A Pseudorandom Number Algorithm for Generating Individual Bank Account Numbers

John Effiong Usen*; Emmanuel Emmanuel Asuk; Ede, Jeremiah Ejimne

University of Cross River State, Calabar, Cross River State, Nigeria

***Corresponding Author**

DOI: https://doi.org/10.51584/IJRIAS.2025.100500011

Received: 12 April 2025; Review: 19 April 2025; Accepted: 25 April 2025; Published: 29 May 2025

ABSTRACT

This research work was an attempt at developing an algorithm which generates bank account numbers for customers of any given bank whilst ensuring enhanced randomness and longevity in the numbers generated. The attempt at doing this began with a review of some existing pseudorandom number generators with a view to understanding their advantages and lapses. Thereafter, the Mid-Square Method (MSM) and the Linear Congruential Generator (LCG) were improved and reformulated. We then combined both reformulated algorithms to form what was termed a Combined Pseudorandom Number Generator (CSRNG). The proposed technique became a new algorithm used for generating unique bank account numbers, because it was made up of three independent components. The first and second components were blocks of four (4) digits generated using MSM, and based on arbitrarily chosen seeds, while the third component was a block of two (2) digits generated with the LCG. The results in each case of the combination were summarized as bank account numbers with better longevity and enhanced randomness (or very reduced predictability).

Key word: Pseudorandom Number Generators, Mid-Square Method (MSM), Linear Congruential Generator (LCG), Combined Pseudorandom number Generator (CSRNG)

INTRODUCTION

Background of the study

In life, events (and occurrences) may occur randomly or with a patterned sequence [4; 12]. Events that usually occur with patterned sequence are those whose future occurrence could be predicted based on data on their previous occurrences [12]. Events such as: rainfall, sunshine, and natural disasters (volcanic eruptions, tsunamis, and hurricanes) fall under this category of events with patterned sequence (i.e., predictable events). Random events, however, are unpredictable events – events whose future reoccurrence cannot be predicted via historical data on its previous occurrences [4].

In order to protect and preserve the balance of nature, man has developed the know-how on using historical data on the activities and patterns of predictable events to forecast the time, place, and impact, of their future reoccurrence [2]. However, whereas man may prefer to make use of historical data on the activities and patterns of predictable events to forecast future reoccurrences to certain levels of accuracy, man eschews this preference when it has to do with activities that require high levels of unpredictability, as utilizing patterns could defeat the very essence of such activities [2; 5]. These activities range from the development of passwords of all sorts, Personal Identification Number (PIN), Bank Verification Number, to bank account numbers.

Generating absolutely random numbers is naturally impossible. As such, generating passwords, Personal Identification Number (PIN), Bank Verification Number, and account numbers for banks, require the use of algorithms called pseudorandom number generators. Pseudorandom number generators (PRNGs) are computer algorithms or programs written for (and utilized in) probability and statistics applications when large quantities

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue V May 2025

of random digits are required [2]. A lot of such programs and algorithms produce endless strings of single-digit numbers, usually in the decimal system [1; 8; 7].

In statistical computing, a number of pseudorandom numbers have been developed for generating numbers with high degree of randomness. These generators consist of: mid-square generator, Linear Congruential Generator, combined linear generator, random number stream, multiple recursive generators, etc., and their applications in the telecommunication, banking, and security-inclined industries cannot be overemphasized as such industries need a high level of randomness and longevity in their respective number strings, without which they could experience loss in revenue, especially from continuous system hacks [3; 5].

In the field of statistical computing, the development of newer and better generators remains a never-ending task, as hackers continue to explore newer techniques bypassing security systems [6; 15]. As it is statistically founded that combining more than one algorithm improves randomness, this research targets the development of a new generator for generating individual bank account numbers for customers of any bank.

