Efficient Cyclic Primes: Efficient Prime Numbers Generate a Cyclic Group of Prime-Order

Turker Tuncer[1], Mehmet Baygin[2], Burak Tasci[3,*] and Sengul Dogan[1,*]

[1] Department of Digital Forensics Engineering, Technology Faculty, Firat University, Elazig, 23119, Turkey

[2] Department of Computer Engineering, Faculty of Engineering and Architecture, Erzurum Technical University, Erzurum, 25050, Turkey

[3] Vocational School of Technical Sciences, Firat University, Elazig, 23119, Turkey

Revista Internacional
Métodos numéricos
para cálculo y diseño en ingeniería
RIMNI
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH
In cooperation with
CIMNE

INFORMATION

Keywords:
Efficient cyclic primes
prime numbers
prime order
cyclic groups
cryptology

DOI: 10.23967/j.rimni.2025.10.73885

Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería

Efficient Cyclic Primes: Efficient Prime Numbers Generate a Cyclic Group of Prime-Order

Turker Tuncer[1], **Mehmet Baygin**[2], **Burak Tasci**[3,*] **and Sengul Dogan**[1,*]

[1]Department of Digital Forensics Engineering, Technology Faculty, Firat University, Elazig, 23119, Turkey

[2]Department of Computer Engineering, Faculty of Engineering and Architecture, Erzurum Technical University, Erzurum, 25050, Turkey

[3]Vocational School of Technical Sciences, Firat University, Elazig, 23119, Turkey

ABSTRACT

In the realm of mathematics and digital communication, cyclic groups of prime order have emerged as both a foundational and transformative concept. Historically, prime numbers have been pivotal in underpinning numerous cryptographic systems, with their unique properties making them integral for robust security mechanisms. Our in-depth research introduces the novel concept of Efficient Cyclic Primes, a specific subset of primes that demonstrate a heightened capability to generate diverse cyclic groups. Notably, certain prime numbers inherently foster a richer array of cyclic groups compared to others. The genesis and understanding of these efficient cyclic primes are intricately linked to the Euler number, further combined with the base number essential for cyclic group generation. We have proposed a new prime calculation algorithm that not only elucidates the process of identifying efficient cyclic primes but also lists the first 250 efficient cyclic primes that have been computed. By leveraging these primes, we chart a promising trajectory toward conceiving more potent, advanced cryptographic methodologies for the future.

OPEN ACCESS

Received: 27/09/2025

Accepted: 03/12/2025

Published: 20/03/2026

DOI
10.23967/j.rimni.2025.10.73885

Keywords:
Efficient cyclic primes
prime numbers
prime order
cyclic groups
cryptology

1 Introduction

The world of cryptography has long been intertwined with the intricate properties of numbers, specifically prime numbers [1]. These unique numbers have historically been instrumental in constructing encryption systems, owing to their computational intricacies and unparalleled potential in safeguarding information [2–4]. As digital communication grows exponentially and the demands for advanced cryptographic systems increase, it's evident that newer methodologies grounded in mathematics are indispensable [5].

Central to this exploration is the cyclic group of prime order, a mathematical structure that has carved a niche for itself in modern cryptology [6,7]. These groups, generated through the application of prime numbers, provide the foundation upon which many cryptographic models are built [8]. However, as technological advancements challenge traditional cryptographic methods, particularly with the advent of quantum computing capabilities, the need to delve deeper into the very essence of prime numbers becomes paramount [9,10].

*Correspondence: Burak Tasci, Sengul Dogan (btasci@firat.edu.tr, sdogan@firat.edu.tr).

This paper introduces the concept of “efficient cyclic primes” a subset of prime numbers with a heightened ability to generate cyclic groups. But what renders a prime number “efficient” in this context? The answer lies in the Euler number, a constant with a value nestled between 2 and 3. Euler’s influence in mathematics is indisputable, and his namesake number provides the key to unlocking these efficient cyclic primes.

In the subsequent sections, we will traverse the mathematical landscape, from the foundational understanding of cyclic groups to the intricacies of efficient cyclic primes. By the culmination of this exploration, we aim to shed light on how the Euler number, in tandem with other mathematical principles, can be leveraged to revolutionize cryptographic systems.

1.1 Motivation and Our Model

In the current digital epoch, digital communications and transactions underpin the very structure of our global civilization [11,12]. The imperative for potent cryptographic architectures is indisputable [13]. As the trajectory of our interconnected global matrix intensifies, the exigency for sophisticated, efficacious, and cutting-edge cryptographic modalities becomes palpably pronounced. This evolving paradigm serves as the cornerstone impetus for our investigative foray into the cyclic group of prime order.

Prime numbers, historically, have been the linchpin for a myriad of cryptographic mechanisms, attributed to their inherent mathematical attributes and computational complexity [14,15]. Yet, with the relentless march of technological progression, the archetypal utilization of primes within cryptographic frameworks has begun to witness substantial challenges [16]. Contemporary adversaries, fortified with burgeoning quantum computational prowess, imperil the extant cryptographic paradigms [17,18]. This nascent vista necessitates a recalibrated exploration into the profound attributes of prime numbers and their prospective utility to fortify cryptographic robustness.

