

Juxtaposition operation to obtain Latin squares

M.I. García-Planas^{*1} and D. Roca-Borrego¹

¹Departament de Matemàtiques, Universitat Politècnica de Catalunya, Barcelona, Spain

**Short
Communication**

Received: XX May 2017

Accepted: XX December 20XX

Online Ready: XX December 20XX

Abstract

With the power that has taken the information technologies, one has developed the study and research about cryptography, and cryptanalysis, in which Latin squares are ideal candidates for being used in cryptographic systems because the Cayley tables of the finite groups are Latin squares. This fact has awakened a new interest in the study of Latin squares by applying them to the study of code theory and error correcting codes. They also play a significant role in the statistical theory of experimental design.

In this work, we develop an algorithm for the generation of Latin squares based on the juxtaposition operation defined for effect.

Keywords: Latin square; cyclic matrices; Generation of latin squares

2010 Mathematics Subject Classification: 68N15; 15A36

1 Introduction

The Latin squares have been studied for centuries; their origin goes back at least to the surroundings of the year 1000 in which the Arab and Indian communities used them as amulets or talismans. Also in the thirteenth century, the philosopher Ramon Llull (1232-1315) introduces in his text *Ars Demonstrativa* (1283) four Latin squares of order 4, using as symbols fire, air, water and earth, in an attempt to explain the world through combinatorial numbers [Bonner (1985)]. However, the first formal definition was given by Leonhard Euler [Euler (1782)], who was interested in solving the problem of the 36 officers.

The Latin squares appear naturally in algebra, for example, when constructing the summation tables on the sets of classes of remainder modulo m .

In recent years, the rise of information technologies has led to the development of cryptography, a science that studies the creation of secret codes, and cryptanalysis aimed at deciphering those codes, in which Latin squares are ideal candidates for being used in cryptographic systems because the Cayley tables of the finite groups are Latin squares. This fact has awakened a new interest in the study of Latin squares by applying them to the study of code theory and error correcting codes. They also play a significant role in the statistical theory of experimental design.

^{*}Corresponding author: E-mail: maria.isabel.garcia@upc.edu

Following Ritter in [Ritter (2003)] a Latin square can be seen as a stream cipher combiner, Koscielny in [Koscielny (2002)] developed some routines using the Maple 7 package to generate 256-order Latin squares. For use in cryptography in addition to the Latin square-building algorithm, an algorithm for extracting symbols from one Random shape must be done. When it is intended to create a Latin square generation algorithm, to obtain an encryption or decryption algorithm for a secure communication protocol, it must be such that the Latin square is generated quickly and from time to time. This construction should not represent an overload of time or resources (memory, hard disk, etc.) in the protocol. Generally, simple Latin square generation algorithms are of exponential order, so the problem requires a certain complexity to be efficient. Gallego in [Gallego (2014)], proposes two implementations, one giving weight to clarity versus efficiency and the other giving weight to efficiency versus clarity, highlighting the difficulty in achieving these two properties at the same time.

The exact number of Latin squares of order $n \geq 11$ has not been known so far (for more information, see [Hulpke, Kaski, Östergård (2011)], [Kolesova, Lam, Thiel (1990)] and [McKay, Meynert, Myrvold] for example). However, there have been different lower bounds for this number. Therefore it is still interesting to create algorithms that generate Latin squares.

The objectives of this work are on the one hand to compile the existing information in the literature on the Latin squares and on the other is to outline some techniques for a specific computer tool for the generation of Latin squares.

In Dénes, Keedwell (1974) some constructions of latin squares are summarized, in particular the construction of Latin squares preserving some subsquare conditions. By simulating an ergodic Markov chain with uniform stationary distribution over the space of n -order Latin squares, Jacobson and Matthews in Jacobson and P. Matthews (1996), have discussed some methods to generate Latin squares.

2 Preliminaries

A Latin square is an $n \times n$ -order matrix L whose elements belong to a finite set A of cardinal n and each of them appears exactly once in each row and each column of L . The set A is usually considered as the set of the firsts n natural numbers and receives the Base set name of the square and n its order.

Example 2.1. A 6-order latin square can be:

1	2	3	4	5	6
4	5	6	1	2	3
6	1	2	3	4	5
3	4	5	6	1	2
5	6	1	2	3	4
2	3	4	5	6	1

A special case of Latin squares is the so-called Latin reduced squares. That are those in which its first row and its first column the numbers from 1 to n taken as symbols, appear in the natural order.

