







SYSTEMATIC MAPPING STUDY ON FAST FACTORIZATION USING PARALLEL OR DISTRIBUTED PROCESSING APPLIED TO CRYPTANALYSIS

Estudio de mapeo sistemático sobre factorización rápida utilizando procesamiento paralelo o distribuido aplicado al criptoanálisis

Jhon-Alejandro Melo 
Universidad del Cauca (Popayán-Cauca, Colombia). 
alejandromelo@unicauca.edu.co

Siler Amador-Donado 
Ph. D. (c) Universidad del Cauca (Popayán-Cauca, Colombia). 
samador@unicauca.edu.co

César-Jesús Pardo-Calvache 
Ph. D. Universidad del Cauca (Popayán-Cauca, Colombia). 
cpardo@unicauca.edu.co

Received / Recibido: 20/07/2024

Accepted / Aceptado: 25/09/2024



ABSTRACT

Cryptography is one of the branches of research within computer security and cybersecurity, it provides security to the stored information and travels between devices. Cryptanalysis, in turn, studies the weaknesses within cryptography, thus allowing improving constants about cryptographic algorithms. Currently there are several algorithms that allow to keep information secure, one of them is RSA (Rivest, Shamir and Adleman), which is used in digital certificates implemented in some communication protocols. However, there is no algorithm capable of deciphering that type of algorithms yet; therefore, the objective of this study is to support other researchers in the area of cryptanalysis. This rapid factorization study using parallel or distributed processing contains 6 research questions that allow us to deepen the use of this type of processing to speed up the execution times of the algorithms. The results made it possible to show that by using this type of processing, factoring time can be reduced.

Keywords: Cryptanalysis; distributed processing; factoring; parallel processing.

RESUMEN

La criptografía es una de las ramas de investigación dentro de la seguridad informática y la ciberseguridad, proporciona seguridad a la información almacenada y viaja entre dispositivos. El criptoanálisis, a su vez, estudia las debilidades dentro de la criptografía, permitiendo así mejorar las constantes en los algoritmos criptográficos. Actualmente existen varios algoritmos que permiten mantener la información segura, uno de ellos es RSA (Rivest, Shamir y Adleman), que se utiliza

Esta edición se financió con recursos del Patrimonio Autónomo Fondo Nacional de Financiamiento para la Ciencia, la Tecnología y la Innovación, Francisco José de Caldas, Minciencias

How to cite: J-A. Melo, S. Amador-Donado, C-J. Pardo-Calvache, "Systematic Mapping Study on Fast Factorization Using Parallel or Distributed Processing Applied to Cryptanalysis", *Revista Facultad de Ingeniería*, vol. 33, no. 69, e17935, 2024. <https://doi.org/10.19053/01211129.v33.n69.2024.17935>

en certificados digitales implementados en algunos protocolos de comunicación. Sin embargo, aún no existe un algoritmo capaz de descifrar este otro tipo de algoritmos; por lo tanto, el objetivo de este estudio es apoyar a otros investigadores en el área del criptoanálisis. Este estudio de factorización rápida mediante procesamiento paralelo o distribuido contiene 6 preguntas de investigación que nos permiten profundizar en el uso de este tipo de procesamiento para acelerar los tiempos de ejecución de los algoritmos. Los resultados permitieron demostrar que, mediante el uso de este tipo de procesamiento, se puede reducir el tiempo de factorización.

Palabras clave: Criptoanálisis; factorización; procesamiento distribuido; procesamiento paralelo.

1. INTRODUCTION

Cryptography is crucial for information security, from the Caesar cipher—named after Julius Caesar—to modern cryptographic algorithms like RSA. These systems have evolved to protect data storage, processing, and transmission over networks. Today, privacy and confidentiality in online operations depend on encryption algorithms; for example, RSA uses the HyperText Transfer Protocol Secure (HTTPS) to secure data transmission, whether personal or financial. A 2022 study by Entrust showed that 62% of companies use encryption strategies, reflecting a 19% increase since 2018, and highlighting the importance of continued research on these algorithms [1].

Cryptanalysis studies methods for deciphering encrypted messages without having the key. The factorization of semi-prime numbers is a vital approach to compromising the security of public-key algorithms like RSA. The security of RSA relies on the difficulty of factoring very large composite numbers into their prime factors. For example, if asked to factor the number 15, one can easily identify 3 and 5 as its prime factors. However, for a number as large as RSA-2048, for example <https://acortar.link/c48jLQ>, which has 617 digits, finding its prime factors is extremely difficult, even for a non-quantum supercomputer. Advances in factorization techniques, such as the Multiple Polynomial Quadratic Sieve (MPQS) and the General Number Field Sieve (GNFS), require longer encryption keys to maintain security.

This study analyzes cryptanalysis in the factorization of semi-prime numbers through a systematic mapping of the literature. Factorization methods for numbers with more than 10 digits were identified, including the use of artificial intelligence. Although parallel processing has improved execution times, the complexity remains high. Therefore, it is important to explore how emerging technologies can assist. The document is organized as follows: methodology, results and discussions, and conclusions.

