Impossible Differential Cryptanalysis on DVB-CSA

Kai Zhang*, Jie Guan, Bin Hu

Information Science and Technology Institute

Zhengzhou 450000, China

[e-mail: zhkai2010@139.com, guanjie007@163.com, hb2110@126.com]

*Corresponding author: Kai Zhang

Received March 13, 2015; revised October 15, 2015; accepted March 13, 2016; published April 30, 2016

Abstract

The Digital Video Broadcasting-Common Scrambling Algorithm is an ETSI-designated algorithm designed for protecting MPEG-2 signal streams, and it is universally used. Its structure is a typical hybrid symmetric cipher which contains stream part and block part within a symmetric cipher, although the entropy is 64 bits, there haven't any effective cryptanalytic results up to now. This paper studies the security level of CSA against impossible differential cryptanalysis, a 20-round impossible differential for the block cipher part is proposed and a flaw in the cipher structure is revealed. When we attack the block cipher part alone, to recover 16 bits of the initial key, the data complexity of the attack is $O(2^{44.5})$, computational complexity is $O(2^{22.7})$ and memory complexity is $O(2^{10.5})$ when we attack CSA-BC reduced to 21 rounds. According to the structure flaw, an attack on CSA with block cipher part reduced to 21 rounds is proposed, the computational complexity is $O(2^{21.7})$, data complexity is $O(2^{43.5})$ and memory complexity is $O(2^{10.5})$, we can recover 8 bits of the key accordingly. Taking both the block cipher part and stream cipher part of CSA into consideration, it is currently the best result on CSA which is accessible as far as we know.

Key Words: Hybrid Symmetric Cipher; Impossible Differential Cryptanalysis; DVB-CSA

This work was supported by the National Natural Science Foundation of China under Grant No.61202491, 61272041, 61272488, 61402523 and 61572516.

1. Introduction

DVB-CSA, which is short for Digital Video Broadcasting Common Scrambling Algorithm, has been used to protect content in MPEG2 (Such as digitally transmitted Pay-TV). In May 1994, it was accepted by the DVB consortium. During 1994 to 2002, the algorithm is confidential and only accessible under a NDA (Non-Disclosure Agreement) from European Telecommunications Standards Institute custodian (ETSI). For CSA, implementation in software is forbidden for security reasons. In 2002, a software program called FreeDec, which has CSA implemented in software, appeared on the internet and it was quickly reverse-engineered, by then the details of the CSA algorithm was to the public.

In 2004, based on the idea of guess-and-determine attack, *Ralf-Phillip Weinmann* and *Kai Wirt* presented an analysis on the stream cipher part of CSA(CSA-SC)[1] and the complexity of the attack is less than 245, based on their cipher representation and predicated on the state cycle structure of one of the registers during keystream generation. In ICCSA 2005, based on the idea of fault attack, *Kai Wirt* presented an attack on the block cipher part of the algorithm(CSA-BC)[2] by introducing a random error in the last round, the attack can be applied to the whole algorithm in real time, but the attack assumption is too strong and it is hard to achieve. In 2009, *Simpson* et al. pointed an error in [1], what's more, the total complexity of the attack after correction is worse than exhaustive search. Otherwise, *Simpson* modified the representation of CSA-SC and presented time memory trade-off attacks on CSA-SC [3], the results are as follows:

	Data	Memory	Time
State Recovery	225	239	250
Key Recovery	248.5	253	253

In 2011, using the idea of rainbow tables, *Erik Tews*, *Julian Walde*, and *Michael Weiner* proposed a time memory trade-off attack for 48 bit of entropy version of CSA [4]. In this reduced version, the effort needed for an exhaustive search reduces from 264 to 248 trial decryptions and this version can be broken in real time, using standard PC hardware if precomputed tables are available. However, these precomputations will cost several years on a standard PC. The authors also pointed that "Using the 64 instead of 48 independent bits for a key would render time memory tradeoffs inefficient". In 2015, *Kai Zhang* and *Jie Guan* proposed a distinguishing attack on the CSA-SC based on the idea of slide resynchronization attack [15], according to the distinguishing attack, the 64 bit initial key can be recovered with computational complexity of $O(255)$.

Impossible differential cryptanalysis was independently proposed by Knudsen to attack the DEAL cipher [5] and further by Biham et al. against Skipjack [6]. The basic idea of impossible differential cryptanalysis is using the impossible differential to sieve some key bits. There are usually two phases for a typical impossible differential cryptanalysis, impossible differentials constructing phase and candidate key sieving phase. Impossible differential cryptanalysis has proven to be very effective against a wide variety of ciphers [8-14].

