CIPHERING WITH CHAINED DISCRETE CHAOTIC MAPS

Ing. Rostislav HUČKA, Doctoral Degreee Programme (3)
Dept. of Radio Electronics, FEEC, BUT
E-mail: hucka@feec.vutbr.cz

Supervised by: Prof. Vladimír Šebesta

ABSTRACT

Direct chaotic transformation offers very effective way to encrypt and decrypt data. This paper describes the usage of more discrete chaotic maps to create an algorithm, which is able to use long keys and variable key length. Designed algorithm has very good ratio between complexity and security.

1 INTRODUCTION

The original idea [2] of ciphering with discrete chaotic maps used only one chaotic system created from a skew-tent map with a special quantisation described in the same article too. There is used a discrete chaotic transformation defined by the map. There is required to repeat whole chaotic transformation for several times due security reasons. This system can be only hardly modified to use long enough keys (128 bit and more) or even variable length of keys.

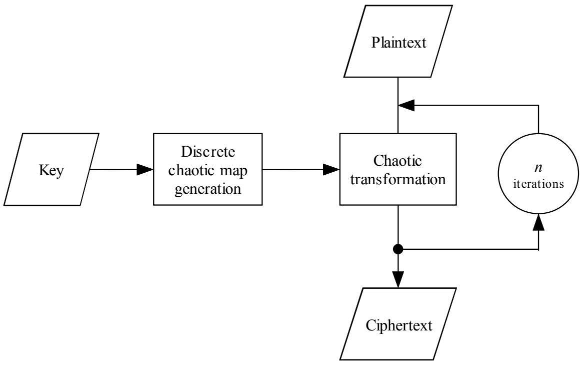

Fig. 1: *The original ciphering algorithm*

This algorithm could be modified to reach the goals above according following scheme (fig. 2). Instead usage of the same chaotic transformation in several iterations we can create more chaotic processes and use them one after another. Those chaotic transformations are created according same scheme, but with different subkeys, so there is required to split long input key to suitable number of suitable long keys. There is also required to split input data into segments, each segment comes through the ciphering process separately. This approach to plaintext was also used in the original cryptosystem design too.

The choice of the segment size has pivotal impact on the realization of the whole ciphering process. For a small length of the segments is possible to use tables to perform calculations, so the overall speed of the entire process is rapidly increased. Every chaotic transformation is performed by simple indexation instead much more complicated calculation and rounding from a chaotic map. Further the usage of the tables allows us to use almost any discrete chaotic map, which defines 1:1 transformation, so it assigns exactly one output value to each input value. The set of input and output values is identical too. Arbitrary ciphering table can be easily numerically inverted, so we can use even those methods of discrete chaotic map generation, which prevents us to perform an analytic map inversion [1].

We have to also design the algorithm to perform enough chaotic transformations to make final correlation between plaintext and ciphertext as low as is required for security.

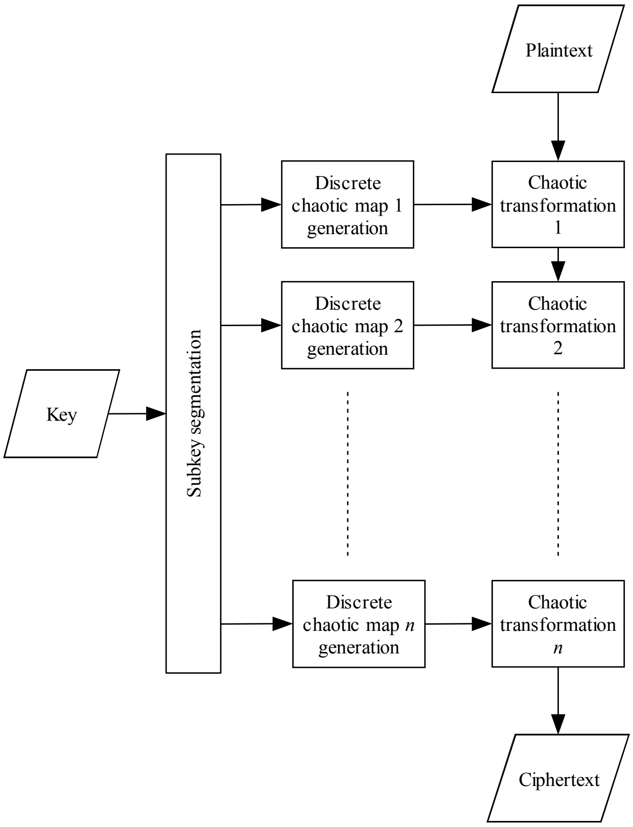
Plaintext
Key
Subkey segmentation
Discrete chaotic map 1 generation
Chaotic transformation 1
Discrete chaotic map 2 generation
Chaotic transformation 2
Discrete chaotic map *n* generation
Chaotic transformation *n*
Ciphertext

