Related-Key Cryptanalysis of Midori *

David Gérault and Pascal Lafourcade
firstname.lastname@udamail.fr

University Clermont Auvergne

Abstract. Midori64 and Midori128 [2] are lightweight block ciphers, which respectively cipher 64-bit and 128-bit blocks. While several attack models are discussed by the authors of Midori, the authors made no claims concerning the security of Midori against related-key differential attacks. In this attack model, the attacker uses *related-key differential characteristics*, *i.e.*, tuples $(\delta_P, \delta_K, \delta_C)$ such that a difference (generally computed as a XOR) of δ_P in the plaintext coupled with a difference δ_K in the key yields a difference δ_C after r rounds with a good probability. In this paper, we propose a constraint programming model to automate the search for optimal (in terms of probability) related-key differential characteristics on Midori. Using it, we build related-key distinguishers on the full-round Midori64 and Midori128, and mount key recovery attacks on both versions of the cipher with practical time complexity, respectively $2^{35.8}$ and $2^{43.7}$.

Key Works: Midori, Related-Key Attack, Constraint Programming.

1 Introduction

The increasing usage of embedded devices led to a lot of research on how to adapt existing cryptographic primitives for the low power and energy constraints associated with the internet of things. Lightweight block ciphers follow this trend, and aim at providing energy efficient ways to ensure confidentiality for fixed size block messages. In 2015, the authors of [2] consider the challenging task of minimizing the energy cost for a lightweight block cipher. They proposed a lightweight symmetric block cipher scheme called *Midori*, composed of two versions Midori64 and Midori128, which respectively cipher 64- and 128-bit message blocks.

In this paper, we challenge the related-key security of both versions of Midori. In the related-key model, introduced independantly by Biham [3] and Knudsen [12], the attacker is allowed to require the encryption of messages of his choice under the secret key, but also under other keys which have a relation to the original one. For instance, if K is the secret key, the attacker can require the

*This research was conducted with the support of the FEDER program of 2014-2020, the region council of Auvergne, and the Digital Trust Chair of the University of Auvergne.

encryption of a message m under K, but also under another key K^*, computed as $K \oplus \delta_K$, where $\oplus$ is the XOR operation and δ_K is a bit string chosen by the attacker. In an ideal block cipher, the distribution of the resulting ciphertext difference should be uniform and independant from the input difference. However, in real ciphers, there exist *related-key differential characteristics*, *i.e.,* difference propagation patterns, which happen with higher probabilities. In a related-key differential attack, the attacker requires the encryption of message pairs satisfying a difference δ_P, under keys satisfying a difference δ_K, expecting an output difference δ_C.

One of the main applications of related-key cryptanalysis is finding collisions on hash functions built from block ciphers (*e.g.,* with the Davies-Meyer construction). For instance, the hash function used in Microsoft's Xbox was broken due to the related-key vulnerability of the underlying block cipher TEA [19], leading to a hack of the system [20]. Related-key attacks were not taken into account in the design of Midori, and the authors made no claim on its security in this model. As Midori is designed for embedded devices, it could however be used to build a hash function, which motivates scrutinizing its security in the related key setting.

The search for related-key differential characteristics is however difficult. Following the idea of [9], we use constraint programming (CP) to tackle this problem. In this programming paradigm, instead of providing an imperative algorithm, the programmer describes the problem to be solved as a set of variables linked together by *constraints* (for instance, $x + y = 10$), and the exploration of the search space is left to the solver. While an overwhelming part of the cryptanalysis literature relies on custom algorithms, we believe that a more generic approach is very promising. In particular, as shown in [9], constraint programming seems less error prone than custom code.

Contributions:

- We provide constraint programming models to find optimal related key differential characteristics on both versions of Midori.
- Using our models, we give the optimal $R - 1$ rounds related-key differential characteristics of both versions of Midori, with probability 2^{-14} for Midori64, and 2^{-38} for Midori128.
- We then mount practical time key recovery attacks requiring $2^{35.8}$ operations with 20 related keys for Midori64, and $2^{43.7}$ encryptions with 16 related keys for Midori128.
- We also provide a related-key distinguisher of probability 2^{-16} for Midori64 and 2^{-40} for Midori128.

Related Work: Most results in the literature using constraint programming for the cryptanalysis of block ciphers use Mixed Integer Linear Programming (MILP). In [14], the authors use MILP to mount a linear cryptanalysis on a stream cipher and on the block cipher AES, both in the single key setting. In [18] and [17], the authors use MILP to find the best related-key differential characteristics on several bit oriented block ciphers, but they do not treat Midori. As opposed to MILP, CP supports *table constraints* defining tuples of authorized

Type	Rounds	Data	Time	Reference
Midori64				
Impossible differential	10	$2^{62,4}$	$2^{80,81}$	[6]
Meet-in-the-middle	12	$2^{55.5}$	$2^{125,5}$	[13]
Invariant subspace*	full(16)	2	2^{16}	[11]
Related-key differential	14	2^{59}	2^{116}	[7]
Related-key differential	**full(16)**	$2^{23.75}$	$2^{35.8}$	**Section 5**
Midori128				
Related-key differential	**full(20)**	$2^{43.7}$	$2^{43.7}$	**Section 5**

Table 1. Summary of the attacks against Midori.

values, which provides a rather efficient way to model the non linear SBs. To the best of our knowledge, only [9] uses classical CP instead of MILP. The authors present a model for finding optimal related-key differential characteristics against AES, using a method similar to the one presented in this paper.

The existing attacks against Midori are summed up in Table 1.

In [6], the authors propose an impossible differential attack on 10 rounds of Midori64. In [13], Li Lin and Wenling Wu describe a meet-in-the-middle attack on 12-round Midori64. In [11], the authors exhibit a class of 2^{32} weak keys which can be distinguished with a single query. Assuming a key from this class is used, then it can be recovered with as little as 2^{16} operations, and a data complexity of 2^1. Finally, a related-key cryptanalysis of Midori64 is performed in [7]. It covers 14 rounds and has a complexity of 2^{116}, as opposed to $2^{35.8}$ for ours. This difference is due to their differential characteristics being far from optimal.

As for Midori128, to the best of our knowledge, no cryptanalysis on it has been published yet. We fill this gap by mounting a key recovery attack on the whole cipher, requiring $2^{43.7}$ encryptions.

Outline: In Section 2, we give a brief description of Midori. We then remind the basics of related-key cryptanalysis and introduce our notations in Section 3. We present our CP models in Section 4. Finally, we detail our results in Section 5, before concluding in the last section.