2. METHODOLOGY

2.1. Definition of Search Objectives and Research Questions

A systematic mapping is a process that allows the collection, categorization, and structuring of existing information on a topic of research interest [2]. For the design of this systematic mapping, the protocol proposed by Petersen et al. in [3] was used as a reference along with the guidelines presented by Kitchenham and Charters in [4] and Budgen et al. in [5]. The following activities were carried out: (i) Apply a GQM approach; (ii) Define a search and selection strategy; (iii) Conduct a review; (iv) Generate a review report. [Figure 1](#) presents a more detailed diagram of the process.

To effectively direct the present systematic mapping, the following search objectives (OB) have been defined:

- OB1.** Identify the types of solutions proposed in the selected primary studies and group them to determine the most relevant ones.
- OB2.** Analyze the main contributions to rapid factorization, considering the types of solutions proposed to identify those with the greatest impact.
- OB3.** Support academics, cryptanalysts, and other interested parties in researching rapid factorization by presenting the challenges encountered in the selected primary studies.

Based on the search objectives, three (3) research questions (P) have been defined, as presented in Table 1. Each question is mapped to the proposed objectives, along with its respective motivation.

For research question P1, the types of solutions described in Table 2 were defined.

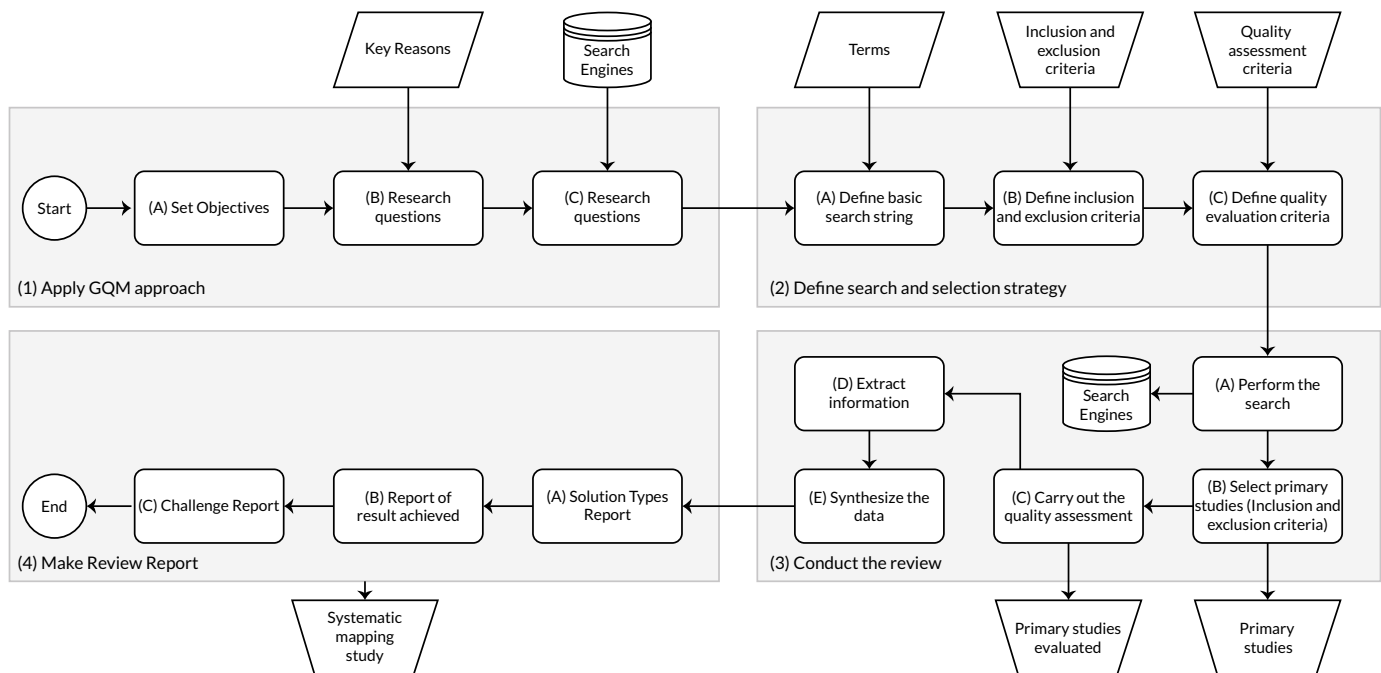


Figure 1. Stages of the process for systematic mapping. Suggested by [6].

Table 1. Research questions.

Id.	Research question	Motivation	Objectives
P1	What types of solutions have been proposed?	Identify the types of contributions described in Table 2 for the selected studies.	OB1
P2	What results have been achieved with the proposed solutions?	Identify the impact of the proposed solutions based on the results obtained.	OB2
P3	What challenges does researching the topic involve?	Determine the challenges that researching the topic presents for the scientific community.	OB3

Table 2. Types of solutions.

Type of solution	Description
Parallel processing	Articles that use parallel processing to accelerate the factorization process.
Distributed processing	Articles that use distributed processing to accelerate the factorization process.
Performance evaluation	Articles that propose experiments or evaluations to measure the performance of factorization algorithms.
Low-level instructions	Articles that use low-level instructions, such as assembly code or SIMD instructions, to accelerate the factorization process.
Artificial intelligence	Articles that use artificial intelligence to enhance factorization processes.
Code optimization	Articles that propose improvements to optimize the steps of factorization algorithms.

