On a Conjecture Connecting Twin Primes and the Sequence A051451

Mike Winkler

Fakultät für Mathematik, Ruhr-Universität Bochum, Germany

mike.winkler@ruhr-uni-bochum.de

May 28, 2026

Abstract

This paper discusses an empirical conjecture originally formulated in September 2013 concerning the distribution of twin primes in relation to OEIS sequence A051451. The conjecture asserts that, for every $n > 2$, there exists a twin prime pair $(p, p+2)$ with $p < a(n)$ such that $(a(n)+p,\, a(n)+p+2)$ is again a twin prime pair. Furthermore, an empirical sub-exponential upper bound for the minimal prime p is proposed and analyzed. We report an extended probable-prime verification up to $n = 1000$, finding no counterexample and including a numerical analysis of the tightest cases and the observed margins below the proposed envelope. The conjecture is analyzed in the framework of Schinzel's Hypothesis H, the Hardy–Littlewood prime k-tuple conjecture, and the Bateman–Horn heuristic.

1 Introduction

This paper investigates a restricted twin prime problem. Let

$$b(m) = \operatorname{lcm}(1, 2, \ldots, m)$$

be the usual least-common-multiple sequence. We write $a(n)$ for the n-th distinct value assumed by $b(m)$, that is for the OEIS sequence A051451 [1]. Equivalently, if q_n denotes the n-th prime power, with 1 included as the first value, then

$$a(n) = b(q_n) = \operatorname{lcm}(1, 2, \ldots, q_n).$$

Thus repeated values of $b(m)$ are omitted. For example, $b(5) = b(6) = 60$, but the value 60 occurs only once in A051451. The first terms are

$$a(n) = 1, 2, 6, 12, 60, 420, 840, 2520, 27720, 360360, \ldots .$$

In September 2013, the following conjecture was formulated and later recorded in OEIS [2, 1].

Conjecture 1. *For every index $n > 2$, there exists a twin prime pair $(p, p+2)$ with $p < a(n)$ such that $\big(a(n)+p,\ a(n)+p+2\big)$ is also a twin prime pair.*

The condition $p < a(n)$ is retained from the original formulation. In the computed range it is restrictive only for the smallest indices; from $n = 10$ onward, the empirical bound considered below is already far smaller than $a(n)$.

Additionally, an empirical upper bound for the minimal prime p was proposed in the original computation [2]. In the present notation this bound is written as

$$p < e^{\sqrt{n}+c_0}, \qquad c_0 = 4.022931736\ldots .$$

In the original 2013 formulation, the numerical constant c_0 was written as $\log_\pi 100$. No theoretical significance is attached here to this particular representation. Throughout the paper, log denotes the natural logarithm.

n	$a(n)$	p
3	6	5
4	12	5
5	60	11
6	420	11
7	840	17
8	2520	29
9	27720	17
10	360360	149

Table 1: The first eight values satisfying Conjecture 1.

This conjecture was empirically verified in the original computation for all values up to $n = 200$, a point where $a(200)$ is an integer comprising 449 decimal digits. Even when $a(n)$ is large, the corresponding minimal prime p can remain comparatively small [2]. The present computation extends the verified range to $n = 1000$. The full output is available on the author's homepage [8].

2 Number Theoretic Context

The computational data can be interpreted in terms of prime constellations and their associated local factors.

2.1 Prime Constellations and Hypothesis H

For fixed n, Conjecture 1 demands the simultaneous primality of the four linear polynomials

$$f_1(p) = p, \qquad f_2(p) = p+2, \qquad f_3(p) = p+a(n), \qquad f_4(p) = p+a(n)+2.$$

Equivalently, it asks for prime values represented by the translated pattern

$$H_n = \{0, 2, a(n), a(n) + 2\}.$$

The relevant local obstruction is admissibility. For $n > 2$, the integer $a(n)$ is even and divisible by 3. Hence all four elements of H_n are congruent to 0 modulo 2, and modulo 3 the tuple occupies only the residue classes 0 and 2. For every prime $q \geq 5$, the tuple has only four elements and therefore cannot occupy all residue classes modulo q. Thus no prime modulus blocks the simultaneous primality of the four forms.

