ANALYSIS OF THE DVB COMMON SCRAMBLING ALGORITHM

Ralf-Philipp Weinmann and Kai Wirt
Technical University of Darmstadt
Department of Computer Science
Darmstadt, Germany
{weinmann,kwirt}@cdc.informatik.tu-darmstadt.de

Abstract The Common Scrambling Algorithm (CSA) is used to encrypt streams of video data in the Digital Video Broadcasting (DVB) system. The algorithm cascades a stream and a block cipher, apparently for a larger security margin. In this paper we set out to analyze the block cipher and the stream cipher separately and give an overview of how they interact with each other. We present a practical attack on the stream cipher. Research on the block cipher so far indicates it to be resistant against linear and algebraic cryptanalysis as well as simple slide attacks.

Keywords: Block cipher, stream cipher, cryptanalysis, DVB, pay-tv

1. Introduction

The DVB Common Scrambling Algorithm is an ETSI-specified algorithm for securing MPEG-2 transport streams such as those used for digitally transmitted Pay-TV. It was adopted by the DVB consortium in May 1994, the exact origin and date of the design is unclear. Until 2002, the algorithm was only available under a Non-Disclosure Agreement from an ETSI custodian. This NDA disallowed and still disallows licensees to implement the algorithm in software for “security reasons”. The little information that was then available to the public is contained in an ETSI Technical Report [European Telecommunications Standards Institute, 1996] and patent applications [Bewick, 1998], [Watts et al., 1998]. This changed in the Fall of 2002, when a Windows program called FreeDec appeared which implemented the CSA in software. It was quickly reverse-engineered and details were disseminated on a website [Pseudonymous authors, 2003].

For keying the CSA, so called control words are used. These control words are provided by a conditional access mechanism, which generates them from encrypted control messages embedded in the transport stream. Conditional access mechanisms vary between broadcasters and can be more easily changed than the actual scrambling algorithm. Ex-
amples for commonly used conditional access mechanisms are Irdeto, Betacrypt, Nagravision, CryptoWorks etc. A new common key is usually issued every 10–120 seconds. The great relevance of CSA lies in the fact that every encrypted digital Pay-TV transmission in Europe is secured using this algorithm. A practical break of CSA would thus affect all broadcasters and could not be remedied by changing the conditional access mechanism.

The scrambling algorithm can be seen as the layering of two cryptographic primitives: a 64-bit block cipher and a stream cipher. Both ciphers employ a common key; the stream cipher uses an additional 64-bit nonce, the origin of which we will discuss later.

In this paper we investigate the two ciphers independently and show weaknesses. Although we do not present a break of the scrambling algorithm we present a known-plaintext attack on the stream cipher and show preliminary results on the block cipher.

The rest of this paper is organized as follows: Section 2 defines the notation used. In Section 3 we describe the two ciphers and how they are combined in the CSA. Our attack on the stream cipher as well as a presentation of properties of the block cipher follow in Sections 4 respectively 5. Section 6 concludes.

## 2. Definitions

In the rest of this paper we use the following notation:

- \( K \)  the common key. A 64 bit key used for both the stream and the block cipher
- \( k_i \) denotes the \( i \)-th bit of \( K \)
- \( K^E \) denotes the expanded key which is derived through the key schedule of the block cipher
- \( SB_i \) is the \( i \)-th 8-byte block of the scrambled data stream
- \( SB_0 \) is used as nonce in the stream cipher
- \( CB_i \) is the \( i \)-th 8-byte block of stream cipher output
- \( IB_i \) intermediate blocks. See Figure 3.1 for details.
- \( DB_i \) is the \( i \)-th 8-byte block of descrambled data
- \( R \) denotes the residue and
- \( SR \) is used for the scrambled residue
- \( IV \) an initialization vector. Always equals the zero block.
- \( \text{rol} \) bitwise rotation to the left by one bit
- \( \| \) denotes concatenation
- \( t_i \) state of the stream cipher after \( i \) clocks
- \( t_{-31} \) is the starting state, \( t_0 \) the state after the initialization
- \( I^A \) is an additional input for the stream cipher generated from the nonce
- \( t_n \) denotes the cycle length, i.e. the smallest number \( j - i \) for which \( t_j = t_i \)
- \( t_s \) is the length of a small cycle, i.e. the smallest number \( j - i \) for which the feedback shift register 1 has the same value in \( t_j \) and \( t_i \)
3. Description

3.1 Cascading the Block and the Stream Cipher

The scrambling algorithm can be seen as a cascade of a block cipher and a stream cipher. Both ciphers use the same 64-bit key $K$, which is called the common key. We will now describe how the block and the stream cipher are combined. Figure 3.1 depicts the descrambling process.