Usually, an impossible differential is constructed by miss-in-the-middle method, more precisely, if the input difference is α, with probability one we can go forward for several rounds to get the internal difference γ, at the same time, from the output difference β, with probability one we can go backward for several rounds to get another internal difference δ, if there are contradictions between γ and δ, an impossible difference $\alpha \nrightarrow \beta$ is constructed.

In this paper, we firstly find a 20-round impossible differential for CSA-BC, then an impossible differential attack on 21-round CSA-BC is proposed, to recover 16 bits of the key, the data complexity is $O(2^{44.5})$, computational complexity is $O(2^{22.7})$ and memory complexity is $O(2^{10.5})$. On the other hand, we find a structure flaw on CSA, using this flaw we can extended part of the attack on CSA-BC to the whole algorithm, when we attack CSA with block cipher part reduced to 21 rounds, the computational complexity is $O(2^{21.7})$, data complexity is $O(2^{43.5})$ and memory complexity is $O(2^{10.5})$, we can recover 8 bits of the key accordingly.

This paper is organized as follows. In Section 2, we give an explanation for the notations used in this paper. In Section 3, we give a description on CSA algorithm. In Section 4, on one hand, we propose an impossible differential attack on 21-round CSA-BC, one the other hand, we reveal a flaw on the structure of CSA and propose an impossible differential attack on CSA with block cipher part reduced to 21 rounds, followed by conclusion in Section 5.

2. Notation

In this paper, we will use the following notations:

- K The common key. A 64 bit key used for both the stream and the block cipher;
- K^E The expanded key which is derived through the key schedule of the block cipher;
- k_i The i-th bit of K;
- SB_i The i-th 8-byte block of the scrambled data stream, SB_0 is used as nonce in the stream cipher;
- CB_i The i-th 8-byte block of stream cipher output;
- IB_i The i-th 8-byte intermediate blocks;
- DB_i The i-th 8-byte block of descrambled data;
- R The residue less than 8-byte;

- *SR* The scrambled residue;
- *IV* An initialization vector;
- | Denotes concatenation.

3. Description of CSA

Algorithm CSA can be regarded as two layer encryptions, firstly a block cipher encryption and then a stream cipher encryption. The two ciphers share the same 64-bit key *K*, and the key is called the common key. Figure 1 below depicts the encryption process of CSA. To encrypt the payload of an *m*-byte packet, the message is divided into eight bytes each which are denoted as blocks (DB_0, DB_1, ···, $DB_{n\text{-}1}$). The last block whose length is not a multiple of eight bytes is called residue (Denoted as R).

The block cipher part of CSA is used in CBC mode with reversed order, and the output of the last block IB_0 is used as a nonce for the stream cipher part. Then the encrypted blocks for the block cipher part are then XORed with the keystream generated by the stream cipher part of CSA, the residue of the block cipher part directly XORed with the keystream without block cipher encryption.

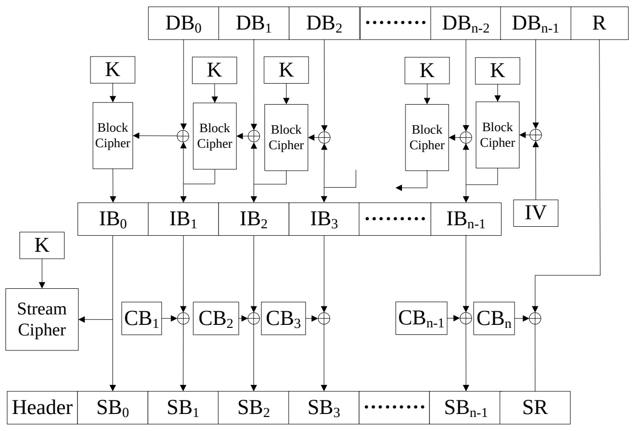

Fig. 1. Structure of CSA

As CSA-SC hardly relates to our work, we will not introduce the details of it here, for more details on the structure of CSA-SC we refer the readers to the reference [1].

CSA-BC is an iterated block cipher, it operates on eight-byte blocks of data, and the key of CSA-BC is the 64-bit common key *K*. First of all, the 64-bit common key *K* is expanded into a 448-bit running key which will be used as the round key for CSA-BC encryption. Then the message is encrypted with the same round transformation ϕ, the input of ϕ is 8-type vector and one single byte expanded key, and the output of ϕ is an 8-type vector. There are altogether 56 times iterations of ϕ for CSA-BC, the details of the CSA-BC encryption and decryption round functions are depicted in Figure 2 and Figure 3 below.

CSA-BC includes two parts: The Key Schedule algorithm and round function ϕ, next we will introduce them separately.

