The turmite-based cryptographic algorithm

The turmite-based cryptographic algorithm

L O Fadieieva, I Ye Makarenko and P V Merzlykin

Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine

E-mail: ipmcourses@gmail.com

Abstract. The novel turmite-based cryptography algorithm has been designed and implemented. The turmites ability to generate pseudo-random number series makes them promising for cryptographic applications. At the same time, most turmites-related researches concentrate on their mathematical properties and generally don't consider possible applications. Lack of effective implementations of turmites-based cryptographic algorithms makes this research topical. The properties of the proposed algorithm have been examined. The frequency analysis resistance and avalanche criterion have been estimated. The results demonstrate that turmites-based algorithms may be used in cryptography and this application deserves attention and further examination.

1. Introduction

The number of computer crimes grows constantly. A lot of people use Internet-banking, e-mail, social networks, and electronic passports, which hold personal information. Smartphones and computers keep synchronized with cloud storage documents, often owned by a third party, so users security depends on the third party integrity. Recent years events show that personal information leaks may cause material and moral damage for owners. Hence nowadays, information protection issues go far beyond the military and corporate scope. Virtually everyone, consciously or unconsciously, deals with data and personal information, which may be abused and compromised. Therefore, cryptography is presently one of the leading scopes of Computer Science.

Until now, cryptography has been employed to ensure the confidentiality of messages (that is, encryption) via message transformation from a clear into an encoded form and vice versa. Thus, it will be impossible for a third party to read the message without secret knowledge (namely, the key required to decrypt the message. In the first two decades of the 21 century, the scope of cryptography has expanded and now includes not only the secret message transmission, but the sender identification and recipient authentication, digital signatures, interactive verification, secure communication techniques, etc.

A turmite is a Turing machine that has infinite two-dimensional cells array, current state, and orientation in space. Turmites may be divided into those with relative and absolute orientation. Relative turmites have their internal orientation. Program instructions change their orientation relatively: “left”, “forward”, “right”, and “turn around”. An example of such a turmite is Langton’s Ant [9].

Let’s consider the rules of the Ant movement. We have endless plane with black and white square cells and the Ant situated in some cell. At each step it can move in any one of the four neighboring cells. The Ant moves according to the following rules:

- at a black square turn 90° counter-clockwise, change the cell color to white, then move forward one square;

- at a white square turn 90° clockwise, change the cell color to black, then move forward one square.

These simple rules cause quite complicated behavior: after a certain period of quite random moves, the Ant always begins to build a pattern of 104 steps, which is repeated infinitely, regardless of the initial coloring of the field [3].

The quality of the encryption algorithm depends mainly on its strength, i.e. cryptanalysis resistance. If a successful attack requires unattainable computational resources, unattainable number of overheard messages or unattainable time, then such an algorithm is considered to be strong. In most cases, it is impossible to prove mathematically the algorithm strength, while it's often possible to prove the cryptographic algorithm vulnerability.

2. Results and discussion

2.1. The algorithm designs

Despite the fact that Langton's Ant have been a research subject for a long time [1], [2], [6], [7], [9], it is rarely considered to be used in cryptography. However, some researches imply that it generates usable pseudorandom number sequences [4], [8].

While studying the possibility of Langton's Ant cryptography application, we have found the only implementation [10], but it has a number of shortcomings. This implementation takes into account just digital images encryption and tends to modify primarily less significant bits (LSB). It may cause the serious cryptographic strength reduction. Furthermore, while multiple Ants act simultaneously, one of them can annihilate another.

Evidently, the cryptographic Langton's Ant application has not been adequately studied). Turmites have great potential in cryptography, and need further study. Taking into account the aforesaid, the following requirements for our algorithm have been introduced:

- the algorithm should be capable of encrypting across-the-board data;
- the algorithm should change all bits with the same probability, not only LSB;
- the data should be encrypted by several turmites;
- each turmite should have its area and not block the actions of others.

Our implementation of the algorithm has the following scheme:

- split the file with input data into blocks of the same size;
- translate each block into a bit matrix that will be used as a turmite's field (bit values "0" and "1" represent two cell colors of the original Langton's algorithm);
- select randomly the starting positions of the turmites;
- further, according to the rules of the Langton's ant movement, perform the bitwise encryption of data.

We implemented the periodic boundary conditions, so after jumping over block's edge, turmite appears at the opposite side. It ensures the Ant's unfettered run on the plane.