Schinzel's Hypothesis H predicts that such an admissible system of irreducible linear polynomials assumes prime values simultaneously for infinitely many integers p [3]. In this sense, the conjecture is compatible with the standard general conjectures on prime-producing polynomial systems. Hypothesis H, however, only predicts infinitude for each fixed n. It does not by itself explain why the first suitable value of p should be small.

2.2 The Hardy–Littlewood Singular Series

The expected density of prime constellations is governed heuristically by the Hardy–Littlewood prime k-tuple conjecture and, more generally, by the Bateman–Horn conjecture [4, 5]. A critical component of this density estimate is the singular series $\mathfrak{S}(H_n)$ for

$$H_n = \{0, 2, a(n), a(n) + 2\}.$$

It is given formally by

$$\mathfrak{S}(H_n) = \prod_{q \text{ prime}} \frac{1 - \frac{\nu(q)}{q}}{\left(1 - \frac{1}{q}\right)^4},$$

where $\nu(q)$ denotes the number of distinct residue classes modulo q occupied by the elements of H_n.

The defining feature of A051451 is the high divisibility of its terms by small prime powers. Consequently, for many small primes q one has

$$a(n) \equiv 0 \pmod{q}.$$

For such primes, the two translated pairs $\{0, 2\}$ and $\{a(n), a(n) + 2\}$ coincide modulo q, so that the tuple occupies only the residue classes 0 and 2 modulo q. Thus $\nu(q) = 2$ rather than the generic value 4 for these local moduli.

For a prime $q \geq 5$ dividing $a(n)$, the corresponding local factor is therefore

$$\frac{1 - \frac{2}{q}}{\left(1 - \frac{1}{q}\right)^4}.$$

Compared with the generic case $\nu(q) = 4$, the gain in the local factor is

$$\frac{1 - \frac{2}{q}}{1 - \frac{4}{q}} = \frac{q-2}{q-4}.$$

For the first few primes this gives

$$q = 5 : \ 3, \qquad q = 7 : \ \frac{5}{3}, \qquad q = 11 : \ \frac{9}{7}.$$

This illustrates quantitatively why divisibility of $a(n)$ by small primes increases the expected density.

In the concrete case $a(5) = 60$, the tuple $\{0, 2, 60, 62\}$ occupies only the residue classes 0 and 2 modulo 5. Hence $\nu(5) = 2$. This reduction of $\nu(q)$ increases the local factor in the singular series and therefore raises the expected density of admissible prime quadruplets. Thus the terms of A051451 give rise to large local density factors.

3 Analysis of the Upper Bound

The proposed upper bound

$$p < e^{\sqrt{n}+c_0}, \qquad c_0 = 4.022931736\ldots,$$

is an empirical upper envelope for the computed values. Reordering the inequality gives

$$\log(p) - \sqrt{n} < c_0.$$

It is convenient to measure the remaining distance to the proposed bound by

$$M(n) = c_0 - \Big(\log p - \sqrt{n}\Big).$$

The proposed inequality is therefore equivalent to $M(n) > 0$.

For the original range $3 \le n \le 200$, statistical analysis of the experimental data gives

$$\frac{1}{198} \sum_{n=3}^{200} \Big(\log p - \sqrt{n}\Big) \approx 0.6728.$$

Equivalently, the average margin below the proposed envelope is approximately

$$c_0 - 0.6728 \approx 3.3501.$$

The tightest case in this range occurs at $n = 35$, where $p = 20507$ [2]. Here,

$$\log(20507) - \sqrt{35} \approx 4.0124,$$

and hence

$$M(35) \approx 0.0105.$$

The proposed envelope is not a theorem.

4 Extended Computation up to $n = 1000$

The computation was extended from the previously verified range $n \le 200$ to $n \le 1000$.

For every

$$3 \le n \le 1000,$$

a prime p satisfying the required twin prime conditions was found within the proposed bound

$$p < \exp\Big(\sqrt{n} + c_0\Big).$$

No counterexample occurred in this range, subject to the probable-prime nature of the primality tests described in the Appendix. The final entry of the computation is

$$n = 1000, \qquad p = 5436467, \qquad M(1000) \approx 20.1371,$$

while $a(1000)$ has 3250 decimal digits [8]. Throughout the computed range, the largest observed value of $p(n)$ is still below $3.1 \cdot 10^8$. Thus the data suggest that $\log p(n)$ grows much more slowly than $\log a(n)$ in the tested range.

