Mystery Cipher 1 – Specification

December 2019

Nils Kopal and Christian Bender

CrypTool Project, nils.kopal@cryptool.org, christian.bender@cryptool.org

1 Introduction

Substitution-permutation networks (SPN) are easy to build block ciphers consisting of substitutions, permutations, and XOR-operations. Many modern ciphers, like AES (Daemen and Rijmen, 1999), are SPNs and are built using the aforementioned building blocks. SPNs usually consist of several rounds performing the same steps again and again. For each round, a different round key is used. To decrypt an SPN-encrypted text, the ciphertext is given to the reversed network. Here, the inverse S- and P-boxes are used and the round keys are given in the reversed order.

In this document, we specify the "Mystery Cipher 1" (MC1). MC1 is a 3-round SPN (toy-)cipher with a total key size of 64 bit and a block size of 16 bit. It is intended for educational purposes, e.g. applying modern cryptanalysis techniques to it. We also present this cipher in different MysteryTwister C3 (MTC3 Team, 2019) challenges, for example for attacking it with differential cryptanalysis.

The rest of the document is structured as follows: First, we introduce the notation used in this document. After that, each building block of the cipher is presented. Then, we specify the complete encryption and decryption processes, and give a set of test vectors. After that, we give a set of test vectors. Finally, we present some basic cryptanalysis and, additionally, literature, which is helpful for cryptanalyzing the cipher.

2 Notation

We use the following notation throughout this document:

- Plaintext is P, e.g. a 16-bit plaintext
- Ciphertext is C, e.g. a 16-bit ciphertext
- Key is K, e.g. a 64-bit key
- Round keys K_0, K_1, ..., K_n, e.g. four 16-bit round keys
- XOR is the bit-wise exclusive or of two bit-strings
- S-box is a substitution box
- P-box is a permutation box

3 Building Blocks

MC1 consists of three main building blocks: (1) a 4-bit wide S-box, (2) a 16-bit wide P-box, and (3) an XOR operation of the 16-bit round key with plaintext or ciphertext. In the following, we present each building block in detail.

3.1 S-box (4 bits)

A substitution-box (S-box) substitutes a given bit input sequence to an output bit sequence. The 4-bit S-box takes four bits as input and returns four bits as output. The S-box is the non-linear part of the cipher. It is defined as follows:

```
S-box =
        {
          10, 05, 15, 08,
          11, 00, 03, 07,
          01, 13, 09, 12,
          06, 14, 02, 04
        }
```

The S-box contains 16 different output values for the 16 different input values. For example, if the input of the S-box is 0, the output is 10. Another example: If the input of the S-box is 8, the output of the S-box 1.

Consequently, the reverse S-box is defined as follows:

```
S-box_R =
        {
         05, 08, 14, 06,
         15, 01, 12, 07,
         03, 10, 00, 04,
         11, 09, 13, 02
        }
```

3.2 P-box (16 bits)

The P-box permutates a 16-bit input to a 16-bit output. It is defined as follows:

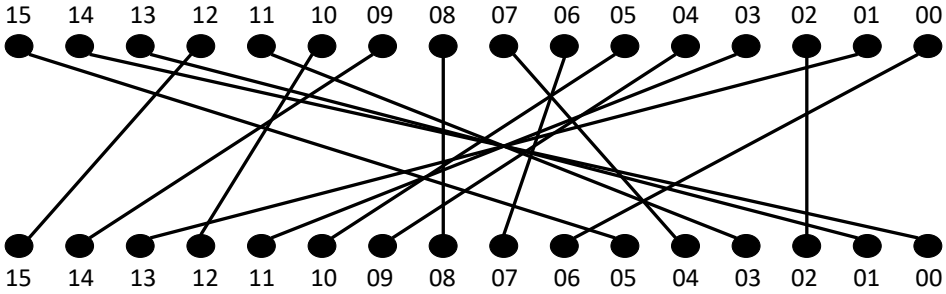

Fig. 1. P-box of MC1

Here, bit 0 is mapped to bit 6, bit 1 is mapped to bit 13, bit 2 is mapped to bit 2, and so on.

Consequently, the reverse P-box can be constructed by reading the P-box from bottom to top. Thus, bit 0 is mapped to bit 14, bit 1 is mapped to bit 13, bit 2 is mapped to bit 2, and so on.

3.3 XOR

The XOR is defined as the exclusive-or of the plain- or ciphertext with a round key. Thus, $C_i = P_i \oplus K_i$ and $P_i = C_i \oplus K_i$.

4 Encryption and Decryption

This section defines the complete cipher based on the aforementioned building blocks.