The algorithm of creating the keys deserves particular attention since it depends on several parameters and its size increases in proportion to the file size growth. The encryption key to one turmite consists of the starting position, the starting orientation, and the number of steps. The decryption key contains the final position, the final orientation, and the number of performed steps. The encryption key is partially defined by user, while the decryption key is calculated by algorithm.

Figure 1 shows the algorithm for the decryption key.

2.2. Experimental results

The examination of the algorithm's properties allowed us to determine the optimal encryption parameters. The first thing we investigated was the number of Ant's steps, since it is known that after a certain period of rather random motion Langton's Ant always begins to build a recurrent pattern of 104 steps that repeats indefinitely, regardless of the initial field state [3].

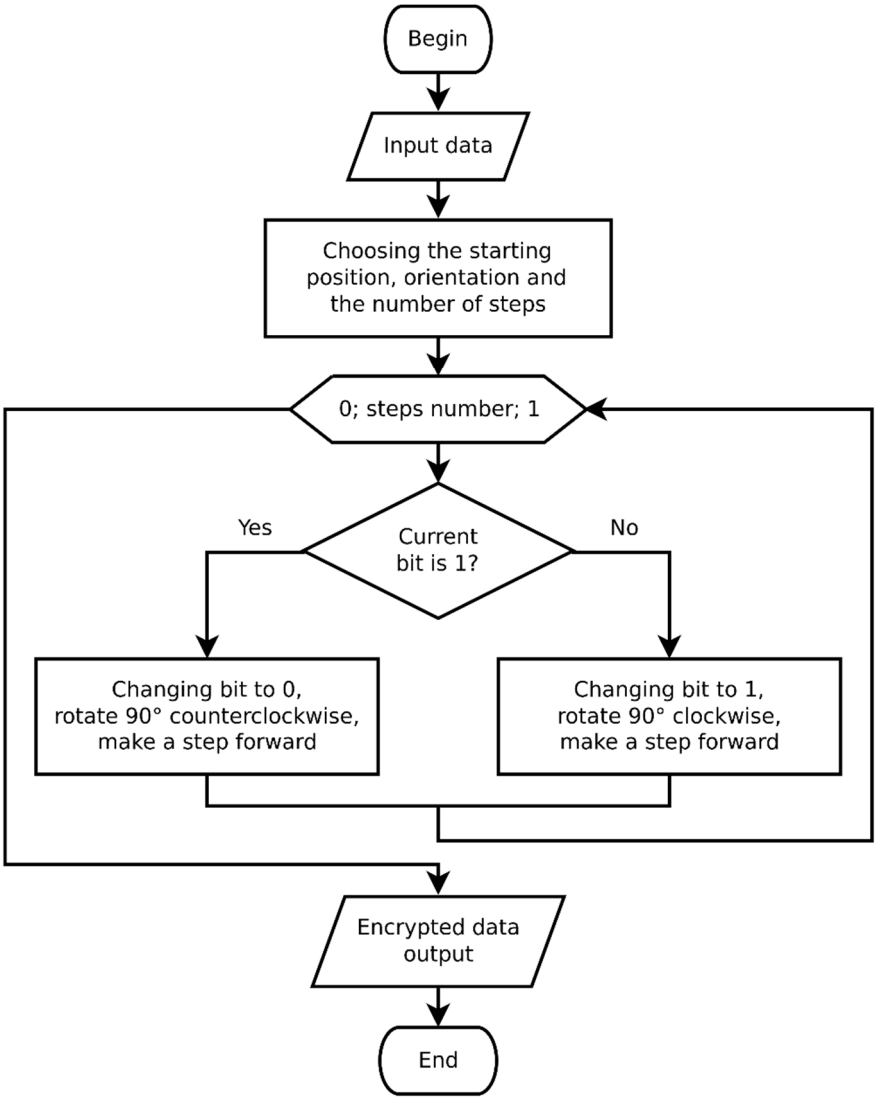

Figure 1. The flowchart of the algorithm.

To test this phenomenon, the bit change matrix has been introduced. We took blocks of different sizes and checked how many steps in each block should be done to encrypt about 80% of the data. For more accurate results, the test has been applied to multiple files of different types.

Figure 2 shows the dependence of steps number on block size.

It can be seen that steps number increases with block size growth. And this increasing is more intensive for blocks larger than 100 bits. Therefore, for encryption it is preferable to use blocks of size fewer than 100 bits.