2.2. Search strategy

The following activities were carried out to search for articles: (i) identification of key terms through document review and consultation with cryptography experts; (ii) combination of terms using “AND” and “OR;” (iii) refinement of the search string; (iv) adaptation of the string to different search engines; and (v) definition of the time frame and inclusion/exclusion criteria. The refined search string was (“factoring large numbers” OR “factoring large integers”) AND (“fast” OR “speed up”) AND (“cryptography” OR “cryptanalysis”) AND (“sieve” OR “sieving”) AND (“parallel” OR “distributed” OR “optimized”). A period from January 2010 to December 2022 was defined. Searches were conducted in Google Scholar, ACM Digital Library, IEEE Digital Library, ScienceDirect, Scopus, and Springer Link, adapting the string to each database. Details of the adaptations are presented in [Table 3](#), and findings are shown in [Table 8](#), including discarded articles and those with no available access.

2.3. Inclusion and exclusion criteria

For the selection of relevant articles, a two-level review was conducted: (Level 1) review of the title; (Level 2) review of the abstract, introduction, and conclusions. To obtain relevant articles, only those studies that met at least one of the inclusion criteria described in [Table 4](#) and did not meet any of the exclusion criteria listed in [Table 5](#) were selected. Subsequently, to achieve the objectives and identify the primary studies, a (Level 3) review of the full text was made.

2.4. Quality assessment criteria

To measure the quality of the selected primary studies and determine their relevance to rapid factorization using parallel or distributed processing, a questionnaire containing twelve (12) questions was created with a scoring system of three values (-1, 0, +1), as described in [Table 6](#). Each article can receive a quality score ranging from -12 to +12. It is important to clarify that a low-quality score does not imply exclusion but rather helps in ranking articles by relevance for future research.

2.5. Data extraction

The following link <https://acortar.link/UPrRAU> presents the summary sheet that ensured a uniform data extraction strategy for all articles, making it easier to classify the information. This sheet summarizes the most important aspects to be considered for each article.

Table 3. Adaptation of the basic search string in the databases.

No.	Search string	Source
1	("factoring large numbers" OR "factoring large integers") AND ("fast" OR "speed up") AND ("cryptography" OR "cryptanalysis") AND ("sieve" OR "sieving") AND ("parallel" OR "distributed" OR "optimized")	Google Scholar
2	[[All: "factoring large numbers"] OR [All: "factoring large integers"]] AND [[All: "fast" OR [All: "speed up"]]] AND [[All: "cryptography" OR [All: "cryptanalysis"]]] AND [[All: "sieve" OR [All: "sieving"]]] AND [[All: "parallel" OR [All: "distributed" OR [All: "optimized"]]] AND [Publication Date: (01/01/2010 TO 07/31/2022)]	ACM Digital Library
3	("factoring large numbers" OR "factoring large integers") AND ("fast" OR "speed up") AND ("cryptography" OR "cryptanalysis") AND ("sieve" OR "sieving") AND ("parallel" OR "distributed" OR "optimized")	IEEE Digital Library
4	("factoring large numbers" OR "factoring large integers") AND ("fast" OR "speed up") AND ("cryptography" OR "cryptanalysis") AND ("sieving") AND ("parallel" OR "distributed")	ScienceDirect
5	("factoring large numbers" OR "factoring large integers") AND ("fast" OR "speed up") AND ("cryptography" OR "cryptanalysis") AND ("sieve" OR "sieving") AND ("parallel" OR "distributed" OR "optimized")	Scopus
6	("factoring large numbers" OR "factoring large integers") AND ("fast" OR "speed up") AND ("cryptography" OR "cryptanalysis") AND ("sieve" OR "sieving") AND ("parallel" OR "distributed" OR "optimized")	Springer Link

Table 4. Inclusion criteria.

Id.	Inclusion criteria (IC)
IC1	Studies whose main topic is rapid factorization using distributed algorithms.
IC2	Studies related to rapid factorization using distributed algorithms.
IC3	Studies that have been published in scientific journals and postgraduate theses.

Table 5. Exclusion criteria.

Id.	Exclusion criteria (EC)
EC1	Duplicate articles or studies.
EC2	Studies that address the topic superficially or do not address the research topic.
EC3	Books or book chapters.
EC4	Undergraduate theses.
EC5	Studies published before 2010.
EC6	Studies in languages other than English.

Table 6. Criteria for evaluating the quality of primary studies.

Id.	Criterion	Scoring for responses		
		+1	0	-1
C1	The study focuses on investigating rapid factorization of numbers using parallel or distributed processing.	Yes	Partially	No
C2	The study provides a clear description of the research problem.	Yes	Partially	No
C3	The study follows a structured and well-founded research process.	Yes	Partially	No
C4	The study provides clear definitions of rapid factorization and/or distributed processing.	Yes	Partially	No
C5	The study proposes a set of techniques to reduce the factorization time.	Yes	Partially	No
C6	The study proposes a method for evaluating factorization times.	Yes	Partially	No
C7	The study clearly and comprehensively presents the results obtained after validating its proposal.	Yes	Partially	No
C8	The study clearly presents the research contributions to industry and/or academia.	Yes	Partially	No
C9	The study clearly describes the discussion of the limitations of the research process and the analysis of the results obtained.	Yes	Partially	No
C10	The study clearly describes future work or research alternatives.	Yes	Partially	No
C11	The study has been published in a relevant journal, conference, or symposium. The journal classification was based on the quartile rankings proposed by Scimago (https://www.scimagojr.com), and the conference and symposium rankings were based on the Computing Research & Education (CORE) rankings (http://portal.core.edu.au/conf-ranks/).	Highly relevant (quartile Q1 for journals and A* for conferences and symposia).	Relevant (quartiles Q2 and Q3 for journals and A or B for conferences and symposia).	Not relevant (quartile Q4 for journals and C or unclassified for conferences and symposia).
C12	The study has been cited by other authors (according to the Google Scholar citation index).	Has been cited by more than ten authors.	Between one and ten authors.	Has not been cited to date.