Statement of the problem

In recent decades in Nigeria, the activities of financial crimes and cybercrimes have skyrocketed, particularly with regards to hackings of bank accounts – whether individual or corporate [10]. The EFCC (Economic & Financial Crimes Commission), SSS (State Security Service), and NPF (Nigeria Police Force), being government agencies that are saddled with the responsibilities of combating financial crimes and cybercrimes, have continued in their efforts of curbing activities of the culprits, on one hand; whereas, on another hand, financial and banks have routinely continued to upgrade their systems in an effort to reduce or avoid these attacks [13]. So far, these efforts have produced reactionary results, rather than preventive results, as the hackers find it easy to hack accounts due to the predictability of the number sequence in the account numbers, alongside the carelessness of account owners disclosing of their BVN (Bank Verification Number). This research is an attempt to address this issue via the formulation and application of a more robust pseudorandom number generator that overcomes the identified flaw.

Aim and objectives of the study

This study aims to formulate a pseudorandom number generator which efficiently generates individual bank account numbers. In line with achieving the stated aim, the objectives of the study are to: (i) analyze a variety of already existing pseudorandom number generators, (ii) propose a formulated pseudorandom number generator, and thereafter (iii) apply the proposed formulated pseudorandom number generator to generating a set of individual bank account numbers.

MATERIALS AND METHODS

Linear Congruential Generator (LCG)

According to [9], Linear Congruential Generator (LCG) is an algorithm which generates a sequence of pseudorandomized numbers calculated with a discontinuous piecewise linear equation. The method represents one of the oldest and best-known pseudorandom number generator algorithms. The theory underlying the LCG is relatively easy to understand, and is easily implementable and fast, particularly on computers which can provide modular arithmetic aided by storage-bit truncation. LCG is a generator which has a transfer function of the following type:

$$f(x) = (ax + c)\, mod\, m$$

where $a < m$ is the multiplier, $c < m$ is the increment, $x < m$ is the modulus, and $a, c, x, m \in N$ (the set of all natural numbers), and $f(x)$ is such that

$$x_n = (ax_{n-1} + c)\, mod\, m$$

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue V May 2025

Typically, $c < m$ and m are chosen to be relatively prime (that is, they are prime numbers). But a is chosen such that $\forall\ x \in N,\ ax\ mod\ m \neq 0$. However, if $c = 0$, the generator is often called a multiplicative congruential generator (MCG) or Lehmer RNG. If $c \neq 0$, the method is called a mixed congruential generator. More so, when $c \neq 0$, mathematicians would call the recurrence an affine transformation, not a linear one, but the misnomer is well-established in computer science.

Mid-Square Method (MSM)

Mid-Square Method, among other methods of statistical computing, in statistics, is a generator that gives pseudorandom numbers; and it was invented by John von Neumann, and was described at a conference in 1949. To generate a sequence of n -digit pseudorandom numbers, an n -digit starting value is created and squared, producing $2n$ digits, leading zeroes are added to compensate. The middle n digits of the result would be the next number in the sequence, and returned as the result. This process is then repeated to generate more numbers [14; 16].

The value of n must be even in order for the method to work – if the value of n is odd then there will not necessarily be a uniquely defined "middle n -digit" to select from. For a generator of n -digit numbers, the period can be no longer than 8^n. If the middle n -digit are all zeroes, the generator then outputs zeroes, the generator then outputs zeroes forever. If the first half of a number in the sequence is zeroes, then the subsequent numbers will be decreasing to zero. While these runs of zeroes are easy to detect, they occur too frequently for this method to be of practical use. The Mid-Square Method can also be stuck on a number other than zero [11].

The algorithm for the Mid-Square Method is given as follows:

Step 1: Start with an n -digit natural number (the seed).

Step 2: Compute the square of the n -digit natural number.

Step 3: Take the middle n digits for the next n -digit natural number.

The proposed combined PRNG

The proposed combined PRNG referred to this study is an algorithm formed by the combination of a modified version of the MSG and the LCG. This proposed algorithm is presented below:

Step 1: For the first block, start with an $n_0^{(1)}$-digit natural number (the seed).

Step 2: Compute the square of the $n_0^{(1)}$-digit natural number.

Step 3: Take the middle $u_i^{(1)}$ digits in the output as the next seed.

Step 4: For the second block, start with an $n_0^{(2)}$-digit natural number (the seed).

Step 5: Compute the square of the $n_0^{(2)}$-digit natural number.

Step 6: Take the middle $u_i^{(2)}$ digits in the output as the next seed.