Now let's test how the number of modified bits varies depending on the quantity of steps. We introduce some coefficient n, which shows the ratio of the encrypted bits number to the number of steps. It will help estimate the optimal block size, since the more n is, the more efficiently algorithm works.

The table 1 below shows the dependence of the modified bits number on the number of steps for different block sizes.

In order to see whether the Ant moves all over the block or its movement localized at some area, it have been built the matrix of modified bits that shows which bits were visited by Ant.

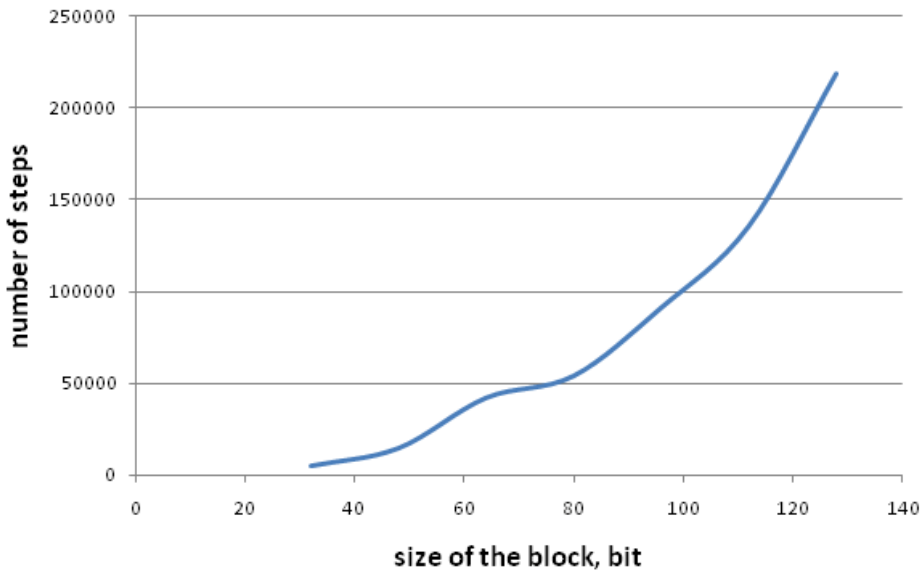

Figure 2. The dependence of the number of steps on the size of the block.

Table 1. The dependence of the modified bits number on the number of steps for different block sizes.

Block size	The number of modified bits	The number of steps	*n*
32 × 32	819	5104	0.16
48 × 48	1843	14774	0.12
64 × 64	3276	42128	0.08
80 × 80	5120	54072	0.09
96 × 96	7372	91155	0.08
112 × 112	10035	135439	0.07
128 × 128	13107	218522	0.06

Figures 3–4 show three matrices of modified bits for different types of data (from left to right: text, graphics, and sound). Black color marks the areas visited by Ant, and white cells indicate unvisited areas.

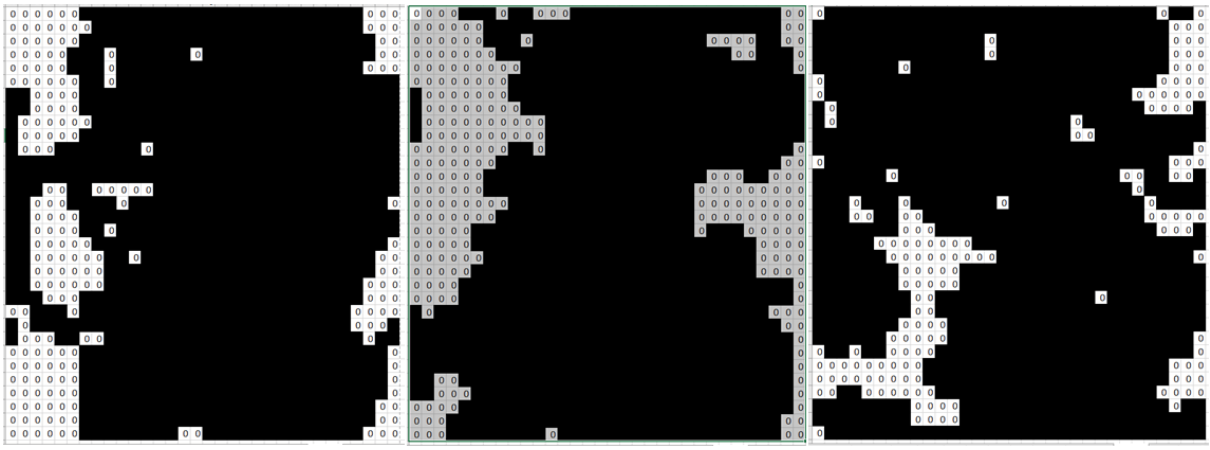

Figure 3. The matrices of modified bits distribution for block size 32 × 32.