3. RESULTS AND DISCUSSIONS

It is important to clarify that the search and selection of articles began by entering the search string in Google Scholar, which was also the data source that yielded the most results. For other sources, most of the articles found were not selected due to EC2. As a result, as shown in Table 7, the total number of selected articles that met at least one IC and did not meet any EC was 15, except for 2 articles selected through backward snowballing. A total of 309 articles were excluded. The compendium of primary articles resulting from the search is presented in Table 8 along with their references, so that readers can explore them further. The results of the quality assessment are presented in Table 9, highlighting that only 2 articles (A6, A13) achieved the highest rating with 9 points each. It is also noteworthy that most articles scored above 1 point, with 5 articles scoring 5 points.

Table 7. Search results.

#	Data source	Found	Relevant	Primary Selected
1	Google Scholar	246	42	15
2	ACM Digital Library	2	0	0
3	IEEE Xplore	1	0	0
4	Science Direct	5	0	0
5	Scopus	2	0	0
6	Springer Link	68	3	0
7	Backward snowballing	2	2	2
TOTAL		326	47	17

Table 8. Compendium of primary articles resulting from the search.

Id	Title	No. of citations	Year	Reference
A1	An Empirical Comparison of the Quadratic Sieve Factoring Algorithm and the Pollard Rho Factoring Algorithm	0	2021	[7]
A2	Parallel structured gaussian elimination for the number field sieve	2	2021	[8]
A3	Fermat factorization using a multi-core system	3	2020	[9]
A4	Factoring integers with a brain-inspired computer	9	2018	[10]
A5	An improved parallel block Lanczos algorithm over GF (2) for integer factorization	12	2017	[11]
A6	Use of SIMD-based data parallelism to speed up sieving in integer-factorizing algorithms	13	2017	[12]
A7	Efficiency of RSA key factorization by open-source libraries and distributed system architecture	3	2017	[13]
A8	Fast cryptanalysis of RSA encrypted data using a combination of mathematical and brute force attack in distributed computing environment	2	2017	[14]
A9	Impact of Optimization and Parallelism on Factorization Speed of SIQS	2	2016	[15]
A10	Strategy of Relations Collection in Factoring RSA Modulus	0	2016	[16]
A11	A parallel line sieve for the GNFS Algorithm	1	2014	[17]
A12	SIMD-Based Implementations of Sieving in Integer-Factoring Algorithms	1	2013	[18]
A13	An integrated parallel GNFS algorithm for integer factorization based on Linbox Montgomery block Lanczos method over GF (2)	13	2010	[19]
A14	Nios II hardware acceleration of the epsilon quadratic sieve algorithm	3	2010	[20]
A15	Application of bio-inspired algorithm to the problem of integer factorization	23	2010	[21]
A16	Parallel Multiple Polynomial Quadratic Sieve on Multi-core Architectures	6	2007	[22]
A17	Integer Factorization by a Parallel GNFS Algorithm for Public Key Cryptosystems	18	2005	[23]

3.1. P1: What types of solutions have been proposed?

As shown in Table 10, 45% (10 articles) of the studies propose the use of parallel processing to accelerate the factorization process. Among these articles, A5, A9, and A10 use code optimization, while A6 employs low-level instructions to enhance the sieving performance. Articles A2, A3, A11, A13, A16, and A17 use parallel processing in one or more common steps of algorithms such as MPQS, GNFS, and Fermat. 18% (4

articles) of the studies use code optimization to improve the methods or functions of each factorization algorithm. Studies A5, A9, A10, and A14 use both code optimization and parallel processing or low-level instructions. 14% (3 articles) of the studies use low-level instructions that provide faster response times between software and hardware, thus accelerating factorization processes. A6 and A14 also employ parallel processing and code optimization, while A12 focuses solely on low-level instructions. Regarding artificial intelligence (AI), which represents 9% (2 articles), A4 and A15 use neuromorphic computing and genetic algorithms, proposing a different type of solution where improvements to algorithms are commonly found. Similarly, performance evaluations account for 9% (2 articles). A1 and A14 evaluate the performance of algorithms or tools for factoring large numbers. It is worth noting that all articles include a section on evaluation or experiments for their solutions. Finally, distributed processing represents 5% (1 article). Shende et al. in A14 apply distributed processing using mobile agents across several servers to divide processing among multiple machines, thereby reducing factorization times.