The key schedule A bit permutation on 64-bit vector is denoted as ρ. Then the running key $K^E = (k_0^E, k_1^E, \cdots, k_{447}^E)$ can be calculated using the following recursive algorithm.

$$k^E_{0,\ldots,63} = k_{0,\ldots,63}$$

$$k^E_{64i,\ldots,64i+63} = \rho(k^E_{64(i-1),\ldots,64i-1}) \oplus 0x0i0i0i0i0i0i0i0i \qquad 1\leq i\leq 6$$

The expression $0x0i0i0i0i0i0i0i0i$ can be regarded as a hexadecimal constant.

The permutation ρ is illustrated in the table below:

Table 1. Permutation ρ

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(i)$	17	35	8	6	41	48	28	20	27	53	61	49	18	32	58	63
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\rho(i)$	23	19	36	38	1	52	26	0	33	3	12	13	56	39	25	40
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
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

The Round Function ϕ For the 8-byte vector of each round permutation, $S=(s_0,\cdots,s_7)$ represents the input of the round function in arbitrary round, then the output of the round function ϕ refresh the state of S with the following function.

$$\phi(s_0,\ldots,s_7,k) = (s_1, s_2 \oplus s_0, s_3 \oplus s_0, s_4 \oplus s_0, s_5, s_6 \oplus \pi'(k \oplus s_7), s_7, s_0 \oplus \pi(k \oplus s_7))$$

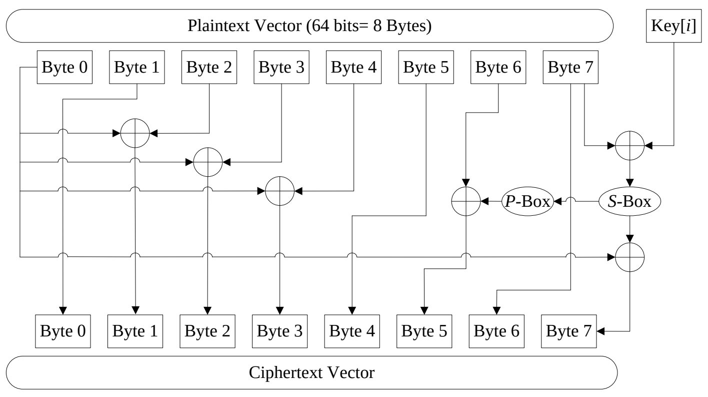

Fig. 2. Structure of the Round Function ϕ

Correspondingly, the inverse round transformation ϕ^{-1} can be illustrated as follows:

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

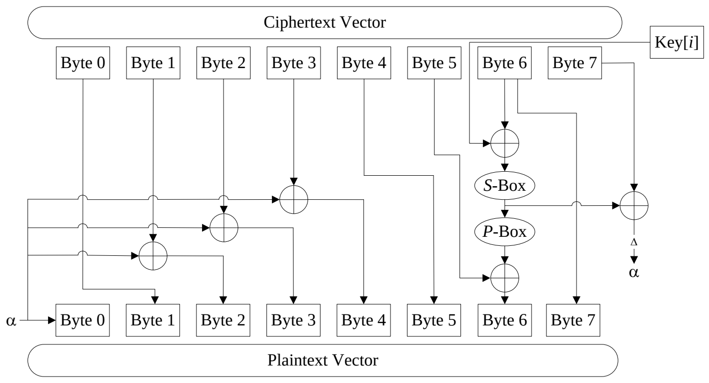

Fig. 3. *Structure of the Inverse Round Function* ϕ^{-1}

The round function ϕ applies two nonlinear permutations π and π', the relation between these two permutations is $\pi'=\sigma\circ\pi$, among which $\sigma=\begin{pmatrix}0&1&2&3&4&5&6&7\\1&7&5&4&2&6&0&3\end{pmatrix}$.

Permutation π can be viewed as an *S*-Box below:

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	3A	EA	68	FE	33	E9	88	1A	83	CF	E1	7F	BA	E2	38	12
01	E8	27	61	95	0C	36	E5	70	A2	06	82	7C	17	A3	26	49
02	BE	7A	6D	47	C1	51	8F	F3	CC	5B	67	BD	CD	18	08	C9
03	FF	69	EF	03	4E	48	4A	84	3F	B4	10	04	DC	F5	5C	C6
04	16	AB	AC	4C	F1	6A	2F	3C	3B	D4	D5	94	D0	C4	63	62
05	71	A1	F9	4F	2E	AA	C5	56	E3	39	93	CE	65	64	E4	58
06	6C	19	42	79	DD	EE	96	F6	8A	EC	1E	85	53	45	DE	BB
07	7E	0A	9A	13	2A	9D	C2	5E	5A	1F	32	35	9C	A8	73	30
08	29	3D	E7	92	87	1B	2B	4B	A5	57	97	40	15	E6	BC	0E
09	EB	C3	34	2D	B8	44	25	A4	1C	C7	23	ED	90	6E	50	00
0A	99	9E	4D	D9	DA	8D	6F	5F	3E	D7	21	74	86	DF	6B	05
0B	8E	5D	37	11	D2	28	75	D6	A7	77	24	BF	F0	B0	02	B7
0C	F8	FC	81	09	B1	01	76	91	7D	0F	C8	A0	F2	CB	78	60
0D	D1	F7	E0	B5	98	22	B3	20	1D	A6	DB	7B	59	9F	AE	31
0E	FB	D3	B6	CA	43	72	07	F4	D8	41	14	55	0D	54	8B	B9
0F	AD	46	0B	AF	80	52	2C	FA	8C	89	66	FD	B2	A9	9B	C0

Table 2. Permutation π

Remark: The figures in the table are hexadecimal, lower nibble is on horizontal and upper is on vertical.

