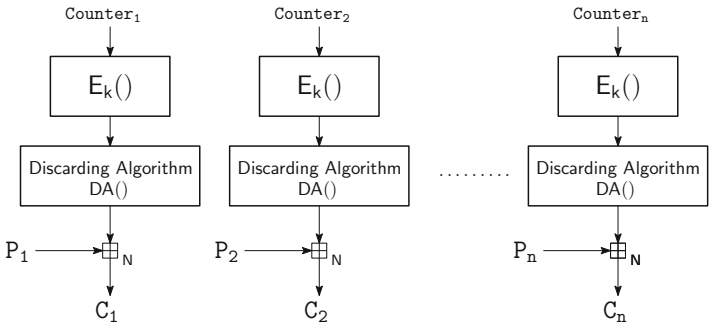

Fig. 1. Encryption of eSPF.

Algorithm 3. Enc $\mathtt{SPF}_r^N(K, M, T, Tw)$

input : Key K, Message M, Counter T, Tweak Tw
output : Ciphertext C

Initialize two NULL strings Q, Q';
Initialize $\ell \leftarrow |M|_N$;
$state \leftarrow \mathsf{STATE}(T)$;
KA($state$,K);
TA($state$,Tw);
for $j \leftarrow 1$ **to** $r-1$ **do**
  SubBytes($state$);
  ShiftRow($state$);
  MixColumns($state$);
  KA($state$,K_j);
  TA($state$,Tw);
SubBytes($state$);
ShiftRow($state$);
KA($state$,K_j);
TA($state$,Tw);
string $Q' \leftarrow$STRING($state$);
$Q \leftarrow$DA(Q');
for $i \leftarrow 0$ **to** $(\ell - 1)$ **do**
  $C[i] \leftarrow (M[i] \boxplus_N Q[i])$;
return C;

Algorithm 4. DA(S)

input : String S
output : String S'

Initialize a string S'=NULL;
For $i \leftarrow 1$ **to** n
  if $S[i] \in \Sigma$
    $S' = S'||S[i]$;
  else
    S' ;
return S';

The Operating Mode of eSPF. We adopt the Counter Mode [19] of operation for eSPF so that we can handle arbitrary length messages. For a large message block, the message will be divided into sub-blocks and the eSPF routine is invoked internally to generate the corresponding output block for a sequence of counters. The ciphertext will be concatenation of all the output blocks (Fig. 1).

The main advantages of counter-mode are parallel encryption/decryption and no requirement of padding, i.e., no length extension. However, *malleability* is the major limitation of this mode. This constraint is applicable to other block cipher modes like CBC, OFB etc. as well [31]. This limitation can be handled by using an additional message authentication protocol in our eSPF scheme, the design and analysis of which is currently beyond the scope of this work.

Algorithm 3 shows the encryption process of eSPF construction. The only difference between encryption and decryption will be the use of modular subtraction in place of modular addition.

3.2 Discarding Algorithm DA()

Let $\Sigma = \{0, 1, 2, \ldots, N-1\}$ be the alphabet set of format size N. Let $\Sigma' = \{0, 1, 2, \ldots, N'-1\}$, where Σ' is the alphabet set containing all the elements

of $\mathbb{F}_{p^b}$ and $N' > N$. Since, each *state* of E_k contains n-symbols, the output of E_k is a string of n symbols. Let, the string Q' be the output of E_k, i.e., $Q' = q'_0 q'_1 q'_2 \dots q'_{n-1}$ such that $q'_i \in \Sigma'$, for $0 \leq i \leq n-1$. The output of *discarding algorithm* $\mathsf{DA}()$ will be a string Q, i.e., $Q = \mathsf{DA}(Q) = q_0 q_1 q_2 \dots q_{n'-1}$ such that $q_i \in \Sigma$ and $n' \leq n$.

We are interested in finding the probability of occurrence of an arbitrary integer $a \in \Sigma$ at the k-th trial after the occurrence of elements of $(\Sigma' - \Sigma)$ in the first $k-1$ trials.

Let X be the success event defined for the occurrence of a specific symbol $a \in \Sigma$ and the corresponding probability be $p_a = \frac{1}{N'}$. Let Z be a failure event defined as the occurrence of a symbol of set $(\Sigma' - \Sigma)$ and the corresponding probability p is $\frac{N'-N}{N'}$.

For $a \in \Sigma$, $\mathsf{Pr}[X = a] = p^{k-1} p_a$.