Table 10 presents the analysis and summary of the different types of solutions found in the reviewed literature on large number factorization. These types of solutions include parallel processing, artificial intelligence, code optimization, low-level instructions, performance evaluation, and distributed processing. Each type of solution is listed along with the corresponding articles that use it and the percentage represented in the total number of reviewed articles.

Table 10 presents the analysis and summary of the different types of solutions found in the reviewed literature on large number factorization. These types of solutions include parallel processing, artificial intelligence, code optimization, low-level instructions, performance evaluation, and distributed processing. Each type of solution is listed along with the corresponding articles that use it and the percentage represented in the total number of reviewed articles.

Table 9. Quality rating for each article.

Article	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	Total
A1	-1	0	1	0	-1	1	1	0	1	0	-1	-1	0
A2	1	0	-1	0	1	0	1	0	0	-1	-1	0	0
A3	1	1	-1	1	1	1	1	0	0	0	0	0	5
A4	-1	1	-1	1	1	1	0	0	0	0	1	0	-1
A5	1	0	-1	1	1	1	1	0	0	-1	1	1	5
A6	1	1	-1	1	1	1	1	0	1	1	1	1	9
A7	0	1	-1	1	0	1	1	0	0	-1	-1	0	1
A8	1	1	-1	1	1	1	1	0	1	-1	-1	0	4
A9	1	1	-1	0	1	1	1	0	1	0	-1	0	4
A10	1	1	-1	0	1	1	1	1	1	0	0	-1	5
A11	1	1	-1	1	1	0	1	0	1	-1	0	1	5
A12	0	1	-1	0	1	1	1	0	1	0	0	0	4
A13	1	1	-1	1	1	1	1	0	1	1	1	1	9
A14	0	1	-1	-1	1	0	0	0	1	0	-1	0	0
A15	-1	0	-1	-1	1	0	1	0	1	1	1	1	3
A16	1	1	-1	1	1	1	1	0	1	0	-1	0	5
A17	1	1	-1	1	1	1	1	0	1	0	0	1	7

Table 10. Classification of primary articles by type of solution.

Type of solution	Article	%
Parallel processing	A2, A3, A5, A6, A9, A10, A11, A13, A16, A17	45
Artificial intelligence	A4, A15	9
Code optimization	A5, A9, A10, A14	18
Low-level instructions	A6, A12, A14	14
Performance evaluation	A1, A7	9
Distributed processing	A8	5

3.2. P2: What results have been achieved with the proposed solutions?

This study identified 17 articles with various proposals focused on the factorization of large numbers. Primary studies are categorized into different types of solutions, with parallel processing showing the highest percentage (45%). This technique is commonly used in factorization algorithms. This section provides a more detailed exploration of the types of solutions found.

In the parallel processing solutions, A2, A5, and A13 focus on accelerating sparse matrix solutions, a key component in some factorization algorithms. These studies use parallel processing and code optimization to speed up cryptanalysis. A3 proposes a parallel implementation of the Fermat factorization method, demonstrating its significantly faster performance compared to sequential approaches. A6 and A14, in addition to parallel processing, employ low-level instructions that execute more rapidly, enhancing screening in algorithms such as the Quadratic Sieve (QS), Multiple Polynomial Quadratic Sieve (MPQS), and Number Field Sieve (NFS).

A9 compares the sequential and parallel versions of the Self-Initialization Quadratic Sieve (SIQS) algorithm, showing superior performance for the parallel version. A10 introduces a new parallel module for collecting relations to find B-smooth numbers for the screening process in algorithms. A11 implements parallel screening in the NFS algorithm, evaluating its performance compared to the sequential version, highlighting the advantages of the parallel algorithm. A16 proposes parallelizing the MPQS algorithm using parallel symbolic computations, demonstrating reduced factorization times with more processors. A17 presents a parallel implementation of the NFS algorithm on a SUN cluster, achieving favorable execution times. Similarly, A8 uses mobile agents to distribute QS algorithm processing, showing that using three machines reduces factorization time.

In a different approach, A4 and A15 focus on artificial intelligence. A4 uses neuromorphic computing to propose a neuromorphic screening method for identifying B-smooth numbers, while A15 introduces a genetic algorithm for integer factorization, successfully factoring numbers up to 8 digits. A12 employs low-level instructions to accelerate screening in the MPQS and NFS algorithms, achieving speedups of 15% to 40%. Finally, A1 and A7 are dedicated to performance evaluation, with A1 focusing on QS and Pollard's rho (PR) algorithms, and A7 on factorization tools like MSieve, GGNSF, and CADO-NFS. Although other studies also include evaluations and experiments, these two specifically concentrate on performance assessment.

3.3. P3: What is the main challenge of factoring large numbers?

There are various challenges in researching fast factorization, with the selected studies revealing interrelated issues concerning computational complexity and algorithm optimization. The increase in

computational effort required as numbers grow becomes a key point, exacerbated by the demand for faster sieving. In the context of parallel implementation, obstacles include synchronization between processors, memory optimization, and the efficient selection of algorithms and parameters. Proper memory allocation, managing calculation complexity, effective synchronization, and process efficiency are critical factors to address. [Table 11](#) details each of these challenges in the selected studies.