Table 2 summarizes the numerical behavior in blocks of indices. The column "minimum margin" gives the smallest observed value of $M(n)$ in the corresponding range, and the next column records where this minimum is attained. The column "average margin" gives the average value of $M(n)$ over that range.

The margins become substantially larger for larger n. In the block $901 \le n \le 950$, the smallest margin is already 15.4244, attained at $n = 914$. In the block $951 \le n \le 1000$, the smallest margin is 15.7846, attained at $n = 960$. Thus the final two blocks in Table 2 are far from the proposed envelope.

At the opposite end, the largest margin in the computed range occurs at

$$n = 991, \qquad p = 360947, \qquad M(991) \approx 22.7066,$$

where $a(991)$ already has 3215 decimal digits. This illustrates that the minimal prime $p(n)$ may remain very small even when the translation parameter $a(n)$ is large.

Table 3 lists the tightest cases in the full range $3 \le n \le 1000$. The case $n = 35$ remains the tightest case. In fact, all ten tightest cases still occur for $n \le 100$.

Over the full range $3 \le n \le 1000$, the average value of

$$\log p - \sqrt{n}$$

is approximately

$$-6.7493.$$

Equivalently, the average margin below the empirical bound is approximately

$$10.7723.$$

Compared with the original 2013 range $3 \le n \le 200$, whose average margin is about 3.3501, this is substantially larger. Thus the extension from the original computation to

Range of n	min. margin	at n	avg. margin	largest observed p	digits of $a(n)$
$3 \le n \le 50$	0.0105	35	2.7929	20507	1–66
$51 \le n \le 100$	0.3894	51	2.3744	317087	69–178
$101 \le n \le 150$	1.2230	123	3.5577	1339691	181–310
$151 \le n \le 200$	1.7139	155	4.6530	3829139	313–449
$201 \le n \le 250$	3.3440	203	6.0734	7045109	452–599
$251 \le n \le 300$	4.2157	286	6.6919	18232649	602–757
$301 \le n \le 350$	5.5506	306	8.6086	11167727	760–917
$351 \le n \le 400$	6.4738	356	8.8572	16023101	920–1075
$401 \le n \le 450$	7.2226	426	9.9880	37510157	1079–1245
$451 \le n \le 500$	8.2409	452	10.8514	25249241	1248–1416
$501 \le n \le 550$	9.3982	516	11.3829	47859101	1420–1592
$551 \le n \le 600$	10.3365	590	12.3595	64122197	1595–1769
$601 \le n \le 650$	10.6637	602	13.2141	88016231	1773–1947
$651 \le n \le 700$	12.2566	651	14.3258	49387757	1949–2127
$701 \le n \le 750$	12.5226	732	14.7741	114472409	2130–2308
$751 \le n \le 800$	13.0829	769	15.1431	128424089	2312–2496
$801 \le n \le 850$	13.7621	822	16.4516	171268037	2499–2684
$851 \le n \le 900$	15.1510	860	17.3640	125269607	2688–2869
$901 \le n \le 950$	15.4244	914	17.4627	179200589	2873–3057
$951 \le n \le 1000$	15.7846	960	18.1997	304920167	3060–3250

Table 2: Blockwise numerical summary for the extended computation up to $n = 1000$.

$n = 1000$ strengthens the computational evidence for the conjecture, but it also changes the interpretation of the constant c_0. In the tested range, this constant appears less like a permanently sharp threshold and more like an empirical envelope whose sharpness is governed by a small number of early exceptional values, especially $n = 35$.

For comparison, we also record a descriptive polynomial comparison curve for the computed data. Over the range $100 \le n \le 1000$, the sharper empirical estimate

$$p(n) < \exp(\sqrt{n} + 2.8)$$

still holds, with the tightest case occurring at $n = 123$, where the logarithmic margin below this sharper comparison curve is about $8.5 \cdot 10^{-5}$. Moreover, throughout the full tested range $3 \le n \le 1000$ one has the simpler empirical bound

$$p(n) < 0.8\, n^3.$$

The curve $0.8\, n^3$ is included only as a graphical comparison curve for the computed data. No conjectural significance is attached to this particular coefficient or exponent.