2 Description of Midori Encryption Scheme

Both versions of Midori, Midori64 and Midori128, use 128-bit keys. In both versions, the blocks are treated as 4×4 matrices of words of m bits, with $m = 4$ for Midori64 and $m = 8$ for Midori128. The encryption process consists in applying a round function that updates an internal state S, represented as

$$S = \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}$$

Fig. 1. Representation of the state in Midori.

*Note that this attack only works if a key from the weak class is used

shown on Figure 1 (where the s_i are 4-bit words for Midori64 and 8-bit words (bytes) for Midori128), for a given number of rounds R. For Midori64, R is equal to 16, whereas for Midori128 R is 20[†].

The round function is composed of the following consecutive operations:

SubCell (SB) substitutes every cell of the state, using a non linear *Substitution Box*, denoted Sbox. The Sbox of Midori64 is given as example in Figure 2(a). For Midori128, 4 different Sboxes are used (one for each line of the state)[‡].

ShuffleCell (SC) operates a permutation of the cells of the state. On input $(s_0, \ldots, s_{15})$, it applies the following permutation:
$(s_0, s_{10}, s_5, s_{15}, s_{14}, s_4, s_{11}, s_1, s_9, s_3, s_{12}, s_6, s_7, s_{13}, s_2, s_8)$.

MixColumns (MC) multiplies the state by the symmetric matrix given in Figure 2(b), thus applying a linear transformation on each column independently. It has the quasi-MDS property if $MC(0,0,0,0) = (0,0,0,0)$ or $|X| + |MC(X)| = 0$ or $|X| + |MC(X)| \geq 4$, where $|X|$ denotes the number of non-zero words in a column X of the state.

KeyAdd (KA) is a XOR between S and a round key derived from the initial key.

The Midori encryption process works as follows: an initial KeyAdd, using the whitening key WK, is applied. Then, the round function is executed $R-1$ times. Finally, a final SubCell is applied to the resulting state, and a new KeyAdd is performed, again using WK. The round key derivation is very straightforard : the key for each round i is obtained by XORing the initial key with a predefined 4×4 constant matrix α_i. For Midori64, the 128-bit key is considered as two 4×4 matrices of 4-bit words K_0 and K_1, and WK is computed as $K_0 \oplus K_1$. The round key for round i is computed as $K_{i \mod 2} \oplus \alpha_i$. For Midori128, K is a single 4×4 bytes matrix, and $WK = K$. The round key for round i is then simply computed as for Midori64: $K \oplus \alpha_i$.

3 Related-Key Cryptanalysis

Differential cryptanalysis studies the propagation of the differences, generally computed as a XOR, between two plaintexts ciphered with the same key. Related-key cryptanalysis, which was independently introduced by Biham [3] and Knudsen [12], additionally considers the case where the two plaintexts are ciphered

[†]The full specification is presented in [2].

[‡]The Sboxes of Midori 128 are given in [11]

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$SB(x)$	c	a	d	3	e	b	f	7	8	9	1	5	0	2	4	6

(a) The Sbox of Midori64.

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

(b) The MixColumns matrix of Midori.

Fig. 2. Midori description.

with different keys. A tuple $(\delta_{in}, \delta_K, \delta_{out})$ is an *n-rounds related-key differential* for a keyed round function f_K, for which f_K^i denotes the output after round i (starting from 0), if for some plaintext P and key K it holds that $f_K^{n-1}(P) \oplus f_{K\oplus\delta_K}^{n-1}(P \oplus \delta_{in}) = \delta_{out}$. Similarly, if $X_i^{P,K}$ denotes the *internal state* of the round function with inputs P and K at round i, a tuple $(\delta_{in}, \delta_K, \delta_{X_0} \dots \delta_{X_{n-1}}, \delta_{out})$ is an *n-rounds related-key differential characteristics* if $(\delta_{in}, \delta_{out}, \delta_K)$ is an n-rounds related-key differential and, for all i from 0 to $n-1$, it holds that $X_i^{P,K} \oplus X_i^{P\oplus\delta_{in}, K\oplus\delta_K} = \delta_{X_i}$.

The differences δ_X are composed of differential words, defined as $\delta_X[i][j] = X[i][j] \oplus X'[i][j]$, where $X[i][j]$ (resp. $X'[i][j]$) denotes a word at position i, j (where $(i, j) \in [0; 3]^2$) of a matrix X (resp. X').

The probability $p = Pr[(\delta_{in}, \delta_K) \rightarrow \delta_{out}]$ denotes the probability that a related-key differential $(\delta_{in}, \delta_k, \delta_{out})$ holds, *i.e.*, for P and K drawn uniformly at random, $f_K(P) \oplus f_{K\oplus\delta_K}(P \oplus \delta_{in}) = \delta_{out}$.

Note that, by definition, for the linear parts L of the cipher, we have $L(P) \oplus L(P \oplus \delta) = L(\delta)$, for any P and δ. On the other hand, for the non linear parts NL, $NL(P) \oplus NL(P \oplus \delta)$ is generally different from $NL(\delta)$. Hence, to handle the non linear parts of block ciphers (namely the Sboxes), related-key differential cryptanalysis usually uses a Differential Distribution Table (DDT) to derive the probability $Pr[\delta_{in} \rightarrow \delta_{out}]$ that for a random word w, $SB(w) \oplus SB(w \oplus \delta_{in}) = \delta_{out}$. For any differential words δ_{in} and δ_{out}, $DDT[\delta_{in}][\delta_{out}]$ gives the number of words w satisfying this relation, and the probability is computed as $\frac{DDT[i][j]}{2^{|W|}}$, where $|W|$ denotes the bit length of the words. When the Sboxes are bijective[§], they do not introduce nor remove differences. More formally, it holds that for any word w, $SB(w) \oplus SB(w \oplus \delta) \neq 0$ if $\delta \neq 0$, and $SB(w) \oplus SB(w \oplus \delta) = 0$ if $\delta = 0$. Said otherwise, for a given Sbox, $Pr[0 \rightarrow 0] = 1$. Hence, the probability of a related-key differential characteristic is only affected by *active Sboxes*, *i.e.*, Sboxes which have a non-zero difference at their input. Thus, the probability p for a related-key differential characteristic to hold for random P and K is computed as the product of the probabilities associated with each the active Sboxes it contains. We have $p = \prod_{i=1}^{x} p_i$, where p_i denotes the probability that the transition $\delta_{ini} \rightarrow \delta_{outi}$, defined by the related-key differential characteristic, holds for the i^{th} active Sbox (among x). Since the complexity of a related-key differential key recovery attack is directly related to the probability of the related-key differential that is used, characteristics with the least possible active Sboxes are generally the most interesting. The crucial point of this type of cryptanalysis is then to determine high probability related-key differential characteristics.