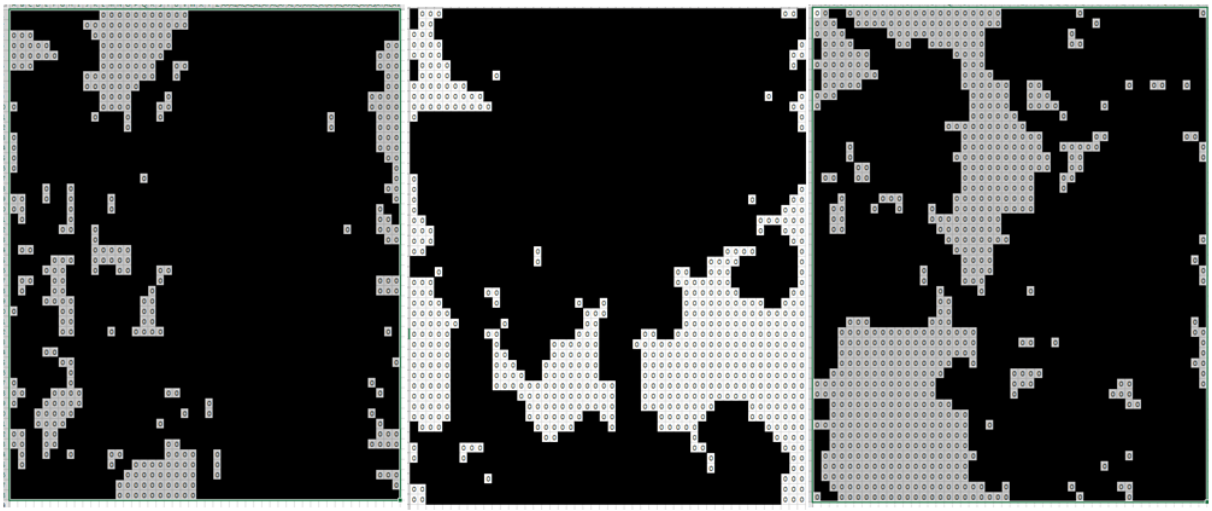

Figure 4. The matrices of modified bits distribution for block size 48 × 48.

As it is seen, that the Ant moves all over the block rather than concentrating in one area.

Therefore, we may conclude that the step number and the block size are dependent on each other. The smaller the block, the fewer step number we need to cover it. The most effective appears to be the 32 × 32 block because it has the largest value of *n* parameter, namely 0.16. Hence, for further research we used the 32 × 32 block and 5104 steps.

To examine the quality of encryption, we considered the encrypted data byte distributions. It is known that the letter frequency in languages corresponds to the normal distribution, and it often becomes the vulnerable place in cryptographic algorithms. So, for security issues, it is necessary that the character distribution in the output file corresponds to the uniform distribution. To check the byte distribution, we used the Pearson's chi-squared test.

As a result, the Pierson's criterion equals 1.05 for encrypted text files, 3.46 for graphic files, and 1.03 for sound files. It is much less than the critical value, which is more than 15. Therefore, we can accept the hypothesis that the distributions are uniform.

We also calculated the correlation coefficient to ensure that there is no dependence between input and output data. The correlation coefficient is -0.03 for text files, 0.67 for image files, and 0.1 for audiofiles. Hence, we can infer that there is correlation between input and output data stored in graphic files.

To resolve this issue, we increased twofold the step number and calculated the Pearson's criterion and correlation coefficients again. As a result, we obtained the following values of Pearson's criterion: 0.98 for text files, 1.47 for images, and 1.07 for audio. It is much less than the critical value, which is more than 15. Therefore, the distributions can be considered uniform. The new values of correlation coefficient were -0.03 for text files, 0.01 for images, and -0.06 for audio. The obtained results allow us to imply that there is no correlation between input and output data, and the output data corresponds to the uniform distribution. Therefore, the algorithm should be resistant to frequency analysis.

Afterwards, we have tested if the algorithm satisfies the avalanche criterion [5]. It does if all output bits change with a probability of 50%, provided a single input bit is changed. Figure 5 shows the bit difference matrix for two encrypted files that had just one different bit before encryption. In this matrix, "1" (black cell) means that the bit is different in two files after encryption, and "0" means that the bit is the same in both files.