Fig. 2: *A new ciphering algorithm, which uses chaotic transformation chaining*

2 CIPHERING PROCESS

For the chaotic transformation in each step can be used arbitrary discrete chaotic map M, which meets following conditions:

$$\begin{gathered} M : X \to Y, X, Y \in \{1, 2, ..., m\} \\ y = M(x) \\ \forall x \in \{1, 2, ..., m\}, x \neq M(x) \end{gathered} \qquad (1)$$

Every used map is created according subkey k_i, which has to have the same size for every used map. Generally there can be used also maps generated in different ways, the only condition is the same size of the subkey, but in the one map could be used also more than one subkey.

Plaintext is splitted into segments and each segment passes through the ciphering algorithm independent on others. The choice of the segment size is essential for further design of algorithm. To allow the optimal performance is good to fulfill

$$m = 2^{8k},\ k \in \mathbf{Z} \qquad (2)$$

If it isn't possible, then is necessary to add some padding into segmentation phase, but it is also a certain performance hit. The choice of the segment size also depends on the chaotic map, which is used to perform chaotic transformation. If we can't create inverted discrete chaotic map $\widetilde{M}$ satisfying (3) analytically,

$$x = \widetilde{M}(M(x)) \qquad (3)$$

so we have to realize numeric inversion. Considering memory requirements for tables is highly recommended to limit the segment size to 24-bit or less. If we calculate both forward and reverse chaotic transformation directly (not using tables), the segment size could be larger.

The ciphering process for one segment p_i is described below

$$c_i = M_1(M_2(\ldots M_n(p_i))) \qquad (4)$$

And the reverse deciphering process

$$d_i = \widetilde{M}_1(\widetilde{M}_2(\ldots \widetilde{M}_n(c_i))),\ \text{where} \qquad (5)$$

p_i plaintext segment

c_i ciphertext segment

d_i decrypted data segment ($p_i = d_i$)

$\forall j \in [1..n],\ x \in \{1, 2, \ldots m\},\ x = \widetilde{M}_j(M_j(x))$

3 PRACTICAL EXPERIMENT

For the first numeric realization was chosen the segment size not so large, only 16-bit. Whole algorithm then doesn't require much memory for ciphering tables. If we use 256-bit keys, the overall required memory space is only 2 MB. Due table usage there was chosen a 16-bit segmentation for subkeys too. Each discrete chaotic map M_j and $\widetilde{M}_j$ is created according [2].

$$\begin{gathered} M_j(x) = \left\lceil \tfrac{m}{k_j} x \right\rceil,\ x \in [1, k_j] \\ M_j(x) = \left\lfloor \tfrac{m}{m-k_j}(m - x \right\rfloor + 1,\ x \in (k_j, m] \end{gathered} \qquad (6)$$

$$\begin{gathered} \widetilde{M}_j(x) = x_1,\ g(x) = x,\ \tfrac{x_1}{k_j} > \tfrac{m-x_2}{m-k_j} \\ \widetilde{M}_j(x) = x_2,\ g(x) = x,\ \tfrac{x_1}{k_j} \leq \tfrac{m-x_2}{m-k_j} \\ \widetilde{M}_j(x) = x_1,\ g(x) = x + 1 \\ x_1 = \left\lfloor \tfrac{1}{m} k_j x \right\rfloor \\ x_2 = \left\lceil (\tfrac{k_j}{m} - 1)x + m \right\rceil \\ g(x) = x + \left\lfloor \tfrac{k_j x}{m} \right\rfloor - \left\lceil \tfrac{k_j x}{m} \right\rceil + 1 \end{gathered} \qquad (7)$$

Despite the chosen maps are available inverted maps as analytic formula, there were used the tables. Without tables the performance of this algorithm is poor so it conflicts with the general requirements. If we use tables the performance of this algorithm as almost the same as in original algorithm with one key, one table and many iterations.

4 CONCLUSION

Resulting source code is written in C language and it is freely available [5] under GPL license. No special optimization of I/O operations was done in the source code in order to increase performance. The usage of chained chaotic processes has positive impact (fig. 3) to lowering of the correlation coefficient between plaintext and ciphertext in dependence to number of iterations (and overall key length too).

Correlation coefficient between plaintext and ciphertext decreases faster than in original ciphering algorithm. The small increase of the correlation coefficient for 9 and more iteration is a parasite effect, which was also discussed in [2]. It also occurs in an algorithm with only one chaotic map used .

