Distinguishing Attack on Common Scrambling Algorithm

Kai Zhang and Jie Guan
Zhengzhou Information Science and Technology Institute, China

Abstract: *Common Scrambling Algorithm (CSA) has been used to encrypt european cable digital television signals since 1994. Although, the key size of CSA is small, up to now, there haven't any effective crypto results which can break the algorithm totally. Based on the idea of slide resynchronization attack, a distinguishing attack which can distinguish the keystream of the Stream Cipher (SC) from a purely random sequence with computational complexity of O(2^{15}) is proposed. According to the distinguishing attack, the 64 bit initial key can be recovered with computational complexity of O(2^{55}).*

Keywords: *Digital video broadcasting common scrambling algorithm, distinguishing attack, slide resynchronization attack, hybrid cipher.*

Received August 31, 2012; accepted February, 23, 2014; published online August 17, 2014

1. Introduction

The Digital Video Broadcasting Common Scrambling Algorithm (DVB-CSA) has been used to encrypt european cable digital television signals since 1994. As part of the MPEG-TS standard, it is virtually used for all conditional access systems in digital television. The algorithm has never been officially revealed and was only available under a non-disclosure agreement from an ETSI custodian before 2002. The agreement didn't permit the algorithm to be implemented in software for "security reasons". In 2002, a software program called FreeDec appeared on the internet, which implemented CSA in software, then the algorithm was reverse engineered quickly by hackers and the details of the CSA was accessed to the public.

Weinmann and Wirt [10] proposed a guess and determine attack [8] with complexity less than 2^{45}. Wirt [11] presented a fault attack on the block cipher which can be launched without regarding the Stream Cipher part (SC). Although, the complexity of this attack is within seconds, the condition of the attack model is too strong to be compared. In [8] Simpson figured out some mistakes in the description of the SC part of CSA and presented time-memory tradeoff attacks on the stream cipher, the results are as follows [10]:

Table 1. Results of DMT attacks on CSA-SC.

	Data	Memory	Time
State Recovery	2^{25}	2^{39}	2^{50}
Key Recovery	$2^{48.5}$	2^{53}	2^{53}

Twes *et al.* [9] presented a time memory trade off attack against 48 bit key CSA, the result showed that CSA can be broken in real time if the precomputed tables are available and calculating these precomputations need over years and the author claimed that "when the algorithm using 64 instead of 48 independent bits for a key would render time tradeoffs inefficient", this is another tradeoff method for CSA cryptanalysis.

Distinguishing attack was firstly proposed in [1]. In a distinguishing attack, the adversary tries to judge whether a given sequence stems from a known cipher or a purely random sequence. A distinguishing attack may not be as strong as a key-recovery attack, but it can provide some undesired information leakage to the adversary. Related key cryptanalysis was first introduced by Biham [3], it is a type of chosen key attacks, in which the relationship between the keys used is known. The slide resynchronization attack was firstly proposed on the cryptanalysis of Grain [6] which is a related key attack using the slide property of the key. This cryptanalytic tool shows great potential in cryptanalysis for some stream ciphers [2, 4, 5, 7, 12] whose initialization phase and key stream generating phase are similar.

In this paper, we propose a distinguishing attack at the basis of "slide resynchronization attack" which can distinguish the key stream of the SC part from a purely random sequence with computational complexity of $O(2^{15})$. According to the distinguishing attack, we can recover the full 64 bit key with computational complexity of $O(2^{55})$.

2. Description of CSA

CSA is a typical hybrid cipher which can be seen as a cascade of a block cipher and a SC. Both ciphers use the same 64 bit key K, which is called the common key. The following description of CSA refers the descriptions in reference [8, 10]. For scrambling, the plaintext is divided into blocks of 64 bit length (8 byte) and the sequence of blocks is encrypted with a custom block cipher in CBC mode using reverse order, then the ciphertext of the block cipher are encrypted with the key stream generated by the stream part of the cipher, the residue is directly XORed to the key stream of the SC and the last output of the block cipher is used as *IV* for the SC. Figure 1 depicts in the scrambling process how the block and the SC are combined.