Encryption/Decryption A plaintext $P=(p_0,\ldots,p_7)$ is encrypted according to:

$$S^0=P$$
$$S^r=\phi(S^{r-1},(k_{8r},\ldots,k_{8r+7})) \qquad \forall 1\le r\le 56$$

$$C = S^{56}$$

And the $C = (c_0, \ldots, c_7)$ is the ciphertext produced.

Similarly, the decryption process is as follows:

$$S^0 = C$$

$$S^r = \phi^{-1}(S^{r-1}, (k_{448-8r}, \ldots, k_{455-8r})) \qquad \forall 1 \le r \le 56$$

$$P = S^{56}$$

4. Impossible Differential Attack on CSA

Impossible differential cryptanalysis is a technique using the differential characters which never occur to eliminate the false keys. Through analyzing the structure and round function of CSA-BC, we find that a single active byte can be kept to 7 rounds at most. Combining this character, we construct a 20-round impossible differential with miss-in-the-middle technique and propose an attack for 21-round CSA-BC.

4.1 20-round impossible differential for CSA-BC

Proposition 1 $(0|0|0|0|0|0|\alpha|0) \overset{20r}{\nrightarrow} (0|\beta|\beta|\beta|0|0|0|\beta)$ is a 20-round impossible differential for CSA-BC.

Proof Firstly, study the differential diffusion character from encryption side.

Define $\Delta^{(i)}$ represents the differential at the ith round, $\Delta^{(0)}$ represents the differential of the plaintext. Suppose $\Delta^{(0)}=(0|0|0|0|0|0|\alpha|0)$, among which $\alpha \neq 0$. According to the structure of CSA-BC, we can get the differential for 1 to 12 rounds:

$$\Delta^{(1)}=(0|0|0|0|0|\alpha|0|0);$$

$$\Delta^{(2)}=(0|0|0|0|\alpha|0|0|0);$$

$$\Delta^{(3)}=(0|0|0|\alpha|0|0|0|0);$$

$$\Delta^{(4)}=(0|0|\alpha|0|0|0|0|0);$$

$$\Delta^{(5)}=(0|\alpha|0|0|0|0|0|0);$$

$$\Delta^{(6)}=(\alpha|0|0|0|0|0|0|0);$$

$$\Delta^{(7)}=(0|\alpha|\alpha|\alpha|0|0|0|\alpha);$$

$$\Delta^{(8)}=(\alpha|\alpha|\alpha|0|0|\Delta_{p\circ s}(\alpha)|\alpha|\Delta_s(\alpha));$$

$$\Delta^{(9)}=(\alpha|0|\alpha|\alpha|\Delta_{p\circ s}(\alpha)|\alpha\oplus\Delta_{p\circ s\circ s}(\alpha)|\Delta_s(\alpha)|\alpha\oplus\Delta_{s\circ s}(\alpha));$$

$$\Delta^{(10)}=(0|0|0|\alpha\oplus\Delta_{p\circ s}(\alpha)|\alpha\oplus\Delta_{p\circ s\circ s}(\alpha)|\Delta_s(\alpha)\oplus\Delta_{p\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))|\alpha\oplus\Delta_{s\circ s}(\alpha)|\Delta_s(\alpha\oplus\Delta_{s\circ s}(\alpha)));$$

$$\Delta^{(11)}=(0|0|\alpha\oplus\Delta_{p\circ s}(\alpha)|\alpha\oplus\Delta_{p\circ s\circ s}(\alpha)|\Delta_s(\alpha)\oplus\Delta_{p\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))|\alpha\oplus\Delta_{s\circ s}(\alpha)\oplus\Delta_{p\circ s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))| \\ \Delta_s(\alpha\oplus\Delta_{s\circ s}(\alpha))|\Delta_{s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha)));$$

$$\Delta^{(12)}=(0|\alpha\oplus\Delta_{p\circ s}(\alpha)|\alpha\oplus\Delta_{p\circ s\circ s}(\alpha)|\Delta_s(\alpha)\oplus\Delta_{p\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))|\alpha\oplus\Delta_{s\circ s}(\alpha)\oplus\Delta_{p\circ s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))| \\ \Delta_s(\alpha\oplus\Delta_{s\circ s}(\alpha))\oplus\Delta_{p\circ s\circ s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))|\Delta_{s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))|\Delta_{s\circ s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha)));$$

Then, study the differential diffusion character from decryption side.

