UDC 004.056.55+004.421.5

Knight Pseudorandom Walk Manifold for Scrambling Multidimensional Data

Vadim Romanuke
Professor, DrSc
ORCID: 0000-0001-9638-9572
v.romanyuk@vtei.edu.ua
romanukevadimv@gmail.com

Vinnytsia Institute of Trade and Economics
of State University of Trade and Economics

Keywords:
data scrambling;
knight pseudorandom walk;
break-in probability;
similarity rate.

The knight open tour problem is to build a sequence of knight moves covering a chessboard completely, without repetitions, where the starting position and ending position are always different. A solution of the knight open tour problem is similar to a sequence of pseudorandom numbers used to map data into non-readable yet usable information. The knight open tour problem has a manifold of solutions for a starting position of the knight depending on the chessboard size. Solutions of the knight open tour problem, which appear like a random series of knight positions, i. e. a pseudorandom walk, are used to further improve balance of the scrambling simplicity and productivity, where the main indicators are the break-in probability and similarity index. The break-in probability is dramatically decreased by taking into account a knight pseudorandom walk manifold for a starting position on a given chessboard. A pessimistic estimation of the break-in probability for an 8×8 chessboard is less than 10^{-16}. A similarly expected estimation for an $8\times8\times8$ chessboard is less than 10^{-26}. A distinct knight pseudorandom walk (out of a manifold of pseudorandom walks) is built online by a given seed integer for a pseudorandom number generator. The scrambled data vector is built online as well in linear runtime complexity. Meanwhile, the similarity index is acceptable, rapidly dropping as the chessboard size is increased (for bigger multidimensional data arrays). A knight pseudorandom walk is determined by the chessboard size, the starting position, the way to vectorize the knight pseudorandom walk, and the pseudorandom number generator seed allowing to specifically move the knight onward through situations with multiple possible moves of the knight. The knight-open-tour scrambler has 10^{16} to 10^{21} times lower break-in probability compared to an ordinary pseudorandom number generator, depending on the chessboard size and the starting position of the knight.

DOI: 10.31558/2786-9482.2024.1.1

Introduction

The main technical purpose of data privacy is to achieve sufficiently low likelihood of deliberately intruding, spoofing, hacking, delaying, distorting, etc. [1, 2]. In particular, data is scrambled by altering or shuffling its entries in accordance with an algorithm whose return is difficult to decipher while the scrambled data is still usable for legitimate demonstration [3, 4]. Within personal and limited access use, the algorithm must be unknown for attackers, or at least it must be unlikely to determine it within reasonable time [5, 6]. Another plausible requirement is to maintain the algorithm as simple as possible due to its frequent application should not affect the operation speed [7, 8].

Basically, the requirement of scrambling algorithm simplicity often prevents from using shufflers based on pseudorandom number generators [9]. The latter utilize rather complex routines for generating a stream of pseudorandom numbers, but the main reason is the seed change [10, 11]. Although the range for the initial value of a pseudorandom sequence is sufficiently wide, the information about the seed change should be constantly sent to the recipient that increases the likelihood of break-in, if the type of the pseudorandom number generator is known [12]. On the other hand, cipher algorithms and cryptographic hashes are high-quality pseudorandom number generators, but generally they are considerably slower than non-cryptographic random number generators [13, 14]. Therefore, it is still desirable to develop a much wider list of simple and fast pseudorandom number generators for tasks of scrambling, regardless of the publicity of its use.

Goal and tasks

Issuing from the growing demands for security and reliability of data privacy and storage policy, the goal is to suggest a new scrambling technique which would provide improved cryptographic properties compared to a peer scrambling technique. The improved cryptographic properties imply decreasing the likelihood of break-in along with decreasing similarity between the initially given data and the scrambled data.

One of the simplest algorithms having been studied for possible scrambling is based on building solutions to the knight open tour problem [15, 16]. In fact, there are a few such algorithms, both exact and heuristic, which have slightly differing computational efficiency and coverage (some solutions may be omitted by a heuristic, whereas they are found by an exact approach, although not always within a reasonable time). A solution of the knight open tour problem is similar to a sequence of pseudorandom numbers used to map data into non-readable yet usable information. The first task is to describe the solution application. The second task is to estimate its main computable parameters for the worst-case scenario compared to an ordinary generator of pseudorandom numbers. The break-in probability and similarity rate will be estimated under the same case along with estimating operation speed and other limitations.

Pseudorandom walk

Denote by

$$\mathbf{U} = [u_I]_F \tag{1}$$

an N-dimensional array of data, where

$$F = \bigtimes_{n=1}^{N} M_n \tag{2}$$

is the format of the array,

$$I = \{i_n\}_{n=1}^{N} \tag{3}$$

is its indexation with indices

$$i_n \in \left\{\overline{1, M_n}\right\} \quad \forall n = \overline{1, N}, \tag{4}$$