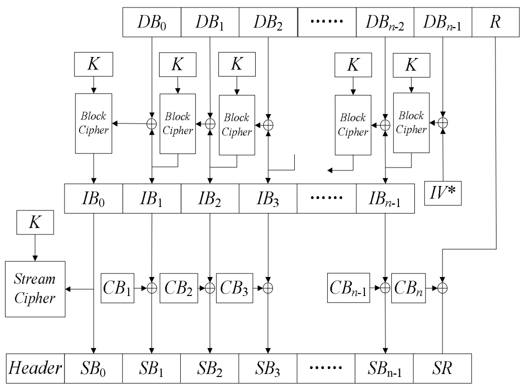
DB_0 DB_1 DB_2 ······ DB_{n-2} DB_{n-1} R
K
Block Cipher
IB_0 IB_1 IB_2 IB_3 ······ IB_{n-1} IV^*
Stream Cipher
CB_1 CB_2 CB_3 CB_{n-1} CB_n
Header SB_0 SB_1 SB_2 SB_3 ······ SB_{n-1} SR

Figure 1. Structure of CSA.

The CSA-SC consists of two FSRs and a FSM which are denoted as FSR-A, FSR-B and FSM-C in Figure 2. FSR-A and FSR-B each have ten stages, with each stage containing one 4 bit word. FS-C consists of two stages, each containing a four bit word and a single carry bit. Here is the structure of CSA-SC during the key stream generating process. Registers *A* and *B* are 40 bit registers, *p*, *q* and *c* are 1 bit registers and others are all 4 bit registers.

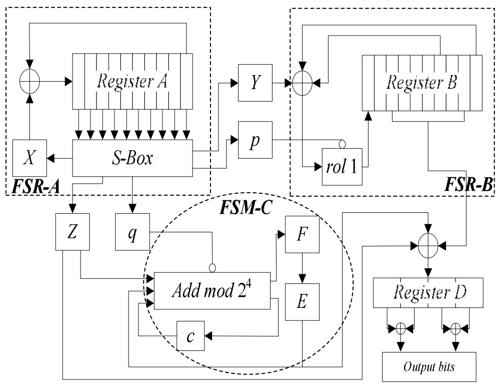

Figure 2. Keystream generating mode of CSA-SC.

There are two modes of SC: The initialization mode and the key stream generating mode. During the initialization process, all of the registers are set to zero, then the common key $K=k_0, k_1, k_2, \ldots, k_{63}$ are loaded into the registers *A* and *B* according to the equations below (Denote $A=A_{0,\ j},\ldots, A_{9,\ j}$; $B=B_{0,\ j},\ldots, B_{9,\ j}$, $0\leq j\leq 3$, A_i and B_i represent 4 bit states $A_{i,\ 0},\ldots, A_{i,\ 3}$ and $B_{i,\ 0},\ldots B_{i,\ 3}$ respectively):

$$A_{i,j}=\begin{cases} if\ (i\leq 7) & k_{4\times i+j} \\ else & 0 \end{cases}$$
$$B_{i,j}=\begin{cases} if\ (i\leq 7) & k_{32+4\times i+j} \\ else & 0 \end{cases} \tag{1}$$

Denote *IV*= (*IV* [7], *IV* [6], *IV* [5], *IV* [4], *IV* [3], *IV* [2], *IV* [1], *IV* [0]), for 32 rounds initialization process, each 4 rounds uses a byte of *IV*, the turn is from the least significant byte to the most significant byte. In every odd cycle number the internal vector I^A is the high nibble of *IV*[*i*] and I^B is the low nibble. In even cycles the nibbles are used the other way round.

1. The register *A* is updated as follows:

$$A'_0=\begin{cases} A_9\oplus X\oplus D\oplus I^A & \textit{Initialization} \\ A_9\oplus X & \textit{Keystream generating} \end{cases} \tag{2}$$

The new state of register *A* is $A'=(A'_0, A_0, \ldots, A_8)$.

2. The register *B* is updated as follows:

$$B'_0=\begin{cases} B_6\oplus B_9\oplus Y\oplus I^B & \textit{Initialization} \\ B_6\oplus B_9\oplus Y & \textit{Keystream generating} \end{cases} \tag{3}$$

The new state of register *B* is as follows:

$$B'=\begin{cases} (B'_0, B_0, \ldots, B_8) & p=0 \\ (rol\,(B'_0), B_0, \ldots, B_8) & else \end{cases} \tag{4}$$

3. The registers *X*, *Y*, *Z*, *p* and *q* are updated through Boolean functions of F_i, $0\leq i\leq 13$, which can be seen as *S*-Boxes, the input variables and corresponding F_i are shown in Table 2 below:

Table 2. Corresponding input and output variables of *S*-Boxes.