From figure 5 we can see that two similar files after encryption have 92 equal bits, which is 18% of file size. To satisfy the avalanche criterion, we had to slightly modify the algorithm. To make the Ant's movement random, we decided to choose its initial position and direction randomly instead of determining by user's key. In that way, we implemented the cryptographic salt.

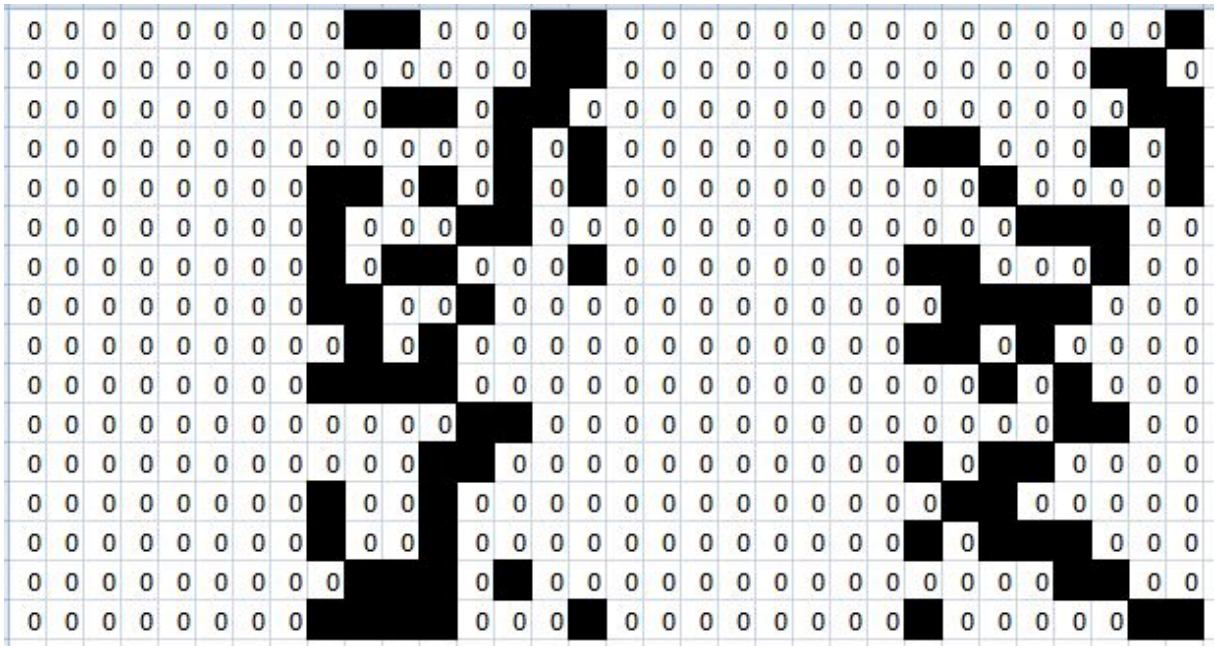

Figure 5. The bit difference matrix.

The new bit difference matrix is shown in figure 6.