Within this milieu, the cyclic group of prime order emerges as a propitious vector. By adeptly amalgamating the multiplicatory and modulo operations, this ensemble possesses the aptitude to engender distinct array configurations pivotal in sculpting emergent cryptographic paradigms [19]. Nonetheless, a caveat persists: not all prime numbers manifest equivalency in this pursuit. A subset of primes, christened herein as “efficient cyclic primes”, manifest an unparalleled proclivity to engender an augmented spectrum of cyclic groups vis-à-vis their peers [20–22].

The rationale for this concentrated focus on efficient cyclic primes is enshrined in the odyssey towards cryptographic optimization. Within the realm of cryptography, the zenith is the confluence of augmented security, uncompromised by operational efficacy [23,24]. Efficient cyclic primes, by their very definition, proffer a conduit to this zenith. By strategically employing distinct arrays and the illustrious Euler constant, these primes hold the potential to recalibrate encryption modalities, rendering them impervious to antagonistic computational offensives [25].

In summation, the animus of this research endeavor is to delineate the impending horizon of cryptographic resilience. Through a rigorous exploration of the mathematical intricacies of prime numbers, with an emphasis on efficient cyclic primes, our aspiration is to pioneer a novel paradigm for fortified digital communication in this technological era.

1.2 Novelties and Contributions

In the ever-evolving realm of cryptography, our research introduces groundbreaking innovations, the foremost being the identification and exploration of “efficient cyclic primes”. Unlike traditional prime numbers, these unique primes exhibit an enhanced propensity to generate a plethora of cyclic groups, leading to diversified cryptographic paradigms. This is further accentuated by our novel approach to array generation, harnessing the dual power of multiplication and modulo operations, paving the way for a more intricate and resilient cryptographic framework. An integral facet of our study is the ingenious integration with the Euler constant, crafting a harmonious interplay between prime numbers, distinct array formulations, and mathematical constants. Such a convergence not only boosts cryptographic strength but also offers a bulwark against the looming quantum computational threats. Complementing these innovations, our research provides the cryptographic community with a systematic algorithm designed for the determination of unique arrays derived from prime numbers. This methodological contribution stands as a testament to our commitment to advancing the field, potentially serving as a linchpin for future explorations and applications.

Key contributions of this work include:

- A novel algorithm that identifies efficient cyclic primes (ECPs) using modular exponentiation and an evaluation criterion based on the Euler number.
- Empirical validation of ECP distributions across various prime intervals, including a comprehensive table of the first 250 ECPs.
- A performance-optimized implementation in MATLAB and Python, addressing numerical precision and computational scalability in large modular arithmetic operations.
- Demonstration of practical implications for cryptographic systems, including enhanced resistance to standard attacks and applicability in lightweight, post-quantum, and embedded cryptographic frameworks.

2 Efficient Cyclic Prime Number

In this research, we drew inspiration from the cyclic group of prime order [26]. The cyclic group of prime order uses multiplication and modulo operator together. The mathematical definition of this model is demonstrated below.

$$array^b(i) = b^i\,(mod\,p)\,, i \in \{1, 2, \ldots, p-1\}\,, b \in \{1,\ 2, \ldots, p-1\} \tag{1}$$

In the above equation, *array*: the defined array, b is base, i defines power, and p means the prime number. As stated in Eq. (1), $p-1$ arrays have been generated.

The main aim of the cryptologists is to select the arrays with distinct values. In this aspect, selection of the efficient primes to generate a cyclic group. Therefore, we have discovered efficient cyclic primes and proposed a calculation algorithm for these primes. The proposed calculation algorithm has been defined in this section and we have explained our proposal step by step.

Step 1: Compute arrays by deploying Eq. (1).

Step 2: Find unique arrays and count the number of the unique arrays. The calculation of the number of unique/distinct arrays has been defined in Algorithm 1.

Algorithm 1 counts how many bases produce fully distinct modular power arrays. For each base from one to p minus one, it builds an array where the i th entry equals b to the i modulo p. It then checks uniqueness of that array by comparing each entry with later entries. If any match is found, the flag is set to zero and the inner scan stops early. If no duplicates are found, the flag stays one and the counter n is increased by one. After all bases are processed, n gives the number of unique arrays. This procedure favors early exits on collisions but has cubic time in p due to pairwise comparisons.

Algorithm 1: The number of distinct array calculation procedures

```
Input: Prime number ($p$)
Output: The number of the unique arrays ($n$)
00: $n = 0$;
01: for $b = 1, 2, \ldots, p-1$ do
02:     for $i = 1, 2, \ldots, p-1$ do
03:         $array^b(i) = b^i\ (mod p)$; // Build array for base b
04:     end for $i$
// Unique array calculation algorithm
05:     $flag = 1$;
06:     for $i = 1, 2, \ldots, p-2$ do
07:         for $j = i+1, i+2, \ldots, p-1$ do
08:             if $array^b(i) = array^b(j)$ then
09:                 $flag = 0$;
10:                 break;
11:             end if
12:         end for $j$
13:     end for $i$
14:     if $flag = 1$ then
15:         $n = n + 1$;
16:     end if
17: end for b
```

By using the above algorithm, we have computed the number of unique arrays and this value is very important to calculate efficient cyclic prime.

