Modular Patterns in Semiprimes: Empirical Observations and Conjecture on Factor Sums and Deterministic Congruence Patterns in Semiprimes Enabling Optimised Factorization

Abstract:

I report an empirical derived and theoretically motivated analysis of modular patterns in composite integers, with a focus on semiprimes. For any odd semiprime N(possibly all N $\in \mathbb{Z}$), the results indicate the existence of $N-1$congruences of the form $p+q \equiv r(\mathrm{mod} m)$, where p and q are factors of N, $m \in \{2, \dots, N-1\}$ as 1 and N are trivial and always $N \equiv 0\, mod(1\ or\ N)$, and each residue r belongs to a restricted, well-structured subset R_m. Empirical experiments suggest that these residue constraints are non-random, deterministic and encodes all the necessary information about the factor pair (p, q). I formalize this observation as a conjecture and provide preliminary reasoning for its generality. These results point toward a potentially deterministic structure in the modular representation of factor sums, and potentially speedup the factorisation of any N and offering a new perspective on the arithmetic properties of semiprimes and composite numbers. I invite further mathematical verification and formalization.

Keywords: *Semiprimes, modular arithmetic, factorization, number theory.*

1. Introduction:

The factorization of composite integers, particularly semiprimes, is a classical problem in number theory with deep theoretical and practical significance [1]. Semiprimes — numbers that are the product of exactly two prime factors — play a central role in cryptography, computational number theory, and the study of arithmetic structures [2]. Most known factoring methods rely on probabilistic searches or smoothness assumptions [3–5]. In contrast, this study explores whether intrinsic modular structures in semiprimes encode deterministic information about their factors. Despite extensive research, the development of efficient deterministic methods for factorization remains a major challenge [6].

In this work, I report a novel observation concerning the modular structure of semiprime factor sums. Specifically, for odd semiprime number (N), I find that there exist (N-1) modular congruences of the form $p + q \equiv r (\mathrm{mod} m)$where p and q are factors of N, m$\in\{1,2,\ldots,N-1\}$, and r belongs to a restricted residue set (R_m). Computational experiments indicate that these residues are not randomly distributed but rather follow structured patterns that encode information about the factor pair (p, q).

These observations, while currently supported by empirical analysis and numerical reasoning, suggest the existence of a deterministic arithmetic structure underlying the modular behaviour of factor sums. I formalize this as a conjecture — the *Semiprime modular restriction Conjecture* — and provide preliminary theoretical arguments to motivate its generality. This framework may offer new perspectives on semiprime arithmetic and invites further mathematical investigation and formal proof.

By combining computational data analysis with modular arithmetic reasoning, this work bridges empirical and theoretical approaches, highlighting patterns in composite numbers that have previously remained under explored in factoring approaches. I anticipate that these findings may stimulate and lead to rigorous mathematical formalization.

2. Preliminaries / Methods

2.1. Definitions

This section formally defines the core mathematical concepts and notations used throughout this paper.

Semiprime (N): A semiprime is a natural number N that is the product of exactly two prime numbers, p and q.

$$N = p * q$$

This work specifically focuses on odd semiprimes, which implies that both prime factors p and q must be odd primes. Trivial factors (1 and N) are excluded from consideration, so I assume $2 < p \leq q < N$.

Factor Sum (S_N): For a given semiprime $N = p * q$, the factor sum, denoted as S_N, is the sum of its two prime factors.

$$S_N = p + q$$

This sum is the central quantity of interest in my modular analysis.

Modulus (m) and Congruence: For any integer $N > m \geq 1$, referred to as the modulus, the congruence relation $a \equiv b \ (mod\ m)$ holds if and only if m divides the difference $a - b$. In this work, I analysed the behaviour of the factor sum S_N under various moduli m in $\{1, 2, \ldots, N-1\}$, expressed as:

$$S_N \equiv r_i \ (mod\ m)$$

where r_i is the residue.

Residue Set (R_m): For a given modulus m, the Residue Set, denoted R_m, is the subset of all possible integer remainders in the set $\{0, 1, \ldots, m-1\}$. My central conjecture posits that for any given semiprime N, the true residue $r\_m = (p+q)\ mod\ m$ must belong to a highly restricted, deterministically defined subset of this complete set. This subset, which I also refer to as the restricted residue set, is what makes this approach novel.

Quadratic Factoring Methods (Fermat's Factorization): As a point of reference, classical factorization methods often implicitly relate to the factor sum S_N especially using ($x^2 - bx + N = 0$) where b is the factor sum. Well-known example, Fermat's factorization method for an odd composite N seeks to find integers x and y such that:

$$N = x^2 - y^2 = (x-y)(x+y)$$

By setting $p = x - y$ and $q = x + y$, the factor sum is directly related to the variable

$$x : S_N = p + q = (x-y) + (x+y) = 2x$$

This establishes a direct connection between the factor sum S_N and the search for a solution in quadratic factorization approaches. The value of x is typically searched starting from $\lceil\sqrt{N}\rceil$ or 2*.$\lceil\sqrt{N}\rceil$ based on the method used.