M_n is the length of dimension n, $M_n \in \mathbb{N} \setminus \{1\}$, and u_I is a numerical or symbolical entry indexed by (3), (4). The total number of entries in array (1) is:

$$L = \prod_{n=1}^{N} M_n . \tag{5}$$

A scrambler s must take only the starting position:

$$I_1 = \left\{i_n^{(1)}\right\}_{n=1}^{N} \text{ by } i_n^{(1)} \in \left\{\overline{1, M_n}\right\} \quad \forall n = \overline{1, N} \tag{6}$$

for an algorithm A, whereupon the scrambled data is an array

$$\mathbf{Y} = s\left(\mathbf{U}, A, I_1\right). \tag{7}$$

Obviously, the number of entries of array (7) is (5), but its format may be changed from (2) into

$$G = \bigtimes_{b=1}^{B} K_b \tag{8}$$

by

$$L = \prod_{b=1}^{B} K_b . \tag{9}$$

So,

$$\mathbf{Y} = [y_J]_G \tag{10}$$

is the scrambled B-dimensional data array of format (8) with indexation

$$J = \{j_b\}_{b=1}^{B} \tag{11}$$

and indices

$$j_b \in \left\{\overline{1, K_b}\right\} \quad \forall b = \overline{1, B}, \tag{12}$$

by dimension b of length K_b, $K_b \in \mathbb{N} \setminus \{1\}$, where

$$\{y_J\}_{\forall J} = \{u_I\}_{\forall I} . \tag{13}$$

In particular, $B = 1$, i. e. array (1) is scrambled into a vector (7) of length (9), where

$$\mathbf{Y} = \left[y_j\right]_{1\times L} \ (j = \overline{1, L}). \tag{14}$$

Another possibility, less applicable than (14), is dimensionality reduction. For instance, an $M_1 \times M_2 \times M_3$ matrix (that can represent some color image for $M_3 = 3$) is scrambled into an $M_1 \times M_2 M_3$, $M_1 M_2 \times M_3$, or $M_1 M_3 \times M_2$ matrix, regardless of possible transposition. In vectorizing-based applications, an $M_1 \times M_2 \times M_3$ matrix data is scrambled into a $1 \times M_1 M_2 M_3$

vector (obviously, a column vector is also possible).

The knight open tour problem whose solutions appear like a random series of knight positions [15, 16], i. e. a pseudorandom walk, is a promising approach to scrambling. The knight starts its open tour at a horizontal position x_1 and a vertical position y_1 on a chessboard of size $C \times C$, where $x_1 \in \left\{\overline{1, C}\right\}$ and $y_1 \in \left\{\overline{1, C}\right\}$. Then the knight moves onward according to an algorithm. In this way, the chessboard is completely covered by the knight, and a definite sequence of C^2 integers indicating the successive positions of the knight is formed. This sequence can be formed in two ways, each of which has two versions. If the chessboard squares are enumerated by the number of the knight move, then a matrix

$$\mathbf{B} = [c_{ik}]_{C \times C},$$

where c_{ik} is the number of the knight move at chessboard square $\{i, k\}$ ($c_{ik} = 1$ for $i = y_1$, $k = x_1$), is vectorized either by concatenating its columns or concatenating its rows. On the other hand, a vector

$$\mathbf{M} = \left[m_j\right]_{1 \times C^2}$$

is formed, where either

$$m_j = C \cdot (i - 1) + k \text{ for } c_{ik} = j \tag{15}$$

or

$$m_j = C \cdot (k - 1) + i \text{ for } c_{ik} = j. \tag{16}$$

So, the knight pseudorandom walk can be represented either by the vectorized matrix $\mathbf{B}$ or vector $\mathbf{M}$. It is easy to see that the two versions of the vectorized matrix $\mathbf{B}$ are directly connected. Knowing one version, the other one is obtained via a simple permutation pattern. The two versions of vector $\mathbf{M}$ by (15) or (16) are directly connected as well, where one version is obtained from the other one via another simple pattern which is the inverse to either (15) or (16).

Scrambler estimations and parameters

The break-in probability can be estimated as follows. The knight open tour problem has a great deal of solutions for a starting position of the knight depending on the chessboard size $C \times C$. Denote this number by $S_A(C; x_1, y_1)$. There are C^2 starting positions on the chessboard of size $C \times C$. For a definite chessboard size and a definite starting position, there are four versions of the pseudorandom walk for a distinct solution out of $S_A(C; x_1, y_1)$ solutions. Hence, an estimation of the break-in probability is

$$P_{\text{break-in}} = \frac{1}{4C^2 \cdot S_A(C; x_1, y_1)}. \tag{17}$$

On an 8×8 chessboard, for instance, there are more than 10^{14} knight open tours for a starting position, i. e. $S_A(8; x_1, y_1) > 10^{14}$ [17]. Therefore, the break-in probability (17) is

$$P_{\text{break-in}} = \frac{1}{4 \cdot 8^2 \cdot S_A(8; x_1, y_1)} = \frac{1}{256 \cdot S_A(8; x_1, y_1)} < 10^{-16} \quad (18)$$

for an 8×8 chessboard. The ending position of the knight cannot be $\{x_1, y_1\}$ but distinct open tours can end at the same position. However, number $S_A(C; x_1, y_1)$ depends on the starting position. If it is closer to a corner of the chessboard, this number is less than the number of knight open tours starting closer to the chessboard center. This is explained by the knight at a corner has fewer ways to move onward. Thus, many tours have a zigzag pattern on the chessboard borders (Figure 1). If the chessboard size is increased, the pattern may remain (Figure 2).