There is a Latin square of order n for any positive integer n . It is easy to construct a Latin square of any order n by placing in the first row the elements of the base set (for example the first n natural numbers) and the other rows we obtain them by cyclically displacing the elements of the base set. Thus far it is known the number $l(n)$ of all Latin reduced squares of order n , where $n \leq 11$ [McKay, Wanless (2005)]. There is a relation between the number of reduced Latin squares and Latin squares $L(n)$ given by $L(n) = n!(n-1)!l(n)$ (when $n \geq 2$). Then, the knowledge of the number of reduced Latin squares, permit us to count the exact number of all Latin squares of order $n \leq 11$. The explicit number can be found in [OEIS-A002860 (2011)].

2.1 Generation of Latin squares

Given an n -order Latin square, it is possible to obtain from it, Latin squares of order any multiple of the order of the given one, using *juxtaposition operation* introduced in the following manner.

Let $A_1, \dots, A_r$, be r Latin squares of order n where the respective base sets are renamed them of the form $\{1, \dots, n\}, \{n+1, \dots, 2n\}, \dots, \{(r-1)n+1, \dots, rn\}$.

Definition 2.1. The juxtaposition of these squares is defined as the matrix:

$$A_1 * A_2 * \dots * A_r = \begin{array}{|c|c|c|c|c|} \hline A_1 & A_2 & \dots & A_{r-1} & A_r \\ \hline A_r & A_1 & \dots & A_{r-2} & A_{r-1} \\ \hline \vdots & \vdots & \dots & \vdots & \vdots \\ \hline A_2 & A_3 & \dots & A_r & A_1 \\ \hline \end{array}$$

That is, each row is obtained by cyclically rotating the Latin squares of the previous row, passing the first Latin square to the last place. By construction, this square is a Latin square, and the order is nr .

Proposition 2.1. The juxtaposition of the n -order Latin squares $A_1 * A_2 * \dots * A_r$ is a Latin square

Example 2.2. Let $A_1 = A_2 = A_3$ three identical two-order square Latin with

$$\begin{array}{|c|c|} \hline 1 & 2 \\ \hline 2 & 1 \\ \hline \end{array}$$

Renaming we obtain

$$A_1 = \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 2 & 1 \\ \hline \end{array} \quad A_2 = \begin{array}{|c|c|} \hline 3 & 4 \\ \hline 4 & 3 \\ \hline \end{array} \quad A_3 = \begin{array}{|c|c|} \hline 5 & 6 \\ \hline 6 & 5 \\ \hline \end{array}$$

Then

$$A_1 * A_2 * A_3 = \begin{array}{|c|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 2 & 1 & 4 & 3 & 6 & 5 \\ \hline 5 & 6 & 1 & 2 & 3 & 4 \\ \hline 6 & 5 & 2 & 1 & 4 & 3 \\ \hline 3 & 4 & 5 & 6 & 1 & 2 \\ \hline 4 & 3 & 6 & 5 & 2 & 1 \\ \hline \end{array}$$

Another way to generate easily computable Latin squares is the operation that we will call *Kronecker's composition* because of its similarity to the left Kronecker product of matrices operation

Given two square matrices A_1 and A_2 of respective orders n_1 and n_2 respectively.

Definition 2.2. The Kronecker composition of these square matrices is defined in the following manner:

Writting

$$A_1 = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n_1} \\ a_{21} & a_{22} & \dots & a_{2n_1} \\ \vdots & \vdots & & \vdots \\ a_{n_1 1} & a_{n_1 2} & \dots & a_{n_1 n_1} \end{pmatrix}, \quad \text{and} \quad A_2 = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n_2} \\ b_{21} & b_{22} & \dots & b_{2n_2} \\ \vdots & \vdots & & \vdots \\ b_{n_2 1} & b_{n_2 2} & \dots & b_{n_2 n_2} \end{pmatrix}$$

$$A_1 \circ A_2 = \begin{pmatrix} a_{11}A_2 & a_{12}A_2 & \dots & a_{1n_1}A_2 \\ a_{21}A_2 & a_{22}A_2 & \dots & a_{2n_1}A_2 \\ \vdots & \vdots & & \vdots \\ a_{n_11}A_2 & a_{n_12}A_2 & \dots & a_{n_1n_1}A_2 \end{pmatrix} =$$

$$\begin{pmatrix} a_{11}, b_{11} & \dots & a_{11}, b_{1n_2} & & a_{1n_1}, b_{11} & \dots & a_{1n_1}, b_{1n_2} \\ \vdots & & \vdots & \dots & \vdots & & \vdots \\ a_{11}, b_{n_21} & \dots & a_{11}, b_{n_1n_1} & & a_{1n_1}, b_{n_21} & \dots & a_{1n_1}, b_{n_1n_1} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{n_11}, b_{11} & \dots & a_{n_11}, b_{1n_2} & & a_{n_11}, b_{11} & \dots & a_{n_11}, b_{1n_2} \\ \vdots & & \vdots & \dots & \vdots & & \vdots \\ a_{n_11}, b_{n_21} & \dots & a_{n_11}, b_{n_1n_1} & & a_{n_1n_1}, b_{n_21} & \dots & a_{n_1n_1}, b_{n_1n_1} \end{pmatrix}$$

Proposition 2.2. Let A_1, A_2 be two matrices of orders n_1 and n_2 respectively, corresponding to Latin squares. Then The Kronecker composition of these matrices $A_1 \circ A_2$, is an n_1n_2 -order Latin square.