For scrambling the payload of an $m$-byte packet, it is divided into blocks $(DB_i)$ of 8 bytes each. If an adaption field was used, it is possible that the length of the packet is not a multiple of 8 bytes. Thus the last block is $n < 8$ bytes long and shall be called residue.

The sequence of 8-byte blocks is encrypted in reverse order with the block cipher in CBC mode, whereas the residue is left untouched. The last output of the chain $IB_0$ is then used as a nonce for the stream cipher. The first $m$ bytes of keystream generated by the stream cipher are XORed to the encrypted blocks $(IB_i)_{i \geq 1}$ followed by the residue.

3.2 The Stream Cipher

3.2.1 Overview. The stream cipher is built of two feedback-shift-registers and a combiner with memory. The overall layout is shown in Figure 3.2.1. The registers $p$, $q$ and $c$ are bit registers. All other registers are 4 bit wide.

The stream cipher operates in one of two modes. The first one is the initialization mode in which the starting state of the cipher is set up. The second one is the generating mode in which the cipher produces two pseudo-random bits per clock cycle.
3.2.2 **Key Schedule.** The cipher uses the common key $K$ and the first scrambled block of the transport stream $SB_0$ as a nonce to set up the initial state. At first all registers of the cipher are set to 0. Then the common key $K = k_0, \ldots, k_{63}$ is loaded into the shift registers $A := a_{0,j}, \ldots, a_{9,j}$ and $B := b_{0,j}, \ldots, b_{9,j}$ with $0 \leq j \leq 3$ according to the following rule:

\[
a_{i,j} = \begin{cases} k_{4i+j} & i \leq 7 \\ 0 & \text{else} \end{cases} \quad \quad b_{i,j} = \begin{cases} k_{32+4i+j} & i \leq 7 \\ 0 & \text{else} \end{cases}
\]

In the following $a_i$ and $b_i$ denote the 20 4-bit registers $a_{i,0}, \ldots, a_{i,3}$ and $b_{i,0}, \ldots, b_{i,3}$ respectively.

Hereafter the cipher is in initialization mode. It uses $SB_0$ and the feedback register $D$ as input and performs 32 clock cycles to calculate the starting state. The inputs for feedback shift registers 1 and 2 are derived from $SB_0$:

\[
(I^A, I^B) := \begin{cases} (SB_0 \div 2^4, SB_0 \mod 2^4) & \text{in state } t_i, i \in \{-31, -29, \ldots\} \\ (SB_0 \mod 2^4, SB_0 \div 2^4) & \text{else} \end{cases}
\]

Thus in every odd cycle number $I^A$ is the high nibble of $SB_0$ whereas $I^B$ is the low nibble. In even cycles the nibbles are used the other way.
round. See below for the equations which update the internal cipher state.

### 3.2.3 Generation Mode.

**Feedback shift register 1.** The feedback $a'_0$ of shift register $A$ is calculated as

$$
a'_0 := \begin{cases} 
a_9 \oplus X & \text{if not in init mode} \\
a_9 \oplus X \oplus D \oplus I^A & \text{else} 
\end{cases}
$$

The next value $A'$ for register $A$ is then given by

$$A' := (a'_0, a_0, \ldots, a_8)$$

**Feedback shift register 2.** The feedback $b'_0$ of shift register $B$ is given by

$$b'_0 := \begin{cases} 
b_0 \oplus b_9 \oplus Y & \text{if not in init mode} \\
b_0 \oplus b_0 \oplus Y \oplus I^B & \text{else} 
\end{cases}
$$

and the new value $B'$ for $B$ is

$$B' := \begin{cases} 
(b'_0, b_0, \ldots, b_8) & p = 0 \\
(\text{rol}(b'_0), b_0, \ldots, b_8) & \text{else}
\end{cases}
$$

**Other registers.** New values for the other registers, namely $X$, $Y$, $Z$, $p$ and $q$ are derived from seven $5 \times 2$ S-Boxes. Table 1 shows which bits from shift-register $A$ are used as input for the S-Boxes and how the new register values are constructed. The S-Boxes itself are shown in Table 4. Table 6 gives an algebraic description of the S-Boxes, with $a$ being the most significant input bit and $c$ the least significant.