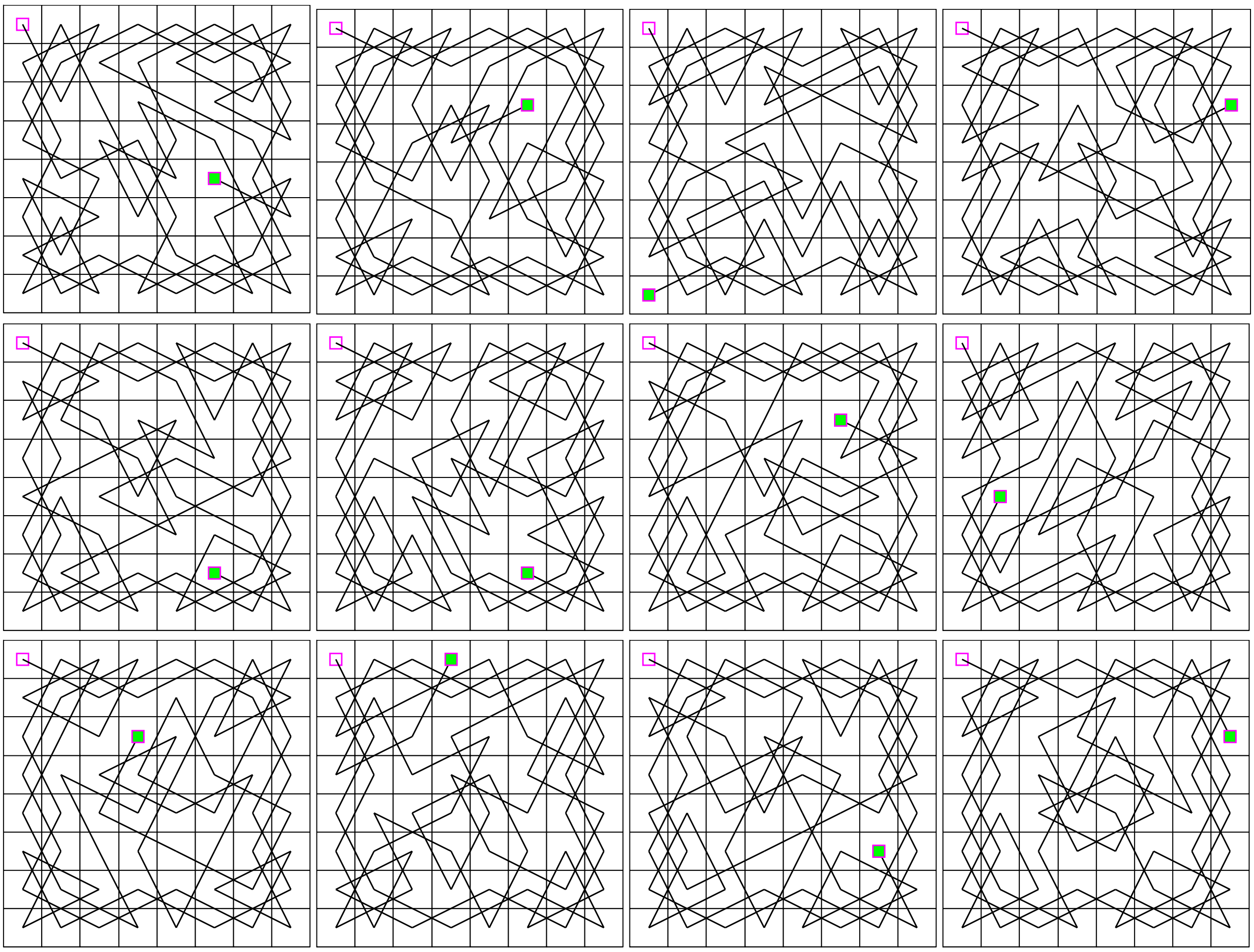

Figure 1. A set of 12 distinct pseudorandom walks on an 8×8 chessboard, each of which starts at the same square (from the top left corner), where the zigzag border pattern is seen

Nevertheless, the zigzag pattern becomes less influential on bigger chessboards. It is quite apparent by comparing pseudorandom walks in Figure 2 to those in Figure 1. This means it is more likely that number $S_A(2C; x_1, y_1)$ varies relatively less across all possible $4C^2$ starting positions on a $(2C) \times (2C)$ chessboard than, say, number $S_A(C; x_1, y_1)$ varying across all possible C^2 starting positions on a $C \times C$ chessboard. Therefore, bigger chessboards are more efficient to produce higher rates of randomness.

