Midori

Naimat Ali Khan[1], Anuj Verma[2] and Shreyas K Shastry[3]

[1] Indian Institute of Technology Bhilai, Raipur, India, naimatk@iitbhilai.ac.in
[2] Indian Institute of Technology Bhilai, Raipur, India, anujv@iitbhilai.ac.in
[3] Indian Institute of Technology Bhilai, Raipur, India, shreyas@iitbhilai.ac.in

Abstract. **Midori**, an **AES** lightweight block cipher, which has two variants namely, Midori64 and Midori128, **"Midori"** is the Japanese word for Green and the Block cipher **Midori** was designed to perform encryption and decryption by consuming optimal energy by the circuit per bit for the aforementioned processes.

This cipher was designed to perform the process of encryption and decryption by just making slight adjustments to the circuit so as to minimise the energy consumption by the circuit by re-utilizing most of the components of the circuit thus reducing energy overheads and reducing the required area for the circuit. This cipher has been shown to have better energy management than ciphers such as **PRINCE**.

In this paper, we discuss the working and functionality of this cipher and also apply various cryptanalysis techniques on this cipher.

Keywords: AES · Lightweight Cryptography · Midori64 · Midori128

1 Introduction

As the field of Lightweight Cryptography has received a lot of interest recently as evident from the numerous cipher proposals that have been made recently such as **CLEFIA**, **KATAN**, **PRESENT**, **Piccolo** to name a few. However, Advanced Encryption Standard (**AES**) remains as the standard when it comes to practical lightweight encryption. As the need for **IoT** securtiy and security for **RFID** tags and other low resource devices rises, finding a suitable crypto-system which can provide the required security and also can perform encryption and decryption with the limited resources available within a reasonably short period of time, lightweight cryptosystems are incorporated into these devices as they fulfill the requirements of these systems.

The efficiency of a lightweight design depends on the aspects such as, area of the circuit, throughput and most importantly energy.
The design choices of these systems consider aspects such as energy-efficiency and the related trade-offs associated with it. The factors that affect the energy-efficiency are clock frequency, architecture, loop rolling and choosing proper components for the construction of the system.

Usage of 4×4 almost MDS binary matrices is preferred over 4×4 MDS matrices as the almost MDS matrices are more efficient in terms of area and signal delay when compared to MDS matrices. It can be noted that the branch number(smallest nonzero sum of active inputs and outputs of the matrix) of almost MDS and MDS matrices are 4 and 5 respectively. Due to this almost MDS requires more number of rounds to ensure security of

the cipher. To overcome this, optimal cell-permutation layers are to be implemented which improve diffusion speed and increase the number of active S-Boxes in each round with lower implementation overheads which drastically improves the minimum number of active S-Boxes in each round which results in faster diffusion than the ShiftRow permutation method.

Midori is an AES block cipher which has 128-bit secret key with 64/128 bit blocks. This cipher outperforms ciphers like **PRINCE** and **NOEKEON** in the aspects mentioned before and is thus well suited to be used in systems which run on a tight energy budget such as **RFID** tags, sensor nodes, medical implants and battery operated portable devices.

2 Midori Cipher specifications

Now that we have established the considerations for the low energy design, we can get into the actual block cipher specifications.
Midori is a family of variant of **SPN**(Substitution Permutation Network) which as the name suggests has S-layer and P-layer and uses a 4 x 4 matrix as state matrix. It mainly consists of two variants namely:

- **Midori64**: It has a block size of $n = 64$.
- **Midori128**: It is the other variant with block size of $n = 128$.

Both the variants have similar 4 x 4 state matrix, but the size of each cell m is different in both of them.
Midori64 has cell size of $m = 4$ bits, while **Midori128** has cell size of $m = 8$ bits.
Suppose we define the cell elements of the state matrix as s_i, then we can say that $s_i \in \{0,1\}^m$. Now we just load a 64-bit or a 128-bit plaintext P accordingly with respect to the variant of **Midori** being used.

	block size(n)	key size	cell size(m)	number of rounds
Midori64	64	128	4	16
Midori128	128	128	8	20

Now as we got the rough idea on what **Midori** is we can now see the underlying **S-boxes** and also **Matrices**.
S-box: Here we use two types of bijective S-boxes namely Sb_0 and Sb_1, where $Sb_o, Sb_1 : \{0,1\}^4 \rightarrow \{0,1\}^4$.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
Sb_0	3	5	9	a	8	e	2	f	1	b	d	7	0	6	c	4
Sb_1	3	5	9	a	8	e	2	f	1	b	d	7	0	6	c	4

On a quick glance, we see that both Sb_0 and Sb_1 have involuntary property.Here, Sb_0 and Sb_1 are utilized in **Midori64** and **Midori128** respectively.

