"HENRI COANDA"
AIR FORCE ACADEMY
ROMANIA

GERMANY

"GENERAL M.R. STEFANIK"
ARMED FORCES ACADEMY
SLOVAK REPUBLIC

INTERNATIONAL CONFERENCE of SCIENTIFIC PAPER
AFASES 2011
Brasov, 26-28 May 2011

SELF-SYNCHRONIZATION TECHNIQUES USED TO ENCRYPT DATA STRINGS OF TELEPHONE AND DATA SUBSCRIBERS SYSTEMS

Constantin GROZEA, Ionuț-Gabriel TÂRŞA

* Military Equipment and Technologies Research Agency, Bucharest, Romania,

Abstract: *Encryption systems running at low speeds can benefit of the advantages offered by the self-synchronization schemes. Using this kind of mechanism the complex problem of traditional synchronization methodology (using a synchronization equipment at both transmission and reception end) can be avoided. This paper proposes the implementation of a software emulator aimed to analyze the self-synchronization techniques functionalities. It can be used both in didactic purposes and in the implementation phase of communications or encryption systems.*

Keywords: *self-synchronization, software emulator, encryption systems*

I. INTRODUCTION

String encryption systems are classified into synchronized and self-synchronized systems.

Synchronized systems:

A synchronous encryption string is a structure (P, C, K, L, E, D), where:

• P, C, K are finite nonempty sets whose elements are called "plain text", "cipher text" and "key";

• L is a non-empty finite set called „alphabet string obtained from the encryption key";

• $g : K \rightarrow L^+$ is a pseudo-random string generator: for

$$\forall k \in K, g(k) = k_1k_2k_2 \ldots \in L^+ \qquad (1)$$

is a set of encryption key obtained from the (theoretically infinite);

• $\forall z \in L$ there is an encryption rule $e_z \in E$ and a decryption rule $d_z \in D$ so:

$$\forall x \in P, d_k(e_k(x)) = x \qquad (2)$$

Encryption process fluid with a synchronous system can be represented as an automaton described by relations:

$$q_{i+1} = \delta(q_i, k), zi = g(q_i, k), y_i = h(z_i, x_i) \quad (3)$$

where q_0 is the initial state - which can be determined from the key k, δ is the state transition function, g is the function that produces encryption string day, and h is the function that produces cipher text output y_i based of plaintext x_i and the encryption string z_i.

• A block encryption scheme can be viewed as a particular case of string encryption, which $\forall i \geq 1, z_i = K$.

Self-synchronized systems:

An encryption system is self-synchronized fluid (or "synchronous") where the key generation function depends on a fixed number fluid character previously encrypted.

So such a system behavior can be described by the equations:

$$q_i = (y_{i-t},\ldots,y_{i-1}), z_i = g(q_i, k), y_i = h(z_i, x_i) \quad (4)$$

where $q_0 = (y_{-t}, y_{-t+1}, \ldots, y_{-1})$ is the initial (known), k is the key, g is the function key generating fluid and h is the output function encrypts the plaintext x_i. The most popular systems are self-synchronized with the linear response registers, used to generate sequences of pseudo-random numbers (cryptography) and to generate cyclic codes [1].

II. POSSIBLE SOLUTIONS FOR ENCRYPTION / DECRYPTION SYNCHRONIZATION PROCESS

When the original information to be protected is digital, the synchronization process of encryption / decryption can be done in ways that lead to self-synchronize or synchronize systems.

Encryption/decryption synchronization process through methods which leads of synchronized systems:

This implies that the information divisions submitted by the equipment are recognized in the same form at the reception [2]. For this reason the strategy and implementation of reliable synchronization of this process is of prime importance.

The main components of a communications links include: telephony terminals, multiplexers / switches, BEU = Bulk Encryption Unit, radio-relays (cable or fiber) and telephone handsets. A bundle including such equipment is shown in Fig. 1.

There are possible solutions to the synchronization process of encryption / decryption equipment used between pairs of the same type (e.g. between radio relay between multiplexers), these methods are usually standardized and not subject to this work. Further discussion will be restricted to possible solutions for synchronization process encryption / decryption which refers to the class encryption equipment, which are optional and communications links are used to protect information exchanged.