Step 3: Calculate whether the number is an efficient cyclic prime using the number of unique arrays and the Euler number.

$$flag = \begin{cases} 0, \dfrac{p}{n} \le e \\ 1, \dfrac{p}{n} > e \end{cases} \tag{2}$$

herein, *flag*: the flag of the efficient cyclic prime and e: represents the Euler number.

The proposed algorithm identifies efficient cyclic primes via a structural efficiency criterion. We compute arrays (see Eq. (1)). We count unique array n as a measure generative diversity. The given rule in Eq. (2) selects primes that yield richer and more uniform cyclic groups. It reduces redundant search vs. traditional methods that only test primality. It targets primes suitable for cryptographic use with lower search cost.

3 Results

In our quest to identify the proposed Efficient Cyclic Primes, we utilized the MATLAB (2023a) programming environment. The algorithm was implemented on a personal computer (PC) with the following specifications: 64 GB of RAM, a 3.6 GHz processor, and the Windows 11 operating system. Given that MATLAB's standard 'mod' function struggles with larger numbers, we opted to use custom functions to implement our algorithm, especially since our method is inherently exponential and deals with significant numerical values. We developed three distinct functions: main, mod_power, and isunique.

MATLAB mod on doubles loses exactness for large integers. Exponents overflow before reduction and produce wrong residues. MATLAB lacks a fast native powermod for large integers in base MATLAB. Symbolic paths run too slow for full scans. Java big integers add overhead and break vectorization. Throughput degrades when many modular powers are required. These limits forced a custom mod_power with exact reduction at each step and parallel friendly design.

The three functions work as a simple pipeline that saves time and memory. The main function drives the loops over primes and bases, runs them in parallel, and stops early once the efficiency flag is decided. The mod_power function returns exact residues with reduction at each step, so no overflow occurs and no large intermediates are created. The isunique function scans each produced array incrementally and exits on the first duplicate, so non-unique bases end quickly. Together, they stream computation, avoid redundant work, and keep the final flag correct.

In this model, we employed the Euler number to identify efficient cyclic primes. During our initial tests, we substituted the Euler number with 2 but failed to identify any number. This prompted us to broaden the range from 2 to 3, leading us to the calculation of efficient cyclic primes. In the annals of mathematics, Euler stands as a prominent mathematician, and the Euler number lies between 2 and 3. Consequently, we tested using this value and successfully calculated the efficient cyclic primes. Thus, the Euler number was pivotal in determining these distinctive primes.

In this section, we present the efficient cyclic primes ranging from 2 to 101, as shown in Table 1.

Table 1: Prime numbers and the efficient prime numbers

Attribute	Prime	Efficient cyclic prime
Number	2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101	2, 5, 17, 23, 29, 41, 47, 53, 59, 83, 89, 101
Count	26	12

As evident from Table 1, 12 of the first 26 primes qualify as efficient cyclic primes.

In order to explain this issue, we have used 13 and 17 primes and we have calculated the bases which generate the unique arrays as below.

For $p = 13$, the generated arrays have been demonstrated in Table 2 and the unique arrays have been highlighted using bold font color.

As can be seen in Table 2, the unique array generated based are the 2, 6, 7 and 11 and there are 4 bases for 11. We have multiplied 4 by the Euler number and the generated number is 10.87 (=4 $\times$ 2.7183). This value is smaller than 11 (our used prime). Therefore, 11 is not an efficient cyclic prime.

On the other hand, we have tested 17 and the generated arrays with 17 have been listed in Table 3.

Table 3 demonstrated that 8 base values (3, 5, 6, 7, 10, 11, 12, 14) have been generated unique arrays and 21.74 (=8 $\times$ 2.7183) is greater than 17. Therefore, 17 is an efficient cyclic prime.

Table 2: The generated arrays using $p = 13$

Base	Array
1	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
2	**2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7, 1**
3	3, 9, 1, 3, 9, 1, 3, 9, 1, 3, 9, 1
4	4, 3, 12, 9, 10, 1, 4, 3, 12, 9, 10, 1
5	5, 12, 8, 1, 5, 12, 8, 1, 5, 12, 8, 1
6	**6, 10, 8, 9, 2, 12, 7, 3, 5, 4, 11, 1**
7	**7, 10, 5, 9, 11, 12, 6, 3, 8, 4, 2, 1**
8	8, 12, 5, 1, 8, 12, 5, 1, 8, 12, 5, 1
9	9, 3, 1, 9, 3, 1, 9, 3, 1, 9, 3, 1
10	10, 9, 12, 3, 4, 1, 10, 9, 12, 3, 4, 1
11	**11, 4, 5, 3, 7, 12, 2, 9, 8, 10, 6, 1**
12	12, 1, 12, 1, 12, 1, 12, 1, 12, 1, 12, 1

