

Towards Lightweight Error Detection Schemes for Implementations of MixColumns in Lightweight Cryptography

Anita Aghaie, Mehran Mozaffari Kermani, Reza Azarderakhsh

Abstract—In this paper, through considering lightweight cryptography, we present a comparative realization of MDS matrices used in the VLSI implementations of lightweight cryptography. We verify the MixColumn/MixNibble transformation using MDS matrices and propose reliability approaches for thwarting natural and malicious faults. We note that one other contribution of this work is to consider not only linear error detecting codes but also recomputation mechanisms as well as fault space transformation (FST) adoption for lightweight cryptographic algorithms. Our intention in this paper is to propose reliability and error detection mechanisms (through linear codes, recomputations, and FST adopted for lightweight cryptography) to consider the error detection schemes in designing *beforehand* taking into account such algorithmic security. We also posit that the MDS matrices applied in the MixColumn (or MixNibble) transformation of ciphers to protect ciphers against linear and differential attacks should be incorporated in the cipher design in order to reduce the overhead of the applied error detection schemes. Finally, we present a comparative implementation framework on ASIC to benchmark the VLSI hardware implementation presented in this paper.

I. INTRODUCTION

Research on error detection of primitives in the hardware VLSI structures of cryptographic algorithms has been center of attention in prior work [1]-[11]. In addition, cipher designers construct the MixColumn transformation by a linear diffusion layer with maximum branch number, known as maximal distance separable (MDS) matrices. We also mention that the MDS matrices applied in the MixColumn (or MixNibble) transformation of ciphers to protect ciphers against linear and differential attacks should be incorporated in the cipher design in order to reduce the overhead of the applied error detection schemes.

To motivate the urgency of embedding error detection as part of the design cycle, we briefly go over the complications of adopting fault FST method for lightweight ciphers (it is a motivation to our proposed criteria in the following sections in terms of utilizing MDS matrices as the mapping function). This method which suggests a generic FST mapping for

data storage during encryption/decryption operations of AES-like ciphers increases the security of expensive redundancy countermeasures against fault attacks [6]. FST is utilized to make “fault collision” difficult during attacks on the classic redundancy-based countermeasures, i.e., the adversary would have challenge in injecting same fault in the storage registers having both original and spatial/time redundancy structures. The countermeasures based on recomputations could fail to detect the occurrence of a fault as long as the adversary could inject the same fault in both the original and redundant computations (biased fault model makes it easier). The countermeasures based on recomputations can be used in conjunction with encoding schemes which nullify the effect of the bias in the fault model by FST, thwarting both these attack schemes.

To investigate the importance of using/reusing MDS matrices in lightweight block ciphers, we have applied this method to the KLEIN cipher. Implementing the “naive” spatial redundancy of KLEIN, we need 232 occupied slices on Virtex-7 (xc7vx330t) with high area overhead. Moreover, we have implemented the spatial redundancy with FST method that applies MixNibble as a mapping function W through this redundancy and using InvMixNibble as its inverse W^{-1} . We note that although KLEIN allows two types of decryption through (a) using encryption transformations but utilizing modes of operations and (b) reverse transformations, this might not be the case for other lightweight block ciphers and that adds to the complications of using FST in lightweight cryptography. Our implementations show that higher area, i.e., 239 occupied slices, is achieved, as expected, due to the W function. Applying the pipeline method, which utilizes MixNibble operation as W function, improves the spatial redundancy algorithm metrics with occupying just 235 slices on Virtex-7.

Our intention in this paper is to propose reliability and error detection mechanisms (through linear codes, recomputations, and FST adopted for lightweight cryptography) to consider the error detection schemes in designing *beforehand* taking into account such algorithmic security. The MDS matrices applied in the MixColumn (or MixNibble) transformation of ciphers to protect ciphers against linear and differential attacks should be incorporated in the cipher design in order to reduce the overhead of the applied error detection schemes.