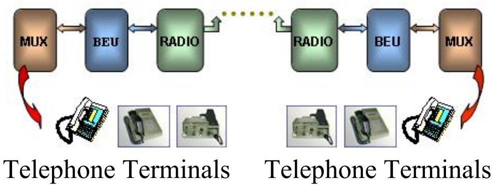


Figure 1. Typical communication system

BEU is equipment that is usually placed between the multiplexers and radio relays at both ends of the link (radio-relays may not be used if the connection is through cable or fiber optic) devices are also called as the group encryption coding information of the group. Speed data flow process can vary from 256 kbps to 2048 kbps (for primary PCM multiplex) to 32 Mbps (for the third multiplex). But encryption devices are commonly used only for primary multiplexing.

There are two methods commonly used for synchronization between BEU:

- *permanent synchronization*;
- *initial synchronization.*

<u>Permanent synchronization</u>: consists in the exchange of information for the synchronization between BEUs. This method has the great advantage of the independence process of synchronization to other equipment because the timing is always observed and restored without outside intervention.

<u>Permanent synchronization has two variants</u>:

The first option assumes that there are unused bits in the data stream processing. These bits can be used for synchronization. The performance of this method are dictated by the percentage of free bits in the range of existing data. If this percentage is small, the synchronization process is slow and therefore inefficient. The second way to achieve permanent synchronization of data flow speed is increased by adding periodic timing information. In this way synchronization bit rate can be chosen to give a reasonable time synchronization. The big disadvantage of both methods is permanent synchronization hardware complexity because it involves decoding the sequence data structure, free bits identifying and processing or, alternatively, dividing the data stream into chunks and their alternate synchronization information for the composition of these clubs flow at higher speed. These operations involve memory 'elastic' and 'special' hardware and that the costs are high.

<u>Initial synchronization</u>: is a much simpler method that uses as a basis for synchronization information provided by other equipment in the system, particularly the

multiplexer. The principle of this method is based on an continuous synchronization between the multiplexers, which allows synchronization on the basis of the information provided BEU. In each case the notes are out of sync in the multiplexers, encryption means that the process must be restarted BEU, so multiplexers resynchronization command to give a BEU. After completion of BEU synchronization, synchronization can be achieved and multiplexers. BEU initial synchronization is performed only at the beginning of encryption and the information is then kept watch, transmitted or retrieved from the data string. This method requires less complex equipment and is therefore much cheaper. Encryption subscriber equipment (e.g. telephone handsets) may use the same principles to synchronize the process of encryption / decryption as encryption equipment group.

Synchronization process methods that lead to self-synchronized systems:

It is considered as an example system block encryption: Cipher FeedBack or CFB [3]. CFB and form a cryptographic algorithm encryption / decryption of strings as equivalent to a self-synchronized algorithm for block ciphers. The sequence of operations for this encryption system is shown below:

$$C_i=E_K(C_{i-1}) \oplus P_i;$$
$$P_i=E_K(C_{i-1}) \oplus C_i;$$
$$C_0=IV. \qquad (5)$$

If a ciphertext block cipher is completely lost, the receiver will resynchronize CFB, combined with a shift register can be used as input for block ciphers.

To transform in a CFB encryption string equivalent to a self-sync cryptographic algorithm that will sync to any multiple of x bit lost, then we will start by initializing a shift register with an initialization vector (IV) size block cipher. This will be encrypted as a block cipher and the most significant x bits of the result will be modulo-two (XOR) with the most significant bit plain text x to produce x-bit cipher text. These x-bit output will then be transferred to shift register and the encryption process is repeated for the next x-bit identical to a plain text (Figure 2).

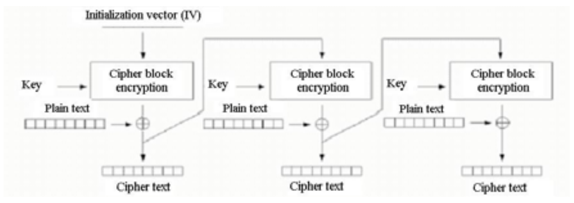


Figure 2. CFB encryption

The decryption process will be similar to the encryption and begin initialization vector, modulo summing operation of encryption and two (XOR) with the most significant x bits of the ciphertext output, resulting in the output x-bit plain text. Then, the x-bit cipher text will be transferred to shift register and the process will continue in the same way (Figure 3).