Step 7: If the square of $n_i^{(1)}$ and $n_i^{(2)}$ respectively give numbers that are one less than the expected number of digits, then begin your result with a zero, and so on. More so, if an output starts with a zero digit, then replace such with the largest digit in the last output. Continue the procedure as it were, if otherwise.

Step 8: Implement the LCG with an appropriate seed to generate the third block of $n_i^{(3)}$-digits. Should the digits in each output be less than the required digits, then add a number of zero-digits. But if the digits are more than the required number of digits then truncate at the required number.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue V May 2025

Justification for the proposed combined PRNG

Tendencies abound that the proposed divided block randomization technique could ensure longevity in the generation of individual bank account numbers as it overcomes the traditional problems of the MSM with the modification proposed in the study. More so, the algorithm of the proposed method surely improves randomness in individual bank account numbers since the combination of the proposed blocks makes predictability of the digits in each block almost impossible.

RESULTS

In order to implement the proposed combined PRNG, we have used $n_0^{(1)} = 2546$; $n_0^{(2)} = 9873$; $n_0^{(3)} = 8$ as seeds (to aid our illustration) respectively, for the mid-square generator (MSG) and Linear Congruential Generator (LCG).

In generating a unique account number with the arbitrarily chosen seeds above, we start the implementation, for $i = 0,1,2,\ldots,9$, for the first block of four (4) digits as illustrated in Table 1. Next, we proceed to the implementation, for $i = 0,1,2,\ldots,9$, of the second block of four (4) digits as illustrated in Table 2.

Next, we obtain the numbers in the third (and final) block by implementing the LCG component of our proposed algorithm using the third arbitrarily chosen seed (say, $n_0^{(3)} = 8$, $i = 0,1,2,\ldots,9$). Here, we use the transfer function of the LCG

$$x_i = (ax_{i-1} + c)\, mod\, m$$

in which case we choose $a = 2 < m$ as the multiplier, $c = 7 < m$ (relatively prime) as the increment, $m = 13$ (relatively prime) is the modulus. This gives the following:

$$x_1 = (2(8) + 7)\, mod\, 13 = 23\, mod\, 13 = 10 = 10$$

$$x_2 = (2(10) + 7)\, mod\, 13 = 27\, mod\, 13 = 1 = 01$$

$$x_3 = (2(1) + 7)\, mod\, 13 = 9\, mod\, 13 = 9 = 09$$

$$x_4 = (2(9) + 7)\, mod\, 13 = 25\, mod\, 13 = 12 = 12$$

$$x_5 = (2(12) + 7)\, mod\, 13 = 31\, mod\, 13 = 5 = 05$$

$$x_6 = (2(5) + 7)\, mod\, 13 = 17\, mod\, 13 = 4 = 04$$

$$x_7 = (2(4) + 7)\, mod\, 13 = 15\, mod\, 13 = 2 = 02$$

$$x_8 = (2(2) + 7)\, mod\, 13 = 11\, mod\, 13 = 11 = 11$$

$$x_9 = (2(11) + 7)\, mod\, 13 = 29\, mod\, 13 = 16 = 16$$

The results for the 3 blocks are finally combined in the order: first block digits (first MSM output), second block (second MSM output), and third block (LCG output). Table 3 displays the 10-digit account numbers generated using our proposed combined PRNG. FIG. 1 shows the plots of each generated block against seed progressions for the four sets of blocks generated.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue V May 2025