MC1 takes a 16-bit plaintext block P as input and returns a 16-bit ciphertext block C as output. Furthermore, it uses four 16-bit round keys (K_0, K_1, K_2, and K_3) as input. Thus, a 64-bit key is split into four round keys, e.g. $K = K_0 \| K_1 \| K_2 \| K3$, where $\|$ is the bitwise-concatenation of each of the round keys.

The overall cipher is defined as follows:

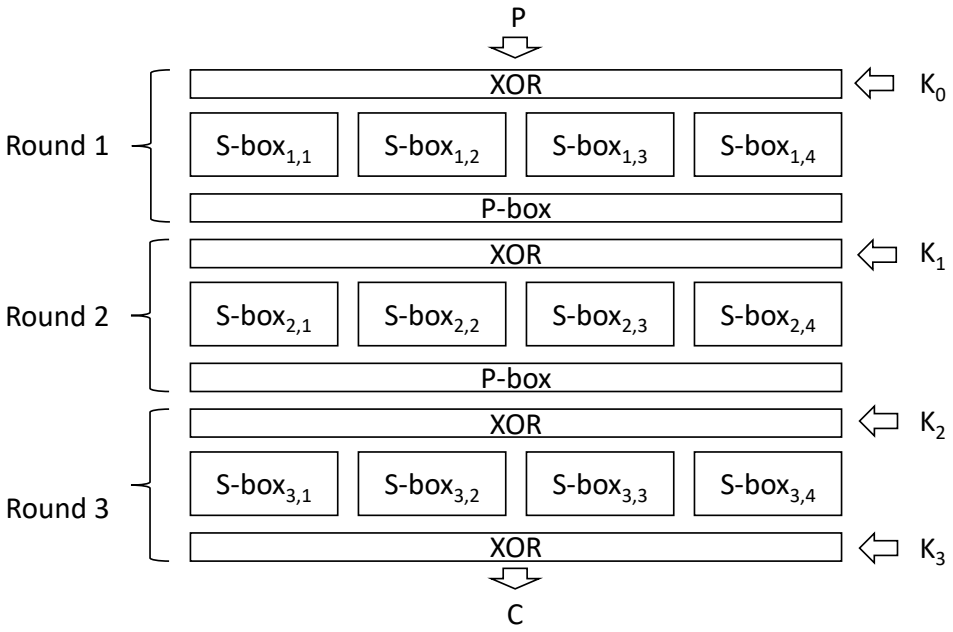

Fig. 2. Mystery Cipher 1

The first two rounds consist of three steps (XOR, substitution, and transposition):

1. XOR: $a_i = C_{i-1} \oplus K_i$, where C_{i-1} is the output of the previous encryption round
2. On each 4-bit part of a_i, we apply the 4-bit S-box in parallel. The result b_i is the bit-wise concatenation of the four S-box outputs
3. Permutation: C_i =P-box(b_i)

In the last round, we omit the transposition but additionally apply a final XOR of the last round key K_3 on C_3. Clearly, the input of the first round is the plaintext P and the output C_3 of the last round XORed with the last round key K_3 is the output ciphertext C.

To decrypt a given text, we use the inverse S-box and the inverse P-box. Also, we apply the round keys in the reversed order.

In our visualization in Figure 2 the S-boxes are indexed with i and j, where i is the number of the round and j is the number of the S-box offset. So, the first S-box is S-box$_{1,1}$, the second S-box is S-box$_{1,2}$, and so on. Regardless of the indexes, the S-boxes are always the same one S-box as defined in Section 3.1.

5 Test Vectors

In this section, we give some test vectors that are intended for testing implementations of the MC1. P, C, and K are given as 16-bit integers (little endian).

```
test vector #1
K: 0, 0, 0, 0
P: 0
C: 57615

test vector #2
K: 0, 0, 0, 0
P: 1500
C: 9548

test vector #3
K: 1, 2, 3, 4
P: 2
C: 60017

test vector #4
K: 1, 2, 3, 4
P: 4
C: 25977

test vector #5
K: 1, 2, 3, 4
P: 16
C: 16634

test vector #6
K: 1, 2, 3, 4
P: 256
C: 37483

test vector #7
```

K: 17, 25, 50, 100
P: 4564
C: 7948

6 Cryptanalysis and Literature

In this last section, we first have a look at the keyspace size of the cipher and discuss a brute-force attack on it. After that, we present literature on differential cryptanalysis useful for analyzing this cipher. Also, we give a hint on an example implementation of differential cryptanalysis.