Anita Aghaie is with Embedded Security Group, Horst Gortz Institute for IT-Security, Ruhr-Universitaet Bochum, 44780 Bochum, Germany (email: anita.ghaie@rub.de).

M. Mozaffari Kermani is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620, USA (e-mail: mehran2@usf.edu).

R. Azarderakhsh is with the Department of Computer and Electrical Engineering and Computer Science and is an I-SENSE Fellow, Florida Atlantic University, Boca Raton, FL, USA (e-mail: razarderakhsh@fau.edu).

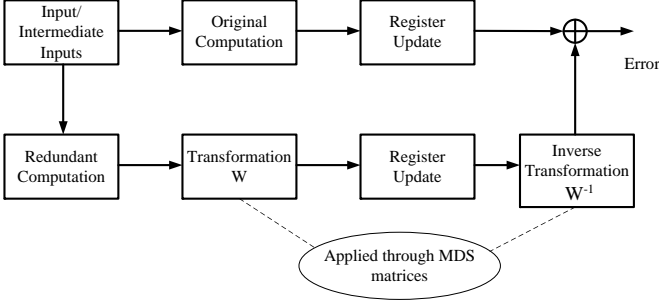


Fig. 1. The FST approach with MDS matrices mapping function.

Algorithm 1 $m \times m$ MDS matrix design criteria.

Inputs: $A : F_2^m \rightarrow F_2^m$; $X = (x_1, \dots, x_n) \in (F_2^m)^n$,

Outputs: Criteria of MDS matrix.

1. Define a linear diffusion $L(X) = (\overset{[i=1]}{1}n \sum L_{1,i}(x_i), \dots, \overset{[i=1]}{1}n \sum L_{n,i}(x_i))$, for $1 \leq i, j \leq n$, and if $L \circ L = X$, then it is involutory.

2. Define the bundle weight of X , $\omega_b(x) = |\{x_i : x_i \neq 0, 1 \leq i \leq n\}|$,

3. Define the branch number of L : $N = \min\{\omega_b(X) + \omega_b(L(x)) | X \in (F_2^m)^n, X \neq 0\}$, and if $N = n + 1$, then it is an MDS matrix.

II. ERROR DETECTION OF VLSI ARCHITECTURES FOR MIXCOLUMN

MixColumn has a significant role to perform as the linear diffusion layer in the encryption and decryption operations over the finite fields. Although, there is a wide range of categories such as circulant, Hadamard, Cauchy, and Hadamard-Cauchy for the MDS matrices to apply in MixColumn, choosing an efficient MDS matrix should be carefully considered in terms of low-cost hardware area, high diffusion speed, and low-latency implementation. One of the common methods to construct lightweight MDS matrices, e.g., circulant, is sparing and compacting in implementation, and then composing it several times in which it provides similar rows in matrices to reduce the hardware implementation cost (number of XOR gates, for instance) like Photon hash functions [12].

The design criteria of MDS matrices, e.g., based on a low Hamming weight polynomial, generate a wide pool of involutory and non-involutory MDS matrices. Moreover, the security of these MDS matrices should be considered carefully during the design phase to improve the security levels.

All types of MDS matrices offer optimal linear diffusion to provide the proper linear part, the MixColumn operation in block ciphers and hash functions, but in general, a compact description for this matrix on which one is better may not be very achievable. The criteria in [13] potentially lead to low number of gates in hardware implementations and small amount of memory usage. The $m \times m$ MDS matrix design criteria are presented in Algorithm 1.

For each of the MDS matrices, we present the multiplication and reduction operations with irreducible polynomials to count the number of XOR gates. The number of XOR gates for

a number of lightweight block ciphers is presented in Table I which also shows the overhead percentages (this table is presented at the end of this section).