Suppose the differential for the ciphertext (i.e. the output differential of the 20 round) is $\Delta^{(20)}=(0|\beta|\beta|\beta|0|0|0|\beta)$, among which $\beta\neq 0$. According to the structure of CSA-BC, we can get the differential for 19 to 12 rounds:

$$\Delta^{(19)}=(\beta|0|0|0|0|0|0|0);$$
$$\Delta^{(18)}=(0|\beta|0|0|0|0|0|0);$$
$$\Delta^{(17)}=(0|0|\beta|0|0|0|0|0);$$
$$\Delta^{(16)}=(0|0|0|\beta|0|0|0|0);$$
$$\Delta^{(15)}=(0|0|0|0|\beta|0|0|0);$$
$$\Delta^{(14)}=(0|0|0|0|0|\beta|0|0);$$
$$\Delta^{(13)}=(0|0|0|0|0|0|\beta|0);$$
$$\Delta^{(12)}=(\Delta_s(\beta)|0|\Delta_s(\beta)|\Delta_s(\beta)|\Delta_s(\beta)|0|\Delta_{p\circ s}(\beta)|\beta);$$

According to the differential character above, we can get the following equation:

$$\begin{aligned}&(0|\alpha\oplus\Delta_{p\circ s}(\alpha)|\alpha\oplus\Delta_{p\circ s\circ s}(\alpha)|\Delta_s(\alpha)\oplus\Delta_{p\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))|\alpha\oplus\Delta_{s\circ s}(\alpha)\oplus\Delta_{p\circ s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))|\\&\Delta_s(\alpha\oplus\Delta_{s\circ s}(\alpha))\oplus\Delta_{p\circ s\circ s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))|\Delta_{s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha))|\Delta_{s\circ s\circ s}(\alpha\oplus\Delta_{s\circ s}(\alpha)))\\&=(\Delta_s(\beta)|0|\Delta_s(\beta)|\Delta_s(\beta)|\Delta_s(\beta)|0|\Delta_{p\circ s}(\beta)|\beta).\end{aligned}$$

Compare the two sides of the equation, we can get a contradiction that $\Delta_s(\beta)=0$ which validates the correctness of the proposition.

■

4.2 Impossible Differential Attack for 21-round CSA-BC

In this subsection, we will propose an impossible differential attack on 21-round CSA-BC. The attack is based on the 20-round impossible differential above with additional one round at the end or at the top which are illustrated in Fig.4 and Fig.5 below.

$$\begin{tikzcd}
\Delta^{(0)}=(0|0|0|0|0|0|\alpha|0) \arrow[d] \\
\boxed{\text{20-Round Impossible Differential Character(Contradiction!)}} \\
\Delta^{(20)}=(0|\beta|\beta|\beta|0|0|0|\beta) \arrow[u] \\
\Delta^{(21)}=(\beta|\beta|\beta|0|0|P(\mu)|\beta|\mu) \arrow[u, "\phi^{-1}"']
\end{tikzcd}$$

Fig. 4. Impossible Differential path used to recover the subkey for the 20th round

The key recover procedure is as follows:

Algorithm 1. Key Recover Attack Algorithm to recover $K^E_{160}-K^E_{167}$ for 21-round CSA-BC

Phase 1 Choose 2^N plaintexts(The plaintexts can be any value), for each plaintext P, consider the structure of P and P' satisfies the following equation:

$$\Delta P=P\oplus P'=(0,0,0,0,0,0,\alpha,0)$$

Among which $\alpha\neq 0$, and one structure proposes $C^2_{2^8}\approx 2^{15}$ pairs of plaintexts P and P'. Encrypt the 2^{N+15} pairs of plaintexts with 21-round CSA-BC, we can get 2^{N+15} pairs of ciphertexts

C and C'.

Phase 2 Choose pairs whose ciphertext differential satisfy the following form:

$$(\beta \mid \beta \mid \beta \mid 0 \mid 0 \mid P(\mu) \mid \beta \mid \mu)$$

Among which $\beta, \mu \neq 0$. As there are 2^{N+15} pairs of plaintexts, the expected number of such pairs is $2^{N+15} \cdot (2^{-8})^3 \cdot (2^{-8})^2 \cdot (2^{-8}) \approx 2^{N-33}$.

Phase 3 Guess the 8-bit value of the round key $K_{160}^E - K_{167}^E$, for every remaining ciphertext pair, compute $\Delta^{(20)}$. If $\Delta^{(20)}=(0|\beta|\beta|\beta|0|0|0|\beta)$, discard the key candidate value of $K_{160}^E - K_{167}^E$, goto Phase 4.

