

<https://doi.org/10.31891/csit-2026-3-14>

Stepan IVASIEV

Associate Professor, Candidate of Technical Sciences, and Associate Professor of the Department of Cybersecurity, Faculty of Computer Information Technologies, West Ukrainian National University
<https://orcid.org/0000-0003-2243-5956>
isv@wunu.edu.ua

Received: 04/06/2026

Accepted: 17/08/2026

Published: 30/09/2026

© Copyright
2026 by the author(s)



This is an Open Access article distributed under the terms of the [Creative Commons CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/)

UDC 004.04

METHOD FOR LARGE INTEGER FACTORIZATION BASED ON THE RESIDUE NUMBER SYSTEM

The paper investigates the problem of factoring large integers, which remains one of the fundamental challenges in modern number theory and public-key cryptography. An analysis of the classical Fermat factorization method is performed, and its computational bottlenecks associated with repeated square computations and exhaustive iteration are identified. To overcome these limitations, a modified version of the Fermat method based on a recurrent computation of the difference of squares is proposed. The proposed modification eliminates repeated exponentiation operations by replacing them with incremental arithmetic operations, thereby creating the mathematical foundation for the application of residue number systems.

Based on the proposed modification, two novel factorization algorithms are developed. The first algorithm employs a key-folding mechanism in the Residue Number System, where sets of admissible quadratic residues are generated for individual moduli and subsequently combined according to the Chinese Remainder Theorem. This approach provides a multistage algebraic filtering procedure that substantially reduces the number of candidate values requiring verification in the positional number system. The second algorithm introduces an offset-lattice approach, in which the search process is transferred from the value space to the iteration-index space. By constructing periodic sets of admissible offsets for groups of moduli, the algorithm performs lattice-based filtering and evaluates only those iterations satisfying all modular constraints, significantly decreasing the number of expensive perfect-square tests.

The computational complexity of the proposed algorithms is analyzed and compared with that of the classical Fermat method. Theoretical analysis demonstrates that both algorithms reduce the effective search space through modular filtering, while the offset-lattice algorithm additionally decreases the number of examined iterations by exploiting the periodic structure of admissible offsets. Owing to the independence of computations for different moduli, both algorithms naturally support parallel implementation, making them suitable for execution on multicore and distributed computing platforms.

Experimental studies performed on integers of various sizes confirm the effectiveness of the proposed approaches. Compared with the classical Fermat method, the developed algorithms reduce the number of candidate checks and improve computational efficiency while preserving the correctness of factorization. The obtained results demonstrate that residue number systems, combined with key-folding and offset-lattice techniques, provide a promising direction for accelerating the factorization of medium-sized integers and may serve as a basis for further research on high-performance modular factorization algorithms.

Keywords— integer factorization, Fermat factorization method, Residue Number System (RNS), Chinese Remainder Theorem (CRT), offset lattices, computational complexity, parallel computing, modular arithmetic.

Introduction

The problem of decomposing a natural number into prime factors is one of the key issues in number theory and has significant practical importance in the field of information security [1–3]. Of particular difficulty is the factorization of large integers that are the product of two large prime factors. Despite the relative simplicity of the multiplication operation for such numbers, the inverse problem of their decomposition is characterized by substantial computational complexity and requires considerable time and resources. It is precisely this asymmetry between direct and inverse operations that underlies the security of modern cryptographic algorithms, in particular RSA. Well-known classical factorization algorithms [4], including modern methods such as the general number field sieve, demonstrate high efficiency only for numbers of limited size. As

the length of the number increases, the complexity of factorization grows superpolynomially, making it practically infeasible for large values in real time. Despite significant progress in this field, the problem of efficient factorization of large integers remains open. The absence of algorithms capable of performing decomposition in polynomial time on classical computing systems necessitates the search for new approaches to its solution [5–6]. In this context, the study of alternative computational methods becomes relevant, particularly those based on the residue number system, which allows increasing the efficiency of arithmetic operations through their parallelization and can potentially be used to develop new factorization algorithms [7].