First cipher, Midori64, can utilize three 4×4 MDS matrices for the MixColumn transformation. We investigate two of them, i.e., the non-involutive MDS matrix (M_B) and the involutive almost MDS matrix (M_C). The former one is the same as the MDS matrix applied in KLEIN (will be shown in more details), and the latter form, M_C , is shown below. Let us denote the input state of MixColumn as A and the output state as R . Then, we have the following:

$$R = M_C \times A \implies \begin{pmatrix} r_0 & r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 & r_7 \\ r_8 & r_9 & r_{10} & r_{11} \\ r_{12} & r_{13} & r_{14} & r_{15} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix}. \quad (1)$$

According to above, we have modulo-2 added three input elements of each column to generate each element of the output matrix (R), in which each output column needs eight XOR gates. Because of the fact that the coefficients of the input state matrix are 0 or 1, we have the number of XOR gates as eight and twelve XOR gates for cumulative column signature and interleaved cumulative column signature, respectively, as shown below.

$$\begin{aligned} r_0 &= a_4 + a_8 + a_{12}, \\ r_4 &= a_0 + a_8 + a_{12}, \\ r_8 &= a_0 + a_4 + a_{12}, \\ r_{12} &= a_0 + a_4 + a_8. \end{aligned} \quad (2)$$

Let us modulo-2 add the first column of the state output matrix to derive the cumulative column signature-based scheme.

$$\begin{aligned} r_0 + r_4 + r_8 + r_{12} &= (0 + 1 + 1 + 1)a_0 + (1 + 0 + 1 + 1)a_4 \\ &\quad + (1 + 1 + 0 + 1)a_8 + (1 + 1 + 1 + 0)a_{12}. \end{aligned} \quad (3)$$

For interleaved cumulative column signature, let us modulo-2 add two even-row elements of the output state, i.e., rows 0 and 2, and two odd-row elements, i.e., rows 1 and 3:

$$\begin{aligned} r_0 + r_8 &= (0 + 1)a_0 + (1 + 1)a_4 + (1 + 0)a_8 \\ &\quad + (1 + 1)a_{12} = a_0 + a_8, \end{aligned} \quad (4)$$

$$\begin{aligned} r_4 + r_{12} &= (1 + 1)a_0 + (0 + 1)a_4 + (1 + 1)a_8 \\ &\quad + (1 + 0)a_{12} = a_4 + a_{12}. \end{aligned} \quad (5)$$

As shown in Table I, we need 4×8 XOR gates in Midori64 with M_C , in which the total number of cumulative column signature gates is 4×3 , and the required XOR gates for interleaved cumulative column signature is 4×2 . Due to the fact that these XOR gates are used in all of the MixColumn transformations similarly, we do not count them in the table.

The last cipher to present the details for the sake of brevity, LED, applies a hardware-friendly MDS matrix for the MixColumn transformation, that is given by:

$$M = \begin{bmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{bmatrix}. \quad (6)$$

Each entity in the input and the output state matrices is a four-bit nibble. As a case study, to compute r_0 (the first element of the resultant matrix R), denoting the bits of the elements of A as a_{ij} for i th bit of j th element, we have:

$$r_0 = 4.a_0 + a_4 + 2.a_8 + 2.a_{12} = x^2.a_0 + a_4 + x.a_8 + x.a_{12}, \quad (7)$$

$$\begin{aligned} r_0 = x^2.[a_{10}x^3 + a_{20}x^2 + a_{30}x + a_{40}] + [a_{14}x^3 + a_{24}x^2 + a_{34}x \\ + a_{44}] + x.[a_{18}x^3 + a_{28}x^2 + a_{38}x + a_{48}] + x.[a_{1c}x^3 \\ + a_{2c}x^2 + a_{3c}x + a_{4c}]. \end{aligned} \quad (8)$$

Using the irreducible polynomial $X^4 + X + 1$ utilized for reductions, one can derive the following for r_0 :

$$\begin{aligned} r_0 = x^3.[a_{2c} + a_{28} + a_{14} + a_{30}] + x^2.[a_{10} + a_{40} + a_{24} + a_{38} \\ + a_{3c}] + x.[a_{10} + a_{20} + a_{34} + a_{18} + a_{48} \\ + a_{1c} + a_{4c}] + 1.[a_{20} + a_{44} + a_{1c} + a_{18}]. \end{aligned} \quad (9)$$