3. Conjectures

3.1. Formal Statement - Semiprime modular restriction Conjecture:

Let $N = pq$ be an odd semiprime, where $p\ and\ q$ are its unknown prime factors. With all or a particular subset of $m \in \{2,3,\dots,N-1\}$ I can always construct a large enough m and its residues $R_m \subset \{0,1,\dots,m-1\}$ which always contains the true $S_N = p+q, S_N\ (mod m) \in R_m$ and the modular congruences can be built solely from the N and no additional information is required to find its factors.

3.2 The key properties of this system are:

i. Self-Contained Construction: The calculation of each residue set $\{R_m\}$ requires only the value of N.

ii. Guaranteed Constraint: The true factor sum is guaranteed to satisfy the congruence $S_N\ (mod m) \in R_m$ for every or all chosen moduli.

The conjecture asserts that the factor sum S_N can be found using any single or several combined moduli $m \in \{2,3,\dots,N-1\}$ and its residues; it is restricted to a small, predictable subset of possibilities. I term the specific set of constraints imposed by $\{R_m\}$ for all moduli m the "modular signature" of the semiprime N.

I further hypothesize that this deterministic structure encodes sufficient information about the factor pair (p,q). By combining the residue sets $\{R_m\}$ across a collection of moduli, one can establish a system of congruences for S_N. This system, potentially solved using methods like the Chinese Remainder Theorem, could dramatically reduce the search space for S_N, creating a pathway to an optimized factorization algorithm that leverages these modular patterns.

4. Preliminary Results and Empirical Support:

4.1. Methodology:

4.1. Precomputation of Modular Signatures

The core of this method is a one-time overhead to generate and store a database of all possible modular restrictions on the factor sum, $S_N = p + q$. This process is performed once for a selected set of moduli or moduli $\{m1, m2, \dots, mk\}$.

4.1.1. Choice of Moduli: The system's effectiveness is enhanced by using a diverse set of moduli. This implementation includes primorials (e.g., 30,210,2310,30030), which are products of the first primes, as well as smaller semiprimes (e,g., 13,17,19,23) and a few composites (e.g., 100,300,360). This variety is lower the computation burden and the explosion of CRT combinations.

4.1.2. The Signature Database: For each modulus m, I generate a key-value map, which I term the modular signature. This map is stored as a JSON file or a similar dictionary structure.

The key is a possible residue of a semiprime modulo m, i.e., $k = N(mod m)$.

The value is the restricted set of allowed residues for the factor sum, $Rm = \{r1, r2, \dots\}$, where one of $r_j \equiv (p + q)(mod m)$.

4.2. Algorithm for Signature Generation

The modular signature for each modulus is generated using the following algorithm:

1. Potential Factor Residues: Since is an odd semiprime, its factors and must be odd. Therefore, the set of all possible factor residues modulo m is defined as $\{1,3,5, \dots, k\}$, where k is the largest, odd integer less than m.
2. Generate Semiprime Residues: All possible residues of $N(mod\ m)$are computed by taking the outer product of the factor residue set with itself: $N_{res} \equiv p \cdot q(mod\ m)$ for all pairs $(p, q) \in P_m \times P_m$.
3. Map to Factor Sum Residues: A dictionary is initialized. For each computed semiprime residue N_{res} , I compute the corresponding factor sum residue $S_{res} \equiv p + q(mod\ m)$ and add it to a set associated with the key N_{res}.
4. Store the Signature: After iterating through all pairs, the resulting dictionary, which now represents the complete modular signature for m, is saved to a file.

The python code and example modular signature files for the computations are available in the following Github repository: https://github.com/chandhruyuva006/Modular_constraint_based_integer_factorisation

4.3. Factorization via Signature Lookup and CRT

Once the database is precomputed, factoring any new large semiprime N becomes an efficient lookup-based process:

4.3.1. Residue Calculation and Lookup: For each modulus in our precomputed set, I calculate the residue $k_i = N(mod\ m_i)$. I then perform a lookup in our database to fetch the corresponding restricted set of factor sum residues, R_{m_i}.