Fermat factorization algorithm

Fermat's factorization algorithm is based on searching for two perfect squares whose difference equals the number being factored [8]. The relation is given by:

$$N = a^2 - b^2 = (a - b)(a + b).$$

The iterative search begins with $a_0 = \lceil \sqrt{N} \rceil$. At subsequent iterations, the values $a = a_0, a_0 + 1, a_0 + 2, \dots$ are examined sequentially. The process continues until the value $b^2 = a^2 - N$ becomes a perfect square. Let $N = pq$, where $p \leq q$. Then the desired values are:

$$a = \frac{p + q}{2}, \quad b = \frac{q - p}{2}.$$

Thus, the number of iterations required by Fermat's method is determined by the difference

$$T_F \approx \frac{p + q}{2} - \lceil \sqrt{N} \rceil.$$

There exist modifications of the algorithm based on representing squares as sums of an arithmetic progression [9]. In this case, to obtain the next square, it is not necessary to perform multiplication; instead, it is sufficient to add the next element of the progression. The recurrent relation can be written as: $d_{k+1} = d_k + \Delta_k$, where $\Delta_k = 2a_k + 1$.

The initial value is defined as $d_0 - N = b^2$. At each iteration, the value is updated as $d_{k+1} = d_k + (2a_k + 1)$ until the condition $d_k = b^2$ is satisfied. This modification makes it possible to avoid operations with high computational complexity and provides a foundation for studying the properties of elements of an arithmetic progression within the residue number system (RNS) [10], which can be used to construct more efficient factorization algorithms.

Factorization algorithm based on key twisting in RNS

The factorization algorithm based on key folding in RNS is characterized by a number of significant advantages conditioned the peculiarities of number representation in RNS and the use of algebraic filtering of the space of possible solutions. Unlike classical methods, which involve direct operations on large integers in a positional numeral system, the proposed approach transfers computations into a multidimensional residue space, where each component corresponds to a separate modulus. This makes it possible to significantly reduce computational costs, since operations on small numbers are performed much faster than analogous operations on large integers. An additional important advantage is that the verification of the quadraticity condition is implemented as a check of residue membership in sets of quadratic residues, which is a computationally inexpensive operation and ensures effective elimination of unsuitable candidates at early stages of the algorithm.

Let the odd composite number to be factored be represented as $N = pq$, where $p, q \in \mathbb{N}, p \leq q$.

To construct factorization keys, consider a system of pairwise coprime moduli:

$$M = \{m_1, m_2, \dots, m_s\}, \gcd(m_i, m_j) = 1.$$

Then each value d_k is represented as a vector:

$$d_k = (d_k \bmod m_1, \dots, d_k \bmod m_s).$$

For each modulus m_i a set of quadratic residues is defined as:

$$Q_{m_i} = \{x^2 \bmod m_i \mid x \in \mathbb{Z}_{m_i}\}.$$

A necessary condition for an iteration to correspond to a potential perfect square is that all residues of the number at this iteration belong to the corresponding sets of quadratic residues:

$$d_k \bmod m_i \in Q_{m_i}, \forall_i$$

During the key construction process, for each modulus a set of admissible residues is formed:

$$K_{m_i} \subseteq Q_{m_i}.$$

Subsequently, sequential "folding" of keys is performed using CRT composition. The computations follow the relation:

$$M_k = \prod_{i=1}^k m_i.$$

The set of admissible values is defined as $K_{M_k} \subseteq \mathbb{Z}_{M_k}$. When adding a new modulus m_{k+1} the Chinese Remainder Theorem (CRT) is applied, yielding:

$$K_{M_k} = \{x \in \mathbb{Z}_{M_{k+1}} \mid x \equiv r \pmod{M_k}, x \equiv s \pmod{m_{k+1}}, r \in K_{M_k}, s \in K_{m_{k+1}}\}.$$

Graphically, the algorithm is illustrated in Fig. 1. A key aspect of the algorithm is the progressive narrowing

of the set of admissible values through the sequential folding of keys in accordance with the Chinese Remainder Theorem. This approach ensures step-by-step integration of information obtained for individual moduli into a unified consistent structure that describes admissible values in an extended modular space.

As a result, a significantly smaller subset of candidates is formed, satisfying all local conditions simultaneously, which reduces the overall number of required checks in the positional numeral system. Thus, the algorithm implements an efficient mechanism of multilevel filtering that combines local constraints for individual moduli into a global constraint over the entire search space.