Let us derive the formulae for just one column signature of MixColumn by modulo-2 adding the first column entries r_0, r_4, r_8 , and r_{12} . One can derive:

$$\begin{aligned} r_4 = 8.a_0 + 6.a_4 + 5.a_8 + 6.a_{12} = x^3.a_0 + \\ (x^2 + x).a_4 + (x^2 + 1).a_8 + (x^2 + x).a_{12}, \end{aligned} \quad (10)$$

$$\begin{aligned} r_8 = B.a_0 + E.a_4 + A.a_8 + 9.a_{12} = (x^3 + x + 1).a_0 \\ + (x^3 + x^2 + x).a_4 + (x^3 + x).a_8 + (x^3 + 1).a_{12}, \end{aligned} \quad (11)$$

$$\begin{aligned} r_{12} = 2.a_0 + 2.a_4 + F.a_8 + B.a_{12} = \\ x.a_0 + x.a_4 + (x^3 + x^2 + x + 1).a_8 + (x^3 + x + 1).a_{12}. \end{aligned} \quad (12)$$

For the cumulative column signature-based scheme, we modulo-2 add the first column entries of matrix R to derive the following signature \hat{P} :

$$\begin{aligned} r_0 + r_4 + r_8 + r_{12} = (4 + 8 + B + 2).a_0 + (1 + 6 + E + 2).a_4 \\ + (2 + 5 + A + F).a_8 + (2 + 6 + 9 + B).a_{12} \end{aligned}$$

TABLE I
NUMBER OF GATES NEEDED FOR THE MIXCOLUMN TRANSFORMATION
AND DERIVING THE PREDICTED SIGNATURES IN DIFFERENT LIGHTWEIGHT
BLOCK CIPHERS

Block cipher	MixCol.	CCS	Inter. CCS
	XOR	XOR	XOR
Midori64 (M_C)	128	176 (37.50%)	160 (25%)
Midori64 (M_B)	256	304 (18.75%)	416 (62.50%)
LED	444	564 (27.02%)	672 (51.35%)
KLEIN (two-nibble)	256	304 (18.75%)	416 (62.50%)

$$= 5.a_0 + B.a_4 + 2.a_8 + 6.a_{12}. \quad (13)$$

After the reduction, one can derive the below formulae as the final form:

$$\begin{aligned} x^3[a_{10} + a_{28} + a_{44} + a_{24} + a_{3c} + a_{2c} + a_{30}] + x^2.[a_{10} + \\ a_{40} + a_{34} + a_{38} + a_{20} + a_{4c} + a_{3c} + a_{1c} + a_{14}] + x.[a_{10} + a_{20} + \\ a_{30} + a_{44} + a_{48} + a_{14} + a_{18} + a_{2c} + a_{24} + a_{4c}] + \\ 1.[a_{20} + a_{44} + a_{1c} + a_{18} + a_{40} + a_{34} + a_{14} + a_{2c}]. \end{aligned} \quad (14)$$

This can be generalized to other columns and thus we have the followings for the second to the fourth columns after the reductions:

$$r_1 + r_5 + r_9 + r_{13} = 5.a_1 + B.a_5 + 2.a_9 + 6.a_{13}, \quad (15)$$

$$r_2 + r_6 + r_{10} + r_{14} = 5.a_2 + B.a_6 + 2.a_{10} + 6.a_{14}, \quad (16)$$

$$r_3 + r_7 + r_{11} + r_{15} = 5.a_3 + B.a_7 + 2.a_{11} + 6.a_{15}. \quad (17)$$

In the following, the other signature-based scheme (interleaved cumulative column signature) is derived through modulo-2 adding the odd-row elements with each other and the even ones as well:

$$r_0 + r_8 = F.a_0 + F.a_4 + 8.a_8 + B.a_{12}, \quad (18)$$

$$r_4 + r_{12} = A.a_0 + 4.a_4 + A.a_8 + D.a_{12}. \quad (19)$$

According to these formulae, we are able to count the number of utilized XOR gates and the cumulative column signature and interleaved cumulative column signature overheads for LED which are presented in Table I. As mentioned above, by default, for each cipher, we need 12 and 8 XOR gates for cumulative column signature and interleaved cumulative column signature, respectively, in addition to what presented, which are omitted for the sake of brevity.