Above this **Midori128** also utilizes four other $8-bit$ sboxes namely $SSb_0, SSb_1, SSb_2, SSb_3$, where we have that $SSb_0, SSb_1, SSb_2, SSb_3 : \{0,1\}^8 \rightarrow \{0,1\}^8$. Now that we have the S-boxes in place lets see how we can derive SSb_i.
So, basically SSb_i mainly consists of input and output bit permutations, in order to keep the involuntary property we take output bit as the inverse of of permutation of the input

bit.
To say in simple words, let us consider p_i as the input bit permutation of each of the SSb_i. And now let us denote $x_{[i]}$ as the $i-th$ bit of x with $x_{[0]}$ being the most-significant bit, then we can see that we can easily denote,

$$p_i(x) = y^{(i)}$$

$$y^{(0)}_{[0,1,2,3,4,5,6,7]} = x_{[4,1,6,3,0,5,2,7]},\ y^{(1)}_{[0,1,2,3,4,5,6,7]} = x_{[1,6,7,0,5,2,3,4]}$$

$$y^{(2)}_{[0,1,2,3,4,5,6,7]} = x_{[2,3,4,1,6,7,0,5]},\ y^{(3)}_{[0,1,2,3,4,5,6,7]} = x_{[7,4,1,2,3,0,5,6]}$$

So, wee see that output bit in SSb_i is just the inverse map of p_i.
Matrix: In order to update the $m-bit$ values x_o, x_1, x_2, x_3 we define a involutive binary matrix **M**: $\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$
The matrix **M** updates the values as:

$$(x_o, x_1, x_2, x_3)^t = M.(x_o, x_1, x_2, x_3)^t$$

This constitutes for the MixColumn function which we will see later.
Note: The calculations are done in $GF(2^8)$.

Now that we have established some of the important terms, we can now jump into the round functions of **Midori**.

Round functions
The round function mainly consists of: S-layer carried out by **SubCell**, P-layer carried out through **ShuffelCell** and **MixColumn** and finally a key-addition layer which **KeyAdd** handles.

We will describe the functionalities of each of the functions separately:
1.**SubCell(S)**:We simply $s_i \leftarrow Sb_o[s_i]$ in **Midori64** and $s_i \leftarrow SSb_{imod(4)}[s_i]$ in **Midori64** for i in range $[0, 15]$.
2.**ShuffelCell(S)**:It applies the permutation as:
$(s0, s1, ..., s15)_{\searrow}(s0, s10, s5, s15, s14, s4, s11, s1, s9, s3, s12, s6, s7, s13, s2, s8)$..
3.**MixColumn(S)**: It updates the values as $(s_i, s_{i+1}, s_{i+2}, s_{i+3})^t \leftarrow M(s_i, s_{i+1}, s_{i+2}, s_{i+3})^t$, for $i = 0, 4, 8, 12$.
4.**KeyAdd(S,RK_i)**:The i-th n-bit round key RK_i is XORed to state S.

Now that we established the round function we can now get into round key generation.

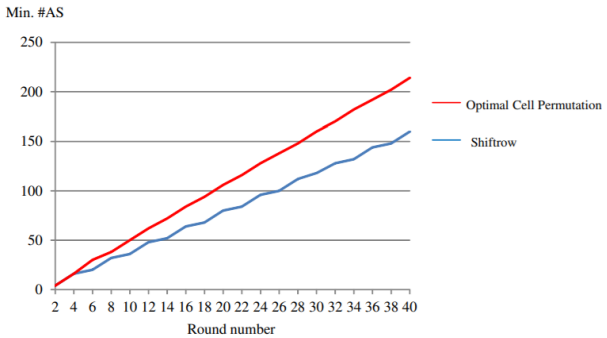
Round Key Generation
For **Midori64** we start up with the 128-bit secret key K, which is then denoted as two 64-bit keys as K_0 and K_1.
Then, we define $WK = K_0 \oplus K_1$. But for **Midori128**, we define $WK = K$ and $RK_i = K \oplus \beta_i$.Where β_i is round constants which we will define later.Round keys generations are same for both variants.