The algorithm implements an iterative reduction of the search space:

$$\mathbb{Z}_{M_S} \rightarrow K_{M_S}, \text{ where } |K_{M_S}| \ll M_S.$$

Thus, the majority of iterations are discarded before performing the computationally expensive operation of number reconstruction.

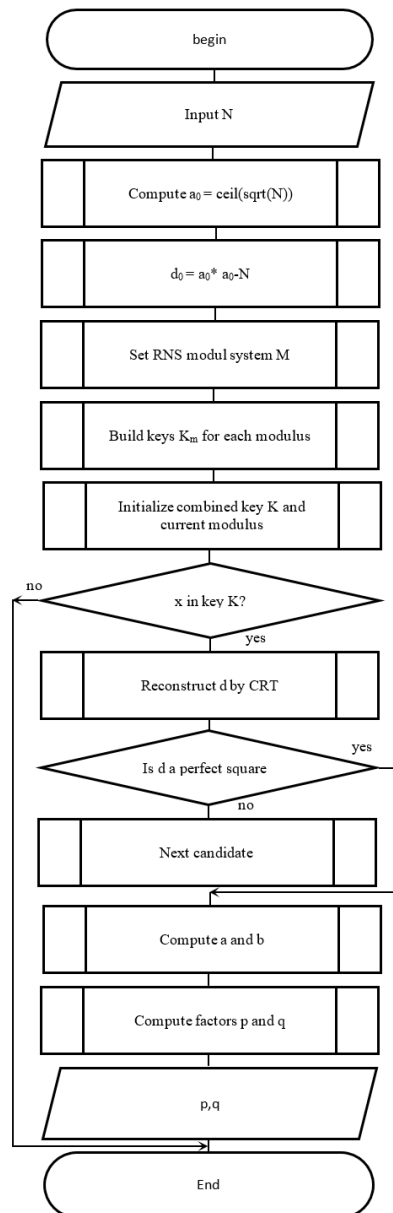


Fig.1. Scheme of the factorization algorithm based on key folding in RNS

Offset lattice-based factorization algorithm

Unlike the previous approach, the search is performed not in the space of values d , but in the space of indices k . The method is defined as follows:

$$d_k = (a_0 + k)^2 - N.$$

From this, we obtain the relation:

$$d_k = k^2 + 2a_0k + (a_0 + k)^2 - N.$$

Hence:

$$d_k \equiv f(k) \pmod{m},$$

where $f(k)$ is a quadratic polynomial over $\mathbb{Z}_m$.

For each modulus, a set of admissible indices is determined for which d_k is a quadratic residue

$$S_m = \{k \in \mathbb{Z}_m \mid f(k) \in Q_m\}.$$

This is the set of indices for which d_k is a quadratic residue. Next, a lattice is constructed that enables filtering of unnecessary iterations.

Let the set of selected moduli be denoted as.

$$M = \prod_{i=1}^s m_i.$$

For each M the following set is formed:

$$S_M = \{k \in \mathbb{Z}_m \mid k \bmod m_i \in S_{m_i}, \forall i\}.$$

This represents the set of admissible indices within the period M .

The selected moduli are divided into two groups. The first group reduces the number of iterations by skipping those whose residues are not quadratic residues, while the second group is used for additional filtering before checking the large integer for being a perfect square.

The first, main group defines the indices used for elimination and is described as:

$$S_{G1} \subset \mathbb{Z}_{M1}.$$

The admissible indices are then defined as:

$$k = t \cdot M_1 + r, r \in S_{G1}.$$

The second, control group $S_{G2} \subset \mathbb{Z}_{M2}$ is used as an additional criterion:

$$k \bmod M_2 \in S_{G2}.$$

For candidates satisfying $k \in S_{G1}$ and $k \in S_{G2}$ the value is computed as

$$d_k = (a_0 + k)^2 - N,$$

and checked against the condition

$$d_k = b^2.$$

The algorithm (Fig.2) implements a sieve in the space of indices and utilizes polynomial structures:

$$f(k) = k^2 + 2a_0k + c.$$

This enables a transition from brute-force enumeration of values to enumeration over residue classes. The conversion to the positional (decimal) number system is performed only at the final verification stage for a given index k . The advantages of the offset-based approach include the use of the polynomial structure of d_k and a reduction in the number of checks, since

$$k \in S_M \Rightarrow |S_M| \ll M.$$

Another advantage is the possibility of precomputing admissible residue classes and the absence of the need to store large sets of values.