The largest prime p found in the range $3 \le n \le 1000$ occurs at

$$n = 986, \qquad p = 304920167, \qquad M(986) \approx 15.8880.$$

rank	n	p	$M(n)$	$p/\exp(\sqrt{n}+c_0)$
1	35	20507	0.0105	0.9896
2	30	10499	0.2411	0.7858
3	51	47807	0.3894	0.6775
4	72	164999	0.4945	0.6099
5	71	109829	0.8424	0.4307
6	85	223757	0.9242	0.3968
7	56	37781	0.9667	0.3803
8	93	317087	0.9997	0.3680
9	55	31847	1.0704	0.3429
10	53	27107	1.0955	0.3344

Table 3: The ten tightest cases for the proposed upper bound in the range $3 \le n \le 1000$.

Thus the largest absolute value of p is not the same as the tightest normalized case. The bound is controlled by $\log p - \sqrt{n}$, not by p alone. This is consistent with the singular-series heuristic: as n grows, the terms $a(n)$ contain more small prime powers, and the local overlap of the pairs $\{0, 2\}$ and $\{a(n), a(n) + 2\}$ modulo small primes increases the expected density of suitable prime quadruplets.

A computational refinement

The preceding computation used a direct search over prime offsets p, with a first check that $p + 2$ is also prime. A simple reformulation of the terms $a(n)$ suggests a more economical variant for larger ranges.

Let q_n be the n-th prime power and let P_n be the largest prime not exceeding q_n. Since $a(n) = \operatorname{lcm}(1, \ldots, q_n)$, the number $a(n)$ is divisible by every prime $\ell \le P_n$. Hence

$$a(n) = c_n P_n\#,$$

where $P_n\# = \prod_{\ell \le P_n} \ell$ is the primorial and c_n contains the additional prime-power contributions. More explicitly, if

$$e_\ell(n) = \max\{e : \ell^e \le q_n\},$$

then

$$c_n = \prod_{\ell \le P_n} \ell^{e_\ell(n)-1}.$$

For example,

$$a(155) = 404779703328000 \cdot 739\#,$$

and

$$a(377) = 191446573294300692534566 4000 \cdot 2297\#.$$

These factorizations are much shorter than the corresponding decimal expansions, since $a(155)$ already has 322 decimal digits and $a(377)$ has 1000 decimal digits.

The same observation also explains a useful sieving effect. If $(p, p+2)$ is a twin prime pair with $p > P_n$, then for every prime $\ell \le P_n$ one has

$$a(n) + p \equiv p \pmod{\ell}, \qquad a(n) + p + 2 \equiv p + 2 \pmod{\ell}.$$

Since p and $p+2$ are primes and $p > P_n$, no prime $\ell \le P_n$ divides either p or $p+2$. Hence such small primes cannot divide either $a(n) + p$ or $a(n) + p + 2$. In a search beyond the present range, it is therefore natural to traverse precomputed twin-prime offsets $(p, p+2)$ rather than all prime candidates p and to use the compact primorial form $c_n P_n\#$ for independent probable-prime testing, for instance with OpenPFGW [9, 10]. This refinement provides an efficient route for extending the numerical range and for independent checks of the computed output.

5 Graphical Representation of the Computed Data

The numerical data can also be represented graphically. The two figures show two normalizations of the same data. Figure 1 gives the normalized form of the original empirical bound, while Figure 2 displays the growth of the minimal primes together with both the original envelope and a purely descriptive polynomial comparison curve. In both figures, the red points mark the ten tightest cases, listed in Table 3, with respect to the original empirical bound.

For each computed index n, let $p(n)$ denote the smallest prime found by the verification procedure, and define

$$L(n) := \log p(n) - \sqrt{n}.$$

Then the proposed empirical bound

$$p(n) < \exp\Big(\sqrt{n} + c_0\Big)$$

is equivalent to

$$L(n) < c_0.$$

Thus Figure 1 gives the most direct visualization of the bound.

Figure 2 displays the comparison in the original scale of $p(n)$, using a logarithmic vertical axis. Besides the original envelope

$$B(n) := \exp\Big(\sqrt{n} + c_0\Big),$$

it also shows the polynomial comparison curve

$$P(n) := 0.8\, n^3.$$

This second curve is included only as a descriptive guide for the computed range.