Thus, the total probability S of getting a can be estimated as:

$$S = \frac{1}{N'} + \frac{N'-N}{N'} \cdot \frac{1}{N'} + \frac{N'-N}{N'} \cdot \frac{N'-N}{N'} \cdot \frac{1}{N'} + \cdots \tag{1}$$

After multiplication with $\frac{N'-N}{N'}$, Eq. 1 can be rewritten as

$$\frac{N'-N}{N'} \cdot S = \frac{N'-N}{N'} \cdot \frac{1}{N'} + \frac{N'-N}{N'} \cdot \frac{N'-N}{N'} \cdot \frac{1}{N'} + \cdots \tag{2}$$

From Eqs. 1 and 2, $S = \frac{1}{N}$.

Without loss of generality, this can be shown for all elements of the set Σ and thus it can be concluded that the *discarding algorithm* does not impact the distribution of occurrence of symbols of Σ, i.e., it will not leak any additional information.

Discarding Rate: By discarding, we mean ignoring a symbol if it does not belong to alphabet set Σ. Let, p_r be the probability of not discarding an output symbol of E_k, i.e., $p_r = \frac{N}{N'}$, then the probability of discarding a symbol is $(1-p_r)$.

Let, Z be a random variable with parameter n and p_r, where n denotes the number of independent trials and p_r is the success probability of the experiment. Then the random variable Z follows binomial distribution. Hence, the probability that Z contains a_0 symbol is $Pr(Z = a_0) = \binom{n}{a_0} {p_r}^{a_0} (1-p_r)^{n-a_0}$ and expected value of a_0 is $n \times p_r$.

To minimize the discarding rate, the field $\mathsf{GF}(p^b)$ and the corresponding set Σ' should be chosen carefully. The small difference between $(N'-N)$ will ensure a higher value for p_r. For example, if $N = 10$, $\mathbb{F}_{11}$ $(p = 11, b = 1, N' = 11)$ is the most suitable option. In this case the average value of a_0 is 14.55, i.e., on an average less than 1.45 symbols out of 16 symbols will be discarded by the discarding algorithm.

4 eSPF for Digits

In this section, we present $\mathtt{eSPF}_{10}^{10}$, which is a concrete instantiation of our construction for digits. We choose, $\mathbb{F}_{11}$, i.e., $\mathsf{GF}(11)$ for our construction. Thus, all the arithmetic operations are done modulo 11.

4.1 The S-Box

The S-box for $\mathbb{F}_{11}$ is shown in Table 1. To choose the S-box mapping, we analyzed all the possible mappings under different criteria such as maximal difference and linear probabilities and hardware implementation. Based on these, an optimal implementation of our S-box with logic gates is as follows:

$$y_0 = \{x_2\bar{x_0} + x_3\}\ y_1 = \{\bar{x_1}\bar{x_2}x_3 + \bar{x_0}\bar{x_1}\}$$
$$y_2 = \{x_0x_1 + \bar{x_1}x_3\}\ y_3 = \{\bar{x_0}\bar{x_1}\bar{x_3} + x_0\bar{x_1}x_2\}$$

where, our S-box can be represented as $y_n = \mathrm{S}[x_n]$. The maximum differential probability and the maximum correlation for this S-box are $2^{-2.45}$ and $2^{-1.45}$ respectively.

Table 1. Representation of S-box for $\mathbb{F}_{11}$.

x	0	1	2	3	4	5	6	7	8	9	10
$S[x]$	2	0	10	6	3	8	9	4	7	5	1

4.2 The ShiftRows

The ShiftRows operation in this construction works exactly like `AES`.

4.3 The Permutation

In [22], Gupta et al. analyzed the format preserving diffusion layers for digits and showed that it is impossible to construct any cryptographically significant 4×4 matrices over the field $\mathbb{F}_{2^4}$ which yields a format preserving set of cardinality 10. Further, for an arbitrary format, non-existence of MDS matrix under some reasonable restrictions has been shown in [12]. Since, our motivation was to use a MDS matrix for optimal efficiency, based on the findings of [12,22], we decided to choose the diffusion layer such that it may violate the format size. The linear diffusion layer for our case is realized by the following 4×4 MDS matrix over GF(11).

$$M = \begin{pmatrix} 1 & 1 & 2 & 5 \\ 5 & 1 & 1 & 2 \\ 2 & 5 & 1 & 1 \\ 1 & 2 & 5 & 1 \end{pmatrix}$$

The branch number of this matrix is 5.