Now we can summarize the algorithm in a nutshell:

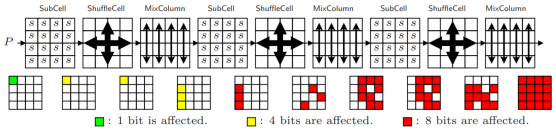
Algorithm $\mathsf{MidoriCore}_{(R)}(X, WK, RK_0, ..., RK_{R-2})$ **:**
$S \leftarrow \mathsf{KeyAdd}(X, WK)$
for $i = 0$ to $R - 2$ do
  $S \leftarrow \mathsf{SubCell}(S)$
  $S \leftarrow \mathsf{ShuffleCell}(S)$
  $S \leftarrow \mathsf{MixColumn}(S)$
  $S \leftarrow \mathsf{KeyAdd}(S, RK_i)$
$S \leftarrow \mathsf{SubCell}(S)$
$Y \leftarrow \mathsf{KeyAdd}(S, WK)$

Design Decisions:
Since, involutive almost MDS(Maximum Distance Separable) matrices have properties that are efficient for implementation but their diffusion speed is very low and also the number of active S-boxes are smaller compared to other cipher which also emply MDS matrices, to counter this the authors have proposed to adopt a way known as optimal cell-permutation layers which is aimed in improving security,diffusion speed and also increase the number of active S-boxes in each round.
So how do we come up with such an apporach?

Apporach for finding optimal cell-permutation layers:
Since its a no brainer for exhaustive search of min number of active S-boxes for among all permutations we use Matus's search mehthod to reduce the search space.
By this search we see that in order to achieve full diffusion in 3-rounds we have to take care of following conditions:
Condition-1: After cell-permutation application we have to make sure that the columns in input cell is mapped to some cell in different column all together.
Condition-2:After cell-permutation application we have to make sure that the columns in input cell is mapped to a cell in same row.
Condition-3:After applying cell-permutation once and also applying it three times inversely each cell in input must be mapped to a cell in the same row.
Now that we have our conditions set-up we can compare this approach with the traditional ShiftRow-type of permutation, we see that the new approach does all the things traditional ShiftRow does but does a bit better than the old approach, hence this optimal permutations achieves security against several attacks while having efficient implementation.



S-box layer:
The properly chosen bit-permutation in beginning and end of S-boxes, which was added to counter the full round attack seen in other ciphers and also keep the involution property, the permutation in such a way that it satisfies the following properties:
1.**Property-1**: Affected 4-bit positions of outputs of an S-box are included in both of two different input groups of the other three S-boxes.
2.**Property-2**:Each input cell affects three cells in the different cell positions from the input.



Now that we have most of the idea on the cipher we can now see some other important functions like Key scheduling and also round constants.

Key scheduling function
In order to save energy **Midori** doesn't employ any key scheduling function, the i-th round key is defined by $L{-}1(K_1) \oplus L{-}1(\beta_{18-i})$, where L^{-1} is inverse of linear layer.
The L^{-1} has the operation defined as $InvShuffleCell.MixColumn$, and $InvShuffleCell$ permutes the state as:
$(s0, s1, ..., s15) \leftarrow (s0, s7, s14, s9, s5, s2, s11, s12, s15, s8, s1, s6, s10, s13, s4, s3)$.
Round Constants
We use a 4 x 4 binary matrix as round constants, which was derived from the hexadecimal encoding of fractional part of π.In order to get further round keys we just bitwise add the least significant bit of round key to previous round constants.

Thus we have a energy efficient key schedule and round constant derivation.

Table 1: The minimum number of deferentially/linearly active S-boxes(AS) of **Midori32** and **Midori64**

Round Number	4	5	6	7	8	9	10	11	12	13	14	15	16
Min. # of AS (Optimal Cell-Permutation)	16	23	30	35	38	41	50	57	62	67	72	72	84
Min # of AS (ShiftRow-type Permutation)	16	18	20	26	32	34	36	42	48	50	52	58	64

Table 2: DDT for Sb_0.