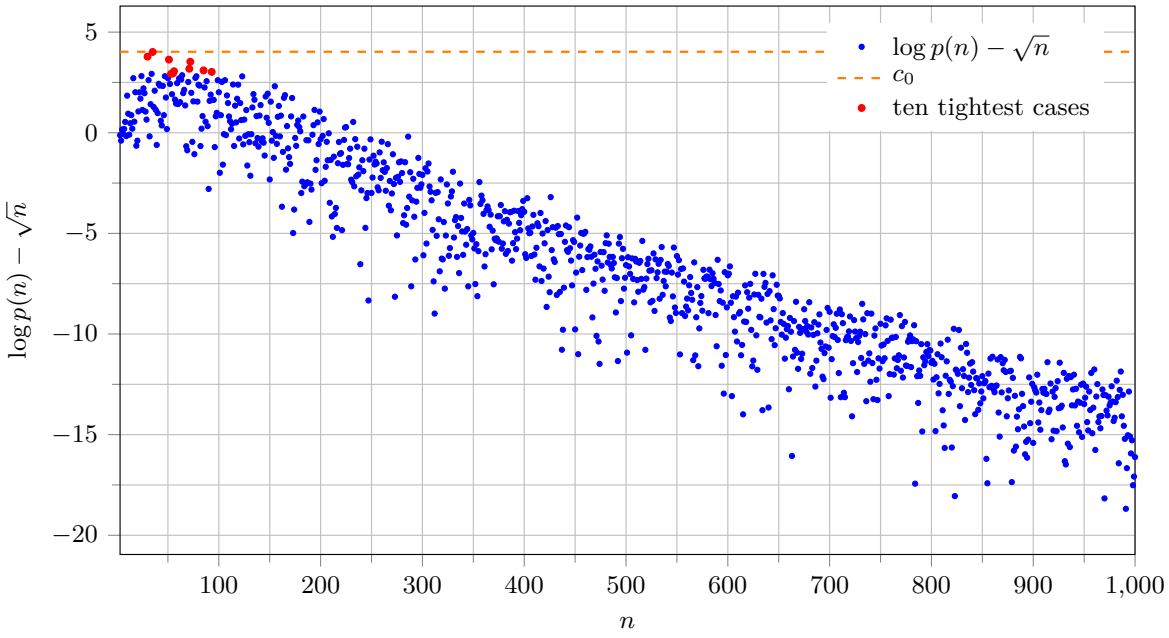

Figure 1: Values of $\log p(n) - \sqrt{n}$ for $3 \le n \le 1000$, together with the empirical threshold $c_0 = 4.022931736\ldots$.