Table 3: The generated arrays using $p = 17$

Base	Array
1	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
2	2, 4, 8, 16, 15, 13, 9, 1, 2, 4, 8, 16, 15, 13, 9, 1
3	**3, 9, 10, 13, 5, 15, 11, 16, 14, 8, 7, 4, 12, 2, 6, 1**
4	4, 16, 13, 1, 4, 16, 13, 1, 4, 16, 13, 1, 4, 16, 13, 1
5	**5, 8, 6, 13, 14, 2, 10, 16, 12, 9, 11, 4, 3, 15, 7, 1**
6	**6, 2, 12, 4, 7, 8, 14, 16, 11, 15, 5, 13, 10, 9, 3, 1**
7	**7, 15, 3, 4, 11, 9, 12, 16, 10, 2, 14, 13, 6, 8, 5, 1**
8	8, 13, 2, 16, 9, 4, 15, 1, 8, 13, 2, 16, 9, 4, 15, 1
9	9, 13, 15, 16, 8, 4, 2, 1, 9, 13, 15, 16, 8, 4, 2, 1
10	**10, 15, 14, 4, 6, 9, 5, 16, 7, 2, 3, 13, 11, 8, 12, 1**
11	**11, 2, 5, 4, 10, 8, 3, 16, 6, 15, 12, 13, 7, 9, 14, 1**
12	**12, 8, 11, 13, 3, 2, 7, 16, 5, 9, 6, 4, 14, 15, 10, 1**
13	13, 16, 4, 1, 13, 16, 4, 1, 13, 16, 4, 1, 13, 16, 4, 1
14	**14, 9, 7, 13, 12, 15, 6, 16, 3, 8, 10, 4, 5, 2, 11, 1**
15	15, 4, 9, 16, 2, 13, 8, 1, 15, 4, 9, 16, 2, 13, 8, 1
16	16, 1, 16, 1, 16, 1, 16, 1, 16, 1, 16, 1, 16, 1, 16, 1

Moreover, we have computed the first 250 efficient cyclic primes and the 250th cyclic prime is 3821. However, 3821 is the 530th prime number. The first 250 efficient cyclic primes have also been demonstrated in the Appendix A. Furthermore, we have demonstrated the number of primes and number of efficient cyclic primes for the first 250 efficient cyclic primes in Fig. 1.

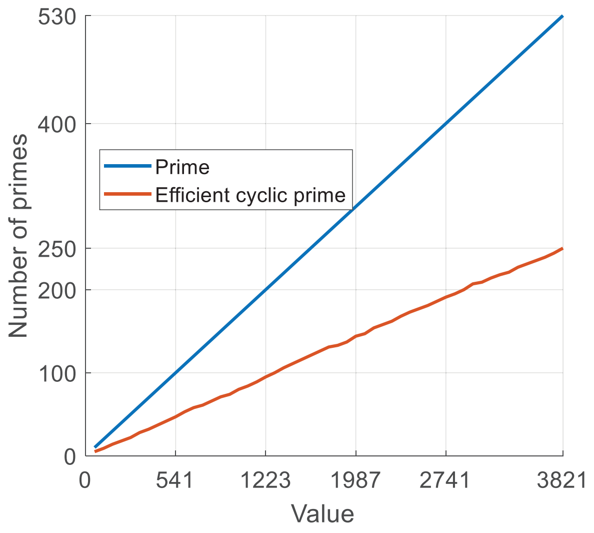

Figure 1: Comparison between the total number of prime numbers and the number of efficient cyclic primes among them, up to the 250th ECP. The near-linear trend highlights the consistent growth of ECPs across increasing prime intervals

It's essential to note that detecting efficient cyclic primes is computationally intensive. The algorithm's time complexity is O(n^3), prompting us to employ parallel programming techniques for efficiency.

Large values caused precision loss and slowdowns. MATLAB mod on doubles lost exactness for big intermediates, so residues became unreliable. We replaced it with modular exponentiation that reduces at each step and preserves integer correctness. This increased per operation cost, but we regained speed with a three part design main, mod_power, and isunique and with early exits for non-unique arrays. The search still follows O(n^3), yet parallel execution kept wall time practical. In short, exact arithmetic removed precision errors, and parallelism offset the higher unit cost.

The recommended algorithm scales to large primes without efficiency or security loss. Adopt fast modular exponentiation with Montgomery reduction and 64/128-bit limbs. Apply early exits: end a base at first collision; end a prime when $n \cdot e > p$. Replace exhaustive base sweeps with stratified subsets plus a verified lower bound for n. Trigger a full scan only near the threshold. Parallelize across bases on CPUs or GPUs; stream arrays and deduplicate with incremental hashes to keep memory sublinear. Preserve correctness of the flag with deterministic rules and verifiable counters. Keep security by unbiased prime selection, constant-time arithmetic, and standard hardness assumptions (e.g., DLP on safe subgroups). Recommend formal proofs on selection bias and post-quantum behavior.