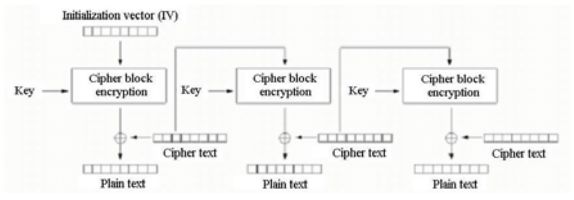


Figure 3. CFB decryption

The decryption process will be similar to the encryption and begin initialization vector, modulo summing operation of encryption and two (XOR) with the most significant x bits of the ciphertext output, resulting in the output x-bit plain text. Then, the x-bit cipher text will be transferred to shift register and the process will continue in the same way:

$$C_i=head(E_k(S_{i-1}),x) \oplus P_i;$$
$$P_i=head(E_k(S_{i-1}),x) \oplus C_i;$$
$$S_i= ((S_{i-1} \ll x) + C_i) \bmod 2^n;$$
$$S_0= IV. \qquad (6)$$

If the decryption is lost x bits of the ciphertext, the plain text can not be reconstructed correctly until the shift register will not again have a state equivalent to that used in encryption, it is time resincronizării block cipher. This will cause the output to a maximum length of a cipher block size, the data can not be decrypted correctly. If the plain text will be an error slip, it will propagate encryption ciphertext and therefore will not be subject to the parallelization process. Conversely, decryption can be parallelized. When performing a decryption of the ciphertext changes will affect two plain text blocks: a change of one bit in a plain text block will be completely corrupted and the next block cipher. Normally, the following plain text block ciphers can be decrypted correctly. Features of CFB encryption system:

• Block ciphers are used only for encryption (decryption is used for all encryption function)

• posts not be filled with additional bits to form multiples of the cipher block size (so that the operations "of fill (padding) to be unnecessary).

CFB encryption system requires an initialization vector IV to be used as initial input block. IV should not be mandatory secret, but it must be unpredictable (unpredictable generated).

CFB system is defined as follows:

<u>La criptarea CFB:</u>

$I_1=IV$; $I_j=LSB_{b-s}(I_{j-1})|C^{\#}_{j-1}$, pentru $j = 2 \ldots n$;
$O_j=CIPH_k(I_j)$, pentru $j = 2 \ldots n$;
$C^{\#}_j= P^{\#}_j \oplus MSB_s(O_j)$, pentru $j = 2 \ldots n$. (7)

<u>La decriptarea CFB:</u>

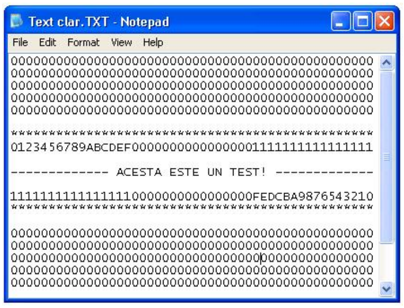
$I_1=IV$; $I_j=LSB_{b-s}(I_{j-1})|C^{\#}_{j-1}$, pentru $j = 2 \ldots n$;
$O_j=CIPH_k(I_j)$, pentru $j = 2 \ldots n$;
$P^{\#}_j = C^{\#}_j \oplus MSB_s(O_j)$, pentru $j = 2 \ldots n$. (8)

III. SOFTWARE EMULATOR

1. Launch AES_CFB1_TEST.exe an emulation of the CFB-AES 128/128 test algorithm [4].

2. Enter two times as 32 hexadecimal characters in the set {"0", "1", ... "9", "A", "B" ... "F"} in the boxes reserved for the work key and initialization vector.

3. Select a plain text file (of any type, text, image, video, executable, etc.)..



```
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000

************************************************
0123456789ABCDEF00000000000000001111111111111111

------------- ACESTA ESTE UN TEST! -------------

1111111111111111000000000000000 0FEDCBA9876543210
************************************************

000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
```

Figure 4. A plain text file example

4. Create a file for saving the encrypted file (eg "text criptat.cri).

5. Click "Encrypt", wait until you see the message "Gata" (Ready) and click OK.

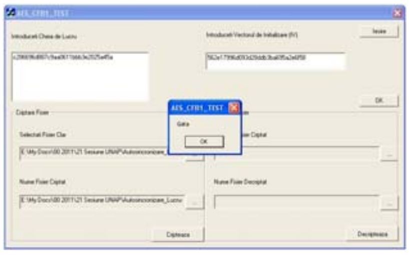

AES_CFB1_TEST
Introduceti Cheia de Lucru
Introduceti Vectorul de Initializare (IV)
Iesire
OK
AES_CFB1_TEST
Gata
OK
Cripteaza
Decripteaza