Phase 4 If $K_{160}^E - K_{167}^E$ can not be uniquely determined, goto Phase 3, else finish the algorithm.

Complexities of the Attack

First of all, let us calculate the complexities to recover the subkey $K_{160}^E - K_{167}^E$. In Phase 4 of Algorithm 1, the number of false keys left after 2^N structure is $(2^8-1)\times(1-2^{-8})^{2^{N-33}}$. As $(2^8-1)\times(1-2^{-8})^{2^{10.5}} \approx 0.88$, with about $2^{43.5}$ structures which pass Phase 3, all false keys can be regarded as being eliminated. In Phase 3, we totally need $2\times2^{10.5}\times2^8\times\{1+(1-2^{-8})+(1-2^{-8})^2+\cdots+(1-2^{-8})^{2^{10.5}-1}\}\approx2^{27.5}$ 1-round decryption, i.e. about $2^{21.7}$ CSA-BC encryption. Otherwise, we need to store $2^{N-33}=2^{10.5}$ pairs of plaintext and ciphertext. At the same time, we need to store 2^8 key candidates.

So, the data complexity to recover $K_{160}^E - K_{167}^E$ is $O(2^{43.5})$, computational complexity is $O(2^{21.7})$ and memory complexity is $O(2^{10.5})$. According to the key schedule we can recover 8 bits initial key k_{32}-k_{39}.

Similarly, we can recover the subkey for the first round, the impossible differential path used is depicted in Fig. 5 below.

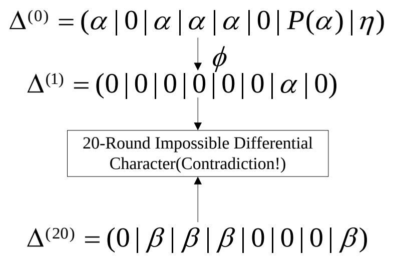
$\Delta^{(0)} = (\alpha \mid 0 \mid \alpha \mid \alpha \mid \alpha \mid 0 \mid P(\alpha) \mid \eta)$
ϕ
$\Delta^{(1)} = (0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid \alpha \mid 0)$
20-Round Impossible Differential Character(Contradiction!)
$\Delta^{(20)} = (0 \mid \beta \mid \beta \mid \beta \mid 0 \mid 0 \mid 0 \mid \beta)$

Figure 5: Impossible Differential path used to recover the subkey for the first round

The key recover procedure is as follows:

Algorithm 2. Key Recover Attack Algorithm to recover $K_0^E - K_7^E$ for 21-round CSA-BC

Phase 1 Choose 2^N ciphertexts(The ciphertexts can be any value), for each ciphertext C,consider the differential structure between C and C' satisfies the following equation:

$$\Delta C = C \oplus C' = (0, \beta, \beta, \beta, 0, 0, 0, \beta)$$

Among which $\beta \neq 0$, and one structure proposes $C_{2^8}^2 \approx 2^{15}$ pairs of plaintexts C and C'. Decrypt the 2^{N+15} pairs of ciphertexts with 21-round CSA-BC, we can get 2^{N+15} pairs of plaintexts P and P'.

Phase 2 Choose pairs whose plaintexts differential satisfy the following form:

$$(\alpha \,|\, 0 \,|\, \alpha \,|\, \alpha \,|\, \alpha \,|\, 0 \,|\, P(\alpha) \,|\, \eta)$$

Among which $\alpha, \eta \neq 0$. As there are 2^{N+15} pairs of ciphertexts, the expected number of such pairs is $2^{N+15} \cdot (2^{-8})^4 \cdot (2^{-8})^2 \approx 2^{N-33}$.

Phase 3 Guess the 8-bit value of the round key $K_0^E - K_7^E$, for every remaining pair of plaintexts, compute $\Delta^{(1)}$. If $\Delta^{(1)} = (0|0|0|0|0|0|\alpha|0)$, discard the key candidate value of $K_0^E - K_7^E$, goto Phase 4, else goto Phase 3.

Phase 4 If $K_0^E - K_7^E$ can not be uniquely determined, goto Phase 3, else finish the algorithm.

Complexities of the Attack

First of all, let us calculate the complexities to recover the subkey $K_0^E - K_7^E$. In Phase 4 of Algorithm 2, the number of false keys left after 2^N structure is $(2^8 - 1) \times (1 - 2^{-8})^{2^{N-33}}$. As $(2^8 - 1) \times (1 - 2^{-8})^{2^{10.5}} \approx 0.88$, with about $2^{43.5}$ structures which pass Phase 3, all false keys can be regarded as being eliminated. In Phase 3, we totally need $2 \times 2^{10.5} \times 2^8 \times \{1 + (1-2^{-8}) + (1-2^{-8})^2 + \cdots + (1-2^{-8})^{2^{10.5}-1}\} \approx 2^{27.5}$ 1-round encryption, i.e. about $2^{21.7}$ CSA-BC encryption. Otherwise, we need to store $2^{N-33} = 2^{10.5}$ pairs of plaintext and ciphertext. What's more, we need to store 2^8 key candidates.