in/out	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	-	-	-	-	-	-	-	-	-	-	-		-	-	-
1	-	2	4	-	2	2	2	-	2	-	-	-	-	-	2	-
2	-	4	-	-	4	-	-	-	-	4	-	-	4	-	-	-
3	-	-	-	-	2	-	4	2	2	2	-	-	-	2	-	2
4	-	2	4	2	2	2	-	-	2	-	-	2	-	-	-	-
5	-	2	-	-	2	-	-	4	-	2	4	-	2	-	-	-
6	-	2	-	4	-	-	-	2	2	-	-	-	2	2	-	2
7	-	-	-	2	-	4	2	-	-	-	-	2	-	4	2	-
8	-	2	-	2	2	-	2	-	-	2	-	2	2	-	2	-
9	-	-	4	2	-	2	-	-	2	2	-	2	2	-	-	-
A	-	-	-	-	-	4	-	-	-	-	4	-	-	4	-	4
B	-	-	-	-	2	-	-	2	2	2	-	4	-	2	-	2
C	-	-	4	-	-	2	2	-	2	2	-	-	2	-	2	-
D	-	-	-	2	-	-	2	4	-	-	4	2	-	-	2	-
E	-	2	-	-	-	-	-	2	2	-	-	-	2	2	4	2
F	-	-	-	2	-	-	2	-	-	-	4	2	-	-	2	4

3 Security Evaluation

3.1 Differential and Linear Cryptanalysis

The DDT and LAT for Sb_0 is show in Table 2 and Table 3 respectively. We can see from the two tables that the maximum differential and linear probability for both the DDT and LAT is $\frac{4}{16}$ i.e. 2^{-2}. Similarly for all the other S-Boxes (i.e. Sb_1, SSb_0, SSb_1, SSb_2 and SSb_3), the maximum differential and linear probability is 2^{-2}.
For Linear and Differential cryptanalysis, we must try to reduce the number of active S-boxes. We have already seen in Table 1, that the minimum number of active S-boxes for variants **Midori32** and **Midori64** are 32 and 64 after round 7 and 13 respectively.
Therefore, after round 7 and 13 for variants **Midori32** and **Midori64**, the maximum differential and linear probability will be $(2^{-2})^{32}$ and $(2^{-2})^{64}$ respectively. This implies that after 7 and 13 rounds of the respective variants, it become computationally hard to do differential or linear cryptanalysis. **Midori32** and **Midori64**, both has sufficient number of rounds in order to prevent differential and linear attacks.

3.2 Integral Attack

First we will look at the properties associated with integral attack:

- **A**: The **All** property hold true when all the elements in a set is different from every other element in the set.
- **C**: The **Constant** properly holds true when all the elements in a set are same.
- **B**: The **Balanced** properly holds true when the combined xor of all the elements of a set it zero.

Table 3: LAT for Sb_0.

in/out	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	+8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	+2	+4	+2	-2	.	+2	.	-2	.	+2	.	+4	-2	.	-2
2	.	+4	.	.	+4	.	.	.	-4	.	.	.	.	+4	.	.
3	.	+2	.	+2	-2	.	+2	+4	+2	-4	-2	.	.	+2	.	+2
4	.	-2	+4	-2	+2	.	-2	.	-2	-4	-2	.	.	-2	.	+2
5	.	.	.	.	.	.	.	.	.	.	-4	-4	.	.	+4	-4
6	.	+2	.	+2	-2	.	+2	-4	-2	.	-2	.	-4	-2	.	+2
7	.	.	.	+4	.	.	-4	.	.	.	.	-4	.	.	-4	.
8	.	-2	-4	+2	-2	.	-2	.	-4	-2	.	+2	+2	.	+2	.
9	.	.	.	-4	-4	.	.	.	-2	+2	-2	-2	+2	+2	-2	+2
A	.	+2	.	-2	-2	-4	-2	.	.	-2	+4	-2	-2	.	+2	.
B	.	.	.	.	.	-4	.	-4	+2	-2	-2	+2	+2	+2	-2	-2
C	.	+4	.	.	.	.	-4	.	+2	+2	-2	+2	+2	-2	+2	+2
D	.	-2	+4	+2	-2	.	-2	.	.	+2	.	+2	-2	+4	+2	.
E	.	.	.	.	.	+4	.	-4	+2	-2	+2	-2	+2	+2	+2	+2
F	.	-2	.	+2	+2	-4	+2	.	.	+2	.	-2	+2	.	+2	+4