**Combiner.** The stream cipher uses a combiner with memory to calculate two bits of output per clock. The memory of the combiner consists of registers $E$, $F$ and $c$. In each cycle a new state for these registers is determined according to

$$(E, F)' := \begin{cases} 
(F, E) & q = 0 \\
(F, E + Z + c \mod 2^4) & \text{else}
\end{cases}
$$

c is unchanged if $q = 0$. Otherwise it is 1 if $E + Z + c \geq 2^4$ and 0 else.

The output of the generator is calculated by $D_2 \oplus D_3 || D_0 \oplus D_1$ where $D := E \oplus Z \oplus B^{\text{out}}$ with $B^{\text{out}}$ given by

- $B_9^{\text{out}} := b_{9,0} \oplus b_{9,1} \oplus b_{9,2} \oplus b_{9,3}$
- $B_8^{\text{out}} := b_{8,0} \oplus b_{8,1} \oplus b_{8,2} \oplus b_{8,3}$
- $B_7^{\text{out}} := b_{7,0} \oplus b_{7,1} \oplus b_{7,2} \oplus b_{7,3}$
- $B_6^{\text{out}} := b_{6,0} \oplus b_{6,1} \oplus b_{6,2} \oplus b_{6,3}$
- $B_5^{\text{out}} := b_{5,0} \oplus b_{5,1} \oplus b_{5,2} \oplus b_{5,3}$
- $B_4^{\text{out}} := b_{4,0} \oplus b_{4,1} \oplus b_{4,2} \oplus b_{4,3}$
- $B_3^{\text{out}} := b_{3,0} \oplus b_{3,1} \oplus b_{3,2} \oplus b_{3,3}$
- $B_2^{\text{out}} := b_{2,0} \oplus b_{2,1} \oplus b_{2,2} \oplus b_{2,3}$
- $B_1^{\text{out}} := b_{1,0} \oplus b_{1,1} \oplus b_{1,2} \oplus b_{1,3}$
- $B_0^{\text{out}} := b_{0,0} \oplus b_{0,1} \oplus b_{0,2} \oplus b_{0,3}$
3.3 The Block Cipher

CSA employs an iterated block cipher that operates bytewise on 64-bit blocks of data and uses a 64-bit key, the common key $K$. Each round of the cipher employs the same round transformation $\phi$, which takes an 8-byte vector along with a single byte of the expanded key as input and outputs an 8-byte vector. This round transformation is applied 56 times. One could also lump together 8 successive rounds of the cipher into a round function $\phi'$ and describe a 7-round cipher which uses 64-bit subkeys; however we feel that the description we give below is more natural and easier to comprehend.

3.3.1 The Key Schedule. Let $\rho$ be the bit permutation on 64-bit strings which is defined in Table 2. The expanded key $K^E = (k_0^E, \ldots, k_{447}^E)$ consists of a total of 448 bits which are recursively computed as follows:

$$k_{0,\ldots,63}^E = k_{0,\ldots,63}$$

$$k_{64i,\ldots,64i+63}^E = \rho(k_{64(i-1),\ldots,64i-1}^E) \oplus 0x0i0i0i0i0i0i0i0i \quad 1 \leq i \leq 6$$

where the expression $0x0i0i0i0i0i0i0i$ is to be interpreted as a hexadecimal constant. We note that the key schedule is entirely $GF(2)$-linear.

3.3.2 The Round Function. At the core of the round transformation $\phi$ are the nonlinear functions $f$ and $f'$. These are distinct