So, the data complexity to recover $K_0^E - K_7^E$ is $O(2^{43.5})$, computational complexity is $O(2^{21.7})$ and memory complexity is $O(2^{10.5})$.

All in all, to recover 16 bits of the key, the total data complexity is $O(2^{44.5})$, computational complexity is $O(2^{22.7})$ and memory complexity is $O(2^{10.5})$. The subkeys k_0-k_7 and k_{32}-k_{39} can be recovered accordingly.

4.3 Structure flaw on CSA

According to the structure of CSA(Fig. 1), as the sequence of 8-byte blocks is encrypted in reverse order with the block cipher in CBC mode, and the last output of the chain IB_0 is used as a nonce for the stream cipher which is directly output as part of the ciphertext, we can get the output of the last block. So if we just induce difference in the first block of the plaintext (DB_0), the difference of other blocks are zero, we can only control the differential of the first block of the plaintext and get the ciphertext of the first block if we want to attack the whole algorithm. This process is depicted in Fig.6 below. We should notice that we can not control the input of the first block.

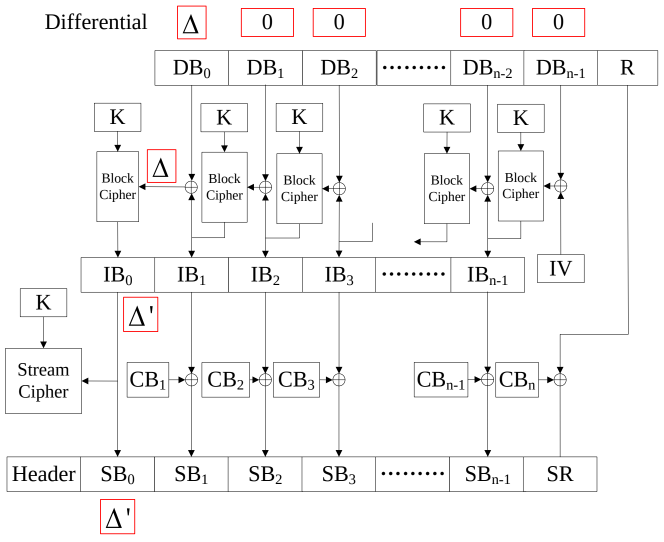

Fig. 6. *A Differential Character of CSA*

The structure flaw above can make some statistical cryptanalytic methods on CSA-BC extend to the cryptanalysis of the whole algorithm, such as differential cryptanalysis, impossible differential cryptanalysis, integral attack and so on. So this flaw is an important problem for CSA which may lead to better cryptanalytic results. Part of the result in section 4.2 can be applied to the whole algorithm according to this flaw.

As the attacker can only control the differential of DB_0 rather than control the actual input of the block cipher, only Algorithm 1 in section 4.1 can be used if we attack the whole algorithm. That is to say, when we attack CSA(with block cipher part reduced to 21 rounds), we can recover 8 bits of the key($k_{32} - k_{39}$) with data complexity $O(2^{43.5})$, computational complexity $O(2^{21.7})$ and memory complexity $O(2^{10.5})$.

Although the result is not as good when compared to the application on CSA-BC, the successful application on CSA indicates that when we design a hybrid symmetric cipher, improper structure design can make security level of a hybrid symmetric cipher reduce to the security level of its stream cipher part or block cipher part, and this indicates that structure design is an important part of the hybrid cipher design which deserves further exploration.

5. Conclusion

Being the ETSI-specified algorithm since 1994, there hasn't any fatal flaw for the 64-bit algorithm CSA so far, the specification and design criteria are still confidential to the public. For the unique and deliberate design of CSA, if we probe into the design criteria and security level of CSA, it will enhance the development of symmetric ciphers obviously. In this paper, a 20-round impossible differential for CSA-BC is presented, and an impossible differential cryptanalysis on CSA-BC reduced to 21 rounds is proposed, to recover 16 bits of the key, the data complexity is $O(2^{44.5})$, computational complexity is $O(2^{22.7})$ and memory complexity is

$O(2^{10.5})$. What's more, we reveal a flaw on the structure of CSA which makes the impossible differential attack can be applied to the whole algorithm. When we attack CSA with block cipher part reduced to 21 rounds, the data complexity is $O(2^{43.5})$, computational complexity is $O(2^{21.7})$ and memory complexity is $O(2^{10.5})$, we can recover 8 bits of the key accordingly. How to make full use of this structure flaw and evaluate CSA against other cryptanalysis techniques is still further to be studied.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported by the National Natural Science Foundation of China under Grant No.61202491, 61272041, 61272488, 61402523 and 61572516.