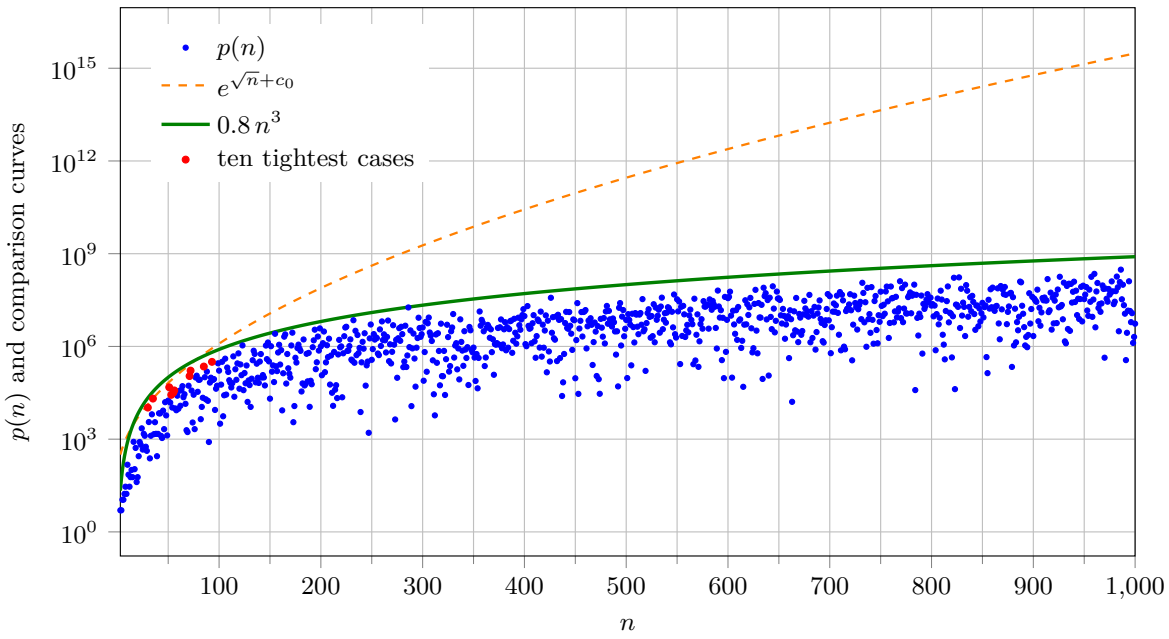

Figure 2: Minimal primes $p(n)$, the original empirical envelope $B(n) = \exp(\sqrt{n} + c_0)$, and the descriptive comparison curve $P(n) = 0.8\,n^3$ for $3 \le n \le 1000$, displayed on a logarithmic vertical scale.

6 Conclusion

The computation up to $n = 1000$ produced no counterexample to Conjecture 1, in the probable-prime sense described in the Appendix. No tighter case for the empirical upper bound was found beyond the early value $n = 35$. Thus the constant $c_0 = 4.022931736\ldots$ appears, in the present data, to be determined mainly by a small number of early exceptional cases.

The extended range changes the interpretation of the bound. Whereas the average margin in the original 2013 range $3 \le n \le 200$ is about 3.3501, the average margin in the full range $3 \le n \le 1000$ is about 10.7723. In particular, the final two blocks in Table 2 have comparatively large margins. The proposed bound therefore remains valid in the computed range, but it becomes less sharp as n increases.

A proof of Conjecture 1 appears to be out of reach with current methods. In particular, standard sieve techniques encounter the parity barrier for problems of twin-prime type [6, 7].

Acknowledgements

I am grateful to the Matheplanet members pzktupel, hyperG, and Primentus for their valuable help with the computations. Their independent checks and optimized computations extended the verification of the conjecture up to $n = 1000$. Further details can be found in the discussion thread accompanying the German forum version of this article [12].

A Julia Verification Script

For reproducibility, the parallel verification script is included below. A machine-readable version may also be kept with the computational output as supplementary material. The script verifies Conjecture 1 computationally for all indices n up to a user-specified limit. It consists of four components.

First, the helper function `is_prime_power` determines whether a given integer k is a prime power, with $k = 1$ treated separately. This is used to generate the sequence `A051451`: the function `get_A051451` iterates over the positive integers and updates a running least common multiple whenever a prime power is encountered, appending each new value to the sequence.

Second, the function `search_for_n` performs the search for one fixed index n. It computes the subexponential search bound limit $= \exp(\sqrt{n} + c_0)$ and searches for the minimal prime $p \ge 3$ such that both $(p, p+2)$ and $\big(a(n)+p,\, a(n)+p+2\big)$ are twin prime pairs. Candidate primes p are traversed using `nextprime`, and primality of the large numbers $a(n) + p$ and $a(n) + p + 2$ is tested using the `Primes.jl` package.

Third, the main function `verify_conjecture` distributes the independent searches for different indices n across Julia threads. Since the search times for different n vary

substantially, the script uses the dynamic thread scheduler when available. It also prints the number of active Julia threads and warns if fewer than four threads are used. The script writes the same output both to the terminal and to a text file. If no output file is specified, a timestamped filename is generated automatically.

For `BigInt` inputs, `Primes.jl` uses probabilistic primality testing with default parameter `reps=25`.[1] Hence the computations should be understood as probable-prime verifications unless independent primality certificates are supplied [11]. For each successful match, the script prints the index n, the minimal prime p, the remaining margin $c_0-(\log p-\sqrt{n})$ relative to the proposed bound, and the number of decimal digits of $a(n)$. No independent primality certificates were generated for the present version. If no valid p is found within the bound, a warning is displayed. Upon completion, the full decimal expansion of $a(\texttt{max\_n})$, the largest value in the sequence, is printed together with its digit count. The output log also records the generation time, the parallel search time, and the total runtime.