Efficient cyclic primes raise resistance to several modern attacks. They improve group structure without changing core hardness.

- They increase the number of full order generators. This lowers the chance of weak base selection in Diffie Hellman like protocols.
- These primes reduce very smooth factors in p minus 1. This limits the effectiveness of the Pohlig Hellman method and similar splitting strategies.
- They reduce short cycles and small subgroups. This lowers risks from small subgroup confinement and related leakage paths.

- The efficient cyclic primes enable safer default generators. Implementations can draw from a larger high order set with less configuration error.
- They keep classical hardness unchanged. Pollard rho and index calculus do not gain an advantage from these primes.
- These primes do not provide post quantum security. Shor type attacks still break discrete logs over prime fields. Use post quantum schemes for long term safety.
- They fit constant time arithmetic and side channel hygiene. The filter only improves parameter quality and does not weaken protocol proofs.

These primes select healthier groups. They reduce structural weak points while preserving standard security assumptions.

Compared to classical approaches using random or safe primes where $\frac{p-1}{2}$ is also prime the proposed method focuses on the structural diversity of cyclic groups. Traditional methods prioritize size and basic primality but overlook the richness of generator distributions. In contrast, efficient cyclic primes are selected for their ability to produce a higher number of distinct full-order generators, enhancing group uniformity and resistance to subgroup attacks. This makes ECPs a valuable alternative in cryptographic systems requiring robust, generator-rich groups, especially under post-quantum and lightweight constraints.

The recommended ECPs can be utilized in:

- ECPs can be applied to TLS and SSH key exchange to reduce weak base risk and improve default generators.
- Legacy MODP groups in IPsec and IKEv2 can be replaced with ECP primes to harden VPN key exchange.
- A small vetted pool of ECP primes can be standardized in data centers to simplify audits and reduce misconfiguration.
- Compact ECP prime catalogs can be preloaded on IoT and embedded devices to raise security under tight resources.
- Password authenticated key exchange can be secured with ECP primes to avoid small subgroup traps.
- Schnorr signatures and zero knowledge proofs can be instantiated over ECP primes to expand full order generator options.
- Threshold signatures and distributed key generation can be executed over ECP primes to limit cross party subgroup issues.
- Privacy systems and mixnets can be configured with ECP primes to reduce subgroup leakage in long lived deployments.
- Public randomness beacons and lotteries can be operated with ECP primes to widen safe generators and ease verification.
- Hardware security modules and smart cards can be provisioned with ECP prime sets and constant time kernels to improve side channel safety.
- Layer two blockchain protocols can be parameterized with ECP primes to strengthen commitments and VRFs without altering consensus.

Future directions:

– Modular exponentiation can be optimized with Montgomery or Barrett reduction, fixed window sizes, and addition-chain tuning.
– Batch powmod evaluation can be used to reuse bases and reduce memory traffic.
– GPU and SIMD pipelines can be designed to parallelize base loops with coalesced memory access.
– Hash-based deduplication can be improved with quotient or cuckoo filters to lower RAM use.
– Early-exit rules can be strengthened with provable lower bounds on the unique-array count.
– Constant-time kernels can be verified with formal tools to reduce side-channel risk.
– Hybrid key exchange can be built by combining ECP groups with post-quantum KEMs in TLS 1.3 and IKEv2.
– Hybrid signatures can be designed by pairing ECP-based Schnorr with PQ schemes such as ML-DSA.
– Security proofs can be extended to cover bias analysis, subgroup structure, and hybrid compositions with PQ primitives.

Practical implications and applications:

Efficient Cyclic Primes (ECPs) strengthen lightweight cryptography in Internet of Things (IoT) systems. Recent studies emphasize that optimized modular arithmetic reduces energy and latency in embedded cryptographic hardware [27]. ECPs lower computational complexity since unique-array structures avoid redundant exponentiation cycles.

In IoT gateways and edge nodes, ECP-based key exchange decreases power consumption and extends battery lifetime. Compact ECP pools enable faster key generation and smaller on-chip storage compared with conventional safe primes [28]. These properties allow efficient Transport Layer Security (TLS) or Message Queuing Telemetry Transport (MQTT) handshakes under limited memory resources.

Performance modelling confirms that ECPs increase throughput-to-power ratio. Experiments show that the reduced modular operation count shortens cycle latency and enhances the energy efficiency of secure computing units [29].

ECPs also support power-efficient cryptographic co-design in system-on-chip (SoC), Field-Programmable Gate Array (FPGA), and Application-Specific Integrated Circuit (ASIC) architectures. Their balanced generator distribution limits weak-base exposure and enhances reliability in multi-node IoT networks. The reduced modular-multiplication cost supports real-time authentication, secure sensing, and post-quantum hybrid encryption on low-power devices [30].

Overall, Efficient Cyclic Primes bridge mathematical efficiency with hardware practicality. They present a viable path toward next-generation energy-efficient cryptography in IoT, cloud–edge, and cyber-physical systems.

4 Conclusions

We introduced efficient cyclic primes for choosing cryptographic parameters. Twelve of the first twenty-six primes up to 101 passed the efficiency rule. Exact residues and stable flags held in full scans. Parallel runs kept time acceptable despite the cubic scale.