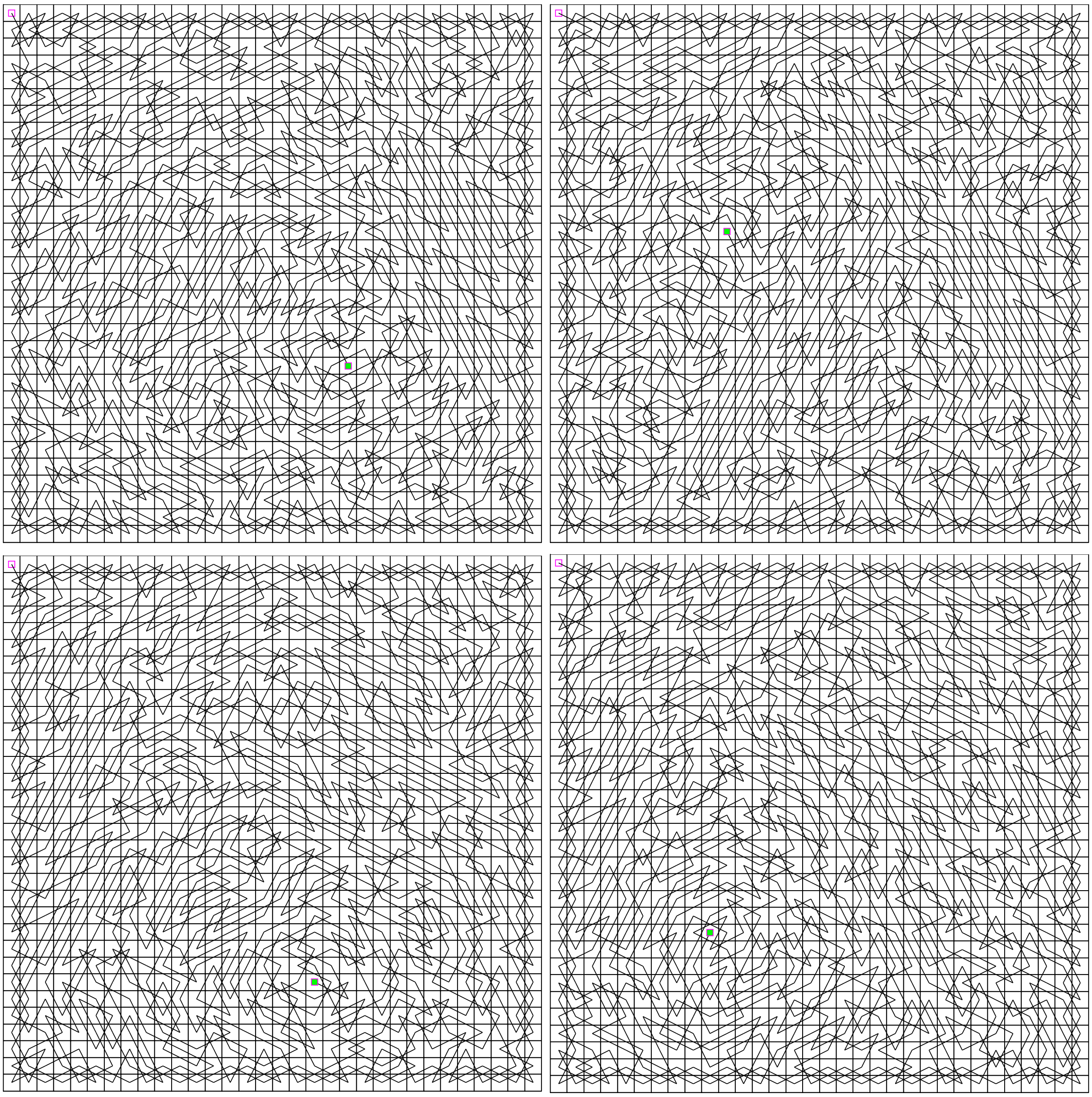

Figure 2. A set of four various pseudorandom walks on a 32×32 chessboard, each of which starts from the top left corner, where the zigzag border pattern is still seen reminiscent of that in Figure 1

So, starting an open tour off one of four corners, either close to it or peculiarly at chessboard squares $\{1, 1\}$, $\{1, C\}$, $\{C, 1\}$, $\{C, C\}$, is the worst-case scenario for break-in probability (17). In such a case, the break-in probability is higher than in any other case. Compared to shuffler-based scrambling, it is important to know a ratio of break-in probability (17) to the shuffler-based scrambling probability (which, obviously, is L^{-1} as the shuffler algorithm is presumed to be known and it can defined by the position of an integer value from 1 to L). This ratio, called the break-in probability rate (BPR) for further consideration, must be estimated under the worst-case scenario with using (18) as reference value. Other important parameters are the similarity index and computation time ratio. The latter is calculated separately for scrambling and descrambling as a ratio of knight-open-tour time to shuffler time. The respective scrambling time rate (STR) and descrambling time rate (DTR) must be close. The similarity index is calculated as the element-wise number of coincidences in multidimensional data array (1) and the scrambled data array (10) divided by L. This index is written as a knight-open-tour scrambler rate (KTSR) and a shuffler scrambler rate (SSR), whereupon a similarity rate (SR) is calculated as a ratio of KTSR to SSR.

Computational experiments are carried out on a single CPU Intel Core i5-7200U@2.50GHz in Matlab 2018a. The data are intended to emulate streaming images, so for this purpose integers between 0 and 255 are generated by a pseudorandom number generator with a known seed. Table 1 presents the mentioned parameters for signed 16-bit integers between 0 and 255 in multidimensional data arrays for $L = C^2$ (here and below the comparative rates are averaged over 100 repetitions), where it is clearly seen that the knight-open-tour scrambler and shuffler scrambler have roughly the same operation speed. SR varies badly, though, for smaller sizes of the chessboard. The estimation of BPR is made with respect to (18), where the tour starts at chessboard square $\{1, 1\}$. Approximately the same parameters hold for numbers with single precision storage (Table 2) and numbers with double precision storage (Table 3) almost independently of the data dimensionality (in Matlab, vectorization or reshaping arrays is executed within a few microseconds), where the numbers are generated starting at the same seed of the pseudorandom number generator. Nevertheless, KTSR and SSR are a few times smaller than those in Table 1.