Finally, as mentioned before, we summarize the number of XOR gates for all the mentioned ciphers in the MixColumn transformation in Table I, in which the overhead percentages are presented. We have not used sub-expression sharing, similar to the S-boxes, for deriving the numbers in Table I; nonetheless, sub-expression sharing can be used to reduce the number of gates at the expense of possible high fan-outs (which might not be tolerated in some cases, requiring repeaters to resolve the problem). Table I and the MixColumn

TABLE II
ASIC TSMC 65-NM SYNTHESIS RESULTS FOR TWO SELECT MixCOLUMN
TRANSFORMATIONS AND THEIR ERROR DETECTION MECHANISMS
(CUMULATIVE COLUMN SIGNATURES: CCS)

Block Cipher	GE of Architectures		
	MixCol.	CCS	Inter. CCS
Midori, M_C	272	330 (21.3%)	317 (16.5%)
LED	575	746 (29.7%)	765 (33.0%)

transformations that we have not considered for the sake of brevity motivate the urgency of considering the overheads beforehand, perhaps as a design factor (we note that other error detection schemes can be considered and the ones provided here are just a subset).

Similar to the S-boxes, we present two metrics for analyzing the results in Table I. The first one is the overhead for our error detection schemes shown in Table I. As seen in this table, Midori64, which applies the M_C matrix, has the lowest-cost for the MixColumn implementation; nevertheless, the cumulative column signature-based scheme overhead of this matrix is more than other ones. Comparing the percent overheads in Table I shows how different they could be with respect to error detection using cumulative column signature (and other schemes for error detection). The second metric is the total number of gates (the original and the add-on detection), e.g., the number of applied logic gates of M_B is not as low as M_C .

Finally, through ASIC synthesis and for two select constructions in Table II, we present the areas for the MixColumn transformations of Midori (M_C) and LED. The benchmarking is done for the error detection architectures using TSMC 65-nm library and Synopsys Design Compiler. Similar to the S-boxes, in order to make the area results meaningful when switching technologies on ASIC, we have provided the NAND-gate equivalency (gate equivalents: GE). This is performed using the area of a NAND gate in the utilized TSMC 65-nm CMOS library which is $1.41 \mu\text{m}^2$. The results are shown in Table II, where the overheads are presented in parentheses (the contrast when comparing this table and Table I is because of the optimizations performed in Design Compiler, noting that we have not performed sub-expression sharing in Table I). The aforementioned metrics/overheads are some of the possible indications to give designers the required criteria to predict low-cost MixColumn implementations for lightweight ciphers.

III. CONCLUSIONS

In this paper, we evaluated the hardware complexities of the MixColumn transformations to propose a framework for design-for lightweight and effective countermeasures for intentional and natural faults in crypto-architectures and to present respective motivations. One can also base the criteria (depending on the objectives) on other performance and implementation metrics, e.g., delay (frequency, throughput, efficiency), power consumption, and energy. Although we chose the MixColumn transformation due to their importance, other less costly transformations can be considered. The results

of our VLSI implementations on ASIC platform shows the diversity of MixColumn in lightweight cryptography, calling for efficient approaches for error detection. Finally, one could consider a subset of fault attacks, differential fault intensity analysis (DFIA), see for instance, [14], [15], [16], which combines differential power analysis with fault injection principles to obtain biased fault models (multi-byte faults cannot be used practically for attacking time redundancy countermeasure implementations, and single-byte fault models are the only viable option for the attackers).