4.4 Key Addition

The key addition transformation is symbol wise modular addition for a state S_i and subkey K_i.

4.5 Tweak Addition

Inclusion of tweak for eSPF is motivated by the domain size of real world applications of FPE algorithms and birthday bound security of the associated block ciphers. Tweak is public and it is used to randomize the instance of the block cipher, i.e., different values of tweak correspond to different families of permutations. Its usage helps in case of FPE algorithms since now the same ciphertext (e.g., if the two credit card numbers will provide the same plaintext for encryption, say middle 6 digits) will look different to the attacker due to different values of the tweak and hence would be indistinguishable. The proposed construction works in counter mode and use of different counter value ensures variability over ciphertext. However, since in the real world applications of FPE algorithms, the length of messages is mostly short (single block messages), same counter value may be used to encrypt different messages. Further, as the domain sizes of various formats are also small, enough variability may not be achieved in some cases. To circumvent this issue, we are using a tweak in our design.

Initiated by the work of Liskov et al. [26], few tweakable block cipher designs have been proposed in literature. In [24], Jean et al. presented the generic TWEAKEY framework that can be used to convert any key alternating block cipher into a tweakable one and proposed three instantiations - Deoxys-BC, JoltiK-BC and KIASU-BC that were the first ad-hoc tweakable block ciphers based on AES.

Injection of tweak in eSPF construction follows the tweak injection method adopted in KIASU-BC [24], i.e., the the tweak will be added to the first two rows of the state. Considering the block size of $\texttt{eSPF}_{10}^{10}$ ($\approx 2^{56}$ and the security, we choose a 60-bit tweak Tw). Two subtweaks $Tw0$ and $Tw1$ will be generated by a Tw using (Algorithm 6). $Tw0$ and $Tw1$ will then be added to the first two rows of the state for each even and odd numbered rounds correspondingly.

4.6 Key Schedule

We propose a new scheduling algorithm (KSA) for $\texttt{eSPF}_{10}^{10}$. The key schedule algorithm takes the 128-bit key K as input and generates $(r+1)$ round subkeys as output. Let K be represented as $k_{127}k_{126}\ldots k_2k_1k_0$. We first divide the K into two bit string of equal size and find $K_0 = \mathsf{STATE}(K \mod 11^{16})$. We iterate Step 5 to Step 9 of the Algorithm 5 to extract the remaining r subkeys. Addition of round constant i provides security against slide attack and the addition operation is chosen to introduce non-linearity. The shift operation ensures that all the bits of K will be used up to round 5. In [3] Bellare *et al.* estimated the lower bound of statistical distance between the uniform distribution on Z_p and the distribution obtained by $b \mod p$ after picking b randomly in Z_a as p/a where $a > p$. We estimate 2^{-72} ($a = 2^{128}$, $p = 11^{16} \approx 2^{56}$) as the statistical distance for digits. This bound suggests that the mod 11^{16} operation does not impact the distributions dramatically.

Algorithm 5. $\mathsf{KSA}(K)$

input : Key K
output: Round Keys $K_0, K_1, \ldots, K_r$

$x_1 \leftarrow k_{127}k_{126}\ldots k_{65}k_{64}$;
$y_1 \leftarrow k_{63}k_{62}\ldots k_1k_0$;
$K_0 \leftarrow \mathsf{STATE}(K \mod 11^{16})$
for $i \leftarrow 1$ **to** r **do**
  $y_i \leftarrow ((y_i \ll 16) + x_i) \oplus i$;
  $x_i \leftarrow (x_i \gg 33) \oplus y_i$;
  $K_i \leftarrow \mathsf{STATE}((x_i||y_i) \mod 11^{16})$;
  $x_{i+1} \leftarrow x_i$;
  $y_{i+1} \leftarrow y_i$;
return $(K_0, K_1, \ldots, K_r)$;

Algorithm 6. $\mathsf{TSA}(Tw)$

input : Tweak Tw
output: Round tweaks $Tw0, Tw1$

$Tw_0 \leftarrow \mathsf{STATE}(Tw \mod 11^8)$;
$Tw \leftarrow (Tw \ll 30)$;
$Tw_1 \leftarrow \mathsf{STATE}(Tw \mod 11^8)$;
return $(Tw0, Tw1)$;

5 Security Analysis

In this section, we evaluate the security of $\mathtt{eSPF}_{10}^{10}$ construction against various standard attacks.