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
$i$ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
\hline
$\rho(i)$ & 17 & 35 & 8 & 6 & 41 & 48 & 28 & 20 & 27 & 53 & 61 & 49 & 18 & 32 & 58 & 63 \\
\hline
$\rho(i)$ & 23 & 19 & 36 & 38 & 1 & 52 & 26 & 0 & 33 & 3 & 12 & 13 & 56 & 39 & 25 & 40 \\
\hline
$i$ & 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 \\
$\rho(i)$ & 50 & 34 & 51 & 11 & 21 & 47 & 29 & 57 & 44 & 30 & 7 & 24 & 22 & 46 & 60 & 16 \\
\hline
$i$ & 48 & 49 & 50 & 51 & 52 & 53 & 54 & 55 & 56 & 57 & 58 & 59 & 60 & 61 & 62 & 63 \\
$\rho(i)$ & 59 & 4 & 55 & 42 & 10 & 5 & 9 & 43 & 31 & 62 & 45 & 14 & 2 & 37 & 15 & 54 \\
\hline
\end{tabular}
\caption{S-Box input and generation of new register values.}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
$S_1$ & $a_{3,0}$ & $a_{0,2}$ & $a_{5,1}$ & $a_{6,3}$ & $a_{8,0}$ \\
$S_2$ & $a_{1,1}$ & $a_{2,2}$ & $a_{5,3}$ & $a_{6,0}$ & $a_{4,1}$ \\
$S_3$ & $a_{0,3}$ & $a_{1,0}$ & $a_{4,1}$ & $a_{5,2}$ & $a_{7,0}$ \\
$S_4$ & $a_{2,3}$ & $a_{0,1}$ & $a_{1,3}$ & $a_{3,2}$ & $a_{7,0}$ \\
$S_5$ & $a_{4,2}$ & $a_{3,2}$ & $a_{5,0}$ & $a_{7,1}$ & $a_{8,2}$ \\
$S_6$ & $a_{2,1}$ & $a_{3,1}$ & $a_{4,0}$ & $a_{6,2}$ & $a_{8,3}$ \\
$S_7$ & $a_{1,2}$ & $a_{2,0}$ & $a_{6,1}$ & $a_{7,2}$ & $a_{7,3}$ \\
\hline
$X$ & $S_4$ & $S_3$ & $S_2$ & $S_1$ & $S_0$ \\
$Y$ & $S_4$ & $S_3$ & $S_2$ & $S_1$ & $S_0$ \\
$Z$ & $S_2$ & $S_1$ & $S_0$ & $S_{i,1}$ & $S_{i,0}$ \\
p & $S_7$ & $S_6$ & $S_5$ & $S_4$ & $S_3$ \\
q & $S_0$ & $S_1$ & $S_2$ & $S_3$ & $S_4$ \\
\hline
\end{tabular}
\caption{Key bit permutation.}
\end{table}
permutations on the set of all byte values and can be seen as the S-Boxes of the cipher. Both permutations have maximum cycle length and are related to each other by a bit permutation \( \sigma \), i.e. \( f' = \sigma \circ f \). This bit permutation maps bit 0 to 1, bit 1 to 7, bit 2 to 5, bit 3 to 4, bit 4 to 2, bit 5 to 6, bit 6 to 0 and bit 7 to 3. See Table 5 for the actual values described by \( f \).

Let \( S = (s_0, \ldots, s_7) \) be the vector of bytes representing the internal state of the block cipher in an arbitrary round. The function \( \phi \) taking the internal state \( S \) from round \( i \) to round \( i + 1 \) can then be defined as

\[
\phi(s_0, \ldots, s_7, k) = (s_1, s_2 \oplus s_0, s_3 \oplus s_0, s_4 \oplus s_0, s_5, s_6 \oplus f'(k \oplus s_7), s_7, s_0 \oplus f(k \oplus s_7))
\]

whereas for decrypting a block of ciphertext we need the inverse function:

\[
\phi^{-1}(s_0, \ldots, s_7, k) = (s_7 \oplus f(s_6 \oplus k), s_0, s_7 \oplus s_1 \oplus f(s_6 \oplus k), s_7 \oplus s_2 \oplus f(s_6 \oplus k), s_7 \oplus s_3 \oplus f(s_6 \oplus k), s_4, s_5 \oplus f'(s_6 \oplus k), s_6)
\]

3.3.3 Encryption/Decryption. Encrypting a plaintext \( P = (p_0, \ldots, p_7) \) is accomplished by

\[
\begin{align*}
S^0 &= P \\
S^r &= \phi(S^{r-1}, (k^E_{8r}, \ldots, k^E_{8r+7})) & 1 \leq r \leq 56 \\
C &= S^{56}
\end{align*}
\]

which yields the ciphertext \( C = (c_0, \ldots, c_7) \). For decrypting this ciphertext the following sequence of operations needs to be carried out:

\[
\begin{align*}
S^0 &= C \\
S^r &= \phi(S^{r-1}, (k^E_{448-8r}, \ldots, k^E_{455-8r})) & 1 \leq r \leq 56 \\
P &= S^{56}
\end{align*}
\]

4. Analysis of the Stream Cipher

In the following we denote with \( t_0 \) the stream cipher’s state after the initialization. That means \( t_{-31} \) is the initial state, in which the common key is loaded in the registers \( A \) and \( B \) respectively. Given this notation we define a full cycle to be the smallest number \( l_w := j - i \) for which the values of all registers in state \( t_i \) are equal to the values in \( t_j \). Also we define a small cycle to be the smallest number \( l_s := j - i \) when the values of \( X \) and \( A \) in state \( t_i \) are equal to the values in \( t_j \).