Figure 5. Example of encryption using the emulator AES_CFB1_TEST.exe

6. View encrypted file.

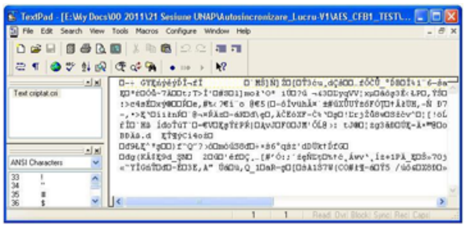


Figure 6. Cipher file for the plain text file example

7. To decrypt encrypted select the same file as input.

8. Create a file to decrypt the file is saved / restored (eg "text decriptat.txt).

9. Click "Decrypt", wait until you see the message "Gata" (Ready) and click OK.

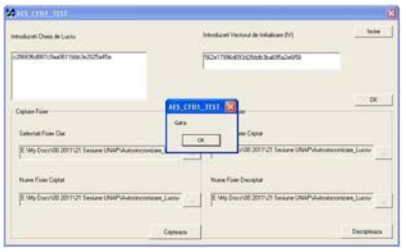
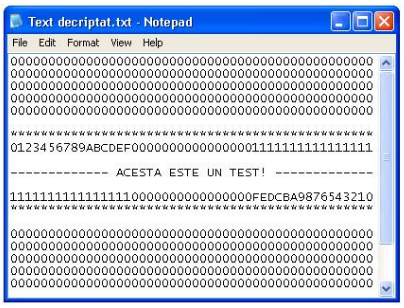


Figure 7. Example of the AES_CFB1_TEST.exe decryption using the emulator

10. View decrypted file / recovered („text decriptat.txt").


Text decriptat.txt - Notepad

```
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000

************************************************
0123456789ABCDEF00000000000000001111111111111111

------------- ACESTA ESTE UN TEST! -------------

1111111111111111000000000000000OFEDCBA9876543210
************************************************

000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
```

Figure 8. Decrypted/recovered text file

11. The encrypted file ("text criptat.cri) is operating three changes that simulate two types of errors as:

- change a few bits of" 1 "to" 0 "and / or vice versa, which means a rate of error bit occurrence BER≠0 (i.e. change a character M in a character N: ... □˙**M**Š] ... → ... □˙**N**Š] ...);

- some bits are deleted, which signifies the emergence of a sliding bit to the left (to watch) (e.g. delete two characters ... :>**c4**sÉ ... → ... :>sÉ ...);

- add a few bits, which means the occurrence of landslides bit to the right (to watch) (eg add 2 characters: ... ôƒmó ... → ... ôƒ**iP**mó...);

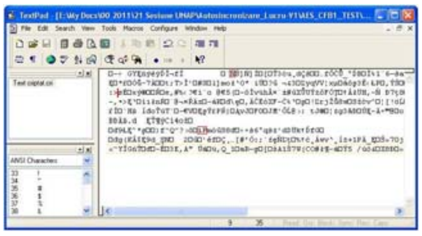
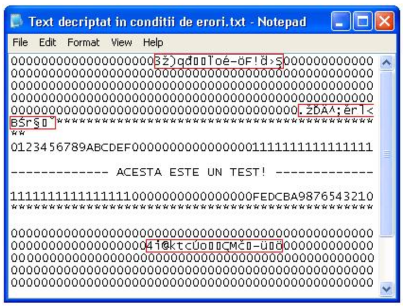


Figure 9. Cipher text file modified to simulate two types of errors

12. Resume decrypt the encrypted file as an input selecting modified to simulate two types of errors considered.

13. View decrypt the file / recovered ((„text decriptat in conditii de erori.txt").


Text decriptat in conditii de erori.txt - Notepad

```
000000000000000000003ž)qđ□□Ťoé–öF!ď›Ş000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000.žĎÄ^;ěrŤ<
BŚr§□˘*******************************************
**
0123456789ABCDEF00000000000000001111111111111111

------------- ACESTA ESTE UN TEST! -------------

1111111111111111000000000000000OFEDCBA9876543210
************************************************

000000000000000000000000000000000000000000000000
00000000000000000004ĭ@ktcÚo□□ÇMč□–ü□ő000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
```

Figure 10. Decrypted text file / restored to the appearance of two types of errors

IV. CONCLUSIONS

The main conclusions about string encryption synchronous systems are:

• (relative to the timing): synchronous encryption systems, equipment butt must synchronize the encryption string to obtain an encryption / decryption correctly. If during transmission are insert or remove bits in the encrypted, then decryption fails, and it can be resumed only on the basis of resynchronization techniques (such as for example resetting the demarcation and placement of specific elements in the text sent at regular intervals).