5.1 Differential and Linear Cryptanalysis

Differential [7] and linear cryptanalysis [27] are two of the most powerful techniques to analyze symmetric-key primitives. To resist the differential and linear attacks, we choose to design our transformations according to the wide trail design strategy [16] and estimate the lower bounds for active S-boxes for different rounds of $\mathtt{eSPF}_{10}^{10}$.

Number of Active S-boxes: The diffusion layer of eSPF uses a 4×4 MDS matrix with branch number 5. Hence, any two round differential/linear characteristic has a minimum of 5 active S-boxes and any four round differential/linear characteristic has a minimum of 25 active S-boxes for eSPF. In Table 2, we mention the number of rounds (r) and the corresponding minimum number of active S-boxes (A_r) for eSPF. In FSE 2006, Granboulan et al. [21] presented a general framework for differential and linear cryptanalysis of block cipher when the block is not a bitstring. A $M \times M$ matrix Δ simulates the behavior of the S-box S over differences by $\Delta(S)_{a,b} = \#\{x|S(x+a) - S(x) = b\}$. The maximum entry of the matrix, i.e., $\mathsf{D}(S)$ is defined as:

$$\mathsf{D}(S) = \max_{(a,b)\neq\{0,0\}} \Delta(S)_{a,b}.$$

The corresponding maximum propagation probability is defined as differential probability, $\mathsf{DP}(S) = \mathsf{D}(S)/M$. The $\mathsf{D}(S)$ is equal to 2 for $\mathtt{eSPF}_{10}^{10}$ and the corresponding maximum $\mathsf{DP}(S)$ is equal to $2^{-2.45}$ ($\frac{2}{11} \approx 2^{-2.45}$).

Table 2. Minimum number of active S-boxes A_r for r rounds of eSPF.

r	1	2	3	4	5	6	8	10	12	16
A_r	1	5	6	25	26	30	50	55	75	100

In order to investigate the security against linear cryptanalysis of the S-box, firstly we calculate the distribution vector $\Lambda_0(S)_{\{a,b\}} = (\#\{x \in \mathbb{F}_{p^b} | \langle a, b|x, S(x)\rangle = u\})_{u\in\{Z\}}$, where $\langle a, b|x, y\rangle = \langle a|x\rangle - \langle b|y\rangle$ and $\langle a, x\rangle$ is scalar product of a and x. The distribution vector represents the behavior of the considered S-box. The random behavior can be defined as: $f_{a,b;u} = \frac{1}{M}\#(x, y) \in \mathbb{F}_{p^b} \times \mathbb{F}_{p^b} | \langle a, b|x, y\rangle = u$. The bias of the S-box represents the difference of behavior of S-box S and the random case and is defined as $\Lambda_S(S)_{a,b;u} = \Lambda_0(S)_{a,b;u} - f_{a,b;u}$. The highest bias measures the non linearity of the S-box. For eSPF_{10}^{10}, the maximal bias is equal to $\frac{2}{11} = 2^{-2.45}$ and the maximum correlation is $2^{-1.45}$.

Based on the above parameters, the probability of any single 6-round differential characteristic of eSPF_{10}^{10} is upper bounded by 2^{-73} and the maximum correlation of a 6-round linear trial is 2^{-43}. These bounds ensure that the data requirement to mount these attacks will exceed the available data $2^{55}(\approx 11^{16})$ for 6-rounds.

5.2 Square Attack

In this section, we describe a 7-round square attack [14] against eSPF. This attack is motivated by the attack shown in [17]. For our 7-round attack, we first construct a 4-round distinguisher. Consider a Λ-set of 11 plaintexts in which the first symbol takes all possible 11 values (active symbol) and the remaining symbols take any constant value that remains same throughout the set. Since, our construction involves tweak addition, in this attack, let us suppose that the attacker uses Λ-sets for the two subtweaks as well, i.e., one symbol of both the subtweaks (position being the same as that of the active symbol in the plaintexts) are active. Considering these, Fig. 2 shows the four round transformations of eSPF construction.