4.1 Observation

The CSA stream cipher’s state consists of 103 bits. This means that the maximum period length is \( 2^{103} \). For cryptographic purposes, one
would expect the cycle to go through a minimum of $2^{80}$ states. Using Floyd’s cycle-finding algorithm however, we observed that after a relatively short preperiod there exist only a few different cycle lengths for different key/nonce combinations; all of these have a length of $l_w < 10^9$, which of course is much smaller than $2^{80}$. When comparing the set of states in several cycles with the same length which where generated by different key/nonce pairs, one notices that these are disjunct; many different cycles with length $l_w$ exist.

On the other hand, taking only $A$ and $X$ in account shows that if two cycles have the same length $l_w$ then $l_s$ is equal too. Moreover the sequence of states in feedback-shift-register 1 is equal. This means that if $l_w$ is equal for two cycles then the registers $A$ and $X$ for these cycles are going through the same values.

We conducted a total of $10^5$ experiments with random key/nonce pairs to determine the most probable period lengths for the state transition function operating on register $A$. Table 3 shows some small cycle lengths $l_s$ together with the number of times $n(l_s)$ we observed a cycle of this length in our test and $a(l_s)$ the average length of the pre-period for a given cycle length.

<table>
<thead>
<tr>
<th>$n(l_s)$</th>
<th>$l_s$</th>
<th>$a(l_s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>36106</td>
<td>22778</td>
<td>152854.6</td>
</tr>
<tr>
<td>24196</td>
<td>97494</td>
<td>83098.3</td>
</tr>
<tr>
<td>18054</td>
<td>121992</td>
<td>27726.2</td>
</tr>
<tr>
<td>15171</td>
<td>42604</td>
<td>65556.8</td>
</tr>
<tr>
<td>3244</td>
<td>25802</td>
<td>17643.8</td>
</tr>
<tr>
<td>1495</td>
<td>108</td>
<td>21051.6</td>
</tr>
<tr>
<td>131</td>
<td>2391</td>
<td>3138.5</td>
</tr>
</tbody>
</table>

In $1.6\%$ of all cases we observed cycle lengths not listed in the above table. For each of these the probability of occurrence must be lower than $0.2\%$. This observation leads to the following attack:

1. Calculate a table $T$ with the states of the small cycles
2. for every state in $T$ do
3. Test if the state is correct
4. Reconstruct the remaining registers
5. end for

It remains to show how one can determine if the state is correct and how the remaining registers can be reconstructed.

### 4.2 Finding the Correct Value for FSR1

The trivial method of finding the correct value for FSR1 is to simply try all possible values. That means that one searches through all states which belong to one of the small cycles. Summing up the number of
states in Table 3 shows that in 98.4% of all cases testing 313,169 possibilities is sufficient; this is far less than the $2^{44}$ possible values for $A$ and $X$.

4.3 Reconstructing the Remaining Registers

The stream cipher’s output is calculated by XORing $Z$, $E$ and $B_{out}$. Since we can now consider $A$ to be known, $Z$ is fully determined. For all possible $2^4$ values of $E$, $F$ and $c$ do the following:

Consider all bits of $B$ at clock cycle $t$ as variables with values in $GF(2)$. Generate a system of equations describing the two output bits at clock cycle $t + k$ as linear equations of bits of these variables. This system is linear since the additional inputs for the feedback shift register are fully determined by $A$ and hence are known. In other words: for every state of $A$ a system of linear equations that fully describes $B$ with respect to $B_{out}$ exists. Therefore this system can be efficiently solved using Gaussian elimination. If the system is inconsistent then the guess for $E$, $F$ and $c$ was wrong and has to be altered.