S-Box	Input Variables	Output
S_X	s_3, s_2, s_1, s_0	$F_6(s_3)\ \|\|\ F_4(s_2)\ \|\|\ F_3(s_1)\ \|\|\ F_1(s_0)$
S_Y	s_5, s_4, s_3, s_2	$F_{10}(s_5)\|\|\ F_8(s_4)\ \|\|\ F_7(s_3)\ \|\|\ )$
S_Z	s_1, s_0, s_5, s_4	$F_2(s_1)\ \|\|\ F_0(s_0)\ \|\|\ F_{11}(s_5)\ \|\|\ F_9(s_4)$
S_p	s_6	$F_{13}(s_6)\ F_5(s_2$
S_q	s_6	$F_{12}(s_6)$

The input variables of the *S*-Boxes are corresponding states of register *A* as follows:

Table 3. Relationship between the internal states and the input of Boolean functions.

Input Variable	x_0	x_1	x_2	x_3	x_4
s_0	$A_{4,0}$	$A_{1,2}$	$A_{6,1}$	$A_{7,3}$	$A_{9,0}$
s_1	$A_{2,1}$	$A_{3,2}$	$A_{6,3}$	$A_{7,0}$	$A_{9,1}$
s_2	$A_{1,3}$	$A_{2,0}$	$A_{5,1}$	$A_{5,3}$	$A_{6,2}$
s_3	$A_{3,3}$	$A_{1,1}$	$A_{2,3}$	$A_{4,2}$	$A_{8,0}$
s_4	$A_{5,2}$	$A_{4,3}$	$A_{6,0}$	$A_{8,1}$	$A_{9,2}$
s_5	$A_{3,1}$	$A_{4,1}$	$A_{5,0}$	$A_{7,2}$	$A_{9,3}$
s_6	$A_{2,2}$	$A_{3,0}$	$A_{7,1}$	$A_{8,2}$	$A_{8,3}$

The algebraic normal forms 5-input boolean functions of F_i, $0\leq i\leq 13$ are presented in the appendix of reference [8].

4. The update of FSM: The FSM include registers *E*, *F* and carry bit Register *c*, they are updated as follows:

$$(E, F)'=\begin{cases} if\ (q=0) & (F, E) \\ else & (F, (E+Z+c)mod\ 2^4) \end{cases} \tag{5}$$

If q=0, c is unchanged. Otherwise, it is 1 if E+F+c≥ 2^4 and 0 else.

5. The output: The SC generates 2 bit-key stream each clock cycle, i.e., $D_2 \oplus D_3 \,||\, D_0 \oplus D_1$, D= (D_3, D_2, D_1, D_0)= $E \oplus Z \oplus B^{out}$, $B^{out} = (B_3^{out}, B_2^{out}, B_1^{out}, B_0^{out})$ and B_i^{out} (0≤i ≤ 3) are defined as follows:

$$\begin{aligned} B_3^{out} &= b_{2,0} \oplus b_{5,1} \oplus b_{6,2} \oplus b_{8,3} \\ B_2^{out} &= b_{5,0} \oplus b_{7,1} \oplus b_{2,3} \oplus b_{3,2} \\ B_1^{out} &= b_{4,3} \oplus b_{7,2} \oplus b_{3,0} \oplus b_{4,1} \\ B_0^{out} &= b_{8,2} \oplus b_{5,3} \oplus b_{2,1} \oplus b_{7,0} \end{aligned} \tag{6}$$

As the block cipher part of CSA hardly relates to our work, we won't introduce the details of the block cipher part of CSA here, for more details on the structure of the block cipher we refer the readers to the reference [11].

3. Distinguishing Attack on CSA-SC

In this section we find related keys and initial value of the CSA-SC. For any (K, IV) pair there exists related (K', IV') pair that generate 1 bit shifted key stream, thus we can lead to a distinguishing attack on SC part of CSA.

As the 64 bit key are filled into the registers directly and the structure of the encryption and the initialization process are similar to each other, which make the distinguishing attack to be possible. The following notations are used throughout the paper.