4.3.2. Construct System of Congruences: This lookup process yields a system of congruences for the true factor sum S_N :

$$\begin{cases} S_N\ (mod\ m_i) \in R_{m_i} \\ S_N\ (mod\ m_i) \in R_{m_i} \\ S_N\ (mod\ m_i) \in R_{m_i} \end{cases}$$

3. Solve for S_N: The **Chinese Remainder Theorem (CRT)** is used to solve this system. By combining the constraints from each modulus, the set of possible candidates for is drastically reduced and the modulus (M_{eff}) might become larger based on the chosen $\{m_i, m_i, \dots m_i\}$ and I have a larger but sparse set of residues $R_{eff}\ where\ S_N \in \{r_i, r_i \dots \dots . r_i\}$. An efficient search through this small solution space reveals the true value of S_N.

4. Final Factor Recovery: upon searching through the current constrained space the S_N the factors and are found by solving the quadratic equation $x^2 - S_N x + N = 0$, whose roots are the desired factors.

4.4. Illustrative Example: Factoring N = 72011

Here I demonstrate the entire process with the semiprime $N = 72011$. For this example, I will use our precomputed modular signatures for two small moduli: a primorial modulus $m_1 = 30$ and a prime modulus $m_2 = 17$.(The choice of the selection of the moduli is solely on the empirical evidence and I have observed that combining a primorial, small prime which is not the factor of the primorial and a composite moduli gave optimal search space reduction. But the stacked use of prime moduli can lead to a CRT combination explosion and may increase the memory usage and the moduli should be carefully stacked with CRT rather than combined at once. The example is chosen simply for demonstrating the CRT combination explosion vs search space reduction tradeoff).

(For the reader's reference, the true factors are 107 and 673, and the true factor sum is $107 + 673 = 780$)

Step 1: Signature Lookup

First, I calculate the residue of N for each selected modulus and perform a lookup in our precomputed signature database.

1. **For $m_1 = 30$:**
   I. I calculate $N\ (mod\ 30) = 72011\ (mod\ 30) = 11$.
   II. I perform a lookup in our signature_30.json or txt file for the key 13. The precomputed signature reveals the restricted set of possible factor sum residues: $R_{30}: \{0, 12, 18\}$.
   III. This gives us our first congruence: $S_N = 0\ mod\ (30)\ or\ 12\ mod(30)\ or = 18\ mod\ (30)$.

2. **For $m_1 = 17$:**
   I. I calculate $N\ (mod\ 17) = 72011\ (mod\ 17) = 16$.
   II. I look up the key 17 in our signature_17.json file. Let's assume the precomputation yielded the set: $R_{17}: \{0, 2, 5, 6, 8, 9, 11, 12, 15\}$. .
   III. This gives us our second congruence: $S_N \in R_{17}: \{0, 2, 3, 7, 8, 9, 10, 14, 15\}$.

(Note: I can verify our true factor sum $S_N = 780$ satisfies these constraints: $0\ (mod 30)$ and $15\ (mod 15)$ or I can say that $S_N \in R_{30}: \{0, 12, 18\}\ and\ R_{17}: \{0, 2, 3, 7, 8, 9, 10, 14, 15\}$

Step 2: Solving the Congruence System:

As 30 and 17 are coprime I can easily solve the CRT by creating all the possible combinations $R_{eff} = \{0, 42 \ldots 270, . . 480\}\ (3 * 8 = 24)$. But now the M_{eff} is also increased to 510. I can see that the R_{eff} is increased by a factor of 8 but the M_{eff} is increased by a factor of 17.

S_N is bounded by $2 * \sqrt{N}\ to\ N$. The start of $S_{N\_start} \approx 536$. From the R_{eff} I serve the possible candidates for the perfect square check and in less than 20 iterations I found the true $S_N = 780$ which satisfies the $S_N = 270\ mod(510)$.

Step 3: Final Factor Recovery

With the candidate value $S_N = 780$, I can find the factors by solving the quadratic equation $x^2 - 780 + 72011 = 0$. Solving the quadratic equation yields the unknown factors $p = 107, q = 683$.

While as a biochemist I can't derive a formal proof of the Semiprime Modular Restriction Conjecture is, the extensive empirical evidence suggests its validity is rooted in fundamental properties of modular arithmetic. I present this conjecture and my methodology as a foundation for further theoretical investigation. **I welcome and invite collaboration from the mathematics community to develop a rigorous proof** and explore the theoretical underpinnings of these observed patterns

5. Discussion

The proposed framework introduces a deterministic modular constraint structure that restricts the possible values of the factor sum $S_N = p + q$ for any semiprime $N = pq$. By precomputing permissible residue classes for each modulus and combining them through the Chinese Remainder Theorem (CRT), we observe that the theoretical search space of possible $S_N$values can be dramatically reduced.

5.1 Search Space Reduction

In classical Fermat-style factorization, the algorithm searches over all integers b such that $b^2 - 4N$ is a perfect square, which leads to a search space of approximately $O(\sqrt{N})$. In contrast, the modular restriction approach partitions this space into a finite number of allowed congruence classes for each modulus m.