Let x_j, y_j, z_j, w_j denote the symbol values in round j after SubBytes, ShiftRows, MixColumns and KeyAddition and stage respectively. Let A[p] denote the p^{th} symbol (column wise) in any intermediate state A where, $0 \leq p \leq 15$. Similarly, $A_j^i[p]$ denotes the p^{th} symbol of i^{th} state A in round j where, (where, $0 \leq i \leq 10$). In the pre-whitening stage, since Λ-sets of plaintexts and subtweaks are in control of the attacker, he chooses the plaintexts P^i and subtweaks TW_0^i (where, $0 \leq i \leq 10$) such that for each i the sum $(P^i + Tw_0^i)$ mod 11 is a constant. The state remains constant until S_1 where the first symbol becomes active again due to addition of the second sub-tweak Tw_1. In, round 2 consider state $S_2[0]$. Due to sub-tweak addition of $Tw_1[0]$, we have:

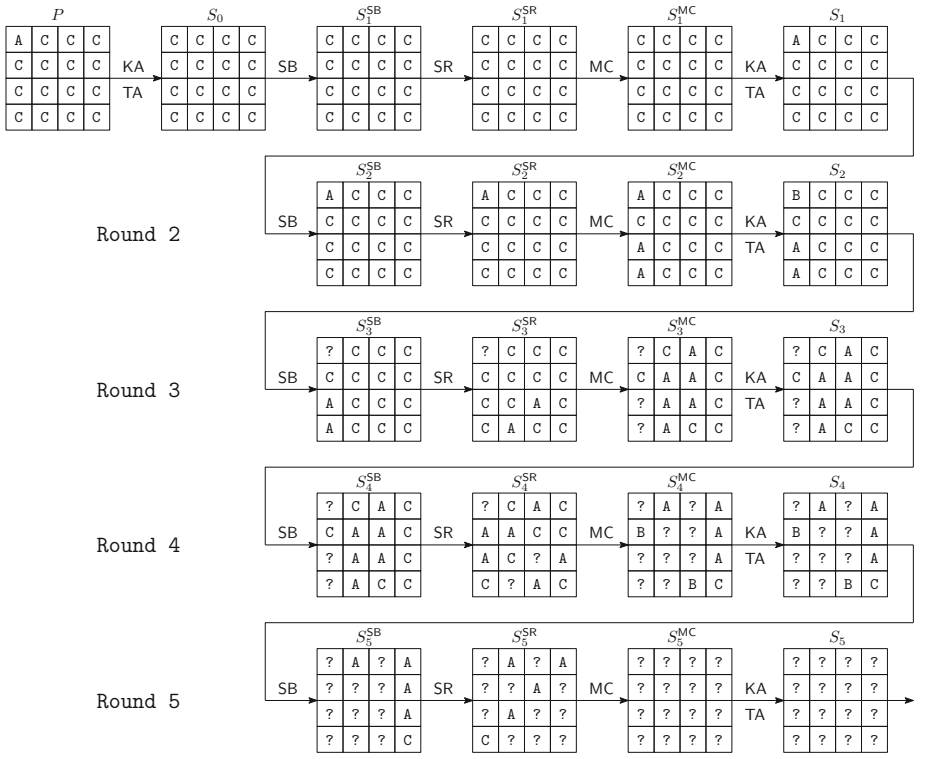

Fig. 2. A five round distinguisher for eSPF. Here A denotes an active symbol, B denotes that mod 10 sum of all values in that symbol is 0 and ? denotes unknown symbol.

$$w_2^i[0] = (z_2^i[0] + Tw_1^i[0]) \bmod 11$$

Since tweak symbol as well as the state symbol are active, if we add all the values in $w_2[0]$, then it can be shown that the sum mod 11 is always 0 as follows:

$$\begin{aligned} w_2^0[0] + w_2^1[0] + \ldots w_2^{10}[0] &= (\sum_{i=0}^{10} z_2^i + \sum_{i=0}^{10} Tw_1^i) \bmod 11 \\ &= (55 + 55) \bmod 11 = 0 \end{aligned}$$