Listing 1: Parallel Julia verification script with logging

```
"""
Verification of the Twin Prime / A051451 Conjecture
Parallelized version with logging.

From terminal:
    julia -t 4 verify_conjecture_parallel_with_log.jl 1000

Optional explicit output file:
    julia -t 4 verify_conjecture_parallel_with_log.jl 1000 output.txt

In Julia REPL:
    include("verify_conjecture_parallel_with_log.jl")  # will ask for
       input
"""

using Primes   # install once with: ] add Primes
using Printf
using Dates

# ---------------------------------------------------------------------
# Helper: check whether k is a prime power, with k=1 included
# ---------------------------------------------------------------------
function is_prime_power(k::Int)::Bool
    k == 1 && return true
    return length(factor(k)) == 1
end

# ---------------------------------------------------------------------
```

[1]The documented false-positive probability is about $0.25^{25} = 4^{-25} = 2^{-50} \approx 8.88 \cdot 10^{-16}$ per probabilistic primality test. The GMP implementation uses trial division, a Baillie–PSW probable-prime test, and Miller–Rabin testing according to the parameter `reps`. This does not constitute a deterministic primality certificate.

```
# Generate the first max_n terms of OEIS A051451.
# --------------------------------------------------------------------------
function get_A051451(max_n::Int)::Vector{BigInt}
    seq    = Vector{BigInt}()
    cur    = big(1)
    number = 1
    while length(seq) < max_n
        if is_prime_power(number)
            cur = lcm(cur, big(number))
            push!(seq, cur)
        end
        number += 1
    end
    return seq
end

# --------------------------------------------------------------------------
# Result type: either a found tuple or a sentinel for "not found".
# --------------------------------------------------------------------------
struct Result
    found  :: Bool
    p      :: Int
    margin :: Float64
    digits :: Int
end

# --------------------------------------------------------------------------
# Logging helpers
# --------------------------------------------------------------------------
function default_output_file(max_n::Int)::String
    timestamp = Dates.format(now(), "yyyymmdd_HHMMSS")
    return "verify_conjecture_parallel_n$(max_n)_$(timestamp).txt"
end

function emit(io::IO, msg="")
    println(msg)
    println(io, msg)
    flush(io)
end

# --------------------------------------------------------------------------
# Inner search function: one index n, entirely thread-local state.
# --------------------------------------------------------------------------
function search_for_n(n::Int, a_n::BigInt, c0::Float64)::Result
    limit  = exp(sqrt(n) + c0)
    digits = length(string(a_n))
    p      = 3

    while p < limit
        if isprime(p + 2)
            if isprime(a_n + p) && isprime(a_n + p + 2)
                margin = c0 - (log(p) - sqrt(n))
```

```
                return Result(true, p, margin, digits)
            end
        end
        p = nextprime(p + 1)
    end

    return Result(false, 0, 0.0, digits)
end

# ---------------------------------------------------------------------
# Parallel verification loop
# ---------------------------------------------------------------------
function verify_conjecture(max_n::Int; output_file::Union{Nothing,
    String}=nothing)
    output_file = isnothing(output_file) ? default_output_file(max_n)
        : output_file

    open(output_file, "w") do io
        total_start = time()

        emit(io, "Verification of the Twin Prime / A051451 Conjecture"
            )
        emit(io, "Parallelized Julia version")
        emit(io, "Output file: $(abspath(output_file))")
        emit(io)

        emit(io, "Generating A051451 up to index $max_n ...")
        generation_start = time()
        a_vals = get_A051451(max_n)
        generation_time = time() - generation_start
        emit(io, @sprintf("Generated A051451 in %.3f seconds.",
            generation_time))
        emit(io)

        c0      = log(100) / log(pi)
        start_n = 3

        nt = Threads.nthreads()
        emit(io, "Using $nt thread(s).")
        script_name = isempty(PROGRAM_FILE) ? "
            verify_conjecture_parallel_with_log.jl" : basename(
            PROGRAM_FILE)
        if nt < 4
            emit(io, "WARNING: The program is running with only $nt
                thread(s). Start Julia with: julia -t 4 $script_name
                $max_n")
        end
        emit(io, "Verifying conjecture for n = $start_n ... $max_n")
        emit(io)

        results = Vector{Result}(undef, max_n)