However, exhaustive search on all possible input differences is not practical for Midori because the size of the input is 128 bits for the key and 64 or 128 bits for the plaintext. Hence, a common method is to solve the problem in two steps (*e.g.,* [5, 8, 9]). The first step does not consider the value of the differential bytes, but only the positions of non-zero differences. During the first step, the differential words are abstracted to a compact representation. In the *compact*

[§]It is the case for most block ciphers, including Midori.

representation, the differential words are abstracted to differential bits. The differential bit Δ representing a differential word δ in the compact representation is defined by $\delta = 0 \Rightarrow \Delta = 0$ and $\delta \neq 0 \Rightarrow \Delta = 1$. We denote a differential word by δ, and a differential bit by Δ.

An n-round *compact related-key differential characteristic* Δ abstracting a related-key differential characteristic $\delta = (\delta_{in}, \delta_k, \delta_{X_0}, \ldots, \delta_{X_{n-1}}, \delta_{out})$ is the tuple $(\Delta_{in}, \Delta_k, \Delta_{X_0}, \ldots, \Delta_{X_{n-1}}, \Delta_{out})$, where $\Delta_{in} = 0$ if $\delta_{in} = 0$ and $\Delta_{in} = 1$ if $\delta_{in} \neq 0$ (and similarly for $\Delta_{X_0}, \ldots, \Delta_{X_{n-1}}$ and Δ_{out}).

The idea of working with two steps is that related-key differential characteristics with the best probabilities are generally the ones with the least active Sboxes. Hence, a lot of filtering can be done by simply starting by working on differential bits and minimizing the number of active Sboxes.

Once compact related-key differential characteristics minimizing the number of active Sboxes are obtained, a second step is run to build full related-key differential characteristics built with differential words. Note that not all compact related-key differential characteristics can be instantiated with differential words. The main reason is that a given input difference to an Sbox can only yield a limited number of output differences. For instance, in Midori64, if $\delta_X = 0x1$ (where $0x$ denotes hexadecimal representation), there exists no word X such that $SB(X) \oplus SB(X \oplus 0x1) = 0x9$, according to the Sboxes of Midori64 given in Figure 2(a). Moreover, the coefficients of the MixColumns matrix cannot be directly taken into account with differential bits, nor can the equalities of the corresponding differential words. This yields transitions that are correct when working on a bit related-key differential characteristic, but not with differential words. Such solutions are said to be *inconsistent*, otherwise, they are *consistent*.

4 Constraint Programming Model

We describe our constraint programming models to find related-key differential characteristics with optimal probability on Midori. This process is decomposed in two steps: the first one aims at lower bounding the number of active Sboxes. It only considers compact related-key differential characteristics. The second step transforms the solutions to Step 1 into word related-key differential characteristics when it is possible. In other words, during Step 1, we simply find the positions of the differences in the related-key differential characteristic, and in Step 2, we assign actual values to these differences.

4.1 Step 1

Variables and objective function: In Step 1, we consider the propagation of differences through the cipher by working on compact related-key differential characteristics. Let n denote the number of times the full round function is applied, *i.e.,* we neglect the initial KeyAdd and the final KeyAdd and SB. For Midori64, $n = 15$, and for Midori128, $n = 19$. When no information about the format is provided, the variables are $n \times 4 \times 4$ binary arrays, *i.e.,* one 4×4 matrix per round. Each of the following variables represent differential bits:

Δ_K represents the differential bits of the key. For Midori64, it is modeled as a $2 \times 4 \times 4$ binary array (as the initial key is composed of two 4×4 matrices). For Midori128, it is represented as a 4×4 binary matrix.

Δ_{SB} represents the state after the SB operation. Note that since this operation does not introduce differences, this variable is somehow redundant. We however use it for readability.

Δ_{KA} represents the state after the KeyAdd operation.

Δ_{MC} represents the state after the MixColumns operation.

Δ_{SC} represents the state after the ShuffleCell operation.

The relations between these variables for a given round r is:

$$\Delta_{SB}[r] \xrightarrow{SC} \Delta_{SC}[r] \xrightarrow{MC} \Delta_{MC}[r] \xrightarrow{KA} \Delta_{KA}[r] \xrightarrow{SB} \Delta_{SB}[r+1]$$

Our aim is to minimize the number of active Sboxes, *i.e.,* Sboxes with non zero differences. Hence, we ask the solver to minimize the sum of all $\Delta_{SB}[r]$, which constitutes our *objective function*:

$$Minimize\left(\sum_{r=0}^{n-1}\sum_{i=0}^{3}\sum_{j=0}^{3}\Delta_{SB}[r][i][j]\right)$$

Constraints: Since we work with differential bits representing the presence or absence of difference, we cannot use the regular XOR operation between such values for KeyAdd nor MixColumns. Let Δ_0 and Δ_1 denote two differential bits. We remind that Δ_0 (resp. Δ_1) is 1 if $\delta_0 \neq 0$ (resp. $\delta_1 \neq 0$). The compact representation contains no information about the actual values of δ_0 and δ_1 when they are non-zero. This abstraction leads us to define the following constraint that describes the xor between several differential bits $x_1, \ldots, x_{q-1}$ where x_q is the result:

$$\mathsf{XOR}(x_1, \ldots, x_q) \Leftrightarrow \{x_1 + \ldots + x_q \neq 1\}$$

where $+$ denotes the integer addition and $x_1, \ldots, x_k \in \{0, 1\}$. Intuitively, it states that the xor of the $q-1$ corresponding words is known to be 0 when all the differential bits are zero, or only one is non zero, but can be either 0 or 1 otherwise.

For ShuffleCell, we simply apply the permutation given in Section 2 to build $\Delta_{SC}[r]$ from $\Delta_{SB}[r]$.
$\Delta_{SC}[r][0][0] = \Delta_{SB}[r][0][0], \Delta_{SC}[r][1][0] = \Delta_{SB}[r][2][2], \Delta_{SC}[r][2][0] = \Delta_{SB}[r][1][1],$
$\Delta_{SC}[r][3][0] = \Delta_{SB}[r][3][3], \Delta_{SC}[r][0][1] = \Delta_{SB}[r][2][3], \Delta_{SC}[r][1][1] = \Delta_{SB}[r][0][1],$
$\Delta_{SC}[r][2][1] = \Delta_{SB}[r][3][2], \Delta_{SC}[r][3][1] = \Delta_{SB}[r][1][0], \Delta_{SC}[r][0][2] = \Delta_{SB}[r][1][2],$
$\Delta_{SC}[r][1][2] = \Delta_{SB}[r][3][0], \Delta_{SC}[r][2][2] = \Delta_{SB}[r][0][3], \Delta_{SC}[r][3][2] = \Delta_{SB}[r][2][1],$
$\Delta_{SC}[r][0][3] = \Delta_{SB}[r][3][1], \Delta_{SC}[r][1][3] = \Delta_{SB}[r][1][3], \Delta_{SC}[r][2][3] = \Delta_{SB}[r][2][0],$
$\Delta_{SC}[r][3][3] = \Delta_{SB}[r][0][2].$

The constraint for MC contains two parts, where r varies from 0 to $n-1$ and j varies from 0 to 3.