This shows that the set of values in the first symbol position after second round tweak addition forms a *balanced* set with probability 1. After SubBytes operation in round 3, the balanced set property is destroyed. Similar explanation can be given till state transformation after ShiftRows in round 4. After MixColumns operation in round 4, we get a completely unknown state. However, at state S_4^{MC}, consider the first column. Here, we have:

$$\sum_{i=0}^{10} z_5^i[0] + \sum_{i=0}^{10} z_5^i[3] = (\sum_{i=0}^{10} y_5^i[0] + \sum_{i=0}^{10} y_5^i[1] + 2\sum_{i=0}^{10} y_5^i[2] + 5\sum_{i=0}^{10} y_5^i[3]) \bmod 11 +$$
$$(\sum_{i=0}^{10} y_5^i[0] + 2\sum_{i=0}^{10} y_5^i[1] + 5\sum_{i=0}^{10} y_5^i[2] + \sum_{i=0}^{10} y_5^i[3]) \bmod 11$$
$$= (2\sum_{i=0}^{10} y_5^i[0] + 3\sum_{i=0}^{10} y_5^i[1] + 7\sum_{i=0}^{10} y_5^i[2] + 6\sum_{i=0}^{9} y_5^i[3]) \bmod 11$$
$$= (2\sum_{i=0}^{10} y_5^i[0] + 0) mod 11 = \text{ Even number}$$

Again in the right hand side of the above equation, since $y_5[1, 2, 3]$ are active cells, their sum over all 11 states is always going to be zero as discussed above. Hence, the additive sum of $Z_5[0] + Z_5[3]$ over all 11 states will always be an even number with probability 1; which will be preserved even after tweak addition in round 4. In the random case, the output will be even with a probability of 6/11. Hence, a valid distinguisher is constructed. This four round attack can be extended up to seven rounds by adding one round in the backward and 2 rounds in the forward directions to recover the secret key.

5.3 Impossible Differential Cryptanalysis

Impossible Differential Cryptanalysis (IDC) [5] uses impossible differential characteristics to eliminate incorrect keys. Since the diffusion layer of `eSPF` construction is very similar to the `AES` algorithm, the basic 4 round impossible characteristics presented in [6] for `AES` algorithm and the proposed construction is same. The input and output characteristics for 4 rounds impossible characteristics is as follows:

$$(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \overset{4R}{\nrightarrow} (0,1,1,1,1,1,1,1,0,1,1,0,1,1,0,1,1)$$

More rounds can be appended before and after the 4 round impossible distinguisher. We found up to 6 rounds characteristics, which can be used for key-recovery attacks, but no such characteristics could be found when the number of round is greater than 6.

5.4 Key Related Attacks

Slide attacks [8] and related-key attacks [4] are the two most important types of key scheduling attacks. Our key scheduling algorithm for `eSPF` adds a round dependent counter in each round to prevent sliding of the subkeys. For related key attack to work, the attacker should be able to identify meaningful relationships between different subkeys so that a related key differential can be constructed over certain rounds. However, the non-linear addition operation and the modular function in our key scheduling algorithms do not allow an adversary to deduce

all the other round keys (and the master key) from one round key by working through the key schedule. The modular function in particular also makes it very hard for an attacker to control the difference propagation through different round keys. Moreover, we also analyzed that each bit of the secret key K is used by the fifth round for all format size 10 or more. Hence, we believe that these features of the proposed KSA are sufficient to resist related key attacks.

5.5 Related Tweak Attack

Launching a related tweak attack to recover the secret key is easier for an attacker compared to related key attack. This is because the tweak value is a public entity and can be chosen by the attacker himself. This allows him to insert differences in the tweak input of the block cipher and construct related tweak differentials. Thus, it is imperative to assess the security of our schemes in this stronger related tweak setting.

We developed an automated program to count the number of active S-boxes and return an upper bound on the probability of the best related tweak truncated differentials. Table 3 lists the number of active S-boxes for the first 8-rounds of eSPF. It can be seen that the probability of any characteristic on more than 6 rounds is not higher than $2^{-36\times 2.45} = 2^{-88.2}$ for eSPF. This bound ensures that the amount of data required to launch the attack will exceed the data available to an attacker (i.e., $11^{16} \approx 2^{53}$). Hence, our construction can resist any related tweak attack of practical complexity if the number of rounds is ≥ 6.

Table 3. Count of active S-box (A_r) and corresponding differential probability (P_r) over different rounds r of eSPF for related tweak differentials.

r	1	2	3	4	5	6	7	8
A_r	0	0	1	5	20	36	50	66
P_r	0	0	$2^{-2.45}$	$2^{-9.8}$	2^{-49}	$2^{-88.2}$	$2^{-122.5}$	$2^{-161.7}$

Considering the attacks discussed above as well as efficiency we recommend $r = 10$.

6 Performance

eSPF was designed with performance implementation costs in mind. In this section, we provide performance comparison of eSPF with FPE designs FFX and SPF.