• changing a bit in the encrypted text (do not remove or add anything) will not affect the decryption of other characters (retarding error).

• an active opponent deleted, inserted, or return the encrypted message components, will result in eleven days and will therefore be detected at the reception. Also, he can make changes to the text encrypted and will be able to see how they will affect the plaintext. So a text encrypted with an encryption system requires synchronous separate mechanisms to ensure authentication and data integrity.

The main conclusions about self-synchronized systems encryption strings are:

• (self-synchronization): since the decryption function h^{-1} depends only on a fixed number of characters previously encrypted, eleven days - possibly resulting inserting or deleting characters encrypted - it can be avoid. Such encryption schemes can restore synchronization property affecting only a finite number of characters in plain text.

• limited error propagation: If the status of a self-synchronized system encryption previous character strings depends on t, then the change (possibly deleting or inserting) a character in the encrypted text can lead to incorrect decryption t characters maximum, then falling back decryption correct.

• dissemination of clear text: As each character in the plaintext affects all encrypted text that follows, any statistical properties are dispersed through the plaintext encrypted. So from this point of view, self-synchronized encryption systems are more robust than those based on synchronized redundant plaintext attacks.

• active Resistance to cryptanalysis: the above properties it is quite difficult to detect an attack came from an active opponent (which may modify, insert or delete characters) self-sync restoring normal decryption phase. Therefore, additional mechanisms are necessary to ensure authentication and data integrity.

Effects of errors for example self-synchronized system of CFB encryption strings:

A bit error is the substitution of a '0' bit for a '1' bit, or vice versa.

In the CFB mode, bit errors in a ciphertext segment affect the decryption of the next *b/s* (rounded up to the nearest integer) ciphertext segments. A bit error may occur, independently, in any bit position in these decrypted segments, with an expected error rate of fifty percent.

Consequently, for the CFB mode, the decryption of the first ciphertext block is vulnerable to the (deliberate) introduction of bit errors in specific bit positions of the IV if the integrity of the IV is not protected.

The same property also holds for the ciphertext segments in the CFB mode; however, for every ciphertext segment except the last one, the existence of such bit errors may be detected by their randomizing effect on the decryption of the succeeding ciphertext segment.

Table 1 summarizes the effects of bit errors in a ciphertext block or IV on the decryption of the ciphertext.

Tab. 1. Summary of Effect of Bit Errors on Decryption

Mode	CFB
Effect of Bit Errors in C_j	SBE in the decryption of C_j RBE in the decryption of $C_{j+1}, \dots, C_{j+b/s}$
Effect of Bit Errors in the IV	RBE in the decryption of $C_1, C_2, \dots, C_j$ *for j with values between 1 şi b/s*

RBE: random bit errors, i.e., bit errors occur independently in any bit position with an expected probability of ½.

SBE: specific bit errors, i.e., bit errors occur in the same bit position(s) as the original bit error(s).

The deletion or insertion of bits into a ciphertext block (or segment) spoils the synchronization of the block (or segment) boundaries; in effect, bit errors may occur in the bit position of the inserted or deleted bit, and in every subsequent bit position. Therefore, the decryptions of the subsequent ciphertext blocks (or segments) will almost certainly be incorrect until the synchronization is restored. When the 1-bit CFB mode is used, then the synchronization is automatically restored b+1 positions after the inserted or deleted bit. For other values of s in the CFB mode, and for the other confidentiality modes in this recommendation, the synchronization must be restored externally.

REFERENCES

1. Ion Angheloiu, "Coding theory".

2. Eugen Borcoci, "Digital Switching Systems".

3. Morris Dworkin, "Recommendation for Block Cipher Modes of Operation - Methods and Techniques" (paper presented at the NIST Special Publication 800-38A, 2001), Chapter 6.3 - The Cipher Feedback Mode, pp. 11-13.

4. Shihong Wang, Huaping Lü, Gang Hu, "A new self-synchronizing stream cipher" (paper presented at Beijing, China, January 11, 2005), Chapter III, Principles to construct efective SSSC, pp. 9-12