Firstly the quasi-MDS property directly gives the following constraint:

$$\left(\sum_{i=0}^{3}\Delta_{SC}[r][i][j] + \Delta_{MC}[r][i][j]\right) \in \{0, 4, 5, 6, 7, 8\}$$

Then, we model the fact that $MC(0, 0, 0, 0) = (0, 0, 0, 0)$ as follows:

$$\left(\sum_{i=0}^{3} \Delta_{SC}[r][i][j] = 0\right) \Leftrightarrow \left(\sum_{i=0}^{3} \Delta_{MC}[r][i][j] = 0\right).$$

The second part directly implements the product of the vector Δ_{SC} with the matrix given in Midori to get Δ_{MC}. It is modeled as follows:

$$\begin{array}{l}
\mathsf{XOR}(\Delta_{SC}[r][1][j], \Delta_{SC}[r][2][j], \Delta_{SC}[r][3][j], \Delta_{MC}[r][0][j]) \\
\mathsf{XOR}(\Delta_{SC}[r][0][j], \Delta_{SC}[r][2][j], \Delta_{SC}[r][3][j], \Delta_{MC}[r][1][j]) \\
\mathsf{XOR}(\Delta_{SC}[r][0][j], \Delta_{SC}[r][1][j], \Delta_{SC}[r][3][j], \Delta_{MC}[r][2][j]) \\
\mathsf{XOR}(\Delta_{SC}[r][0][j], \Delta_{SC}[r][1][j], \Delta_{SC}[r][2][j], \Delta_{MC}[r][3][j])
\end{array}$$

For KA, following the rules of Midori and the XOR constraint described earlier[¶], we have, for r from 0 to $n-1$, and i and j from 0 to 3: For Midori64 :

$$\mathsf{XOR}(\Delta_{MC}[r][i][j], \Delta_K[r \mod 2][i][j], \Delta_{KA}[r][i][j])$$

and for Midori128:

$$\mathsf{XOR}(\Delta_{MC}[r][i][j], \Delta_K[i][j], \Delta_{KA}[r][i][j])$$

4.2 Step 2

Variables: In addition to the variables from Step 1, new ones are introduced to represent the differential words in the whitening key, the plaintext, the result of the initial KeyAdd, and the probabilities for each Sbox. When no information about the format is provided, the following variables are $n \times 4 \times 4$ word arrays, *i.e.,* one 4×4 matrix per round.

δ_K represents the differential words in the key. It is modeled as a $2 \times 4 \times 4$ array of 4-bit words for Midori64, as a 4×4 byte matrix for Midori128.

δ_{SB} represents the state after the SB operation.

δ_{KA} represents the state after the KeyAdd operation.

δ_{MC} represents the state after the MixColumns operation.

δ_{SC} represents the state after the ShuffleCell operation.

δ_{WK} represents the whitening key, which is $\delta_K[0] \oplus \delta_K[1]$ for Midori64, and δ_K for Midori128.

δ_P represents the plaintext and $\delta_{P'}$ the state after the initial KeyAdd.

P is a $n \times 4 \times 4$ matrix used to compute the final probability, where $P[r][i][j] = 0$ if $\delta_{SB}[r][i][j] = 0$, and $log_2(DDT[\delta_{KA}[r][i][j]][\delta_{SB}[r+1][i][j])$ otherwise. For Midori64, the domain of this variable is $\{0, 1, 2\}$, whereas for Midori128 it is $\{0, 1, 2, 3, 4, 5, 6\}$.

To find the optimal related-key differential characteristics, we need to maximize the sum of the P variables, hence our objective function is

$$Maximize \left(\sum_{r=0}^{n-1} \sum_{i=0}^{3} \sum_{j=0}^{3} P[r][i][j] \right)$$

[¶]Note that the XOR operations between the key and constants at each rounds are not taken into account when working at a differential level. This is because the constants are canceled, *i.e.,* for two different keys K^0 and K^1, and a constant c, $(K^0 \oplus c) \oplus (K^1 \oplus c) = K^0 \oplus K^1$.

Constraints: The first constraints aim at linking each variable of the input (from Step 1) to the variables of Step 2: for instance, for δ_{KA}, we have $\forall r \in [0..n-1], \forall i \in [0..3], \forall j \in [0..3]$:

$$\text{if } \Delta_{KA}[r][i][j] == 0 \text{ then } \delta_{KA}[r][i][j] = 0, \text{ else } \delta_{KA}[r][i][j] > 0$$

Similar constraints are defined for the other input variables.

The constraint for SC is exactly the same as in Step 1, except that it is on the δ variables (differential words) instead of the Δ variables (differential bits).

For the other operations, we make use of *table* constraints to model the Sboxes and the XOR operations[‖]. Intuitively, table constraints tell the solver which tuples of values are allowed.

We denote $\mathsf{tupleXOR}$ the set of all tuples (X, Y, Z) that satisfying $Z = X \oplus Y$. Similarly we denote $\mathsf{tupleSB}_S$ the tuples modeling the DDT for the Sbox S, *i.e.,* for every couple of words $\delta_{in}, \delta_{out}$ there is a tuple $(\delta_{in}, \delta_{out}, log_2(DDT_S[\delta_{in}][\delta_{out}]))$, where DDT_S is the DDT of the Sbox S.

We also denote by $\mathsf{TABLE}((x, y, z), SET)$, the constraint that tells the solver that the values x, y and z must for a valid tuple with regards to a given SET.

We define $\mathsf{XORbyte}(x, y, z) := \mathsf{TABLE}((x, y, z), \mathsf{tupleXOR})$ and extend it to $\mathsf{XORbyte}_3(x, y, z, w) := \mathsf{XORbyte}(t, z, w)$, where t is defined by $\mathsf{XORbyte}(x, y, t)$.