The factorization algorithm based on offset lattices is characterized by increased efficiency due to shifting the search process from the space of values to the space of iteration indices. This approach makes it possible to exploit the quadratic structure of the function that describes the difference between the square of the current approximation and the number under consideration, and to construct a set of admissible indices for which the required quadraticity conditions are satisfied. As a result, the search is performed not over all possible values, but only over a restricted subset determined by the conditions of membership in sets of quadratic residues for the selected moduli. This significantly reduces the number of iterations at which the perfect square check is performed, which is the most computationally expensive operation.

An important advantage of the algorithm is the possibility of precomputing offset lattices that define admissible classes of indices within a given period. Such a lattice has a periodic structure, which allows organizing the search as transitions between admissible positions without the need to sequentially analyze each iteration. As a result, a substantial portion of unsuitable values is effectively skipped before performing any costly arithmetic operations in the positional number system.

An additional factor contributing to efficiency is the use of multilevel filtering. Dividing the set of moduli into several groups enables a cascade verification of candidates, where the primary offset lattice forms the main set of admissible indices, and subsequent lattices perform refinement checks. This organization significantly reduces the density of candidates at the final stage, minimizing the number of perfect square checks to the necessary minimum.

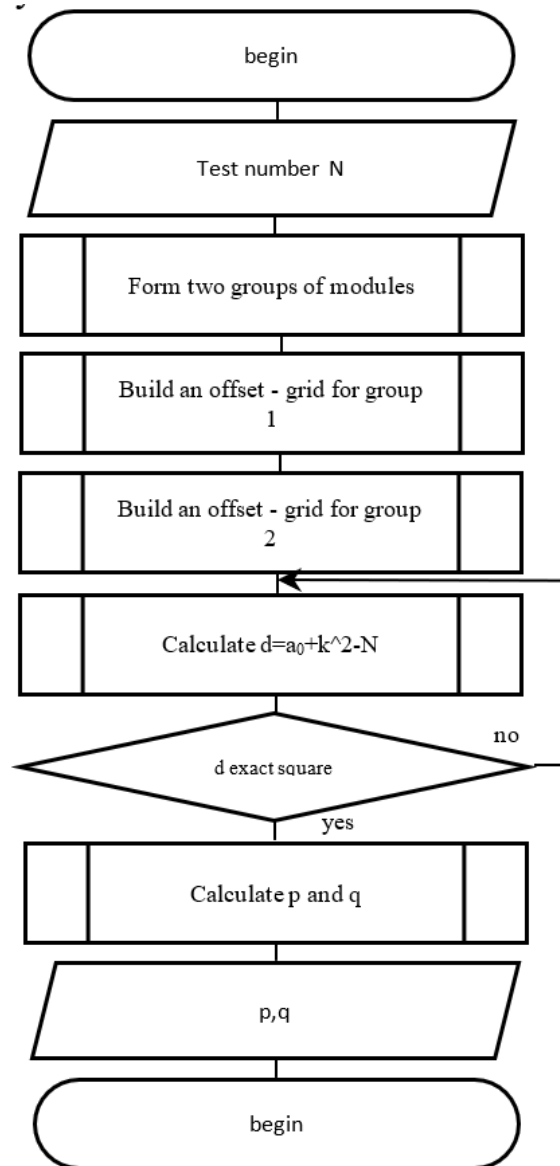


Fig. 2. Scheme of the factorization algorithm based on offset lattices

The algorithm is also well suited for parallel implementation. The construction of admissible offset sets for different moduli can be performed independently, and their combination allows for a hierarchical organization. Furthermore, candidate verification across different periodic intervals is mutually independent, enabling efficient distribution of computations across parallel threads or computing nodes. Altogether, this ensures good scalability of the algorithm and effective utilization of modern multicore and distributed computing systems.

Investigation of the Efficiency of the Proposed Factorization Algorithms

The computational complexity of Fermat's factorization method is assumed to be $O(k)$, where k is the number of iterations, which depends on the difference between the factors forming the product. For the factorization algorithm based on key folding in the RNS, the computational complexity can be expressed as $O(C_{key} + k\rho_{RNS})$. The computational complexity of the offset-lattice-based algorithm is given by $O(C_{key} + k\rho_1\rho_2)$. Figure 3 presents a comparison of the computational complexities of the classical Fermat method and the proposed algorithms.