6.1 Implementation

As eSPF is an AES-like block-cipher, the round-operations are best implemented using *table-lookups*. The reference implementation was done for 32-bit platform, but 64-bit processor support is ubiquitous and the implementation is also faster, so all the results are for the 64-bit implementation. Table 4 shows the implementation results.

Table 4. Execution speed in symbols/second and cycles/symbol for eSPF_{10}^{10}.

Processor	Clock speed	Speed for eSPF_{10}^{10}	
		Symbols/second	Cycles/symbol
Core i7 6700	3.4 GHz	201.2×10^6	16.8
Core i7 4770	3.4 GHz	168.1×10^6	20.2
Core i5 2400	3.1 GHz	44.8×10^6	70.5

The *lookup-table* based implementation of eSPF_{10}^{10} round function required 4 tables with 64 entries of 32-bit integers. The round functions and the MOD (remainder) operation was combined. Each column required 4 table-lookups, as a result there were 16 table-lookups in total. PDEP and PEXT instructions, were used for various bit-manipulation operations needed, significantly improving the performance. Owing to the similarity with SPF for digits (which consists of 14 rounds), the performance gains for eSPF_{10}^{10} would primarily be the result of reduced number of rounds, however the *discarding algorithm* would cause some performance degradation. The expected speedup can be estimated to be $((14 \div 10) \times (10 \div 11))$, which is about 1.27.

6.2 Performance of eSPF_{10}^{10} Compared to FFX with Radix 10

Even for the smallest input sizes, FFX requires 11 invocations of AES-128 to encrypt messages containing about 52 symbols of radix 10. FFX needs 10 AES-128 invocations with a MOD operation which is quite expensive, and an extra AES-128. To test the expected performance, we used the inbuilt 128-bit integer support in gcc to perform MOD operations. On an Intel Core i7 4770 CPU at 3.4 GHz, the MOD operation was taking approximately 184 clock cycles at an average. We also tested an assembly version of a fast 128-bit MOD implementation for MSVC which took 296 clock cycles. On the same machine, it was found out that the AES-128 execution speed was 129 MiB/s by running openssl speed aes-128-cbc, which comes down to about 400 cycles per AES-128 invocation. As FFX uses 10 MOD operations and 11 AES-128 invocations, one FFX encrypt operation should take about $(184 \times 10) = 1840$ cycles for MOD and $(400 \times 11) = 4400$ cycles for AES, so in total it takes about 6240 cycles at least. So, FFX should run at about 120 cycles/symbol $(6240 \div 52)$ at max; this ignores

any other performance loss due to copies, and other operations like $\mathsf{NUM}_{radix}()$, $\mathsf{STR}_{radix}()$ etc. So, eSPF is about **6 times faster** than FFX for similar parameters. Considering traditional applications of FPE such as encryption of SSN and CCN, eSPF would be around **30 and 15 times faster** than FFX respectively.

According to the definition of FFX, a large MOD (of size approximately $radix^{N/2}$, where N is the size of input string) operation is needed. As the MOD can get very large, efficient implementation would need to use *big-integer* libraries, which tend to be significantly slower (can be a few orders of magnitude, depending on parameters) than the AES-128, used. As a result the overall implementation can get very slow. This is not a problem in eSPF.

7 Conclusion

In this work, we present a new efficient format-preserving encryption construction based on substitution-permutation networks. We present a concrete instance of the proposed construction for format size 10. Further, to analyze the security of the presented design, we consider conventional cryptanalytic techniques as well as dedicated attacks. Finally, we compare the efficiency of the presented construction with existing schemes. The construction is approximately ten times faster than existing popular designs such as FFX and BPS for most of practical uses of FPE. A similar construction for other popular format size is an interesting open problem.