The MixColumns operation can then be expressed, according the specification of the cipher, by: $\forall r \in [0..n-1], \forall j \in [0..3]$:

$$\begin{array}{l}
\mathsf{XORbyte}_3(\delta_{SC}[r][1][j], \delta_{SC}[r][2][j], \delta_{SC}[r][3][j], \delta_{MC}[r][0][j]) \\
\mathsf{XORbyte}_3(\delta_{SC}[r][0][j], \delta_{SC}[r][2][j], \delta_{SC}[r][3][j], \delta_{MC}[r][1][j]) \\
\mathsf{XORbyte}_3(\delta_{SC}[r][0][j], \delta_{SC}[r][1][j], \delta_{SC}[r][3][j], \delta_{MC}[r][2][j]) \\
\mathsf{XORbyte}_3(\delta_{SC}[r][0][j], \delta_{SC}[r][1][j], \delta_{SC}[r][2][j], \delta_{MC}[r][3][j])
\end{array}$$

We now define δ_{WK}. For Midori64, δ_{WK} is defined by: $\forall i \in [0..3], \forall j \in [0..3]$:

$$\mathsf{XORbyte}(\delta_K[0][i][j], \delta_K[1][i][j], \delta_{WK}[i][j])$$

For Midori128, we simply have $\delta_{WK} = \delta_K$.

The initial KeyAdd operation then is modeled as: $\forall i \in [0..3], \forall j \in [0..3]$:

$$\mathsf{XORbyte}(\delta_P[r][i][j], \delta_{WK}[i][j], \delta_{P'}[i][j])$$

For the other KeyAdd operations, we have $\forall r \in [0..n-1], \forall i \in [0..3], \forall j \in [0..3]$:

$$\mathsf{XORbyte}(\delta_{MC}[r][i][j], \delta_K[r \mod 2][i][j], \delta_{KA}[r][i][j])$$

for Midori64, and

$$\mathsf{XORbyte}(\delta_{MC}[r][i][j], \delta_K[i][j], \delta_{KA}[r][i][j])$$

[‖] As the operation XOR is not by default implemented in the solver.

for Midori128. We finally model the SB operations. In the model for Midori128, where 4 different Sboxes are used, we use Sbox_i, where i is the number of the line. For readability, the number of the Sbox is omitted in what follows.

The initial SB is modeled as follows: $\forall i \in [0..3], \forall j \in [0..3]$:

$$\mathsf{TABLE}((\delta_{P'}[i][j], \delta_{SB}[0][i][j], P[0][i][j]), \mathsf{tupleSB})$$

Then, for the other rounds, we have $\forall r \in [1..n-1], \forall i \in [0..3], \forall j \in [0..3]$:

$$\mathsf{TABLE}((\delta_{KA}[r-1][i][j], \delta_{SB}[r][i][j], P[r][i][j]), \mathsf{tupleSB})$$

5 Results

Step 1 was implemented in the Minizinc** language, and solved using the solver Chuffed ††, which minimizes the objective function in around 3 hours for Midori64 and in around 10 hours for Midori128‡‡. Step 2 was solved using Choco3 [15], which finds the best related-key differential characteristic (when it exists) for each input from Step 1 within 10 seconds.

The results are given in table 2.

Note that since all Sboxes in our related-key differential characteristics have the best possible probability (2^{-2}), any related-key differential characteristic with more Sboxes has a lower probability.

Version	Number of rounds	Number of Sboxes	Probability
Midori64	15	7	2^{-14}
	16	8	2^{-16}
Midori128	19	19	2^{-38}
	20	20	2^{-40}

Table 2. The results obtained by the solvers, both for full-round and $n-1$ rounds of both versions of Midori. The number of Sboxes is the result of Step 1, and the probability is the result of Step 2.

5.1 Key Recovery Attacks

Our goal is to recover the secret key K. We use an *encryption oracle* $\mathsf{Enc}_K(x, m)$ that encrypts a message m with the key $K \oplus x$. Our attacks are different for Midori64 and Midori128.

** `http://www.minizinc.org/`

†† https://github.com/geoffchu/chuffed

‡‡ We run our experiments on an Intel i7-4790, 3.6 Ghz with 16 GB RAM.

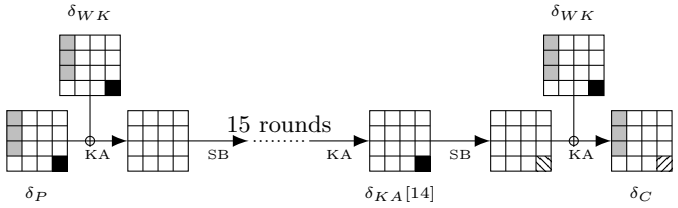

Fig. 3. An example of a related-key differential characteristic provided by the solver.

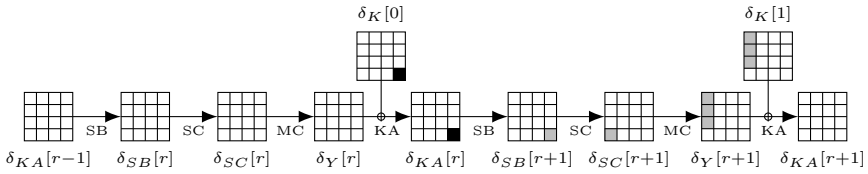

Fig. 4. An optimal 2-rounds related-key differential characteristic from Set 1 for Midori64, where r is an even round. Non-zero differences are represented as black and gray squares. It has 1 active Sbox, for instance with ■ $= 0x1$ and ■ $= 0x2$.

Midori64: For this attack, we first recover WK, one word at a time, using 16 15-rounds related-key differential characteristics with $16 \cdot 2^{19.32} = 2^{23.32}$ operations. Then we use 4 14-rounds related-key differential characteristics to recover $K[0]$ in $2^{35.8}$ operations. By combining them, we obtain $K[1] = K[0] \oplus WK$ and deduce K (composed of $K[0]$ and $K[1]$), for a total complexity of $2^{23.32} + 2^{35.8} \approx 2^{35.8}$.
Recovery of WK: The solvers give 16 different 15-rounds related-key differential characteristics, the corresponding related-key differentials are given in Appendix A. Each of them contains only one non-zero difference at the end of the 15th round (corresponding to $\delta_{KA}[14]$ in Figure 3), all at different positions.

To be complete we need to give all the details our related-key differential characteristics. Minizinc finds optimal 2-rounds patterns with 1 active Sbox in all the odd rounds and none in the even rounds, as described in Figure 4. Then the missing steps in Figure 3 are 7.5 times the characteristic given in Figure 4.