3.4. Main Observations

Systematic mapping allowed for the identification of related works on fast factorization using parallel or distributed processing. After analyzing the results, the following observations are made:

- There is significant interest from the scientific and academic community in this type of research, with various approaches and perspectives presented in [Table 11](#) to tackle the problem. In the fields of engineering and mathematics, extensive and diverse research has been conducted to reduce factorization times. This study reveals several techniques that can assist other researchers in improving the performance of various algorithms.
- Many proposals focus on dividing the problem to reduce factorization times. Numerous studies, as shown in [Table 11](#), utilize parallel processing to enhance the performance of factorization algorithms. Studies that employed this approach demonstrated a reduction in execution time compared to sequential implementations.
- Many studies agree that computational complexity is one of the most relevant challenges, which continuously increases as the numbers to be factored grow in digits.
- Quantum computing could potentially accelerate the execution times of factorization algorithms, as many studies highlight that current resources are inadequate for such demanding processes in terms of processing power and information storage.

4. CONCLUSIONS

In this study, which covers 17 research articles dedicated to the factorization of large numbers using various methods and/or techniques to address the computational challenge of performing time-costly operations, it was evident that most approaches focus on the implementation of parallel processing techniques, representing 45% of the studies. This approach particularly leverages the performance of various processors to accelerate processes such as sieving or solving sparse matrices, which are critical steps in cryptanalysis algorithms like QS, MPQS, and NFS. This approach showed significant improvements in algorithm performance compared to their sequential implementations.

The study provides a diverse perspective on the strategies used by researchers in the field of cryptanalysis, such as code optimization, low-level instructions, AI techniques, and distributed processing. By evaluating the performance of various algorithms and factorization tools, these approaches enrich the research field and should continue to be explored as alternatives to the majority of proposed solutions for accelerating factorization processes. Most of these approaches demonstrated performance improvements that help reduce the factorization times of some algorithms.

Table 11. Challenges in factorization

Article	Main challenges
A1	The study faced challenges related to time constraints, the complexity of large numbers, variability in prime factors, and the comparison of factorization algorithm performance. These factors influence the difficulty of efficiently addressing the factorization of large numbers.
A2	The study faced challenges in concurrency and memory management when implementing a parallel algorithm for NFS and overcame these obstacles through solutions such as localized row allocation and memory optimization techniques.
A3	In the context of this study, the researchers encountered obstacles when employing high-performance computing to accelerate the Fermat Factorization (FF) algorithm. These difficulties included the task of breaking the problem into equally sized parts, efficient communication between processors, and the interdependencies present in some solution steps. Additionally, a lack of previous research on parallelizing the FF algorithm, particularly from the perspective of parallel computing, was identified.
A4	This study encountered difficulties such as peak collisions and synchronization issues in a neuromorphic approach. The factor base partitioning strategy also presented challenges. These complexities highlight the need for further research to overcome these difficulties in factoring large numbers.
A5	The study faced challenges in factoring due to the exponential time required to find the prime factors of a large semiprime number. Issues such as matrix and vector multiplications were encountered in the parallel Block Lanczos algorithm. Additionally, communication and synchronization stages in the parallel implementation took significant time. These challenges were addressed by redesigning the algorithm to improve efficiency.
A6	In this study, challenges such as computational complexity, parallelizing the sieving in factorization algorithms using SIMD, and reducing data packing overhead using online and lattice sieving were faced. Intel SSE2 and AVX instructions were used to overcome these difficulties, achieving acceleration of 5-40%.
A7	This study addressed challenges in choosing the appropriate open-source factorization library for large numbers on a computer cluster. They compared the efficiency of libraries such as Msieve, GGNFS, and CADO-NFS, noting that Msieve was not optimal for sieving but showed improvements when combined with GGNFS. In the cluster implementation, unexpected increases in time were observed, possibly due to inefficient distribution. Additionally, they faced difficulties in estimating the time required to factor large RSA numbers.
A8	The challenge in factoring large numbers is the computational complexity in finding their prime factors. Given the lack of efficient algorithms for this task, especially for numbers resulting from the multiplication of large primes, the process consumes significant time and resources. In the study, the lack of numbers with known factors limited the generalization of results, and scalability faced obstacles due to the exponential increase in computational requirements with larger numbers. In summary, complexity, lack of data, and scalability marked the difficulties in this study.
A9	Factoring large numbers involves increasing computational complexity with the size of the number, which requires more time. This study identified challenges in implementing and optimizing the SIQS algorithm for this task. Factoring numbers with more than 70 digits became slow due to its complexity. Memory management was improved with a modified storage approach. The sieving phase, the longest part, faced difficulties with polynomial calculations and candidate selection. The search for optimal performance for different numbers was addressed by time and memory analysis and adjustments. In summary, the study tackled challenges related to computational complexity, memory management, and optimization of the SIQS algorithm for factoring large numbers.
A10	The difficulties addressed in this study include the computational complexity of factoring large numbers, the challenge of finding prime factors, the need for efficient algorithms and hardware designs, and the limitations of real-world systems.