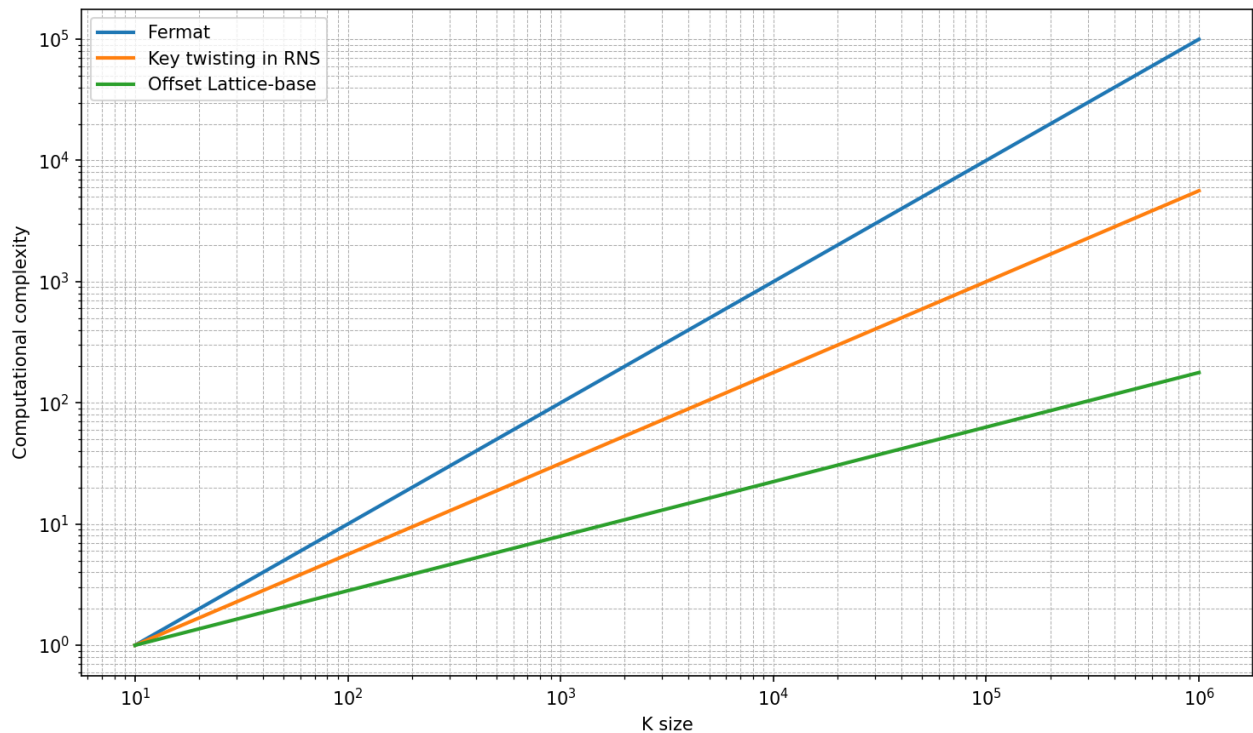


Fig. 3. Comparison of the computational complexities of the classical Fermat method and the proposed algorithms

For experimental evaluation, software was developed that implements both the classical Fermat method and the proposed algorithms. Fig. 4 presents the time characteristics for products up to 32 bits with varying numbers of iterations. The factorization algorithm based on key folding in the residue number system shows better performance due to well-chosen sets of moduli.