REFERENCES

- [1] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Reliable hardware architectures for the third-round SHA-3 finalist Grostl benchmarked on FPGA platform," in *Proc. DFT*, pp. 325-331, 2011.
- [2] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Fault detection structures of the S-boxes and the inverse S-boxes for the Advanced Encryption Standard," *J. Electronic Testing*, vol. 25, no. 4-5, pp. 225-245, 2009.
- [3] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A Lightweight High-Performance Fault Detection Scheme for the Advanced Encryption Standard Using Composite Fields," *IEEE Trans. Very Large Scale Integrated (VLSI) Systems*, vol. 19, no. 1, pp. 85-91, Jan. 2011.
- [4] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *J. Cryptographic Engineering*, vol. 5, no. 3, pp. 153-169, 2015.
- [5] M. Mozaffari Kermani, R. Azarderakhsh, and A. Aghaie, "Fault detection architectures for post-quantum cryptographic stateless hash-based secure signatures benchmarked on ASIC," *ACM Trans. Embedded Computing Syst.* (special issue on Embedded Device Forensics and Security: State of the Art Advances), vol. 16, no. 2, pp. 59:1-19, Dec. 2016.
- [6] S. Patranabis, A. Chakraborty, D. Mukhopadhyay, and P. P. Chakrabarti, "Fault space transformation: A generic approach to counter differential fault analysis and differential fault intensity analysis on AES-like block ciphers," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 5, May 2017.
- [7] M. Mozaffari Kermani, R. Azarderakhsh, C. Lee, and S. Bayat-Sarmadi, "Reliable concurrent error detection architectures for extended Euclidean-based division over $GF(2^m)$," *IEEE Trans. Very Large Scale Integrated (VLSI) Systems*, vol. 22, no. 5, pp. 995-1003, May 2014.
- [8] M. Mozaffari Kermani, R. Azarderakhsh, and A. Aghaie, "Reliable and error detection architectures of Pomaranch for false-alarm-sensitive cryptographic applications," *IEEE Trans. Very Large Scale Integrated (VLSI) Systems*, vol. 23, no. 12, pp. 2804-2812, Dec. 2015.
- [9] M. Mozaffari Kermani, K. Tian, R. Azarderakhsh, and S. Bayat-Sarmadi, "Fault-resilient lightweight cryptographic block ciphers for secure embedded systems," *IEEE Embedded Systems*, vol. 6, no. 4, pp. 89-92, Dec. 2014.
- [10] S. Bayat-Sarmadi, M. Mozaffari Kermani, and A. Reyhani-Masoleh, "Efficient and concurrent reliable realization of the secure cryptographic SHA-3 algorithm," *IEEE Trans. Computers-Aided Design Integr. Circuits Syst.*, vol. 33, no. 7, pp. 1105-1109, Jul. 2014.
- [11] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A High-Performance Fault Diagnosis Approach for the AES SubBytes Utilizing Mixed Bases," in *Proc. IEEE Workshop Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 80-87, Nara, Japan, Sep. 2011.
- [12] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON family of lightweight hash functions," in *Proc. Annual Cryptology*, Springer, Aug. 2011, pp. 222-239.
- [13] D. Augot and M. Finiasz, "Direct construction of recursive MDS diffusion layers using shortened BCH codes," in *Proc. Int. Workshop Fast Software Encryption*, 2014, pp. 3-17.
- [14] N. Farhady Ghalaty, B. Yuce, and P. Schaumont, "Analyzing the efficiency of biased-fault based attacks," *Embedded Systems Letters*, vol. 8, no. 2, pp. 33-36, 2016.
- [15] N. Farhady Ghalaty, B. Yuce, M. M. I. Taha, and P. Schaumont, "Differential fault intensity analysis," in *Proc. FDTC*, 2014, pp. 49-58.
- [16] S. Patranabis, A. Chakraborty, P. Ha Nguyen, and D. Mukhopadhyay, "A biased fault attack on the time redundancy countermeasure for AES," in *Proc. COSADE*, 2015, pp. 189-203.