In order to recover one word of WK, we use the corresponding values of δ_K, δ_P and $\delta_{KA}[14]$ given by the related-key differential characteristics[§§]. First we randomly choose some plaintext P, and query the oracle for $C = \mathsf{Enc}_K(0, P)$, and $C^* = \mathsf{Enc}_K(\delta_K, P\delta_P)$. We compute $\delta_C = C \oplus C^*$. We say that δ_C is *valid* iff $\forall i, j \in [0,3], \delta_{KA}[14][i][j] = 0 \Rightarrow \delta_C[i][j] = \delta_{WK}[i][j]$. If δ_C is valid then we compute $\delta_{SB}[i][j] = \delta_C[i][j] \oplus \delta_{WK}[i][j]$ (where (i, j) is the positions of the non-null difference). We now use the fact that for Midori64 the maximum value in the DDT is 4, *i.e.* every valid δ_C yields at most 4 possible values for $SB[i][j]$ (x in Algorithm 1), to obtain four candidates for $WK[i][j] = SB[i][j] \oplus C[i][j]$. By repeating this process several times we can find the right candidate: it is the

[§§]From δ_K which is composed of $\delta_{K[0]}$ and $\delta_{K[1]}$, we can compute $\delta_{WK} = \delta_{K[0]} \oplus \delta_{K[1]}$.

Input: $\delta_K, \delta_P, \forall k \in [0,14]\delta_{KA}[k], i, j$
$P \xleftarrow{\$} \{0, 2^{64}-1\}$;
$C = Enc_K(0, P)$; $C^* = Enc_K(\delta_K, P \oplus \delta_P)$;
$\delta_C = C^* \oplus C$;
if δ_C *is valid* **then**
  $\delta_{SB}[i][j] = \delta_C[i][j] \oplus \delta_{WK}[i][j]$;
  for $\forall x \in DDTS(\delta_{KA}[14][i][j], \delta_{SB}[i][j])$ **do**
    $WK[i][j] = SB[i][j] \oplus C[i][j]$;
    CPT[WK[i][j]]++;
  end
end

Algorithm 1: How to recover WK in Midori64, where $DDTS(a,b) = \{x : SB(x) \oplus SB(x \oplus a) = b\}$, and CPT a table that is initialized to zero and stores the occurences of possible candidates for WK.

word that has the most occurrences[¶¶]. This is formally described in Algorithm 1. This is done 16 times, one for each word of WK.

Complexity analysis of Algorithm 1: Our aim is to determine how many times we need to repeat our attack in order to have the true key. To determine precisely this value denoted T, we follow the approach given in [16]. It uses the signal to noise ratio S/N introduced by Biham in [4]. It is defined as $S/N = \frac{2^k \cdot p}{\alpha \cdot \beta}$, where k is the number of key bits that we want to recover (in our case, $k = 4$ since we aim to recover a word of 4 bits of the key), p is the probability of the related-key differential characteristic (for us $p = 2^{-14}$), α is the number of key candidates suggested for each good pair (using the DDT, we have $\alpha = 4$), and β is the ratio of the pairs that are not discarded. For β we have $2^{-14} + 2^{-60}$ since 2^{-14} is the probability given by the solvers and 2^{-60} corresponds to the false positives, *i.e.*, pairs having the same difference pattern, with 4 bits of undetermined difference. Then we obtain $S/N = \frac{2^k \cdot p}{\alpha \cdot \beta} = \frac{2^4 \cdot 2^{-14}}{4 \cdot (2^{-14} + 2^{-60})} = \frac{2^{-10}}{2^{-12} + 2^{-58}} \approx 4$. We denote by $\mathsf{P_S}$ the probability to obtain the true key. We use the equation (19) of [16], where Φ denote the density probability function of the standard normal distribution, and Φ^{-1} its inverse: $\mathsf{P_S} = \Phi\left(\frac{\sqrt{T \cdot S/N} - \Phi^{-1}(1-2^{-k})}{\sqrt{S/N+1}}\right)$ (19). Then we can obtain $\mathsf{P_S}$ for given values of T, S/N and α. Note that since we repeat the analysis 16 times (one for each word of WK), we need to have ${\mathsf{P_S}}^{16}$ sufficiently large as well. By numerical approximation we obtain $T = 20 \approx 2^{4.32}$, which gives $\mathsf{P_S} > 0.99$, and ${\mathsf{P_S}}^{16} > 0.99$. Hence, using $T \cdot p^{-1}$ plaintext pairs, we recover a key word with a probability greater than 0.99. The corresponding data complexity is then $16 \cdot 2 \cdot 20 \cdot 2^{14} \approx 2^{23.32}$ chosen plaintexts, as well as 1 related key, for each related-key differential characteristic used.

Recovery of $K[0]$: Using WK previously computed thanks to the 15-rounds related-key differential characteristics, we decrypt the last round of Midori and

[¶¶] Indexes of the cell having the maximum values in the tables CPT.

Input: $\delta_K, \delta_P, \forall k \in [0,14]\delta_{KA}[k], i, j, WK$
$P \xleftarrow{\$} \{0, 2^{64}-1\}$;
$C = Enc_K(0, P)$; $C^* = Enc_K(\delta_K, P \oplus \delta_P)$;
$\delta_C = C^* \oplus C$;
if $\delta_C = \delta_{WK}$ **then**
  $SB = C \oplus WK$; $SB^* = C^* \oplus WK^* \oplus \delta_{WK}$;
  $X = Inv_{SB}(SB)$; $X^* = Inv_{SB}(SB^*)$;
  $\delta_X = X^* \oplus X$;
  $\delta_{SB}[14] = Inv_{SC}(Inv_{MC}(\delta_X))$;
  for $\forall x \in DDTS(\delta_{KA}[13], \delta_{SB}[14])$ **do**
    for $\forall u, v, w \in \{0, 2^4\}^3$ **do**
      $K[0][.][j'] = MC[14][.][j'] \oplus KA[.][j']$;
      $CPT[K[0][.][j']] + +$;
    end
  end
end

Algorithm 2: How to recover $K[0]$ in Midori64, where Inv_{SB} (resp. Inv_{MC} and Inv_{SC}) is the inverse of the Sbox function (resp. MC and SC), $\delta_{WK} = \delta_{K[0]} \oplus \delta_{K[1]}$, CPT a table that is initialized to 0 and stores the occurences of possible candidates for $K[0]$, and j' is the index of the column where the 4 possibles values appeared after the MixColumn. The value of j' is deterministic and only depends of the positions i, j.

obtain the state of the 14th round $\delta_{KA}[14]$. Now we use other four 14-rounds related-key differential characteristics outputted by the solvers, one for each column of $K[0]$, the corresponding related-key differentials are given in Appendix B. They have only one active Sbox in the last round and there is a characteristic for each position of the active word. Hence we obtain the value of $\delta_{KA}[13]$. Similarly as in the case of Midori64, we can use the DDT to obtain 4 possibilities for a word of $SB[13]$. In the encryption function of Midori64, we have to apply the ShuffleCell (which does not influence our attack), then MixColumns which propagates the position of the 4 possibles values into different position. Then we need to guess all the remaining values for these 3 words of 4 bits in the column where the value has been shifted after the ShuffleCell. This leads us to a total of $2^4 \cdot 2^4 \cdot 2^4 \cdot 4 = 2^{14}$ possibilities. Each of these possibilities gives a candidate for 4 words of the key $K[0]$ by xoring the obtained column of $MC[13]$ and the corresponding column of $KA[14]$, as described in Algorithm 2.