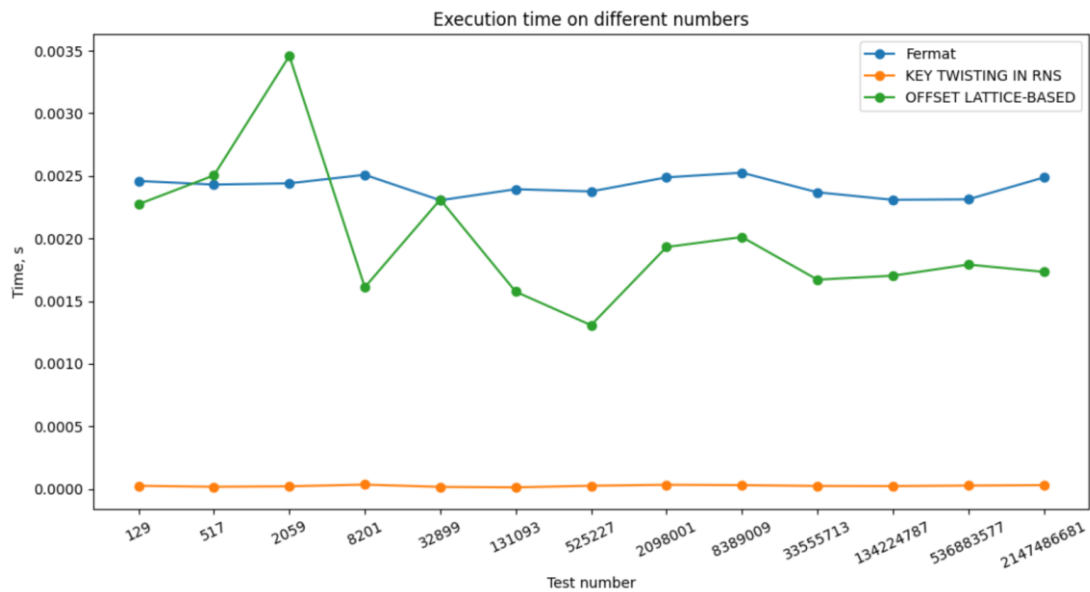


Fig. 4. Time characteristics of factorization algorithms with increasing k for numbers up to 32 bits

Additionally, experiments conducted for 100-bit numbers (Fig. 6) indicate an increase in the efficiency of the offset-lattice-based factorization algorithm, since the construction of large factorization keys required significant time.