6.1 Keyspace Size and Brute-Force Attack

MC1 needs four 16-bit round keys, which are based on a single 64-bit key. As a good practice, the key should be selected randomly. Like with every cipher, MC1 can be attacked (ciphertext-only) with an exhaustive keysearching attack aka a brute-force attack. Here, every possible key is tested by decrypting the given ciphertext and then checking, if we obtained valid plaintext, e.g. using the entropy function. The correct plaintext is assumed to have the lowest entropy value. Clearly, we assume that we have enough ciphertext material, thus, using the entropy function is possible. Since we have a 64-bit cipher, a total of $2^{64} = 18,446,744,073,709,551,616$ keys needs to be tested. Thus, if a single computer can test 20 million keys per second, it would search for about 29,247 years. To reduce the duration of this attack to one year, consequently almost 30 thousand computers are needed to work in parallel and all the time. Thus, a brute-force attack on MC1 is not practical feasible.

6.2 Related Work about Differential Cryptanalysis

MC1 is vulnerable to differential cryptanalysis (Biham and Shamir, 1991). Differential cryptanalysis is a chosen-plaintext attack developed by Eli Biham and Adi Shamir in the 1980s. This attack looks at how differences in input blocks affect the resulting difference of output blocks of a cipher. Using a key recovery attack, it is possible to recover all round keys of a block cipher, if it is not resistant to differential cryptanalysis.

Heys developed a tutorial (Heys, 2002) which can be used to attack MC1.

Also, the book "The Block Cipher Companion" (Knudsen and Robshaw, 2011, p. 109 - 126) is useful for implementing differential cryptanalysis on MC1.

The (German) master's thesis (Bender, 2019) describes in detail the differential cryptanalysis of SPNs.

Within his masters thesis, Bender also developed three tutorials on differential cryptanalysis, which are part of the open-source software CrypTool 2 (Kopal et al., 2014). Figure 3 shows the third tutorial on differential cryptanalysis implemented in CrypTool 2[1].

[1] You can download CrypTool 2 for free from https://www.cryptool.org/

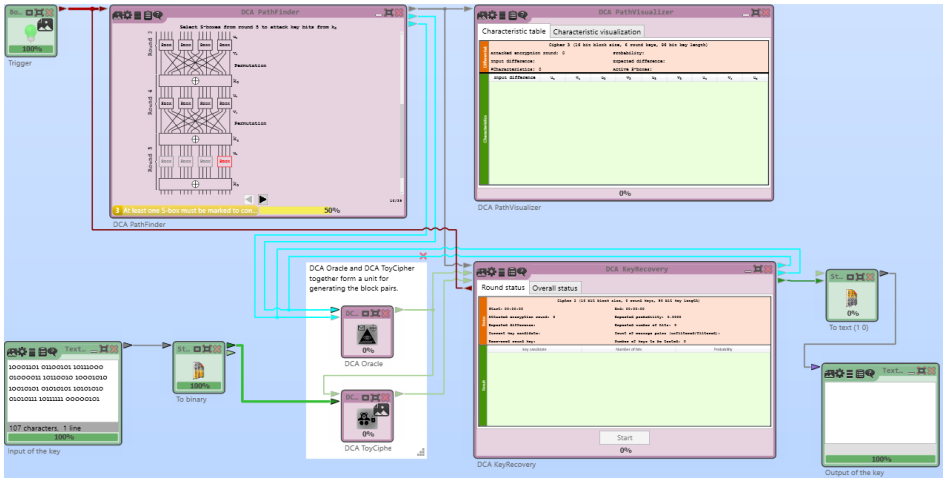

Fig. 3. Screenshot of CrypTool 2 with opened differential cryptanalysis tutorial 3

References

Bender, C. (2019). Master's thesis: Analyse symmetrischer Blockchiffren mittels differenzieller Kryptoanalyse in CrypTool 2. https://www.cryptool.org/en/ctp-education/cryptool-in-scientific-papers.

Biham, E. and Shamir, A. (1991). Differential Cryptanalysis of DES-like Cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72.

Daemen, J. and Rijmen, V. (1999). AES Proposal: Rijndael.

Heys, H. M. (2002). A Tutorial on Linear and Differential Cryptanalysis. *Cryptologia*, 26(3):189–221.

Knudsen, L. R. and Robshaw, M. (2011). *The Block Cipher Companion*. Springer Science & Business Media.

Kopal, N., Kieselmann, O., Wacker, A., and Esslinger, B. (2014). CrypTool 2.0. *Datenschutz und Datensicherheit-DuD*, 38(10):701–708.

MTC3 Team (2019). MysteryTwister C3, The Crypto Challenge Contest. https://www.mysterytwisterc3.org/.