- $S^{(t)}$: The internal state of CSA-SC at time t.
- $A^{(t)}$: The internal state of register A at time t, $A_{i,j}^{(t)}$ (0≤ i≤ 9, 0≤ j≤ 3) represents the jth bit of the ith stage for register A at time t.
- $B^{(t)}$: The internal state of register B at time t, $B_{i,j}^{(t)}$ (0≤ i≤ 9, 0≤ j≤ 3) represents the jth bit of the ith stage of register B at time t.
- $X^{(t)}$, $Y^{(t)}$, $Z^{(t)}$, $E^{(t)}$, $F^{(t)}$, $D^{(t)}$: The internal state of registers X, Y, Z, E, F, D at time t, $X^{(t)}$, $Y^{(t)}$, $Z^{(t)}$, $E^{(t)}$, $F^{(t)}$, $D^{(t}$ (0 ≤ i ≤3) represent the ith bit of corresponding registers at time t.
- $p^{(t)}$, $q^{(t)}$, $c^{(t)}$: The internal state of registers p, q, c at time t.

Before the key initialization process, $A^{(0)}$ and $B^{(0)}$ are initialized with the 64 bit key as follows:

$$A_{i,j}^{(0)} = \begin{cases} if\ (i \le 7) & k_{4\times i+j} \\ else & 0 \end{cases} \qquad B_{i,j}^{(0)} = \begin{cases} if\ (i \le 7) & k_{32+4\times i+j} \\ else & 0 \end{cases} \tag{7}$$

Among which k_i denotes the ith bit of K, other registers are set to zero.

To lead a slide resynchronization attack we should calculate all the state vectors of registers at different times. Firstly calculate state vectors $A^{(1)}$ and $B^{(1)}$ by using key initialization algorithm.

$A^{(0)}$	0	1	2	3	4	5	6	7	8	9
0	k_0	k_4	k_8	k_{12}	k_{16}	k_{20}	k_{24}	k_{28}	0	0
1	k_1	k_5	k_9	k_{13}	k_{17}	k_{21}	k_{25}	k_{29}	0	0
2	k_2	k_6	k_{10}	k_{14}	k_{18}	k_{22}	k_{26}	k_{30}	0	0
3	k_3	k_7	k_{11}	k_{15}	k_{19}	k_{23}	k_{27}	k_{31}	0	0

$\Rightarrow$

$A^{(1)}$	0	1	2	3	4	5	6	7	8	9
0	k'_0	k_0	k_4	k_8	k_{12}	k_{16}	k_{20}	k_{24}	k_{28}	0
1	k'_1	k_1	k_5	k_9	k_{13}	k_{17}	k_{21}	k_{25}	k_{29}	0
2	k'_2	k_2	k_6	k_{10}	k_{14}	k_{18}	k_{22}	k_{26}	k_{30}	0
3	k'_3	k_3	k_7	k_{11}	k_{15}	k_{19}	k_{23}	k_{27}	k_{31}	0

$B^{(0)}$	0	1	2	3	4	5	6	7	8	9
0	k_{32}	k_{36}	k_{40}	k_{44}	k_{48}	k_{52}	k_{56}	k_{60}	0	0
1	k_{33}	k_{37}	k_{41}	k_{45}	k_{49}	k_{53}	k_{57}	k_{61}	0	0
2	k_{34}	k_{38}	k_{42}	k_{46}	k_{50}	k_{54}	k_{58}	k_{62}	0	0
3	k_{35}	k_{39}	k_{43}	k_{47}	k_{51}	k_{55}	k_{59}	k_{63}	0	0

$\Rightarrow$

$B^{(1)}$	0	1	2	3	4	5	6	7	8	9
0	k'_{32}	k_{32}	k_{36}	k_{40}	k_{44}	k_{48}	k_{52}	k_{56}	k_{60}	0
1	k'_{33}	k_{33}	k_{37}	k_{41}	k_{45}	k_{49}	k_{53}	k_{57}	k_{61}	0
2	k'_{34}	k_{34}	k_{38}	k_{42}	k_{46}	k_{50}	k_{54}	k_{58}	k_{62}	0
3	k'_{35}	k_{35}	k_{39}	k_{43}	k_{47}	k_{51}	k_{55}	k_{59}	k_{63}	0

Figure 3. One step transformation of register A and B.