Complexity analysis of Algorithm 2: This time $k = 4 \times 4 = 16$, $p = 2^{-14}$, $\alpha = 2^{14}$, $\beta = 2^{-14} + 2^{-60}$, then $S/N = \frac{2^k \cdot p}{\alpha \cdot \beta} = \frac{2^{16} \cdot 2^{-14}}{2^{14} \cdot (2^{-14} + 2^{-60})} = \frac{2^{-12}}{2^{-14} + 2^{-58}} \approx 4$. Then by numerical approximation we find $T = 28 \approx 2^{4.8}$, which gives $\mathsf{P_S} > 0.99$, and ${\mathsf{P_S}}^4 > 0.99$. It gives us the time complexity of $2 \cdot 4 \cdot 28 \cdot 2^{14} \cdot 2^{14} = 2^{35.8}$ and a data complexity of $2 \cdot 4 \cdot 28 \cdot 2^{14} = 2^{21.8}$.

Midory128: The constraints programming solvers find 16 patterns similar to the one of Figure 5, each of them having a different position for the active Sbox.

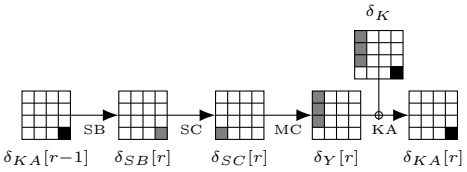

Fig. 5. The optimal 1-round related-key differential characteristic for Midori128. There is 1 active Sbox that can be repeated to cover 19 rounds of Midori128, for instance with ■ = $0x9$ and ■ = $0x1$.

The corresponding related-key differentials are given in Appendix C. Hence, we can build 16 different 19-rounds related-key differential characteristics with 19 active Sboxes each, and each happening with probability 2^{-38}, to recover one word of WK per characteristic. We use a similar technique as for Midori64 for WK, since in Midori128 the key is K exactly WK.

Complexity evaluation: Here, we only need to use 16 related-key differential characteristics, so we want $\mathsf{P_S}^{16} \geq 0.99$. To compute S/N, we use $k = 8$ as we recover a 8-bit word of the key for each related-key differential characteristic, $p = 2^{-38}$, $\alpha = 64$ (according to the DDT), and $\beta = 2^{-38} + 2^{-120}$. With these values, we have $S/N = 4$, and need $T = 25 \approx 2^{4.7}$ to have $\mathsf{P_S}^{16} > 0.99$. Thus, the data complexity of the attack is $2 \cdot 25 \cdot 2^{38} \approx 2^{43.7}$ plaintexts and 16 related keys, and we need $2^{43.7}$ encryptions.

5.2 Related-Key Distinguishers

A related-key distinguisher aims at distinguishing a cipher scheme from a Pseudo-Random Function (PRF) that represents an ideal cipher. We construct two distinguishers for Midori64 and for Midori128.

Midori64: Midori64 has related-key differential characteristics with 8 active Sboxes, all of which can be crossed with the maximal probability 2^{-2}. Following the bound given in [1], we have that only $\frac{\sqrt{2}}{p} = \frac{\sqrt{2}}{2^{-18}} = 2^{18.5}$ are needed to distinguish Midori64 from a PRF, using the distinguisher given in Figure 3. The equivalent complexity to find for a PRF is 2^{26} operations (following the formula given in [10]). Thus, we are able to distinguish Midori64 from a PRF***.

Midori128: As for Midori64, there exist patterns that can be repeated to cover the whole cipher. The optimal ones only contain one active Sbox per round, *e.g.*, Figure 5, hence leading to 20-rounds distinguishers with 20 active Sboxes. Since these Sboxes can be crossed with maximal probability (2^{-2}), the probability of the distinguisher is 2^{-40}. Hence, distinguishing Midori128 from a PRF, using

***In Appendix D, we provide an example of values that satisfy the distinguisher built using the pattern given in Figure 4 with ■ = $0xa$ and ■ = $0xa$.

Input: δ_K, δ_P, δ_C
for $i = 1$ *to* $2^{18.5}$ **do**
  $P \xleftarrow{\$} \{0, 2^{64} - 1\}$;
  $C = Enc_K(0, P)$; $C^* = Enc_K(\delta_K, P \oplus \delta_P)$;
  if $C \oplus C^* = \delta_C$ **then** return 1;
end
return 0;

Algorithm 3: Algorithm for a distinguisher for Midori64.

Algorithm 3, can be done with $\frac{\sqrt{2}}{p} = 2^{39.5}$ encryptions of plaintext pairs whereas the equivalent complexity to find such a structure for a PRF is 2^{52}, again with the formula given in [10].

6 Conclusion

In this paper, we give a practical related-key attack on Midori64, improving the existing key recovery attack from 2^{116} for 14 rounds to $2^{35.8}$ for the full 16 rounds cipher. We also are able to provide the first related-key attack on Midori128 with a complexity of $2^{43.7}$. In order to construct such impressive practical attacks, we model Midori with constraint programming. The constraint programming solvers help us determine the minimal number of active Sboxes in a few hours, and then to derive optimal related-key differential characteristics in a few seconds.

Finally we propose two efficient distinguishers for Midori64 and Midori128. In the future, we aim at exploring how CP can be used to perform some related-key cryptanalysis on other symmetric encryption schemes.

References

1. T. Baignères, P. Sepehrdad, and S. Vaudenay. Distinguishing distributions using chernoff information. In S. Heng and K. Kurosawa, editors, *Provable Security - 4th International Conference, ProvSec 2010, Malacca, Malaysia, October 13-15, 2010. Proceedings*, volume 6402 of *Lecture Notes in Computer Science*, pages 144–165. Springer, 2010.
2. S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni. Midori: A block cipher for low energy. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015.
3. E. Biham. *Advances in Cryptology — EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus, Norway, May 23–27, 1993 Proceedings*, chapter New Types of Cryptanalytic Attacks Using Related Keys, pages 398–409. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

We would like to thank Marine Minier for her valuable advice.