Each individual modulus filters out a significant fraction of invalid candidates, and the intersection of multiple such modular filters further constrains the feasible region of S_N. Theoretically, if all residues across moduli $m_1, m_2, \ldots, m_k$are combined using CRT, the intersection may uniquely identify the true factor sum once the product $M = m_1 m_2 \ldots m_k$ exceeds N.

Thus, the method effectively converts the continuous factor search problem into a discrete modular reconstruction problem. Each new modulus acts as an independent sieve, exponentially pruning the possible candidates as M grows.

5.2 Theoretical Completeness and CRT Reconstruction

From a theoretical standpoint, the modular restriction system is complete: given the full set of moduli up to $N-1$, the corresponding residues contain all necessary information to reconstruct $S_N$exactly via CRT or at least point towards it. This implies that, in principle, the true factors of Ncan always be recovered deterministically without probabilistic assumptions.

Formally, if for every modulus $m \in \{2,3,\dots,N-1\}$the valid residue subset $R_m$is known, then:

$$S_N \equiv r_m (\mathrm{mod} m), r_m \in R_m.$$

By selecting residues $r_m$from each $R_m$and applying CRT, the resulting solution is congruent to S_N modulo the product of all moduli. Once this combined modulus exceeds N or a significant fraction of N, the reconstruction becomes unique, implying the possibility of exact deterministic factorization.

However, while this completeness is partially guaranteed theoretically, practical realizations are constrained by the exponential growth of modular combinations, and the computational overhead of CRT resolution over large moduli sets.

5.3 Computational Complexity and Practical Constraints

The current implementation remains exploratory. For a given modulus m, generating and storing the valid residue sets $R_m$requires $O(m)$operations. When combined across k moduli, naive CRT reconstruction would grow combinatorially as $O(\prod_{i=1}^{k} \mid R_{m_i} \mid)$, which becomes infeasible without optimization.

In practice, only a small subset of low and medium-sized moduli (e.g., below a few thousand) can be used simultaneously. Nevertheless, even this restricted range yields noticeable empirical reductions in the factorization search space, as demonstrated in the timing experiments.

The method is therefore scalable in theory but limited in current implementation. Its efficiency depends on improving:

1. **Residue pruning strategies** — selecting only moduli with high filtering power or low redundancy possibly dynamically selecting moduli based on the number of residues.
2. **Optimized CRT merging** — reusing partial congruences and caching intermediary remainders.
3. **Candidate generation algorithms** — Optimal candidate generating functions without filtering and eliminating the candidates which are not in the residue sets.

5.4 limitations and feasibility

Although the modular constraint approach does not yet outperform classical factoring algorithms in runtime in my empirical analysis, its novelty lies in transforming factorization into a deterministic combinatorial synthesis problem. The framework is extensible: with optimized modular combination and residue pruning strategies, it could, in principle, factor any integer by progressively increasing the combined modulus range.

The proposed modular constraint approach provides a deterministic framework for integer factorization grounded in residue space reduction and CRT reconstruction. While current implementations are limited by computational constraints, the theoretical completeness of the method suggests that, with optimized residue pruning and modular combination strategies, the approach can be scaled to larger semiprimes.

In this sense, the approach may represent a new deterministic pathway to integer factorization, pending the development of efficient methods for handling modular combination growth and residue filtering. Future research will focus on the mathematical structure of these residue intersections, computational pruning heuristics, and theoretical guarantees of convergence speed under modular compression.

7. Acknowledgements

I sincerely thank Mr. Indhirakumar Murugesan, 12-B class members, PSGCAS class members, mathisfun.com members, Mr. Dario Alpern and open-source Python community for the standard library tools that enabled this implementation.

8. Supplementary Information

All the data used in this study is available in the supplementary files.

9. Data and Code Availability

The current implementation of the crude algorithms and modular signature generation code are available in the Github repository : Chandhru Srinivasan. (2025). Modular constraint-based integer factorization (Version 1.0) [Computer software]. GitHub. https://github.com/chandhruyuva006/Modular_constraint_based_integer_factorisation

10. References:

1. Crandall, R. & Pomerance, C. *Prime Numbers*. (Springer New York, New York, NY, 2001). doi:10.1007/978-1-4684-9316-0.

2. Rivest, R. L., Shamir, A. & Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**, 120–126 (1978).

3. Pollard, J. M. A monte carlo method for factorization. *BIT* **15**, 331–334 (1975).

4. Brent, R. P. An improved Monte Carlo factorization algorithm. *BIT* **20**, 176–184 (1980).

5. Lenstra, H. W. Factoring Integers with Elliptic Curves. *Ann. Math.* **126**, 649 (1987).

6. Bressoud, D. M. & Wagon, S. *A Course in Computational Number Theory*. (Key College Publishing, in cooperation with Springer, New York, 2000).