Ciphering algorithm with chained chaotic maps compared to original algorithm with only one chaotic map has low sensitivity to small change of the key. If there was only one bit in the key changed and same plaintext was ciphered with those two keys, there was correlation coefficient between those different ciphertexts greater than 0.9, what is a security risk. This problem can’t be avoided even with a different choice of the used chaotic maps. But correction still can be performed using key modification. Whole key can be ciphered with itself (input key is used both as key and as plaintext) and the resulting ciphertext as key for ciphering user data.

For the overall performance analysis there was measured dependence of the average ciphering speed on number of iterations (and key length). The period necessary to generate tables was included in the measured time. Tables necessary to generate tables is constant (depends only on computer speed) and the period of ciphering itself depends on the number of used chaotic maps. Measured algorithm used both 16-bit subkey and data segment. Performance of this algorithm was done on PC with AMD Athlon XP 2600+ CPU.

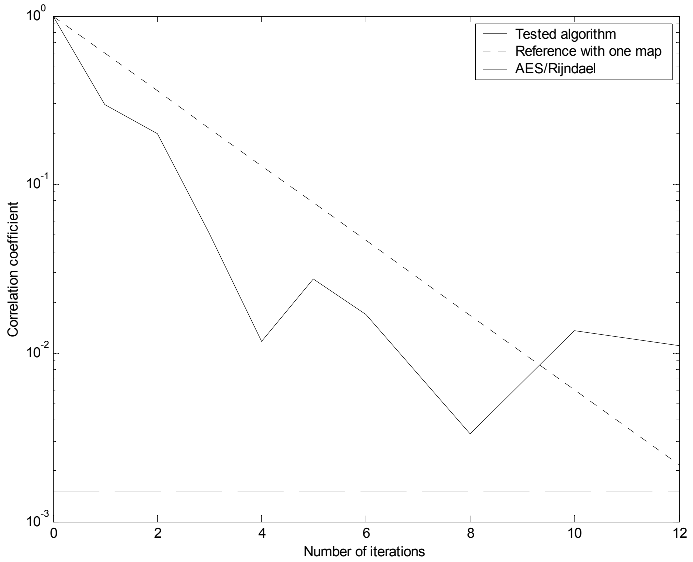

Fig. 3: *Dependence of the correlation coefficient between plaintext and ciphertext on number of iterations*

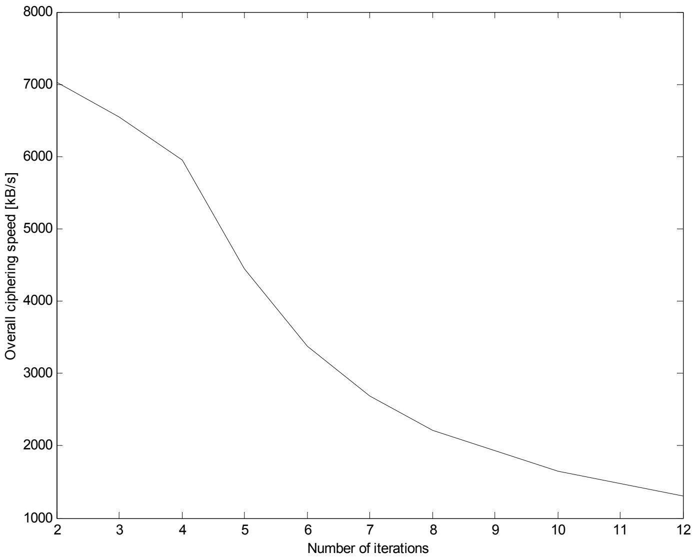

Fig. 4: *Overall ciphering algorithm performance*

ACKNOWLEDGEMENTS

This research has been supported by the research grant GACR (Grant Agency of Czech Republic) No. 102/04/0557 "Development of the digital wireless communication resources." The paper has been prepared as a part of the solution of FRVŠ project No. 1624/2004.

REFERENCES

[1] Hučka, R.: New Approach to Design of the Nonlinear Digital Chaotic Maps. In Radioelektronika 2003 Conference Proceedings. Radioelektronika 2003. Brno, Czech Republic: Institute of Radio Electronics, 2003, p. 103 - 107. ISBN 80-214-2383-8.

[2] Masuda, N., Kazuyuki, A.: Cryptosystems With Discretized Chaotic Maps. In IEEE Transactions on Circuits and Systems - I. 2002, vol. 49, no. 1, 28-40.

[3] Kennedy, M. P., Rovatti, R., Setti, G.: Chaotic Electronics in Telecommunications. New York, CRC Press, 2000, ISBN 0-8493-2348-7.

[4] Schneier, B.: Applied Cryptography. New York, John Wiley & Sons, Inc., 1996. ISBN 0-471-11709-9

[5] Hučka, R.: Využití chaotických map v kryptografii [online]. Brno: FEKT VUT v Brně, 2003. Dostupný z WWW: <http://wes.feec.vutbr.cz/~hucka>.