The last step of the attack is to determine which of the possible solutions for the linear equations system is the correct one. This has to be done because different values for $E$, $F$ and $c$ may lead to a solution of the system. The correct value can be determined simply by running the keystream generator with the calculated state and checking if the output corresponds to the actual output of the generator.

4.4 Results

Some of the generated equations are linearly dependent. Experimentally we derived that for finding a unique solution to the system described above, 60 equations are sufficient.

For carrying out the attack one thus needs to solve approximately $2^{19} \cdot 2^9 = 2^{28}$ systems of linear equations, each of which contains 60 equations in 40 unknowns. Experiments showed that this can be done in less than an hour on a 1.25 GHz PowerPC G4. We stress that our attack leaves much room for improvement. It might be possible to increase our chances at guessing the correct value for $A$ from statistical deviation in the output of the stream cipher. But already our unoptimized version shows that the stream cipher can be broken in a very short time. Also, this attack is well suited to parallelization.

5. Analysis of the Block Cipher

We note that the round function $\phi$ is a weak permutation. Given the inputs $x_1, x_2$ and outputs $y_1 = \phi(x_1, k)$ and $y_2 = \phi(x_2, k)$ of a single round it is trivial to determine the round subkey $k$. The key
5.1 Linear Approximation of the S-Boxes

The maximum bias of both S-Boxes is $\frac{17}{128}$. Trying to find a linear path through several rounds of the cipher we see that the number of active S-Boxes in the path increases exponentially in the number of rounds. Because of this fact and the high number of rounds, the authors believe that classical linear cryptanalysis poses not threat to the cipher.

5.2 Polynomial Interpolation of the S-Boxes

We have interpolated the S-Boxes as polynomials over fields $GF(2^8) = GF(2)[X]/m(X)$ for all $m \in GF(2)[X]$ with $\deg(m) = 8$ and $m$ irreducible. The resulting polynomials are all dense and of maximum degree. Interpolating bit traces of the S-Boxes results in polynomials consisting of 117–137 terms. Two of them are of degree 8, the other 6 of degree 7.

Thus we conclude that both representations are not useful for algebraic cryptanalysis of the cipher.

6. Conclusion

In this paper we described the Common Scrambling Algorithm and presented an analysis of the underlying stream and block cipher parts. We showed that the stream cipher is weak and can be efficiently broken. We also pointed out some properties of the block cipher which eventually could be used in an successful attack. However, since the block cipher uses 56 rounds we believe that such an attack would have to use sophisticated techniques.

Cryptanalyzing both stream and block cipher at the same time seems to be a task too daunting to attempt. Finding special cases where plaintext and corresponding ciphertext can be obtained that is encrypted with only one of the ciphers facilitates easier cryptanalysis. For the stream cipher these are packets with a residue. A sufficiently long adaptation field on the other hand can lead to packets which are only protected by the block cipher.

We believe that extending the attack on the stream cipher to a key recovery is not a trivial task. Since the state update function of the stream cipher is irreversible and nonlinear, the only option we see at this point for recovering the key is to solve a large system of polynomial equations for different nonces and key streams. The nonlinear equations in this system are of the form seen in Table 6.

There are various directions for future research on these topics. First of all the attack presented offers room for further improvements like the reduction of the necessary register guesses. Investigating how to recover
the Common Key $K$ from a known state of the stream cipher is another logical step. Finally, the block cipher needs more scrutiny.

7. Appendix

Table 4. S-Boxes of the stream cipher.

<table>
<thead>
<tr>
<th>Input</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
<th>$S_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>00001</td>
<td>00</td>
<td>01</td>
<td>00</td>
<td>01</td>
<td>00</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>00010</td>
<td>01</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>00</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00011</td>
<td>01</td>
<td>10</td>
<td>10</td>
<td>00</td>
<td>01</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>00100</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>00</td>
<td>11</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>00101</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>00</td>
<td>11</td>
<td>00</td>
</tr>
<tr>
<td>00110</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>00111</td>
<td>00</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>10</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01000</td>
<td>11</td>
<td>01</td>
<td>01</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>01001</td>
<td>10</td>
<td>11</td>
<td>01</td>
<td>10</td>
<td>01</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01010</td>
<td>10</td>
<td>10</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>11</td>
<td>00</td>
</tr>
<tr>
<td>01011</td>
<td>00</td>
<td>01</td>
<td>11</td>
<td>01</td>
<td>01</td>
<td>11</td>
<td>00</td>
</tr>
<tr>
<td>01100</td>
<td>01</td>
<td>00</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>01101</td>
<td>01</td>
<td>01</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>01110</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>00</td>
<td>10</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>01111</td>
<td>11</td>
<td>10</td>
<td>00</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>01</td>
</tr>
</tbody>
</table>

Table 5. S-Box of the block cipher. Output arranged row-wise; lower nibble on horizontal, upper on vertical.