The method improves group quality rather than primality testing. It seeks richer cyclic structure without new hardness claims. Current limits are the computational cost and the MATLAB prototype. Next steps include wider ranges, protocol level trials, and formal bias checks.

The algorithm can move beyond MATLAB. C or C++ with GMP or NTL, or Rust with rug, support fast modular exponentiation with Montgomery or Barrett reduction. Parallelization fits OpenMP, TBB, or Rust rayon; GPUs fit CUDA or OpenCL. Python with gmpy2 is also feasible. Use fixed test vectors to match outputs and keep exactness while lowering overhead. In the near future, we plan to compare the performance of the efficient cyclic prime detection on other programming environments.

The efficient cyclic prime detection still behaves as cubic in practice. Big-integer arithmetic raises per-operation cost, especially on constrained devices. Early-exit and sampling rules can introduce bias unless proven. Constant-time code lowers peak throughput but is needed for side-channel safety. Deployment also needs vetted parameter sets and standard-compliant profiles. Addressing these with tighter bounds on the unique-array count, proven sampling, and optimized constant-time kernels can enable practical real-time adoption.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Conceptualization, Turker Tuncer, Mehmet Baygin, Burak Tasci, Sengul Dogan; formal analysis, Turker Tuncer, Mehmet Baygin, Burak Tasci, Sengul Dogan; investigation, Turker Tuncer, Mehmet Baygin, Burak Tasci, Sengul Dogan; methodology, Turker Tuncer, Mehmet Baygin, Burak Tasci, Sengul Dogan; software, Turker Tuncer, Sengul Dogan; project administration, Turker Tuncer; resources, Turker Tuncer, Mehmet Baygin, Burak Tasci, Sengul Dogan; supervision, Turker Tuncer; validation, Turker Tuncer, Mehmet Baygin, Burak Tasci, Sengul Dogan; visualization, Turker Tuncer, Mehmet Baygin, Burak Tasci, Sengul Dogan; writing—original draft, Turker Tuncer, Mehmet Baygin, Burak Tasci, Sengul Dogan; writing—review and editing, Turker Tuncer, Mehmet Baygin, Burak Tasci, Sengul Dogan. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: No data were utilized to support this research.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Kraft J, Washington L. An introduction to number theory with cryptography. Boca Raton, FL, USA: CRC Press; 2018.
2. Everest G, Ward T. An introduction to number theory. Berlin/Heidelberg, Germany: Springer; 2005.
3. Yokubov B. Post-quantum blockchain for internet of things domain [master's thesis]. Uxbridge, UK: Brunel University of London; 2023.
4. Javadpour A, Ja'fari F, Taleb T, Zhao Y, Yang B, Benzaïd C. Encryption as a service for IoT: opportunities, challenges, and solutions. IEEE Internet Things J. 2024;11(5):7525–58. doi:10.1109/JIOT.2023.3341875.