Overall, the scrambler is defined by four parameters: the chessboard size, the starting position, a specific integer σ_A determining one of $S_A(C; x_1, y_1)$ open tours, and one of four ways to obtain a $1 \times C^2$ vector representing the knight pseudorandom walk. Integer σ_A could be called the walk seed and it is a number between 1 and $S_A(C; x_1, y_1)$:

$$\sigma_A \in \left\{\overline{1, S_A(C; x_1, y_1)}\right\}.$$

This integer predetermines a distinct knight pseudorandom walk by setting the pseudorandom number generator seed to a definite integer. Algorithm A building the open tour online frequently stumbles over situations when there are a few possible onward moves of the knight. One of such moves is further selected by generating a random value and comparing a threshold to this value.

Table 1. Comparative rates for signed 16-bit integers between 0 and 255

C	10	20	30	40	50	60	70	80	90	100	110	120	130	140
KTSR	0.0339	0.0064	0.0046	0.0047	0.0051	0.0042	0.0043	0.0042	0.0042	0.0042	0.004	0.004	0.004	0.004
SSR	0.0134	0.0063	0.005	0.0045	0.0044	0.0039	0.004	0.0041	0.004	0.004	0.004	0.004	0.004	0.004
SR	2.5298	1.0278	0.922	1.044	1.1799	1.0676	1.0672	1.0194	1.0583	1.0474	1.0006	1.0059	1.0046	0.9925
STR	1.007	1.1648	0.9676	1	0.99	0.9961	0.9717	0.9799	0.9953	0.9637	0.9786	0.9966	0.9672	0.9845
DTR	1.0863	1.1381	1.062	1.0022	0.9934	0.957	1.0126	0.9986	0.9858	1.0097	1.008	1.0123	0.9979	1.0134
BPR	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}

Table 2. Comparative rates for numbers with single precision storage

C	10	20	30	40	50	60	70	80	90	100	110	120	130	140
KTSR	0.03	0.0025	0.0011	0.0006	0.0012	0.0003	0.0004	0.0002	0.0002	0.0003	0.0001	0.0001	0.0001	0.0001
SSR	0.0102	0.0027	0.001	0.0007	0.0004	0.0002	0.0002	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
SR	2.9411	0.9345	1.0638	0.9615	2.9411	1.1627	1.9607	1.0309	1.7857	3.2258	1.1111	0.9345	0.8849	1.8867
STR	1.0208	1.0489	0.9749	1.0002	0.9649	0.9911	0.988	0.9781	0.9782	0.9829	0.9705	0.9802	0.9667	0.9877
DTR	1.1964	1.0257	1.0046	0.9985	1.0009	0.9953	1.0032	1.0192	1.0039	1.004	1.0063	0.9963	1.0081	1.0128
BPR	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}

Meanwhile, a scrambling technique can be applied multiple times. Thus, another quadruple of scrambler parameters should be assigned. Table 4 presents the comparative rates for numbers with double precision storage by double scrambling with the same chessboard size, where the tour at the second stage scrambling starts at chessboard square $\{4, 4\}$. In general, these rates do not much differ from those in Tables 1–3, but BPR now is much better. KTSR and SSR are comparable to those in Tables 2 and 3. However, if the second stage scrambling chessboard is of size $(C/2)\times(C/2)$, then KTSR is improved in about six times (Table 5). In addition, STR and DTR, varying between 0.5802 and 0.7547, are much favorable for the knight-open-tour scrambler. Its break-in probability increases, though, due to the smaller second stage scrambling chessboard. The decrement is hardly noticeable, anyway. This is an acceptable tradeoff for decreasing similarity along with speeding up the scrambling (descrambling) process.

Table 3. Comparative rates for numbers with double precision storage

C	10	20	30	40	50	60	70	80	90	100	110	120	130	140
KTSR	0.03	0.0025	0.0011	0.0006	0.0012	0.0003	0.0004	0.0002	0.0002	0.0003	0.0001	0.0001	0.0001	0.0001
SSR	0.0102	0.0027	0.001	0.0007	0.0004	0.0002	0.0002	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
SR	2.9411	0.9345	1.0638	0.9615	2.9411	1.1627	1.9607	1.0309	1.7857	3.2258	1.1111	0.9345	0.8849	1.8867
STR	1.0555	1.0086	0.9855	1.0048	0.9898	0.9811	0.988	0.9822	0.9726	0.9973	0.974	0.9701	0.9859	0.9931
DTR	1.2022	1.0001	1.0542	0.9997	1.0339	0.9759	0.9969	1.0064	1.0168	0.9984	1.0122	1.0023	0.9902	1.0099
BPR	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-16}	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}

Table 4. Comparative rates for numbers with double precision storage by double scrambling