Among Figure 3:

$(k'_3, k'_2, k'_1, k'_0) = D^{(0)} \oplus X^{(0)} \oplus I^A \oplus A_9^{(0)}$, $I^B \oplus B_6^{(0)} \oplus B_9^{(0)}$

We denote the related key K' as follows:

K'=(k'_0, k'_1, k'_2, k'_3, k_0, k_1, k_2 , k_3, …, k_{24}, k_{25}, k_{26}, k_{27}, k'_{32}, k'_{33}, k'_{34}, k'_{35}, k_{32}, k_{33}, k_{34}, k_{35},…, k_{56}, k_{57}, k_{58}, k_{59})

If the following conditions occur:

- *Condition 1*: (k_{28}, k_{29},k_{30}, k_{31}, k_{60}, k_{61}, k_{62}, k_{63})= (0, 0, 0, 0, 0, 0, 0, 0).
- *Condition 2*: $q^{(0)}$=0.
- *Condition 3*: IV= (a, a, a, a, a, a, a, a, a , a, a, a, a, a, a), a∈GF (2^4).
- *Condition 4*: For K', $D^{(31)}$ = a, a∈GF(2^4).

For condition 1, as we don't know whether the (k_{28}, k_{29}, k_{30}, k_{31}, k_{60}, k_{61}, k_{62}, k_{63})= (0, 0, 0, 0, 0, 0, 0, 0), we can realize this by trying all the related keys ΔK, Δ(k_{28}, k_{29}, k_{30}, k_{31}, k_{60}, k_{61}, k_{62}, k_{63})∈GF(2^8), the difference of other bits are all zero, there must be a certain ΔK which make condition 1 occur.

For condition 2, if $q^{(0)}$=0, the register E and F don't update for the moment and are kept zero for t= 0, the probability of $q^{(0)}$= 0 is 1/2. As $q^{(0)}$= F_{12}(k_{10}, k_{12}, k_{29}, 0, 0)= $k_{29}.\overline{k_{10}} \oplus k_{12}$, if we trying all the related keys ΔK, Δk_{12}∈GF(2), the difference of other bits are all zero, it can be easily proven that the condition 2 must within.

For condition 3, the IV can be chosen as (a, a, a, a, a, a, a, a, a, a, a, a, a, a, a), a∈GF(2^4), thus the even cycle and odd cycle share the same I^A and I^B.

As for condition 4, $D^{(31)}$ can be regarded as a random vector, to guarantee the probability, we can choose any of the other bits except these positions:

k_{12}, k_{28}, k_{29}, k_{30}, k_{31}, k_{60}, k_{61}, k_{62}, k_{63}

It can be easily proven that the probability of $D^{(31)}= a$ is about 0.98 if we use 2^6 related key $(1-(1-2^{-4})^{2^6}\approx 0.98)$ To sum up, the total computational complexity of all the conditions is 2^{15} pairs of related keys.

It's easy to see that, for K, the internal states of the cipher during the initialization process are denoted as $S^{(0)}, S^{(1)}, S^{(2)},\cdots, S^{(31)}$, for K', the corresponding states are denoted as $S'^{(0)}, S'^{(1)}, S'^{(2)},\cdots, S'^{(31)}$, if the four conditions above appear, the following equations during the initialization process are kept:

$$S'^{(0)}=S^{(1)}, S'^{(1)}= S^{(2)}, S'^{(2)} = S^{(3)}, \cdots, S'^{(30)}= S^{(31)}$$

During the key generating process, the following relations are kept:

$$S'^{(31)}= S^{(32)}, S'^{(32)} = S^{(33)}, S'^{(33)}= S^{(34)}, \ldots$$

Hence, for K, CSA-SC generates 1 bit shifted of the keystream generated for K'. Thus, the key stream can be distinguished from a purely random sequence with computational complexity of $O(2^{15})$.

Further, if the 1 bit shifted of the keystream generated, according to the ΔK for condition 1, we can determine eight key bits ($k_{28}, k_{29}, k_{30}, k_{31}, k_{60}, k_{61}, k_{62}, k_{63}$). As for condition 1, $k_{29}= 0$, we can determine k_{12} according to the equation $k_{29}\cdot\overline{k_{10}}\oplus k_{12}=0$ and ΔK for condition 2. Hence, we have known 9 key bits which will decrease the exhaustive search complexity from $O(2^{64})$ to $O(2^{55})$. Last of all, we want to stress that as the residue is directly XORed to the key stream of the SC, we can get the key stream generated by the SC easily, which makes our attack can be applied to the whole algorithm.