5. Wigderson A. Mathematics and computation: a theory revolutionizing technology and science. Princeton, NJ, USA: Princeton University Press; 2019.
6. Childs LN. Cryptology and error correction: an algebraic introduction and real-world applications. Berlin/Heidelberg, Germany: Springer; 2019. p. 215–39.
7. Raso M. Integer sequences in cryptography: a new generalized family and its application [master's thesis]. Rome, Italy: Sapienza University of Rome; 2025.
8. Kandar S, Chaudhuri D, Bhattacharjee A, Dhara BC. Image encryption using sequence generated by cyclic group. J Inf Secur Appl. 2019;44:117–29. doi:10.1016/j.jisa.2018.12.003.
9. Rosch-Grace D, Straub J. Analysis of the likelihood of quantum computing proliferation. Technol Soc. 2022;68:101880. doi:10.1016/j.techsoc.2022.101880.
10. Shukla KK, Rai HM, Amanzholova S, Priyanka, Chaudhary A, Sharma G. Exploring advancements, applications, and challenges in the realm of quantum cryptography. In: Next generation mechanisms for data encryption. Boca Raton, FL, USA: CRC Press; 2024. p. 116–45. doi:10.1201/9781003508632-9.
11. Mattiello H. Sailing the X.0 wave theory: navigating the future of civilization. Intell Sustain Manuf. 2024;1(2):10021. doi:10.70322/ism.2024.10021.
12. Tomasello F. From industrial to digital citizenship: rethinking social rights in cyberspace. Theory Soc. 2023;52(3):463–86. doi:10.1007/s11186-022-09480-6.
13. Peralta G, Cid-Fuentes RG, Bilbao J, Crespo PM. Homomorphic encryption and network coding in IoT architectures: advantages and future challenges. Electronics. 2019;8(8):827. doi:10.3390/electronics8080827.
14. Oppliger R. Cryptography 101: from theory to practice. Norwood, MA, USA: Artech House; 2021.
15. Zaman BU. Exploring a dichotomy prime numbers divided by a unique property. Authorea Preprints. 2024;1–3.
16. Taylor PA. Hackers: crime in the digital sublime. London, UK: Psychology press; 1999.
17. Perkovich G, Levite AE. Understanding cyber conflict: 14 analogies. Washington, DC, USA: Georgetown University Press; 2017.
18. Fajinmi JO. Quantum computing in the spotlight: redefining cybersecurity and cryptography. Authorea Preprints. 2025.
19. Beckham O, Oldman G, Karrie J, Craig D. Techniques used to formulate confidential data by means of fragmentation and hybrid encryption. IRJMIS. 2019;6(6):68–86. doi:10.21744/irjmis.v6n6.766.
20. Anastopoulos C, Hu BL. Equivalence principle for quantum systems: dephasing and phase shift of free-falling particles. Class Quantum Grav. 2018;35(3):035011. doi:10.1088/1361-6382/aaa0e8.
21. Higgins PM. Number story: from counting to cryptography. Berlin/Heidelberg, Germany: Springer; 2008.
22. Tan M. The paradoxical and the reversible: towards a critique of pervasive computing and the intelligent nation 2015 (iN2015) Masterplan in Singapore [dissertation]. Singapore: National University of Singapore; 2011.
23. Kim M, Harmanci AO, Bossuat JP, Carpov S, Cheon JH, Chillotti I, et al. Ultrafast homomorphic encryption models enable secure outsourcing of genotype imputation. Cell Syst. 2021;12(11):1108–20.e4. doi:10.1016/j.cels.2021.07.010.
24. Goswami A, Raghavan SS, Chandrashekar K. Cloud guardians: AI-driven cybersecurity in the cloud era. Mughalsarai, India: JEC Publication; 2025.
25. Speva JM. Computer architecture using a thorough, concise, step-by-step approach: a course for newcomers to the information security field. Romeoville, IL, USA: Lewis University; 2006.
26. Jungnickel D. On the uniqueness of the cyclic group of order *n*. Am Math Mon. 1992;99(6):545–7. doi:10.1080/00029890.1992.11995889.
27. Mamvong JN, Goteng GL, Zhou B, Gao Y. Efficient security algorithm for power-constrained IoT devices. IEEE Internet Things J. 2020;8(7):5498–509. doi:10.1109/JIOT.2020.3033435.

28. Singh A, Chawla N, Ko JH, Kar M, Mukhopadhyay S. Energy efficient and side-channel secure cryptographic hardware for IoT-edge nodes. IEEE Internet Things J. 2019;6(1):421–34. doi:10.1109/JIOT.2018.2861324.

29. Gangothri BN, Satamraju KP, Malarkodi B. SensorNet-cipher: a lightweight protocol for security in resource-limited networks. In: Proceedings of the 2024 International Conference on Emerging Smart Computing and Informatics (ESCI), 2024 Mar 5–7; Pune, India. doi:10.1109/ESCI59607.2024.10497441.

30. Ramalingam G, Uthirapathy P. Optimizing secure data transmission in cognitive IoT-WSN: an energy-aware approach with hybrid POA-SCA and block chain technology. Int J Commun. 2025;38(8):e70081. doi:10.1002/dac.70081.

Appendix A

Table A1 shows that the first 250 cyclic primes exhibit near-linear growth in V vs. N. Successive gaps cluster at multiples of 6 with mean 15.34. The minimum is 2, the maximum is 3821, the median is 1628, and the largest jump is 66. This pattern aligns with the fact that primes greater than 3 lie in the 6k ± 1 classes. The dominance of 6k gaps appears stronger than expected in this subset. Verify it with mod-6 frequency and dependence tests, runs tests, and Durbin–Watson on gap sequences. Estimate the local slope of V(N) using moving averages or LOESS. Model the largest gaps with heavy-tail fits such as GEV or lognormal. Compute sliding-window densities and compare with the density of all primes up to 3821.