Table 11. Challenges in factorization (Continued)

A11	In the study of the parallel implementation of the GNFS algorithm for factoring large integers, key obstacles were identified, including the time consumed by the sieving step, inefficiencies in previous implementations due to excessive communication and load imbalance, and the goal of parallelizing all steps of the algorithm. These difficulties were addressed in the study to improve the efficiency and acceleration of the parallel implementation of GNFS in the factoring of large integers.
A12	The researchers faced difficulties in trying to reduce the overhead of packing and unpacking associated with SIMD-based parallelism. While they achieved some acceleration in index calculations, performing it in the subtraction phase proved challenging. It was observed that the frequent data exchange between regular registers and SIMD hindered the benefits of data parallelism. Surprisingly, there were no attempts at SIMD-based parallelization in sieving algorithms found in the literature.
A13	The main challenge in factoring large numbers is the time complexity of algorithms like the General Number Field Sieve (GNFS). Although GNFS can factor numbers with more than 140 digits, it takes several months; the slowest part is the sieving and solving the linear system using the Montgomery Block Lanczos method, which is time-consuming. The challenge is to find more efficient and faster methods for these stages to factor large numbers in a reasonable amount of time.
A14	The central challenge in factoring large numbers is the increasing computational effort as the numbers grow, as calculating polynomials and performing sieving tasks take more time with larger numbers. This study faced similar difficulties, highlighting the increased computational effort, the need to compute various polynomials and perform sieving, and the constraints of hardware and software platforms in terms of performance and portability.
A15	The primary factor in factoring large numbers is the intensification of computational effort as the numbers grow, since polynomial computations and sieving become more prolonged with larger numbers. This study faced similar difficulties, highlighting the increase in computational effort, the need to calculate various polynomials and perform sieving, and the limitations regarding performance and portability of hardware and software platforms.
A16	In this study, challenges related to the unpredictability of data dependencies, irregular data access patterns, and variation in parameters in symbolic computations were identified, thus making it difficult to predict memory and processor usage patterns. Transitioning from shared-memory to distributed computing required redesigning the existing software. Synchronization between threads during the sieving stage introduced delays and communication overhead, and the choice between MPI (Message Passing Interface) with shared memory or multiple threads depended on the length of the number.
A17	The study presented difficulties during the implementation of the GNFS algorithm. Among these, the high memory demand for larger composite numbers was highlighted, thus posing a challenge in terms of memory usage, even for numbers with fewer than 116 digits. Additionally, synchronization issues were noted, as each processor performs sieving on different pairs and the master node waits for the slave nodes to finish, which can affect efficiency. Furthermore, the current difficulty in factoring integers larger than 116 digits was mentioned due to memory requirements and parameter selection that impact the efficiency of the GNFS algorithm.

There are several challenges surrounding research on fast factorization of semi-prime numbers. One of the most significant is the increase in computational effort required as the numbers to be factored grow in bit length. This complexity is further intensified by the volume of operations and results that need to be stored in memory. One of the most relevant challenges in the context of parallel processing is synchronization between processors and memory management, as inadequate control or optimization can impact the factorization process.

Finally, it is worth noting that despite recent advances in computing and mathematics, it is still not possible to factorize numbers as large as a 2048-bit RSA key within a reasonable time frame. The largest number factored to date is RSA-250 in February 2020 [24], an 829-bit number that would have taken about 2700 years to factorize with a single core. However, by utilizing parallel and distributed processing, it was factorized in a few months using multiple machines around the world. Given that this is an exponential problem, researchers in [24] estimate that a 1024-bit number would be 200 times harder to factor. Studies in quantum technology, such as [25] suggest that with a quantum computer, factorization could be achieved in polynomial time, leading to significantly shorter times compared to current algorithms, which have exponential complexity. This poses a challenge for current information security, prompting the development of quantum cryptographic systems [26].

AUTHOR CONTRIBUTIONS

Jhon-Alejandro Melo: Conceptualization, data curation, formal analysis, research, software, visualization and writing - original draft. **Siler Amador-Donado:** Conceptualization, project administration, methodology, supervision and writing - review and editing. **César-Jesús Pardo-Calvache:** Conceptualization, project administration and writing - review and editing.

ACKNOWLEDGMENTS

To the University of Cauca, especially to the SEC (Security, Encryption & Cybersecurity) research group, the GTI research group, and the Altenua-Matdis research group for the support provided in the development of this work. Professors Siler Amador Donado and César Pardo are grateful for the contribution of the Universidad del Cauca, where they currently work as a full professors.

REFERENCES

- [1] Ponemon Institute, *Global Encryption Trends Study: The data is in the cloud, but who's in control?* 2022. Available: <https://www.entrust.com/es/resources/reports/global-encryption-trends-study>
- [2] E. Suescún-Monsalve, J.-C. Sampaio-do-Prado-Leite, C.-J. Pardo-Calvache, "Semi-Automatic Mapping Technique Using Snowballing to Support Massive Literature Searches in Software Engineering," *Revista Facultad de Ingeniería*, vol. 31, no. 60, e14189, May 2022, <https://doi.org/10.19053/01211129.v31.n60.2022.14189>
- [3] K. Petersen, H. Flensburg, R. Feldt, M. Mattsson, S. Mujtaba, *Systematic Mapping Studies in Software Engineering*, 2008.