C	10	20	30	40	50	60	70	80	90	100	110	120	130	140
KTSR	0.02	0.0025	0	0.0019	0.0008	0.0017	0.0002	0.0005	0.0005	0.0005	0.0003	0.0002	0.0004	0.0001
SSR	0.0095	0.0026	0.0014	0.0006	0.0004	0.0003	0.0002	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
SR	2.1052	0.9615	0	2.9126	1.8181	6.0606	1.0101	2.9126	4.4943	4.8076	3.8834	2.9702	6.0606	0.909
STR	1.1184	1.0018	0.9893	0.9946	0.9772	0.9718	0.9729	0.9886	0.9756	0.9822	0.9916	0.9753	0.9861	0.9838
DTR	1.1708	0.9912	1.0369	1.0016	0.9986	0.9978	1.0037	0.9987	1.0128	0.9912	1.0044	1.001	0.9956	1.0101
BPR	10^{-18}	10^{-18}	10^{-19}	10^{-19}	10^{-19}	10^{-19}	10^{-19}	10^{-20}	10^{-20}	10^{-20}	10^{-21}	10^{-21}	10^{-21}	10^{-21}

Table 5. Comparative rates for numbers with double precision storage by double scrambling with $C \times C$ and $(C/2)\times(C/2)$ chessboards

C	10	20	30	40	50	60	70	80	90	100	110	120	130	140
KTSR	0	0	0	0.0013	0.0008	0.0003	0.0002	0.0002	0.0005	0.0005	0.0003	0.0003	0.0002	0.0002
SSR	0.0098	0.0025	0.0011	0.0006	0.0004	0.0003	0.0002	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
SR	0	0	0	2.2471	2.1052	1.0869	0.9523	0.9009	4.4943	4.8543	3.9603	3.9215	4.1666	3.0612
STR	0.7197	0.6731	0.6375	0.6593	0.6351	0.6246	0.6135	0.5992	0.5976	0.6063	0.5927	0.5931	0.5859	0.5802
DTR	0.7547	0.6669	0.6742	0.6499	0.6252	0.6331	0.6288	0.6164	0.6127	0.6047	0.5895	0.5983	0.5855	0.5811
BPR	10^{-17}	10^{-17}	10^{-18}	10^{-18}	10^{-18}	10^{-18}	10^{-18}	10^{-19}	10^{-19}	10^{-19}	10^{-20}	10^{-20}	10^{-20}	10^{-20}

It is noteworthy that the chessboard can be three-dimensional [18]. This leads to further decreasing the break-in probability. Indeed, the knight open tour problem has a far greater deal of solutions for a starting position of the knight on a chessboard of size $C \times C \times C$. For a depth position z_1 on a chessboard of size $C \times C \times C$, where $z_1 \in \left\{\overline{1, C}\right\}$, denote the number of solutions starting off position (cube) $\{x_1, y_1, z_1\}$ by $S_A(C; x_1, y_1, z_1)$. There are C^3 starting positions on the chessboard of size $C \times C \times C$. For a definite chessboard size and a definite starting position, there are four versions of the pseudorandom walk for every face of the chessboard. Having six faces, the number of versions of the pseudorandom walk is $6 \cdot 4^C$, and a raw estimation of the break-in probability is

$$P_{\text{break-in}} = \frac{1}{6 \cdot 4^C \cdot C^3 \cdot S_A(C; x_1, y_1, z_1)} \tag{19}$$

being much lower than (17) for the same number C. However, estimation (19) is made for a one sequence of C layers of the three-dimensional chessboard (successively from layer 1 to layer C, going through a face). Altogether there are $C!$ such sequences. So, a more precise estimation is

$$P_{\text{break-in}} = \frac{1}{6 \cdot 4^C \cdot C! \cdot C^3 \cdot S_A(C; x_1, y_1, z_1)}. \tag{20}$$

It is expected that an $8 \times 8 \times 8$ chessboard has far more than 10^{14} knight open tours for a starting position, so $S_A(8; x_1, y_1, z_1) > 10^{14}$ at least. Therefore, the break-in probability (20) is

$$P_{\text{break-in}} = \frac{1}{6 \cdot 4^8 \cdot 8! \cdot 8^3 \cdot S_A(C; x_1, y_1, z_1)} = \frac{1}{8117488189440 \cdot S_A(8; x_1, y_1, z_1)} < 10^{-26} \tag{21}$$

for an $8 \times 8 \times 8$ chessboard. However, computing knight open tours on a three-dimensional chessboard may run into known computational issues [15, 16, 19], where searching for a specific knight pseudorandom walk may become intractable (when it cannot be completed within reasonable amount of time) due to a significantly deep “path” [20]. After all, the existence of solutions of the knight open tour problem on three-dimensional chessboards has not been proved yet for any size.

Discussion

Given a seed integer for a pseudorandom number generator, a distinct knight pseudorandom walk (out of a manifold of pseudorandom walks) is built online. This means that the scrambled data vector is built online as well, i. e. every next knight move is immediately followed by moving an entry of array (1) to a specific place. As any knight open tour algorithm has linear runtime complexity [15], the knight-open-tour scrambler does not make it any (significantly) longer than other scramblers.

Square chessboards are better than non-squarely-shaped chessboards as they contain richer manifolds of knight pseudorandom walks. Indeed, cornered start positions of the knight produce poorer manifolds. A square chessboard has the fewest corner-like start positions. Contrary to that, a rectangular chessboard has more corner-like start positions closer to the shorter side. Say, a horizontally stretched chessboard has corner-like start positions closer to the left and right sides because the knight is more “squeezed” there having fewer possible moves upwards and downwards than in the direction to the center.