Proof. Suppose that $a_{1i}b_{1j} = a_{\ell k}b_{rs}$, then $a_{1i} = a_{\ell k}$ and $b_{1j} = b_{rs}$, but taking into account that A_1 and A_2 are Latin squares we have that $(1, i) = (\ell, k)$ and $(1, j) = (\ell, k)$ $(1, j) = (r, s)$ $\square$

Remark 2.1. If the base sets of A_1 and A_2 are $\{a_1, \dots, a_{n_1}\}$ and $\{b_1, \dots, b_{n_2}\}$, respectively then, the basis set of $A_1 \circ A_2$ is $\{1, 2, \dots, n_1n_2\}$ obtained after renaming the set

$$\{(a_1, b_1), \dots, (a_1, b_{n_2}), (a_2, b_1), \dots, (a_2, b_{n_2}), \dots, (a_{n_1}, b_1), \dots, (a_{n_1}, b_{n_2})\}$$

following natural order $((a_1, b_1) \rightarrow 1, \dots, (a_1, b_{n_2}) \rightarrow n_2, \dots)$.

For more details on these operations see [Roca-Borrego (2017)].

In a similar way but using right Kronecker product operation Dénes and Keedwell in [Dénes, Keedwell (1974)] define the product of two Latin squares, obtaining different Latin squares because of the Kronecker composition is not a commutative operation.

More information about distinct methods to generate Latin squares can be found in [Jacobson and P. Matthews (1996)].

3 Description of the Algorithm

As mentioned, the calculation of the Latin squares continues to be a reason for investigation and analysis. In this line, it is worth mentioning the development of a new algorithm presented by [Yordzhev (2016)], based on bitwise operations.

Thus, the development of optimisation in the calculation of a random Latin square has been based both on the code previously mentioned and on the theorem of the juxtaposition of Latin squares.

We rely on the bitwise working method for its performance and efficiency in the consumption of the processor. As well as, it is much more efficient and fast than other methods. It is due to the operations are treated directly by the low-level instructions on the adjacent processor.

To make the process more comprehensible, we present here the development of the algorithm for the juxtaposition of 4 Latin squares of order 2 and order 4, obtaining in this way Latin squares of 8x8 and 16x16 orders. We do this to maintain a small order due to the high execution time required by the calculation with high values of n , and thus it can be easily tested.

To perform the juxtaposition of the matrices is needed a module to perform the renaming of the symbols of the base set of the Latin squares. This code has been implemented using the bitwise operations technique. Below is the code developed to perform such renaming.

```
void renombre(){
    int desplazamiento =N;
    int desplazamiento2 = N<<1;
    int desplazamiento3= N<<2;
    for (int i=0; i<N; i++) {
        for (int j=0; j<N; i++){
            L2 [i] [j] = L2 [i] [j] << desplazamiento;
            L3 [i] [j] = L3 [i] [j] << desplazamiento2;
            L4 [i] [j] = L4 [i] [j] << desplazamiento3;
        }
    }
}
```

Once the symbols have been renamed, we proceed to the coherent juxtaposition of the calculated Latin squares, in this case, L, L2, L3 and L4.

The juxtaposition operation is then performed

```
void yux () {
    Renombre ();
    for (int i=0; i<N; i++){
        F [i] [j] = L [i] [j];
        F [j+2] = L[i] [j];
        F [i+4] [j+4] = L[i] [j];
        F [i+6] [j+6] = L [i] [j];

        F [i+2] [j] = L2 [i] [j];
        F [i+4] [j+2] = L2 [i] [j];
        F [i+6] [j+4] = L2 [i] [j];
        F [i] [j+6] = L2 [i] [j];

        F [i+4] [j] = L3 [i] [j];
        F [i+6] [j+2] = L3 [i] [j];
        F [i] [j+4] = L3 [i] [j];
        F [i+2] [j+6] = L3 [i] [j];

        F [i+6] [j] = L4 [i] [j];
        F [i] [j+2] = L4 [i] [j];
        F [i+2] [j+4] = L4 [i] [j];
        F [i+4] [j+6] = L4 [i] [j];

    }
}
```