- [4] B. Kitchenham, S. M. Charters, *Guidelines for performing Systematic Literature Reviews in Software Engineering*, 2007.
- [5] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, *Using Mapping Studies in Software Engineering*, 2008. <https://www.ebse.org.uk>
- [6] E. Nicolás, P. Paredes, C. E. Orozco, C. Pardo, "Análisis del estado del arte acerca de la (in)felicidad en las comunidades de desarrollo de software ágil," *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, vol. 57, pp. 425-437, 2023.
- [7] Z. Li and W. Gasarch, *An Empirical Comparison of the Quadratic Sieve Factoring Algorithm and the Pollard Rho Factoring Algorithm*, 2021. <https://doi.org/10.48550/arXiv.2111.02967>
- [8] C. Bouillaguet, P. Zimmermann, *Parallel Structured Gaussian Elimination for the Number Field Sieve*, 2021.
- [9] H. M. Bahig, H. M. Bahig, Y. Kotb, "Fermat factorization using a multi-core system," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 4, pp. 323-330, 2020. <https://doi.org/10.14569/IJACSA.2020.0110444>
- [10] J. V. Monaco, M. M. Vindiola, "Factoring Integers with a Brain-Inspired Computer," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 3, pp. 1051-1062, 2018. <https://doi.org/10.1109/TCSI.2017.2771533>
- [11] L. T. Yang, Y. Huang, J. Feng, Q. Pan, C. Zhu, "An improved parallel block Lanczos algorithm over GF(2) for integer factorization," *Information Sciences*, vol. 379, no. 2, pp. 257-273, 2017. <https://doi.org/10.1016/j.ins.2016.09.052>
- [12] B. Sengupta, A. Das, "Use of SIMD-based data parallelism to speed up sieving in integer-factoring algorithms," *Appl Math Comput*, vol. 293, pp. 204-217, 2017. <https://doi.org/10.1016/j.amc.2016.08.019>
- [13] E. J. Vuicik, D. Šešok, S. Ramanauskaitė, "Efficiency of RSA Key Factorization by Open-Source Libraries and Distributed System Architecture," *Baltic Journal of Modern Computing*, vol. 5, no. 3, pp. 269-274, 2017. <https://doi.org/10.22364/bjmc.2017.5.3.02>
- [14] V. Shende, G. Sudi, M. Kulkarni, "Fast cryptanalysis of RSA encrypted data using a combination of mathematical and brute force attack in distributed computing environment," in *IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, 2017, pp. 2446-2449. <https://doi.org/10.1109/ICPCSI.2017.8392156>
- [15] D. Breitenbacher, I. Homoliak, J. Jaros, P. Hanacek, "Impact of optimization and parallelism on factorization speed of SIQS," in *20th World Multi-Conference on Systemics, Cybernetics and Informatics*, 2016, pp. 55-62.
- [16] H. Yu, G. Bai, "Strategy of Relations Collection in Factoring RSA Modulus," *Lecture Notes in Computer Science*, vol. 9543, pp. 199-211, 2016. https://doi.org/10.1007/978-3-319-29814-6_16
- [17] S. Daoud, I. Gad, "A parallel line sieve for the GNFS Algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 7, pp. 178-185, 2014. <https://doi.org/10.14569/ijacsa.2014.050727>
- [18] B. Sengupta, A. Das, "SIMD-based implementations of sieving in integer-factoring algorithms," *Lecture Notes in Computer Science*, vol. 8204, pp. 40-55, 2013. https://doi.org/10.1007/978-3-642-41224-0_4

- [19] L. T. Yang, L. Xu, S. S. Yeo, S. Hussain, "An integrated parallel GNFS algorithm for integer factorization based on Linbox Montgomery block Lanczos method over GF(2)," *Computers and Mathematics with Applications*, vol. 60, no. 2, pp. 338-346, 2010. <https://doi.org/10.1016/j.camwa.2010.01.020>
- [20] U. Meyer-Bäse, G. Botella, E. Castillo, A. García, "Nios II hardware acceleration of the epsilon quadratic sieve algorithm," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 7703, e77030M, 2010. <https://doi.org/10.1117/12.849883>
- [21] R. V. Yampolskiy, "Application of bio-inspired algorithm to the problem of integer factorization," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 115-123, 2010. <https://doi.org/10.1504/IJBIC.2010.032127>
- [22] G. Macariu, D. Petcu, "Parallel multiple polynomial quadratic sieves on multi-core architectures," in *9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 59-65, 2007. <https://doi.org/10.1109/SYNASC.2007.21>
- [23] L. T. Yang, L. Xu, M. Lin, "Integer factorization by a parallel GNFS algorithm for public key cryptosystems," *Lecture Notes in Computer Science*, vol. 3820, pp. 683-695, 2005. https://doi.org/10.1007/11599555_65
- [24] F. Boudot et al., "State of the Art in Integer Factoring and Breaking Public-Key Cryptography," *IEEE Security and Privacy Magazine*, vol. 2022, no. 2, e314918. <https://doi.org/10.1109/MSEC.2022.3141918>
- [25] R. Young, P. Birch, C. Chatwin, *A simplification of the Shor quantum factorization algorithm employing a quantum Hadamard transform*, 2018. <https://doi.org/10.1117/12.2309468>
- [26] S. K. Sehgal, R. Gupta, "Quantum Cryptography and Quantum Key," in *International Conference on Industrial Electronics Research and Applications*, 2021, pp. 1-5. <https://doi.org/10.1109/ICIERA53202.2021.9726722>