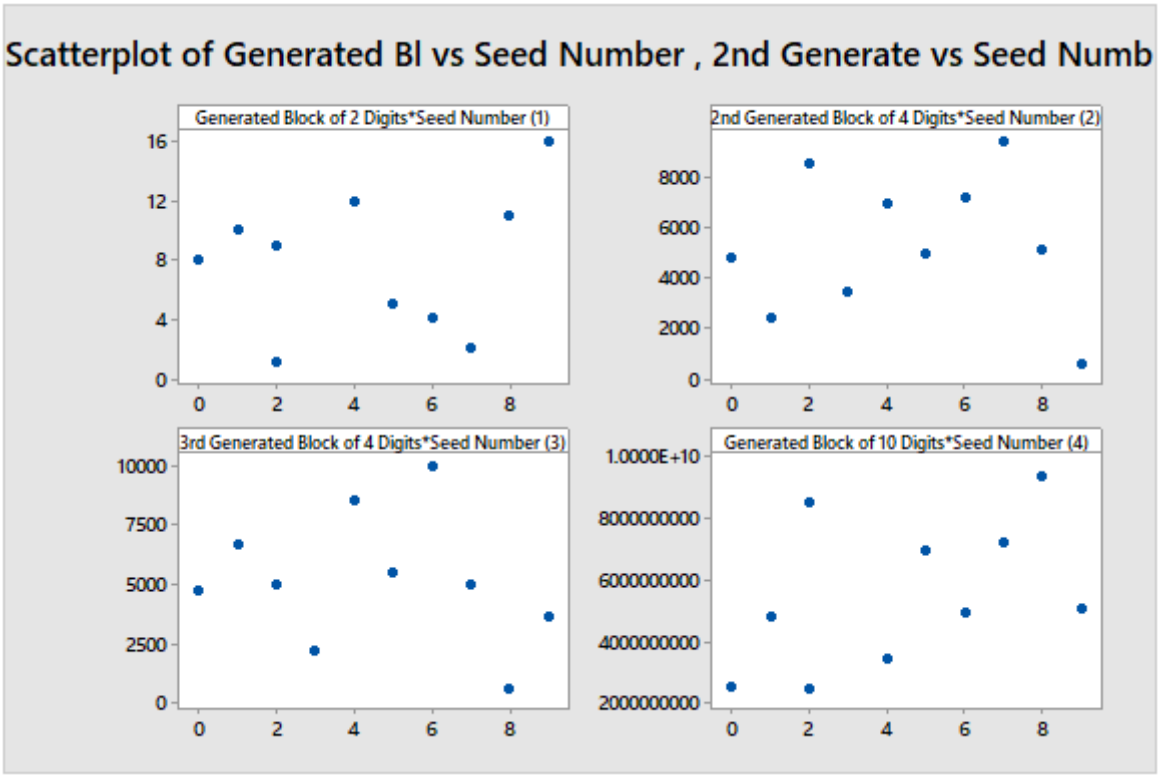

Fig. 1: Plots of Generated Blocks vs Seed Progressions

DISCUSSION

The results have shown that the proposed combined PRNG works effectively in generating the required random account number primarily because of the independent improvements of the Mid-Square Method (MSM) and Linear Congruential Generator (LCG). Such improvements are particularly observed in the initial and second lines of Table 1, as well as the fourth and ninth lines of Table 2.

More so, the account numbers produced by the proposed combined generator are noticeably unpredictable; as such, tendencies are that it could reduce the cases of being hacked into by scammers, whilst improving the uniqueness of an account number for a customer of a specified bank.

CONCLUSION

In conclusion we have: (i) reviewed a variety pseudorandom number generators in terms of their merits and demerits, (ii) reformulated the Mid-Square Method (MSM) and the Linear Congruential Generator (LCG) in order to enhance randomness and longevity of both algorithms, (iii) proposed a Combined Pseudorandom Number Generator (CPRNG) suitable for generating bank account numbers that possess better longevity and randomness guaranteed to high degree of accuracy, and (iv) implemented the proposed technique using arbitrarily chosen seeds.

Table 1 Obtaining numbers in the first block of digits

i	$n_i^{(1)}$	$n_i^{(1)} \times n_i^{(1)}$	$u_i^{(1)}$
0.	**2546**	**06482116**	**4821**
1.	4821	23242041	2420
2.	**2420**	**05856400**	**8564**

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue V May 2025

3.	8564	73342096	3420
4.	3420	11696400	6964
5.	6964	48497296	4972
6.	4972	24720784	7207
7.	7207	51940849	9408
8.	9408	88510464	5104
9.	5104	26050816	0508