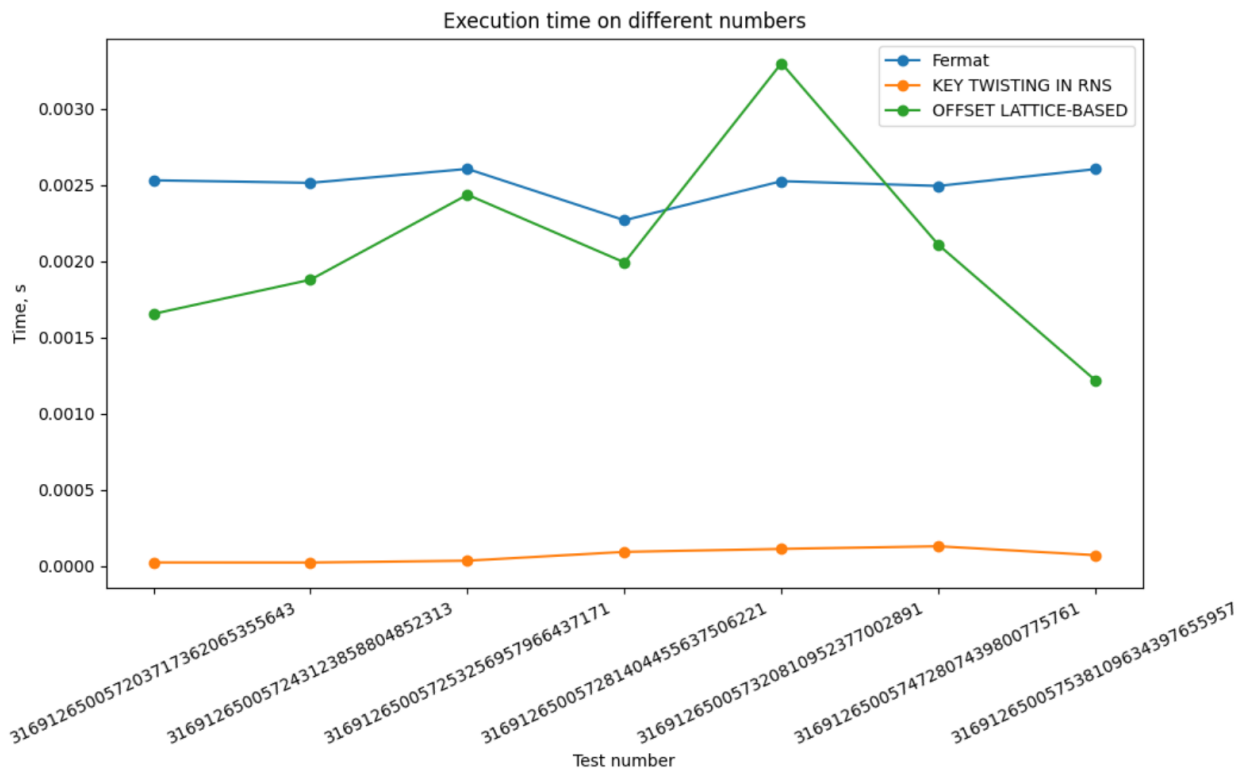


Fig. 5. Execution time of factorization algorithms for 100-bit numbers

The experimental results demonstrate varying effectiveness of the considered approaches for factorizing large integers, depending on the number of iterations required to reach a solution.

Conclusions

As a result of the conducted research, it has been established that the proposed factorization algorithms based on the residue number system and offset lattices offer significant advantages over the classical Fermat method.

Unlike the latter, which relies on sequential enumeration of values and largely depends on the difference between the factors of a number, the developed approaches implement efficient multilevel filtering of the search space, allowing most unsuitable candidates to be discarded at early stages of computation.

The algorithm based on key folding in the residue number system ensures the reduction of the set of admissible values through the consistency of residue representations, while the offset-based algorithm additionally exploits the structure of iteration indices, enabling checks to be performed only for a limited subset of positions. Together, this reduces the number of expensive perfect square verification operations and improves the overall computational efficiency.

Moreover, both approaches demonstrate natural suitability for parallel implementation, ensuring better scalability compared to the classical Fermat method.

ADDITIONAL INFORMATION

DECLARATION ON THE USE OF GENERATIVE ARTIFICIAL INTELLIGENCE TOOLS

During the preparation of this work, the author(s) used ChatGPT (OpenAI) for the purposes of translation, checking grammar and spelling, general formatting, and searching for relevant news for reference. After using this service, the author(s) reviewed and edited the content as needed and take full responsibility for the content of the publication.

REFERENCES

1. M. Karpinski, S. Ivasiev, I. Yakymenko, M. Kasianchuk, and T. Gancarczyk, "Advanced Method of Factorization of Multi-Bit Numbers Based on Fermat's Theorem in the System of Residual Classes," in Proc. 16th Int. Conf. Control, Automation and Systems (ICCAS), Gyeongju, South Korea, 2016, pp. 1484–1486, <https://doi.org/10.1109/ICCAS.2016.7832500>
2. V. Zadiraka, Y. Nykolaychuk, and S. Ivasiev, "The Theory of Factorization Multidigit Numbers," in Proc. Int. Conf. The Experience of Designing and Application of CAD Systems in

Microelectronics (CADSM), Lviv, Ukraine, 2015, pp. 221–225,

<https://doi.org/10.1109/CADSM.2015.7230841>

3. J. P. Buhler, H. W. Lenstra, and C. Pomerance, "Factoring Integers with the Number Field Sieve," in The Development of the Number Field Sieve, A. K. Lenstra and H. W. Lenstra, Eds. Berlin, Germany: Springer-Verlag, 1993, pp. 50–94, <https://doi.org/10.1007/BFb0091539>

4. K. Lenstra, H. W. Lenstra, M. S. Manasse, and J. M. Pollard, "The Factorization of the Ninth Fermat Number," Math. Comput., vol. 61, no. 203, pp. 319–349, Jul. 1993, <https://doi.org/10.1090/S0025-5718-1993-1182953-4>

5. J. D. Dixon, "Asymptotically Fast Factorization of Integers," Math. Comput., vol. 36, no. 153, pp. 255–260, Jan. 1981, <https://doi.org/10.1090/S0025-5718-1981-0606519-9>

6. C. P. Schnorr, "Refined Analysis and Improvements on Some Factoring Algorithms," J. Algorithms, vol. 12, no. 1, pp. 101–127, Mar. 1991, [https://doi.org/10.1016/0196-6774\(91\)90030-J](https://doi.org/10.1016/0196-6774(91)90030-J)