References

[1] R.P. Weinmann and K. Wirt, "Analysis of the DVB Common Scrambling Algorithm," in *Proc. of IFIP International Federation for Information Processing 2005*, Volume 175/2005, pp.195-207, 2005. Article (CrossRef Link).

[2] K. Wirt, "Fault attack on the DVB Common Scrambling Algorithm," in *Proc. of Computational science and its applications-ICCSA 2005*, Volume 3481, pp.511-517, 2005. Article (CrossRef Link).

[3] L. Simpson, M. Henricksen and W.S. Yap, "Improved Cryptanalysis of the Common Scambling Algorithm Stream Cipher," in *Proc. of the 14th Australasian Conference on Information Security and Privacy 2009*, pp.108-121, 2009. Article (CrossRef Link).

[4] E. Tews, J. Walde and M. Weiner, "Breaking DVB-CSA," in *Proc. of West European Workshop on Research in Cryptography 2011*, pp.41-45, 2011. Article (CrossRef Link).

[5] L.R. Knudsen, "DEAL-A 128-bit Block Cipher," *Technical Report 151*, Department of Informatics, University of Bergen, Bergen, Norway, Feb. 1998. Article (CrossRef Link).

[6] E. Biham, A. Biryukov, "A. Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials," in *Proc. of Eurocrypt'99. Berlin: Springer-Verlag, LNCS*, 1999. 1592: pp. 12-23, 1999. Article (CrossRef Link).

[7] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *Journal of Cryptology*, Vol 3, pp. 3-72, 1991. Article (CrossRef Link)

[8] E. Biham and N. Keller, "Cryptanalysis of Reduced Variants of Rijndael," *3rd AES Conference*, 2000. Article (CrossRef Link)

[9] W. Zhang, W. Wu, L. Zhang, D. Feng, "Improved related-key impossible differential attacks on reduced-round AES-192," in *Proc. of Selected Areas in Cryptography (SAC 2006)*, Montreal, Canada, Springer-Verlag, August 17-18, pp. 168-181, 2006. Article (CrossRef Link)

[10] W. Wu, W. Zhang, D. Feng, "Impossible differential cryptanalysis of reduced-round ARIA and Camellia," *Journal of computer science and technology*, 22(3): 449-456, 2007. Article (CrossRef Link)

[11] W. Wu, L. Zhang, W. Zhang, "Improved Impossible Differential Cryptanalysis of Reduced-Round Camellia," in *Proc. of Selected Areas in Cryptography (SAC 2008)*, Springer-Verlag, LNCS vol. 5381, pp. 442-456, 2009. Article (CrossRef Link)

[12] J. Chen, K. Jia, H. Yu, X. Wang, "New Impossible Differential Attacks of Reduced-Round Camellia-192 and Camellia-256," in *Proc. of Information Security and Privacy*, Springer- Verlag, LNCS vol. 6812, pp. 16-33, 2011. Article (CrossRef Link)

[13] C. Du, J. Chen, "Impossible Differential Cryptanalysis of ARIA Reduced to 7 Rounds," in *Proc. of Cryptology and Network Security*, LNCS vol. 6467, pp. 20-30, 2010. Article (CrossRef Link)

[14] S. Li, C. Song, "Improved Impossible Differential Cryptanalysis of ARIA," in *Proc. of Information Security and Assurance*, ISA 2008, pp. 129-132, 2008. Article (CrossRef Link)

[15] K. Zhang, J. Guan "Distinguishing Attack on Common Scrambling Algorithm," *The International Arab Journal of Information Technology*, 12(4), 410-414, 2015.

Kai Zhang received the M.S. degree in cryptology from the Information Science and Technology Institute, Zhengzhou, China, in June. 2013. He is pursuing his Ph.D. degree in cryptology from this university. His main research interests include design and analysis of symmetric ciphers. His works have been published in several refereed journals and he has been serving as a referee for the international journal of Security and Communication Networks in the area of information security and cryptology.

Jie Guan is a professor of Information Science and Technology Institute, Zhengzhou, China. Her main subject interest is cryptography and her main teaching lies in the areas of information systems, the theory of cryptography and quantum computation. She received Ph.D. degree in cryptography from Information Science and Technology Institute, Zhengzhou, China, in 2004.

Bin Hu is a professor of Information Science and Technology Institute, Zhengzhou, China. His main subject interests and his main teaching are Boolean function, information security and cryptology. He received Ph.D degree in cryptography from Information Science and Technology Institute, Zhengzhou, China, in 2008.