<table>
<thead>
<tr>
<th>0x00</th>
<th>0x01</th>
<th>0x02</th>
<th>0x03</th>
<th>0x04</th>
<th>0x05</th>
<th>0x06</th>
<th>0x07</th>
<th>0x08</th>
<th>0x09</th>
<th>0x0A</th>
<th>0x0B</th>
<th>0x0C</th>
<th>0x0D</th>
<th>0x0E</th>
<th>0x0F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>3A</td>
<td>0E</td>
<td>66</td>
<td>eF</td>
<td>eE</td>
<td>30</td>
<td>e9</td>
<td>09</td>
<td>88</td>
<td>01</td>
<td>83</td>
<td>0C</td>
<td>8D</td>
<td>e1</td>
<td>7F</td>
</tr>
<tr>
<td>0x10</td>
<td>e9</td>
<td>61</td>
<td>95</td>
<td>0C</td>
<td>36</td>
<td>e5</td>
<td>70</td>
<td>42</td>
<td>05</td>
<td>82</td>
<td>06</td>
<td>7C</td>
<td>17</td>
<td>5A</td>
<td>30</td>
</tr>
<tr>
<td>0x20</td>
<td>7A</td>
<td>06</td>
<td>47</td>
<td>eC</td>
<td>10</td>
<td>51</td>
<td>8F</td>
<td>0D</td>
<td>3C</td>
<td>0C</td>
<td>5B</td>
<td>67</td>
<td>0D</td>
<td>8C</td>
<td>1B</td>
</tr>
<tr>
<td>0x30</td>
<td>69</td>
<td>eF</td>
<td>0F</td>
<td>03</td>
<td>4E</td>
<td>48</td>
<td>84</td>
<td>3F</td>
<td>84</td>
<td>10</td>
<td>04</td>
<td>0C</td>
<td>5F</td>
<td>5C</td>
<td>0C</td>
</tr>
<tr>
<td>0x40</td>
<td>16</td>
<td>0A</td>
<td>0C</td>
<td>4C</td>
<td>eF</td>
<td>42</td>
<td>0E</td>
<td>4A</td>
<td>0C</td>
<td>55</td>
<td>66</td>
<td>e3</td>
<td>39</td>
<td>93</td>
<td>3E</td>
</tr>
<tr>
<td>0x50</td>
<td>7C</td>
<td>19</td>
<td>42</td>
<td>79</td>
<td>0D</td>
<td>eE</td>
<td>96</td>
<td>0F</td>
<td>80</td>
<td>0E</td>
<td>80</td>
<td>85</td>
<td>63</td>
<td>45</td>
<td>6E</td>
</tr>
<tr>
<td>0x60</td>
<td>75</td>
<td>0A</td>
<td>0A</td>
<td>9A</td>
<td>13</td>
<td>2A</td>
<td>9D</td>
<td>C2</td>
<td>5E</td>
<td>05</td>
<td>5A</td>
<td>1F</td>
<td>32</td>
<td>35</td>
<td>0C</td>
</tr>
<tr>
<td>0x70</td>
<td>29</td>
<td>03</td>
<td>30</td>
<td>0E</td>
<td>79</td>
<td>02</td>
<td>87</td>
<td>01</td>
<td>8B</td>
<td>28</td>
<td>48</td>
<td>05</td>
<td>57</td>
<td>97</td>
<td>04</td>
</tr>
<tr>
<td>0x80</td>
<td>0B</td>
<td>0C</td>
<td>34</td>
<td>CD</td>
<td>8B</td>
<td>04</td>
<td>40</td>
<td>25</td>
<td>0A</td>
<td>0C</td>
<td>76</td>
<td>23</td>
<td>0C</td>
<td>D9</td>
<td>00</td>
</tr>
<tr>
<td>0x90</td>
<td>89</td>
<td>09</td>
<td>0E</td>
<td>4D</td>
<td>09</td>
<td>0A</td>
<td>8D</td>
<td>6F</td>
<td>05</td>
<td>3E</td>
<td>26</td>
<td>47</td>
<td>74</td>
<td>86</td>
<td>6F</td>
</tr>
<tr>
<td>0xA0</td>
<td>8E</td>
<td>50</td>
<td>37</td>
<td>01</td>
<td>28</td>
<td>08</td>
<td>75</td>
<td>0D</td>
<td>17</td>
<td>07</td>
<td>24</td>
<td>08</td>
<td>BF</td>
<td>0F</td>
<td>0B</td>
</tr>
<tr>
<td>0xB0</td>
<td>80</td>
<td>09</td>
<td>21</td>
<td>01</td>
<td>76</td>
<td>91</td>
<td>17</td>
<td>0D</td>
<td>80</td>
<td>0F</td>
<td>0C</td>
<td>80</td>
<td>0F</td>
<td>0A</td>
<td>0F</td>
</tr>
<tr>
<td>0xC0</td>
<td>8D</td>
<td>1F</td>
<td>0E</td>
<td>08</td>
<td>9B</td>
<td>02</td>
<td>20</td>
<td>01</td>
<td>0D</td>
<td>0E</td>
<td>0D</td>
<td>0B</td>
<td>08</td>
<td>7B</td>
<td>59</td>
</tr>
<tr>
<td>0xD0</td>
<td>8B</td>
<td>09</td>
<td>26</td>
<td>8C</td>
<td>03</td>
<td>43</td>
<td>02</td>
<td>72</td>
<td>07</td>
<td>0F</td>
<td>02</td>
<td>80</td>
<td>14</td>
<td>04</td>
<td>55</td>
</tr>
<tr>
<td>0xE0</td>
<td>AD</td>
<td>0F</td>
<td>46</td>
<td>03</td>
<td>0A</td>
<td>08</td>
<td>02</td>
<td>52</td>
<td>02</td>
<td>0C</td>
<td>0A</td>
<td>08</td>
<td>08</td>
<td>08</td>
<td>06</td>
</tr>
<tr>
<td>0xF0</td>
<td>AD</td>
<td>0F</td>
<td>46</td>
<td>03</td>
<td>0A</td>
<td>08</td>
<td>02</td>
<td>52</td>
<td>02</td>
<td>0C</td>
<td>0A</td>
<td>08</td>
<td>08</td>
<td>08</td>
<td>06</td>
</tr>
</tbody>
</table>
Table 6. Algebraic description of the S-Boxes used in the stream cipher.