Table A1: The first 250 cyclic primes

N	V	N	V	N	V	N	V	N	V	N	V	N	V	N	V	N	V	N	V
1	2	26	257	51	587	76	953	101	1301	126	1637	151	2087	176	2459	201	2909	226	3407
2	5	27	263	52	593	77	971	102	1307	127	1667	152	2099	177	2477	202	2927	227	3413
3	17	28	269	53	599	78	977	103	1319	128	1697	153	2111	178	2543	203	2939	228	3449
4	23	29	293	54	617	79	983	104	1361	129	1709	154	2129	179	2549	204	2957	229	3461
5	29	30	311	55	641	80	1013	105	1367	130	1721	155	2141	180	2579	205	2963	230	3467
6	41	31	317	56	647	81	1019	106	1373	131	1733	156	2153	181	2609	206	2969	231	3491
7	47	32	347	57	653	82	1031	107	1409	132	1787	157	2207	182	2621	207	2999	232	3527
8	53	33	353	58	659	83	1049	108	1427	133	1811	158	2213	183	2633	208	3023	233	3533
9	59	34	359	59	677	84	1061	109	1433	134	1823	159	2237	184	2657	209	3041	234	3539
10	83	35	383	60	683	85	1091	110	1439	135	1847	160	2243	185	2663	210	3083	235	3557
11	89	36	389	61	719	86	1097	111	1451	136	1877	161	2267	186	2687	211	3089	236	3581
12	101	37	401	62	743	87	1103	112	1481	137	1889	162	2273	187	2693	212	3119	237	3593
13	107	38	419	63	761	88	1109	113	1487	138	1901	163	2297	188	2699	213	3137	238	3617
14	113	39	431	64	773	89	1151	114	1493	139	1907	164	2309	189	2711	214	3167	239	3623
15	137	40	443	65	797	90	1163	115	1499	140	1913	165	2333	190	2729	215	3203	240	3659
16	149	41	449	66	809	91	1181	116	1511	141	1931	166	2339	191	2741	216	3209	241	3671
17	167	42	461	67	821	92	1187	117	1523	142	1949	167	2351	192	2753	217	3251	242	3677
18	173	43	467	68	827	93	1193	118	1553	143	1973	168	2357	193	2777	218	3257	243	3701
19	179	44	479	69	839	94	1217	119	1559	144	1979	169	2393	194	2789	219	3299	244	3719
20	191	45	503	70	857	95	1223	120	1571	145	1997	170	2399	195	2819	220	3323	245	3761

(Continued)

Table A1 (continued)

N	V	N	V	N	V	N	V	N	V	N	V	N	V	N	V	N	V	N	V
21	197	46	509	71	863	96	1229	121	1583	146	2027	171	2411	196	2837	221	3329	246	3767
22	227	47	521	72	887	97	1259	122	1601	147	2039	172	2417	197	2843	222	3347	247	3779
23	233	48	557	73	929	98	1277	123	1607	148	2063	173	2423	198	2879	223	3359	248	3797
24	239	49	563	74	941	99	1283	124	1613	149	2069	174	2441	199	2897	224	3371	249	3803
25	251	50	569	75	947	100	1289	125	1619	150	2081	175	2447	200	2903	225	3389	250	3821

Note: N: number, V: efficient prime number value.

MATLAB code of the recommended model is given in Algorithm A1.

Algorithm A1: MATLAB code of the recommended model

```
function flag = efficient_cyclic_prime(p)
    counter = 0;
    for a = 1:p-1
        dizi = zeros(1, p-1); % Initialize dizi inside the loop
        for i = 1:p-1
            matris(a, i) = mod_ustel(a, i, p);
            dizi(i) = matris(a, i);
        end
      if unikmi(dizi) == 1
          counter = counter + 1;
      end
    end
    if counter * exp(1) > p
        flag = 1;
    else
        flag = 0;
    end
end
function sayac = unikmi(arr)
    sayac = 1;
    for i = 1:length(arr)-1
        for j = i + 1:length(arr)
            if arr(i) == arr(j)
                sayac = 0;
                return; % Break out of the loop once a repeated element is found
            end
        end
      end
    end
```

(Continued)

Algorithm A1 (continued)

```
function result = mod_ustel(base, expo, modulo)
    if expo == 1
        result = mod(base, modulo);
    else
        result = mod(base, modulo);
        for $i$ = 2:expo
            result = mod(result * base, modulo);
        end
     end
  end
```

Python code of the ECP is given in Algorithm A2.

Algorithm A2: Python code of this model

```
import math
import time
def mod_ustel(base, expo, modulo):
    if expo == 0:
        return 1
    result = base % modulo
    for _ in range(2, expo + 1):
        result = (result * base) % modulo
    return result
def unikmi(arr):
    n = len(arr)
    for i in range($n$):
        for j in range($i$ + 1, $n$):
            if arr[$i$] == arr[$j$]:
                return 0        return 1
def efficient_cyclic_prime($p$):
    """"""
    Number control.
    """"""
     if $p$ <= 1:
         return 0
     if not is_prime($p$):
         return 0
     counter = 0
    for a in range(1, $p$):
        dizi = []
        for i in range(1, $p$): # MATLAB'da 1:$p$-1, Python'da range(1, $p$)
            value = pow($a$, $i$, $p$)
            dizi.append(value)
```

(Continued)

Algorithm A2 (continued)

```
                if unikmi(dizi) == 1:
                    counter += 1
                if counter * math.exp(1) > $p$:
                    flag = 1
                else:
                    flag = 0
                return flag
def is_prime($n$):
        if $n$  < 2:
                return False
        for i in range(2, int(math.sqrt($n$)) + 1):
                if $n$  % $i$  == 0:
                        return False
          return True
# Tests
test_primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43]
results = {}
start_time = time.time()
for $p$  in test_primes:
          flag = efficient_cyclic_prime($p$)
          results[$p$] = flag
end_time = time.time()
# Sonuçları JSON formatında kaydet
output = {
        "test_results": results,
        "execution_time_seconds": end_time - start_time,
}
import json
with open('cyclic_prime_test_results.json', 'w') as $f$:
        json.dump(output, $f$, indent = 4)
print("cyclic_prime_test_results.json dosyasına kaydedildi.")
```