4. E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, London, UK, UK, 1993.
5. A. Biryukov and I. Nikolic. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 322–344. Springer, 2010.
6. Z. Chen and X. Wang. Impossible differential cryptanalysis of midori. *IACR Cryptology ePrint Archive*, 2016:535, 2016.
7. X. Dong. Cryptanalysis of reduced-round midori64 block cipher. Cryptology ePrint Archive, Report 2016/676, 2016. `http://eprint.iacr.org/2016/676`.
8. P.-A. Fouque, J. Jean, and T. Peyrin. Structural evaluation of aes and chosen-key distinguisher of 9-round aes-128. In *Advances in Cryptology - CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 183–203. Springer, 2013.
9. D. Gerault, M. Minier, and C. Solnon. Constraint programming models for chosen key differential cryptanalysis. In *The 22nd Internation Conference on Principles and Practice of Constraint Programming*, Toulouse, France, 2016.
10. H. Gilbert and T. Peyrin. *Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations*, pages 365–383. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
11. J. Guo, J. Jean, I. Nikolić, K. Qiao, Y. Sasaki, and S. M. Sim. Invariant subspace attack against full midori64. Cryptology ePrint Archive, Report 2015/1189, 2015. `http://eprint.iacr.org/`.
12. L. R. Knudsen. *Cryptanalysis of LOKI 91*, pages 196–208. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.
13. L. Lin and W. Wu. Meet-in-the-middle attacks on reduced-round midori-64. Cryptology ePrint Archive, Report 2015/1165, 2015. `http://eprint.iacr.org/`.
14. N. Mouha, Q. Wang, D. Gu, and B. Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Information Security and Cryptology - 7th International Conference, Inscrypt 2011*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.
15. C. Prud'homme, J.-G. Fages, and X. Lorca. *Choco Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016.
16. A. A. Selçuk. On probability of success in linear and differential cryptanalysis. *Journal of Cryptology*, 21(1):131–147, 2008.
17. S. Sun, L. Hu, M. Wang, Q. Yang, K. Qiao, X. Ma, L. Song, and J. Shan. Extending the applicability of the mixed-integer programming technique in automatic differential cryptanalysis. In *Information Security - 18th International Conference, ISC 2015*, volume 9290 of *Lecture Notes in Computer Science*, pages 141–157. Springer, 2015.
18. S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song. Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In *Advances in Cryptology - ASIACRYPT 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.
19. D. J. Wheeler and R. M. Needham. Tea, a tiny encryption algorithm. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings*, pages 363–366, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.
20. ZDNet. New xbox security cracked by linux fans. `http://www.zdnet.com/article/new-xbox-security-cracked-by-linux-fans`.

A 16 Related-Key Differentials for WK for Midori64

In the following table we give the 16 differential found by the solver that we used for recovery WK for Midori64.

n^o	δ_P	δ_K	$\delta_{KA}[14]$
1	1110000000000002	00000000000000021110000000000000	0000000000000002
2	0000110100020000	00000000000200000000110100000000	0000000000020000
3	0000000200000111	00000002000000000000000000000111	0000000200000000
4	0002000010110000	00020000000000000000000010110000	0002000000000000
5	0000022200000010	00000000000000100000022200000000	0000000000000010
6	1011000000200000	00000000002000001011000000000000	0000000000200000
7	0000002011100000	00000020000000000000000011100000	0000002000000000
8	0010000000002202	00100000000000000000000000002202	0010000000000000
9	0000000000001211	00000000000002000000000000001011	0000000000000200
10	0000000003110000	00000000020000000000000001110000	0000000002000000
11	1101020000000000	00000200000000001101000000000000	0000020000000000
12	0100222000000000	01000000000000000000222000000000	0100000000000000
13	0000000011012000	00000000000020000000000011010000	0000000000002000
14	0000000020001110	00000000200000000000000000001110	0000000020000000
15	0000301100000000	00002000000000000000101100000000	0000200000000000
16	2111000000000000	20000000000000000111000000000000	2000000000000000

B 4 Related-Key Differentials for $K[0]$ for Midori64

In the following table we give the 4 related-key differential charateristics found by the solver, that we used for recovery $K[0]$ for Midori64.

n^o	δ_P	δ_K	$\delta_{KA}[13]$
1	0111000000000000	01110000000000001000000000000000	1000000000000000
2	0000101100000000	00001011000000000000100000000000	0000100000000000
3	0000000001110000	00000000011100000000000001000000	0000000001000000
4	0000000000001011	00000000000010110000000000000100	0000000000000100

C 16 Related-Key Differentials for Midori128

In the following table, we give the 16 related-key differential charateristics found by the solver, that we used for recovery K for Midori128. The hexadecimal values of corresponding bytes are separated by a coma for more clarity.

n^o	δ_P	δ_K	$\delta_{KA}[18]$
1	33, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	32, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
2	0, 3, 0, 0, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, 1, 0, 0, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
3	0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1	0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1	0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4	0, 0, 0, 8, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0	0, 0, 0, 9, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0	0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
5	0, 0, 0, 0, 32, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0	0, 0, 0, 0, 33, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0	0, 0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
6	1, 1, 0, 1, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	1, 1, 0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
7	0, 0, 0, 0, 0, 0, 6, 0, 1, 1, 1, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 7, 0, 1, 1, 1, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0
8	0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 9, 9, 9	0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 9, 9, 9	0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0
9	0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 2, 2, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
10	0, 0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0, 6, 0	0, 0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0, 1, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0
11	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 1, 1	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0
12	0, 0, 0, 0, 0, 0, 0, 0, 32, 32, 0, 32, 33, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 32, 32, 0, 32, 1, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0
13	0, 0, 0, 0, 0, 0, 0, 0, 33, 0, 0, 0, 1, 1, 1, 0	0, 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 0, 1, 1, 1, 0	0, 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0
14	0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 8, 0, 0, 0, 0	0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 9, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0
15	1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8	1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9
16	1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0	1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0

D Example of Related-Key Distinguisher for Midori64

In Table 3, we give an example of values that satisfy the related-key distinguisher built using the pattern given in Figure 4 with ■ = $0xa$ and ■ = $0xa$.

	Plaintext (P)	Key (K)	Ciphertext (C)
Y	cdd0776b6777667e	fffefefeefeeffe7 5448eff3eeffffff	99471218a32ea67c
Y^*	6770776b67776674	fffefefeefeeffed fee8eff3eeffffff	33e71218a32ea67c
$\delta_Y = Y \oplus Y^*$	aaa000000000000a	00000000000a aaa000000000	aaa0000000000000

Table 3. A pair of plaintext/key couples following the related key distinguisher on Midori64, where Y can be P, K or C.