References

1. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: *Camellia*: a 128-bit block cipher suitable for multiple platforms — design and analysis. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 39–56. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44983-3_4
2. Bellare, M., Hoang, V.T., Tessaro, S.: Message-recovery attacks on Feistel-based format preserving encryption. Cryptology ePrint Archive, Report 2016/794 (2016). http://eprint.iacr.org/2016/794
3. Bellare, M., Ristenpart, T., Rogaway, P., Stegers, T.: Format-preserving encryption. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 295–312. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05445-7_19
4. Biham, E.: New types of cryptanalytic attacks using related keys. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 398–409. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_34
5. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_2
6. Biham, E., Keller, N.: Cryptanalysis of reduced variants of Rijndael (1999, unpublished manuscript)

7. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-38424-3_1
8. Biryukov, A., Wagner, D.: Advanced slide attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 589–606. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_41
9. Black, J., Rogaway, P.: Ciphers with arbitrary finite domains. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 114–130. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45760-7_9
10. Brier, E., Peyrin, T., Stern, J.: BPS: a format-preserving encryption proposal. NIST. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf
11. Brightwell, M., Smith, H.: Using datatype-preserving encryption to enhance data warehouse security, vol. PP, pp. 141–149 (1997). http://csrc.nist.gov/niccs/1997
12. Chang, D., Ghosh, M., Gupta, K.C., Jati, A., Kumar, A., Moon, D., Ray, I.G., Sanadhya, S.K.: SPF: a new family of efficient format-preserving encryption algorithms. In: Chen, K., Lin, D., Yung, M. (eds.) Inscrypt 2016. LNCS, vol. 10143, pp. 64–83. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54705-3_5
13. Coppersmith, D., Holloway, C., Matyas, S.M., Zunic, N.: The data encryption standard. Inf. Secur. Tech. Rep. **2**(2), 22–24 (1997)
14. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052343
15. Daemen, J., Rijmen, V.: The block cipher Rijndael. In: Quisquater, J.-J., Schneier, B. (eds.) CARDIS 1998. LNCS, vol. 1820, pp. 277–284. Springer, Heidelberg (2000). https://doi.org/10.1007/10721064_26
16. Daemen, J., Rijmen, V.: The wide trail design strategy. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 222–238. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45325-3_20
17. Dobraunig, C., Eichlseder, M., Mendel, F.: Square attack on 7-round Kiasu-BC. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 500–517. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_27
18. Betül Durak, F., Vaudenay, S.: Breaking the FF3 format-preserving encryption standard over small domains. Cryptology ePrint Archive, Report 2017/521 (2017). http://eprint.iacr.org/2017/521
19. Dworkin, M.: NIST Special Publication 800–38A: Recommendation for Block Cipher Modes of Operation-Methods and Techniques, December 2001
20. Dworkin, M.: Recommendation for block cipher modes of operation: methods for format-preserving encryption. NIST Special Publication, 800:38G
21. Granboulan, L., Levieil, É., Piret, G.: Pseudorandom permutation families over Abelian groups. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 57–77. Springer, Heidelberg (2006). https://doi.org/10.1007/11799313_5
22. Gupta, K.C., Pandey, S.K., Ray, I.G.: Format preserving sets: on diffusion layers of format preserving encryption schemes. In: Dunkelman, O., Sanadhya, S.K. (eds.) INDOCRYPT 2016. LNCS, vol. 10095, pp. 411–428. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49890-4_23
23. Hoang, V.T., Morris, B., Rogaway, P.: An enciphering scheme based on a card shuffle. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 1–13. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_1

24. Jean, J., Nikolić, I., Peyrin, T.: Tweaks and keys for block ciphers: the TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_15
25. Lee, J.-K., Koo, B., Roh, D., Kim, W.-H., Kwon, D.: Format-preserving encryption algorithms using families of tweakable blockciphers. In: Lee, J., Kim, J. (eds.) ICISC 2014. LNCS, vol. 8949, pp. 132–159. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15943-0_9
26. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_3
27. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_33
28. Morris, B., Rogaway, P.: Sometimes-recurse shuffle. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 311–326. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_18
29. Morris, B., Rogaway, P., Stegers, T.: How to encipher messages on a small domain. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 286–302. Springer, Heidelberg (2009)
30. Ristenpart, T., Yilek, S.: The mix-and-cut shuffle: small-domain encryption secure against N queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 392–409. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_22
31. Rogaway, P.: Evaluation of Some Blockcipher Modes of Operation. http://www.cryptrec.go.jp/estimation/techrep_id2012_2.pdf
32. Schroeppel, R., Orman, H.: The hasty pudding cipher. AES candidate submitted to NIST, p. M1 (1998)
33. Sheets, J., Wagner, K.R.: Visa Format Preserving Encryption (VFPE). NIST Submission (2011)
34. Spies, T.: Feistel finite set encryption. NIST Submission, February 2008. http://csrc.nist.gov/groups/ST/toolkit/BCM/modes-development.html