4 CONCLUSIONS

With the help of algorithms we obtain Latin squares, for example

4	8	1	2	131072	524288	262144	65536	256	2048	512	1024	64	16	32	128
8	4	2	1	524288	262144	65536	131072	2048	1024	256	512	128	32	16	64
1	2	4	8	65536	131072	524288	262144	512	256	1024	2048	16	64	128	32
2	1	8	4	262144	65536	131072	524288	1024	512	2048	256	32	128	64	16
64	16	32	128	4	8	1	2	131072	524288	262144	65536	256	2048	512	1024
128	32	16	64	8	4	2	1	524288	262144	65536	131072	2048	1024	256	512
16	64	128	32	1	2	4	8	65536	131072	524288	262144	512	256	1024	2048
32	128	64	16	2	1	8	4	262144	65536	131072	524288	1024	512	2048	256
256	2048	512	1024	64	16	32	128	4	8	1	2	131072	524288	262144	65536
2048	1024	256	512	128	32	16	64	8	4	2	1	524288	262144	65536	131072
512	256	1024	2048	16	64	128	32	1	2	4	8	65536	131072	524288	262144
1024	512	2048	256	32	128	64	16	2	1	8	4	262144	65536	131072	524288
131072	524288	262144	65536	256	2048	512	1024	64	16	32	128	4	8	1	2
524288	262144	65536	131072	2048	1024	256	512	128	32	16	64	8	4	2	1
65536	131072	524288	262144	512	256	1024	2048	16	64	128	32	1	2	4	8
262144	65536	131072	524288	1024	512	2048	256	32	128	64	16	2	1	8	4

Finally, we want to remark that the algorithm presented, optimises the time of obtaining a Latin square of order $N = nr$. The generation time of a Latin square is at least polynomial order $O(N^3)$ (depending on the algorithm), while the presented algorithm is of the order $O(n^3 + nr)$.

Remark that, the cost of choosing n symbols and placing them in n rows implies n^2 operations, but if we want to obtain a Latin square, we can not have collisions involving a cost in execution. In the standard approach for the generation of random Latin squares (to be useful in cryptography) of order n , obtained by applying movements to the one obtained by cyclic displacement of the base set, n^3 changes are required, (see [Jacobson and P. Matthews (1996)], [Gallego (2014)]). Therefore, if we want to obtain a Latin square of order rn we need $(rn)^3$ movements. But if we construct it from a Latin square of order n , renaming it r times and juxtaposing it we only require $n^3 + rn$.

References

- Bonner, A. (1985). *Selected Works of Ramon Llull*, Princeton: Princeton University Press, 1 305–568.
- Dénes, J., Keedwell, A.D. (1974). *Latin squares and their applications*, Academic Press, New York-London.
- Euler, L. (1782). *Recherches sur une nouvelle espece de quarres magiques*. *Verh. v. h. Zeeuwsch Genootsch. der Wetensch*, 9 85–239.
- Gallego, I. (2014). *Análisis de algoritmos para generación de cuadrados latinos aleatorios para criptografía TFM*. Universidad Nacional de la Plata.
- Hulpke, A., Kaski, P., Östergård, P.R.J. (2011). The number of Latin squares of order 11, *Math. Comp.* 80 1197–1219.
- Jacobson, M., Matthews, P. (1996). Generating uniformly distributed Latin squares. *J. Combin. Des.* 4(6), 405–437.
- Koscielny, C. (2002). Generating quasigroups for cryptographic applications. *Int. J. Appl. Math. Comput. Sci* 12(4), 559–569.
- Kolesova, G., Lam, C.W.H., Thiel, L. (1990). On the number of 8x8 Latin squares, *J. Combin. Theory Ser. A* 54 143–148.

- McKay, B. D. and Wanless, I. M. (2005). On the number of Latin squares. *Annals of Combinatorics* 9(3), 335–344.
- McKay, BD., Meynert, A., Myrvold, W. (2007). Small Latin squares, quasigroups, and loops, *J. Combin. Des.* 15 98–119.
- OEIS. The On-Line Encyclopedia of Integer Sequences. A002860 - Number of Latin squares of order n . Available: <http://oeis.org/A002860> (Last accessed on April 30, 2017 at 12:45).
- itter, T. (2003). Latin Squares: A Literature Survey. Research Comments from Ciphers (online) <http://www.ciphersbyritter.com/RES/LATSQ.HTM>
- Roca-Borrego, D. (2017). Estudio y análisis de los cuadrados latinos para la optimización del proceso de obtención. Universitat Politècnica de Catalunya, Spain *Unpublished TFG thesis*.
- Yordzhev, K. (2016). Bitwise Operations in Relation to Obtaining Latin Squares. *British Journal of Mathematics & Computer Science* 5, 1–7.

©2017 M.I. García-Planas & D. Roca-Borrego; This is an Open Access article distributed under the terms of the Creative Commons Attribution License <http://creativecommons.org/licenses/by/2.0>, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.