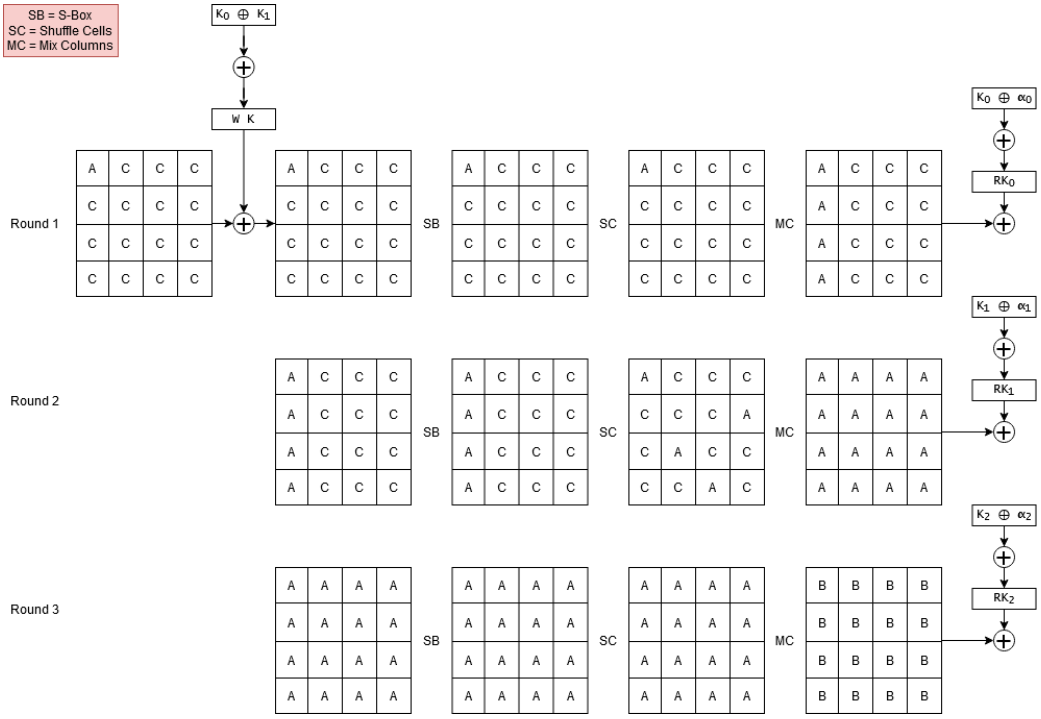


Figure 1: Integral Analysis of Midori Core for 3 rounds.

For Integral attack we took 2^n states of n bit cells, the initial states of the set are such that 1 cell has the property **A**, whereas the other 15 cells has the property **C**. We observed that after 4 Rounds, all the cells of the state has the **B** property. This means that after 4 Rounds none of the cell in state holds any of the property stated above (after 3 rounds the state becomes Balanced for all 16 cells). We did the integral analysis and result is shown in Figure 1. We can see in the figure that all cells holds the balanced property after 3 rounds, if we go any further then the state will lose this balanced property.
Therefore, similar to what we have seen in integral analysis of AES, we can only do key recovery attack upto a maximum of 5. If we go beyond 5 rounds, then we won't be able to deduce the integral property of a cell.
Hence, both **Midori32** and **Midori64** has sufficient number of rounds to prevent integral attacks on the model.

3.3 Other attacks discussed:

There are various other attacks discussed in the literature which we are not going to look into detail:

1. Boomerang-Type Attack
2. Meet-in-the-Middle Attacks
3. Slide Attacks
4. Reflection Attacks

4 Conclusion

In this paper we discussed the Midori block cipher, we discussed two variants of Midori, i.e. **Midori32** and **Midori64**. Midori is designed to consumed low energy, and therefore use components such as S-Box, Mix Columns, Shuffle Cells etc. which make the cipher energy efficient. The designers of Midori has taken careful decisions at every step in order to make the cipher as lightweight as possible.
We also discussed the design for each variant of the cipher and looked at at the algorithms involved in the encryption and decryption using Midori. Midori is an AES block cipher and is a family of variant of Substitution Permutation Network. We looked at various design decisions (such as Round Functions, Key Generation, Optimal Cell Permutation) that make the cipher energy efficient.
Lastly we took the roll of the attacker and tried to launch various attacks on the cipher model. We observed that Midori cipher is robust enough to prevent against all of the attacks discussed in the literature. We looked at Differential/Linear attacks as well as Integral Attacks in detail and concluded that Midori has sufficient number of rounds and properties in order to prevent against these attacks.