\[
\begin{align*}
S_{1,0} &= abc + abc + abd + bde + ab + ae + be + ce + b + d \\
S_{1,1} &= abcd + abde + abc + abd + acd + ade + bcd + bce + \\
&\quad ab + ac + bc + bd + be + cd + ce + de + a + d + e + 1 \\
S_{2,0} &= abce + abc + ade + bce + bde + ab + ac + ce + c + d + 1 \\
S_{2,1} &= abde + abc + abd + abc + acd + cde + cd + ce + b + d + e + 1 \\
S_{3,0} &= ce + de + a + b + d \\
S_{3,1} &= abcd + acde + abc + ab + ae + acd + ace + ade + bcd + bde + \\
&\quad cde + ad + bc + bd + be + cd + ce + a + b + d + e + 1 \\
S_{4,0} &= abcd + abde + acde + abc + abe + bde + ab + ad + ae + bc + \\
&\quad be + de + c + d + 1 \\
S_{4,1} &= abcd + abde + acde + abc + abe + bcd + cde + ad + ab + ae + \\
&\quad de + a + b + c + e + 1 \\
S_{5,0} &= abde + acde + acd + abc + ad + ace + bce + cde + ab + ac + \\
&\quad ae + bd + be + ce + de + c \\
S_{5,1} &= abcd + abce + acd + ab + abc + acd + bcd + bce + \\
&\quad bde + cde + ac + ad + ae + be + cd + ce + de + b + d + e + 1 \\
S_{6,0} &= abcd + abde + acde + acd + ade + bcd + cde + bc + bd + cd + \\
&\quad c + e \\
S_{6,1} &= abc + ade + bce + bde + bc + ce + a + d \\
S_{7,0} &= abde + abd + cde + bc + cd + de + a + b + c + e \\
S_{7,1} &= abcd + abdebc + acde + acd + ade + bde + ac + ae + de + \\
&\quad b + c + d + e
\end{align*}
\]
References