The estimations of BPRs in Tables 1–5 may look too pessimistic, but they are considered as a “lowest” bound of the gain in security provided by the knight-open-tour scrambler. This bound might have been much bettered for bigger chessboards, but the number of the knight open tour problem solutions is itself an open question for such chessboards. However, if to step aside a little from the worst-case scenario, by assuming that another 10 squares along the chessboard dimension decreases the BPR by the factor of 2, a BPR estimate for $C = 140$ is 10^{-20} to 10^{-22}. It is also likely that the estimations by (18) and (21) are better in most non-corner-like cases. Owing to using the ratios, the reported comparative rates are expected to be independent of the hardware. However, it is worth noting that the real-time operation speed of the knight-open-tour scrambler has a limit being determined by the hardware performance. Thus, 19600 double-precision values are scrambled within 150 milliseconds on a single CPU Intel Core i5-7200U@2.50GHz, which is 7.975 Mb/s in terms of the speed. This means that the speed of streaming data intended for scrambling must not exceed 7.975 Mb/s for such a hardware. This limit, being the worst-case scenario, keeps roughly the same owing to the linear runtime complexity of the knight open tour algorithm. Moreover, the limit is quite comparable to the speed of shuffler-based scrambling and scrambling by pseudorandom binary numbers generated with using linear-feedback shift registers. Nevertheless, no limits exist for data privacy and storage purposes, where the data are stationary and the only intention is to store it securely protected.

Whichever the chessboard size is, the number of chessboard squares must be not fewer than the data length by (5). Obviously, the redundant chessboard squares are not used in the case when a multidimensional data array has fewer entries than the number of chessboard squares.

Conclusion

Multidimensional data array (1) is scrambled into array (7) by starting position (6) and algorithm A, whereupon the scrambled data array (10) has format (8) with indexation by (11)–(13). To further improve balance of the scrambling simplicity and productivity, solutions of the knight open tour problem are used. The break-in probability is dramatically decreased by taking into account a knight pseudorandom walk manifold for a starting position on a given chessboard. The number of possible solutions is gigantic for an 8×8 chessboard, let alone bigger chessboards. Thus, a pessimistic estimation of the break-in probability for an 8×8 chessboard is less than 10^{-16}. A similarly expected estimation for an $8\times8\times8$ chessboard is less than 10^{-26}. Meanwhile, the similarity index is acceptable, rapidly dropping as the chessboard size is increased (for bigger multidimensional data arrays).

Compared to an ordinary generator of pseudorandom numbers, the knight-open-tour scrambler shuffles data also, having the same computational efficiency, but it has 10^{16} to 10^{21} times lower break-in probability depending on the chessboard size and the starting position of the knight. A knight open tour problem solution, also referred to as a knight pseudorandom walk, is determined by the chessboard size, the starting position, the way to vectorize the knight pseudorandom walk, and the pseudorandom number generator seed allowing to specifically move the knight onward through situations with multiple possible moves of the knight.

From the practical point of view, the knight-open-tour scrambler has a limited operation speed of 7.975 Mb/s while the data is streamed, whereas there is no such a limitation in securely storing stationary data. The data dimensionality has no impact on the scrambler performance, but the chessboard size should be consistent with the data length. Overall, the knight-open-tour scrambler is mainly intended for private use and corporate body security purposes including business and governmental agencies.

References

1. Pandya, P. (2013). Advanced data encryption. In *Cyber Security and IT Infrastructure Protection* (pp. 325–345). Elsevier. https://doi.org/10.1016/B978-0-12-416681-3.00015-X
2. Pandya, P. (2013). Advanced data encryption. In *Computer and Information Security Handbook* (pp. 1127–1138). Elsevier Inc. https://doi.org/10.1016/B978-0-12-394397-2.00070-2
3. Li, N., Zhang, N., Das, S. K., & Thuraisingham, B. (2009). Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Networks*, *7*(8), 1501–1514. https://doi.org/10.1016/j.adhoc.2009.04.009
4. De Capitani di Vimercati, S., Foresti, S., Livraga, G., & Samarati, P. (2022). Digital infrastructure policies for data security and privacy in smart cities. In *Smart Cities Policies and Financing: Approaches and Solutions* (pp. 249–261). Elsevier. https://doi.org/10.1016/B978-0-12-819130-9.00007-3
5. Waheed, A., Subhan, F., Suud, M. M., Alam, M. M., & Haider, S. (2023). Design and optimization of nonlinear component of block cipher: Applications to multimedia security. *Ain Shams Engineering Journal*, 102507. https://doi.org/10.1016/j.asej.2023.102507
6. Fair, T., Nordfelt, M., Ring, S., & Cole, E. (2005). Spying Basics. In *Cyber Spying* (pp. 45–86). Elsevier. https://doi.org/10.1016/b978-193183641-8/50005-8
7. Stapko, T. (2008). Choosing and optimizing cryptographic algorithms for resource-constrained systems. In *Practical Embedded Security* (p. 149–171). Elsevier. https://doi.org/10.1016/b978-075068215-2.50009-4