```

```
        search_start = time()

        @static if VERSION >= v"1.9"
            Threads.@threads :dynamic for n in start_n:max_n
                results[n] = search_for_n(n, a_vals[n], c0)
            end
        else
            Threads.@threads for n in start_n:max_n
                results[n] = search_for_n(n, a_vals[n], c0)
            end
        end

        search_time = time() - search_start
        emit(io, @sprintf("Parallel search completed in %.3f seconds."
            , search_time))
        emit(io)

        for n in start_n:max_n
            r = results[n]
            if r.found
                emit(io, @sprintf("n=%3d | p=%-10d | margin = %.4f |
                    digits = %d",
                                  n, r.p, r.margin, r.digits))
            else
                emit(io, "n=$n: *** NO MATCH found within the bound!
                    ***")
            end
        end

        a_last  = a_vals[max_n]
        ndigits = length(string(a_last))
        emit(io)
        emit(io, "="^60)
        emit(io, "Full value of a($max_n) [$ndigits digits]:")
        emit(io, "="^60)
        emit(io, string(a_last))
        emit(io, "="^60)

        total_time = time() - total_start
        emit(io)
        emit(io, @sprintf("A051451 generation time: %.3f seconds",
            generation_time))
        emit(io, @sprintf("Parallel search time:    %.3f seconds",
            search_time))
        emit(io, @sprintf("Total runtime:           %.3f seconds",
            total_time))
        emit(io, "Output saved to: $(abspath(output_file))")
        emit(io)
        emit(io, "Done.")
    end
end

```

```
# ------------------------------------------------------------------------
# Entry point: works from terminal and from REPL.
# ------------------------------------------------------------------------
if length(ARGS) >= 1
    max_n = tryparse(Int, ARGS[1])
    output_file = length(ARGS) >= 2 ? ARGS[2] : nothing

    if isnothing(max_n) || max_n < 3
        println("Error: max_n must be an integer >= 3.")
    else
        verify_conjecture(max_n; output_file=output_file)
    end
else
    print("Enter upper limit for n (>= 3): ")
    input = readline()
    max_n = tryparse(Int, input)
    if isnothing(max_n) || max_n < 3
        println("Error: max_n must be an integer >= 3.")
    else
        verify_conjecture(max_n)
    end
end
```

References

[1] OEIS Foundation Inc., *Sequence A051451*, The On-Line Encyclopedia of Integer Sequences, https://oeis.org/A051451, accessed May 2026.

[2] Winkler, M. *A twin prime conjecture*. September 2013, https://mikematics.de/twin_prime_conjecture_2013.pdf, accessed May 2026.

[3] Schinzel, A. and Sierpiński, W. *Sur certaines hypothèses concernant les nombres premiers*, Acta Arithmetica **4**, no. 3, 185–208, 1958.

[4] Hardy, G. H. and Littlewood, J. E. *Some problems of 'Partitio Numerorum'; III: On the expression of a number as a sum of primes*, Acta Mathematica **44**, 1–70, 1923.

[5] Bateman, P. T. and Horn, R. A. *A heuristic asymptotic formula concerning the distribution of prime numbers*, Mathematics of Computation **16**, no. 79, 363–367, 1962. DOI: 10.2307/2004056.

[6] Halberstam, H. and Richert, H.-E. *Sieve Methods*, Academic Press, London, 1974.

[7] Murty, M. R. and Vatwani, A. *Twin primes and the parity problem*, Journal of Number Theory **180**, 643–659, 2017. DOI: 10.1016/j.jnt.2017.05.011.

[8] Winkler, M. *Computational output for the A051451 twin prime conjecture,* $3 \leq n \leq 1000$, May 2026. Ancillary file: https://mikematics.de/A051451_twin_prime_output_1000.txt, accessed May 2026.

[9] Luhn, N. *Prime tuple archive*, https://pzktupel.de/smarchive.php, accessed May 2026.

[10] OpenPFGW. *OpenPFGW primality testing software*, https://sourceforge.net/projects/openpfgw/, accessed May 2026.

[11] JuliaMath, *Primes.jl documentation*, https://juliamath.github.io/Primes.jl/stable/api/, accessed May 2026.

[12] Matheplanet. *Discussion thread accompanying the German forum version of this article*, https://www.matheplanet.de/matheplanet/nuke/html/article.php?sid=2039&mode=&order=0&thold=0#begincomments, accessed May 2026.