Table 2 Obtaining numbers in the second block of digits

i	$n_i^{(2)}$	$n_i^{(2)} \times n_i^{(2)}$	$u_i^{(2)}$
0.	9873	97476129	4761
1.	4761	22667121	6671
2.	6671	44502241	5022
3.	5022	25220484	2204
4.	**2204**	**04857616**	**8576**
5.	8576	73547776	5477
6.	5477	29997529	9975
7.	9975	99500625	5006
8.	5006	25060036	0600
9.	**0600**	**00360000**	**3600**

Table 3 10-Digit Account Numbers Generated via the proposed CPRNG

	1st Block	2nd Block	3rd Block	
i	$u_i^{(1)}$	$u_i^{(2)}$	$u_i^{(3)}$	Account No.
0.	2546	9873	08	2546987308
1.	4821	4761	10	4821476110
2.	2420	6671	01	2420667101
2.	8564	5022	09	8564502209
4.	3420	2204	12	3420220412

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue V May 2025

5.	6964	8576	05	6964857605
6.	4972	5477	04	4972547704
7.	7207	9975	02	7207997502
8.	9408	5006	11	9408500611
9.	5104	0600	16	5104060016

REFERENCES

1. Afflerbach, L. (1990). Criteria for the assessment of random number generators. Journal of Computational and Applied Mathematics, 31, 3-10.
2. Dateu, O., Macovei, C. & Hobincu, R. (2020). Chaos-based cryptographic pseudorandom number generator template with dynamic state change. Applied Sciences, 10(451), 1-17.
3. Edmond, A. R. (1990). The generation of pseudorandom numbers on electronic digital computers. The Computer Journal, 2(4), 181-185.
4. Glette-Iversen, I. & Flage, R. (2024). On unpredictable events in risk analysis. Safety Science, 180(2), 11-25.
5. Jacak, M. M., Jozwiak, P., Niemczuk, J. & Jacak, E. J. (2021). Quantum generators of random numbers. Scientific Reports, 11(2), 1-21.
6. James, F. & Moneta, L. (2020). Review of high-quality random number generators. Computing and Software for Big Science, 4(2), 1-12.
7. Junsangsri, P. & Lombardi, F. (2024). Pseudo-random number generators for stochastic computing (SC): Design and Analysis. IEEE Explore Open Journal of Nanotechnology, 5(3), 57-67.
8. Maheswari, K. M. U., Kundu, R. & Saxena, H. (2018). Pseudorandom number generators algorithms and applications. International Journal of Pure and Applied Mathematics, 118(22), 331-336.
9. Nannipieri, P., Di-Matteo, S., Baldanzi, L., Crocetti, L., Belli, J., Fanucci, L. & Saponara, S. (2021). True random number generator based on Fibonacci-Galois ring oscillators for FPGA. Applied Sciences, 11(8), 2-19.
10. Pereware, T. A. & Digiteme-Batubo, B. N. (2016). Efforts in combating cybercrime and criminality in Nigeria. Information and Knowledge Management, 6(3), 23-28.
11. Priyanka, Hussain, I. & Khalique, A. (2019). Random number generators and their applications: A review. International Journal of Research and Computer Engineering, 7(2), 1777-1781.
12. Sytnik, V. M. & Proskuryakova, L. N. (2024). Expanding foresight methodology to better understand the unknown future and identify hard-to-predict events. European Journal of Futures Research, 21(9), 23-38.
13. Tsado, L., Raufu, A., Ben-Edet, E. & Krakrafaa-Bestman, D. (2023). Combatting the threat of cybercrime in Nigeria: Examining current laws and policies. Journal of Applied and Theoretical Social Sciences, 5(4), 413-430.
14. Van-Niel, K. & Laffan, S. W. (2010). Gambling with randomness: The use of pseudorandom number generators in GIS. International Journal of Geographical Information Science, 17(1), 49-68.
15. Wu, X., Han, Y., Zhang, M., Zhu, S., Cui, S., Wang, Y. & Peng, Y. (2025). Pseudorandom number generators based on neural networks: A review. Journal of King Saudi University – Computer and Information Sciences, 37(3), 1-18.
16. Yu, F., Li, L., Tang, Q., Cai, S., Song, Y. & Xu, Q. (2019). A survey on true random number generators based on chaos. Discrete Dynamics in Nature and Society, Hindawi, 2-11.