8. Sun, P. (2020). Security and privacy protection in cloud computing: Discussions and challenges. *Journal of Network and Computer Applications*, *160*, 102642. https://doi.org/10.1016/j.jnca.2020.102642
9. Bakiri, M., Guyeux, C., Couchot, J.-F., & Oudjida, A. K. (2018). Survey on hardware implementation of random number generators on FPGA: Theory and experimental analyses. *Computer Science Review*, *27*, 135–153. https://doi.org/10.1016/j.cosrev.2018.01.002
10. L'Ecuyer, P. (2006). Chapter 3. Uniform random number generation. In *Handbooks in Operations Research and Management Science. Vol. 13* (p. 55–81). Elsevier. https://doi.org/10.1016/S0927-0507(06)13003-0
11. Plesser, H. E., & Jahnsen, A. G. (2010). Re-seeding invalidates tests of random number generators. *Applied Mathematics and Computation*, *217*(1), 339–346. https://doi.org/10.1016/j.amc.2010.05.066
12. Dunn, W. L., & Shultis, J. K. (2012). Pseudorandom number generators. In *Exploring Monte Carlo Methods* (pp. 47–68). Elsevier. https://doi.org/10.1016/b978-0-444-51575-9.00003-8
13. Bowman, K. P. (2005). Statistics and pseudorandom numbers. In *An Introduction to Programming with IDL* (pp. 227–235). Elsevier. https://doi.org/10.1016/b978-012088559-6/50024-8
14. Ross, S. M. (2023). Random numbers. In *Simulation* (p. 39–45). Elsevier. https://doi.org/10.1016/b978-0-32-385738-3.00008-0
15. Parberry, I. (1997). An efficient algorithm for the Knight's tour problem. *Discrete Applied Mathematics*, *73*(3), 251–260. https://doi.org/10.1016/S0166-218X(96)00010-8
16. Lin, S.-S., & Wei, C.-L. (2005). Optimal algorithms for constructing knight's tours on arbitrary $n \times m$ chessboards. *Discrete Applied Mathematics*, *146*(3), 219–232. https://doi.org/10.1016/j.dam.2004.11.002
17. Weisstein, E. W. Knight Graph. URL: https://mathworld.wolfram.com/KnightGraph.html
18. Bai, S., Yang, X.-F., Zhu, G.-B., Jiang, D.-L., & Huang, J. (2010). Generalized knight's tour on 3D chessboards. *Discrete Applied Mathematics*, *158*, 1727–1731. https://doi.org/10.1016/j.dam.2010.07.009
19. Kyek, O., Parberry, I., Wegener, I. (1997). Bounds on the number of knight's tours. *Discrete Applied Mathematics*, *74*(2), 171–181. https://doi.org/10.1016/S0166-218X(96)00031-5
20. Romanuke, V. V. (2022). Finite uniform approximation of two-person games defined on a product of staircase-function infinite spaces. *International Journal of Approximate Reasoning*, *145*, 139–162. https://doi.org/10.1016/j.ijar.2022.03.005

The manuscript is received – 05/12/2023; accepted – 13/02/2024.

Різноманіття псевдовипадкових блукань шахового коня для скремблювання багатовимірних даних

Вадим Романюк

Анотація

Задача відкритого циклу шахового коня полягає у побудові послідовності ходів шахового коня, яка повністю покриває шахову дошку без повторів, де початкове та кінцеве положення є завжди різними. Розв’язок задачі відкритого циклу шахового коня подібний до послідовності псевдовипадкових чисел, за якими можна відобразити дані у корисну інформацію без можливості її читання. Задача відкритого циклу шахового коня для певного стартового положення має різноманіття розв’язків, кількість яких залежить від розміру шахової дошки. Розв’язки задачі відкритого циклу шахового коня виглядають як його псевдовипадкове блукання або як випадкова послідовність його положень. Ці розв’язки використовуються для подальшого покращення балансу простоти скремблювання та продуктивності, де головними показниками є ймовірність зламу та індекс подібності. Ймовірність зламу суттєво зменшується завдяки різноманіттю псевдовипадкових блукань шахового коня для певного стартового положення на даній шаховій дошці. Песимістична оцінка ймовірності зламу для шахової дошки розміром 8×8 є меншою за 10^{-16}. Аналогічна оцінка для шахової дошки розміром 8×8×8 є меншою за 10^{-26}. Конкретна реалізація псевдовипадкового блукання шахового коня будується в режимі онлайн за заданого початкового цілого для генератора псевдовипадкових чисел. Вектор даних після скремблювання також будується в режимі онлайн за лінійної часової складності. Індекс подібності є прийнятним. Він стрімко падає зі зростанням розміру шахової дошки (для більших масивів багатовимірних даних). Конкретне псевдовипадкове блукання шахового коня визначається розміром шахової дошки, початковим положенням, методом векторизації псевдовипадкового блукання шахового коня, а також початковим цілим для генератора псевдовипадкових чисел. Ці параметри визначають специфічний рух коня у ситуаціях, коли постають множинні варіанти подальшого руху. Скремблер на основі відкритого циклу шахового коня має від 10^{16} до 10^{21} разів меншу ймовірність зламу, порівняно зі звичайним псевдовипадковим генератором, залежно від розміру шахової дошки та початкового положення шахового коня.

Ключові слова: скремблювання даних; псевдовипадкове блукання шахового коня; ймовірність зламу; рівень подібності.