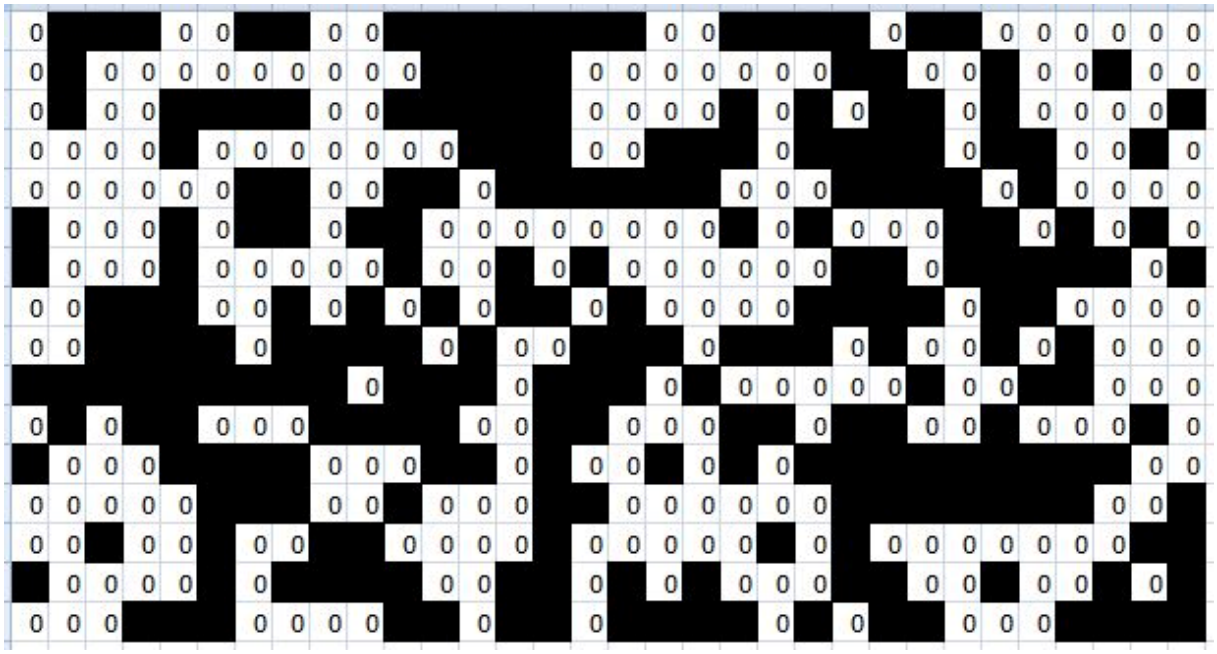

Figure 6. The bit difference matrix after cryptographic salt implementation.

From figure 6 we can see that two similar files after encryption have 239 equal bits, which is 47% of file size. It means that the algorithm meets the avalanche criterion.

3. Conclusions

The analysis of the current cryptographic turmite application shows that the issue remains topical. The attempts to implement the algorithms reveal a number of shortcomings that call into question their

practical usability. Taking it into account, we introduced the functional requirements to our algorithm. The Qt library has been chosen for software implementation because of its cross-platform features and sufficient performance due to optimization capabilities of the C++ compiler.

The analysis of algorithm's properties allowed us to optimize its parameters, such as block size and steps number. The algorithm resistance to the frequency analysis and the avalanche criterion satisfaction have been tested, as well. In the future, the algorithm may be improved by implementing Turmites with different movement rules.

References

[1] Beuret O and Tomassini M 1998 Behaviour of Multiple Generalized Langton's Ants *Proceedings of the Artificial Life V Conference* ed Langton C and Shimohara K pp 45-50 URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.179&rep=rep1&type=pdf

[2] Brady A H 1988 *The Universal Turing Machine: A Half-Century Survey* (Oxford: Oxford University Press)

[3] Darling D 2004 *The Universal Book of Mathematics: From Abracadabra to Zeno's Paradoxes* (New York: John Wiley & Sons) pp 180–1

[4] Dirgová Luptáková I and Pospíchal J 2015 How Random Is Spatiotemporal Chaos of Langton's Ant? *Journal of Applied Mathematics, Statistics and Informatics* **11** 5–13 URL https://doi.org/10.1515/jamsi-2015-0008

[5] Fadieieva L O and Merzlykin P V 2018 The avalanche criterion satisfaction research of the turmite-based cryptographic algorithm *CEUR Workshop Proceedings* **2292** 83–6

[6] Gajardo A, Goles E and Moreira A 2002 Complexity of Langton's ant *Discrete Applied Mathematics* **117** 41–50 URL https://doi.org/10.1016/S0166-218X(00)00334-6

[7] Hamann H, Schmickl T and Crailsheim K 2011 Thermodynamics of emergence: Langton's ant meets Boltzmann *IEEE Symposium on Artificial Life (ALIFE)* pp 62–9 URL https://doi.org/10.1109/ALIFE.2011.5954660

[8] Hosseini S M, Hossein K and Jahan M V 2011 From Chaos to Random Behavior, Generating Random Numbers by Cellular Automata Confusion *CSC'11, The 2011 International Conference on Scientific Computing* pp 262–8

[9] Langton C G 1986 Studying artificial life with cellular automata *Physica D: Nonlinear Phenomena* **22** 120–49 URL http://doi.org/10.1016/0167-2789(86)90237-X

[10] Wang X and Dahai X 2014 A novel image encryption scheme using chaos and Langton's Ant cellular automaton *Nonlinear Dynamics* **79** 2449–56 URL https://doi.org/10.1007/s11071-014-1824-0