4. Remarks on Success Rate

According to the result in section 3, if we trying 2^{15} pairs of K and K', the 1 bit shifted keystream can occur with probability of 0.98 and this probability can be increased to 0.9997 when trying 2^{16} pairs of K and K'. There is also another instance which may misjudge the distinguisher of our attack. When the key candidate can't satisfy the several conditions above, the internal states after 32 rounds of initialization can be viewed as random, so $S'^{(31)}$ is equal to $S^{(32)}$ with probability of 2^{-89}, the probability is so small that it hardly affect out result.

On the other hand, we should get enough key stream bits to distinguish the shifted key stream from the random case, this can be accessed from the residue bits. If we get n residue bits, the success rate can achieve $1-2^{-n}$, when $n=10$, the success rate can achieve 0.999 and the key stream can be easily got from one residue of the cipher (Less than 64 bits). So, the success rate of our attack can be very close to 1.

5. Conclusions

Although, the algorithm CSA only has 64 bit key size, for many *TV* stations only 48 bit of entropy are used [9]. However, there haven't any effective crypto results which can break the algorithm totally up to now. The structure of CSA and the design rationale are well worth studying which may improve the design of the symmetric encryption algorithms significantly. As the initialization process and the key stream generating process are similar, based on the thought of slide resynchronization attack we propose a distinguishing attack which can distinguish the key stream of the SC from a purely random sequence with computational complexity of $O(2^{15})$. At the same time, we can recover 9 bits of the key accordingly, which makes the exhaustive search complexity reduce to $O(2^{55})$. Our attack provides a clue to recover several key bits of CSA, which may be combined with other attacks to reduce the computational complexity further. The ability of CSA to resist other cryptanalysis is further to be studied.

Acknowledgements

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

References

[1] Baigneres T., Junod P., and Vaudenay S, "How Far Can We Go Beyond Linear Cryptanalysis?," *in Proceedings of Advances in Cryptology-Asiacrypt*, Korea, pp. 432-450, 2004.

[2] Berbain C., Gilbert H., and Maximov A., "Cryptanalysis of Grain," *in Proceedings of Fast Software Encryption*, Graz, Austria, pp. 15-29, 2006.

[3] Biham E., "New Types of Cryptanalytic Attacks using Related Keys," *the Journal of Cryptology*, vol. 7, no. 4, pp. 229-246, 1994.

[4] Hassan M. and Al-Shalabi H., "Modified Cryptanalysis of RC5," *the International Arab Journal of Information Technology*, vol. 3, no. 4, pp. 299-302, 2006.

[5] Khazaei S., Hassanzadeh M., and Kiaei M., "Distinguishing Attack on Grain," available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.2357&rep=rep1&type=pdf, last visited 2005.

[6] Kücük O., "Slide Resynchronization Attack on the Initialization of Grain 1.0," available at: http://www.ecrypt.eu.org/stream/papersdir/2006/044.ps, last visited 2006.

[7] Lee Y., Jeong K., Sung J., and Hong S., "Related-Key Chosen IV Attacks on Grain-v1 and Grain-128," *in Proceedings of Australasian*

Conference on Information Security and Privacy, Wollongong, Australia, pp. 321-335, 2008.

[8] Simpson L., Henricksen M., and Yap W., "Improved Cryptanalysis of the Common Scambling Algorithm Stream Cipher," *in Proceedings of Australasian Conference on Information Security and Privacy*, Brisbane, Australia, pp. 108-121, 2009.

[9] Tews E., Walde J., and Weiner M., "Breaking DVB-CSA," *in Proceedings of West European Workshop on Research in Cryptography*, Weimar, Germany, pp. 41-45, 2011.

[10] Weinmann R. and Wirt K., "Analysis of the DVB Common Scrambling Algorithm," *in Proceedings of International Federation for Information Processing Conference*, Windermere, UK, pp. 195-207, 2005.

[11] Wirt K., "Fault Attack on the DVB Common Scrambling Algorithm," *in Proceedings of International Conference on Computational Science and its Applications*, Singapore, pp. 511-517, 2005.

[12] Wu H. and Preneel B., "Resynchronization Attacks on WG and LEX," *in Proceedings of Fast Software Encryption*, Graz, Austria, pp. 422-432, 2006.

Kai Zhang received BS and MS degrees from Zhengzhou Information Science and Technology Institute in 2010 and 2013 respectively. He is studying for PhD degree in cryptography in the same university. His current research interests include design and analysis of symmetric cipher.

Jie Guan is an associate professor of Zhengzhou Information Science and Technology Institute. 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 PhD degree in cryptography from Zhengzhou Information Science and Technology Institute in 2004.