7. S. Shatnawi, "Fermat Factorization Exact Bound and Improvements," J. King Saud Univ. Comput. Inf. Sci., vol. 36, no. 1, Art. no. 101745, 2024, <https://doi.org/10.1016/j.jksuci.2023.101745>

8. M. Seysen, "A Probabilistic Factorization Algorithm with Quadratic Forms," Math. Comput., vol. 48, no. 178, pp. 757–780, Apr. 1987, <https://doi.org/10.1090/S0025-5718-1987-0866107-7>

9. V. Yatskiv, S. Kulyna, P. Bykovyy, T. Maksymyuk and A. Sachenko, "Method of Reliable Data Storage Based on Redundant Residue Number System," 2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Dortmund, Germany, 2020, pp. 1–4, <https://doi.org/10.1109/IDAACS-SWS50031.2020.9297052>

10. V. Yatskiv, S. Kulyna, P. Bykovyy, T. Maksymyuk and A. Sachenko, "Method of Reliable Data Storage Based on Redundant Residue Number System," 2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Dortmund, Germany, 2020, pp. 1–4, <https://doi.org/10.1109/IDAACS-SWS50031.2020.92970>

Степан ІВАСЬЄВ

Західноукраїнський національний університет

МЕТОД ФАКТОРИЗАЦІЇ БАГАТОРОЗРЯДНИХ ЧИСЕЛ НА ОСНОВІ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ

У роботі досліджено задачу факторизації багаторозрядних цілих чисел, яка залишається однією з фундаментальних проблем сучасної теорії чисел та криптографії з відкритим ключем. Проведено аналіз класичного методу факторизації Ферма та визначено його основні обчислювальні обмеження, пов'язані з багаторазовим обчисленням квадратів і послідовним перебором кандидатів. Для усунення цих недоліків запропоновано модифікацію методу Ферма, що базується на рекурентному обчисленні різниці квадратів. Запропонований підхід виключає необхідність багаторазового піднесення до квадрату, замінюючи його послідовністю інкрементних арифметичних операцій, що створює математичне підґрунтя для використання системи залишкових класів.

На основі запропонованої модифікації розроблено два нових алгоритми факторизації. Перший алгоритм реалізує механізм скручування ключів у системі залишкових класів, у якому для окремих модулів формуються множини допустимих квадратичних залишків, що надалі послідовно об'єднуються відповідно до Китайської теореми про залишки. Такий підхід забезпечує багаторівневу алгебраїчну фільтрацію простору можливих значень та істотно зменшує кількість кандидатів, які потребують перевірки у позиційній системі числення. Другий алгоритм ґрунтується на використанні offset-решіток, у яких процес пошуку переноситься із простору значень у простір індексів ітерацій. Побудова періодичних множин допустимих offset-ів для груп модулів дозволяє виконувати перевірку лише тих ітерацій, що одночасно задовольняють усі модульні обмеження, значно скорочуючи кількість дорогих операцій перевірки точного квадрата.

Для запропонованих алгоритмів виконано теоретичний аналіз обчислювальної складності та проведено їх порівняння з класичним методом Ферма. Показано, що обидва алгоритми забезпечують зменшення ефективного простору пошуку за рахунок модульної фільтрації, тоді як алгоритм на основі offset-решіток додатково скорочує кількість досліджуваних ітерацій шляхом використання періодичної структури допустимих offset-ів. Незалежність обчислень для окремих модулів забезпечує природну можливість паралельної реалізації обох алгоритмів, що робить їх придатними для використання на багатоядерних і розподілених обчислювальних системах.

Проведені експериментальні дослідження на числах різної розрядності підтвердили ефективність запропонованих підходів. Порівняно з класичним методом Ферма розроблені алгоритми забезпечують істотне скорочення кількості кандидатів, що перевіряються, та підвищення швидкодії процесу факторизації без втрати коректності отриманих результатів. Отримані результати свідчать, що використання системи залишкових класів у поєднанні з механізмами скручування ключів та offset-решіток є перспективним напрямом прискорення факторизації багаторозрядних чисел і може слугувати основою для подальшого розвитку високопродуктивних алгоритмів модульної факторизації.

Ключові слова: факторизація цілих чисел, метод Ферма, система залишкових класів, Китайська теорема про залишки, offset-решітки, обчислювальна складність, паралельні обчислення, модульна арифметика.