The Riemann hypothesis

A guide to the submitted Lean proof

This guide follows the Lean project linked in Levent Alpöge's November 2026 announcement. It identifies the precise theorem statements, reconstructs the analytic deduction, and traces the arithmetic inputs that would establish the claimed zero-free region. Linked filenames refer to the archived Lean sources in the verification repository. Original announcement

Verification status at drafting. The targeted Yokota run checked seven analytic lemmas using only the three standard Lean axioms. The full build of the submission's `Main` module was still running. The arithmetic inputs to the central estimate remain open in this audit. Live verification record

1 What the submission asserts

`Main.lean` exposes five declarations. The main RH declaration has Mathlib's own type `RiemannHypothesis`. The Dirichlet assertions use Mathlib's `DirichletCharacter.LFunction` at every positive modulus.

Lean declaration	Mathematical content
`riemannHypothesis`	Mathlib's statement of RH.
`riemannZeta_eq_zero_re`	The same zero condition written out in the submission.
`LFunction_ne_zero_of_half_lt_re`	Nonvanishing to the right of the critical line, with the principal character at $s = 1$ excepted.
`LFunction_eq_zero_re`	Every zero of every Dirichlet L-function in $0 < \operatorname{Re} s < 1$ lies on $\operatorname{Re} s = \frac{1}{2}$.
`LFunction_eq_zero_re_of_isPrimitive`	The critical-strip assertion restricted to primitive characters.

See `Statements.lean` and the pinned Mathlib definition of RH. The term GRH in this guide refers to Dirichlet L-functions.

Proof in one line. Construct arithmetic families with controlled cohomology and divisibility, obtain a square-root-scale bound for smoothed prime sums on power intervals, apply a Landau-type zero-free criterion, and reflect zeros by the functional equations.

2 The reduction to right-half-plane nonvanishing

The proof routes RH and the critical-strip assertions through `UpperTheory.EveryCharacter`. In mathematical notation this proposition is

$$\forall N \geq 1,\ \forall \chi \pmod N,\ \forall s \in \mathbb{C}, \qquad \operatorname{Re} s > \tfrac{1}{2},\ (\chi, s) \neq (1,1) \quad \Longrightarrow \quad L(s,\chi) \neq 0.$$

The source uses the disjunction $\chi \neq 1 \vee s \neq 1$. The exception handles the pole under the conventions of the formalized L-function. The equivalent right-half-plane input in the final wrapper is `Claim.HalfPlane`. `EveryCharacter.lean` `HalfPlane.lean`

Primitive and imprimitive characters

For a character χ modulo N, let χ_* be its primitive character of conductor $f \mid N$. Passing from χ_* to χ introduces a finite product of Euler factors of the form $1 - \chi_*(p)p^{-s}$. If $\operatorname{Re} s > 0$, then

$$\left|\chi_*(p)p^{-s}\right| \leq p^{-\operatorname{Re} s} < 1.$$

No such factor vanishes there. Thus a zero of the imprimitive L-function in the open strip gives a zero for the primitive inducing character. The same factorization extends a primitive right-half-plane result to all characters. `ClaimStrip.lean` `Imprimitive.lean`

For primitive ψ, the completed functional equation takes a zero at s to a zero of the completed function for ψ^{-1} at $1 - s$. In the open strip, the gamma factor at s is nonzero, so the original L-zero gives a completed zero. At the reflected point the source rewrites the L-function as the completed function divided by its gamma factor. If $0 < \operatorname{Re} s < \frac{1}{2}$, then $\operatorname{Re}(1-s) > \frac{1}{2}$, and the reflected point contradicts right-half-plane nonvanishing. Zeros with $\operatorname{Re} s > \frac{1}{2}$ contradict it directly. The critical line is the remaining case. `ClaimStrip.lean`

The zeta function

The character of modulus 1 satisfies $L(s,\chi) = \zeta(s)$ in Mathlib. The route through `Formalism.VW.rh_of_every` uses this equality and the zeta functional equation. Mathlib's RH statement explicitly excludes the negative even integers and $s = 1$. The source also uses $\zeta(0) \neq 0$. An alternative reduction, `Claim.zeta`, treats the nonpositive half-plane via the completed zeta function and the established nonvanishing on $\operatorname{Re} s \geq 1$. The reduction audit checked the analytic lemmas underlying both routes. `MainTheorems.lean` `CritLine.lean` `RhOfRight.lean`

3 The analytic criterion on power intervals

Smoothing and normalization

Write $\Lambda(n)$ for the von Mangoldt function. The zeta component is the centered sum

$$S_\zeta(x) = \sum_{n \le x} \Lambda(n)(x/n - 1),$$
$$G_\zeta(x) = S_\zeta(x) - x \log x + (1 + \gamma)x,$$

where γ is the Euler–Mascheroni constant. For a nonprincipal character the complex version of the centered sum replaces $\Lambda(n)$ by $\Lambda(n)\chi(n)$ and adds $xL'(1,\chi)/L(1,\chi)$. The source takes real parts and combines a finite family with positive weights. The cyclic level of a primitive character includes the primitive characters induced by its powers and includes the trivial character. The resulting level sum is `Level.Gm`. `level_family` identifies that sum with the weighted family in the analytic criterion, preserving the zero-free goal. `StatementOC.lean` `Analytic.lean` `ExplicitFormulaErrorOfHostConditions.lean` `LevelFamily.lean`

Why zeros leave uncancelled poles

For $\operatorname{Re} s > 1$ the smoothing factor satisfies

$$\int_n^\infty (x/n - 1)x^{-s-1}\,dx = \frac{n^{-s}}{s(s-1)}.$$

The Dirichlet series for the smoothed sum therefore enters through $-L'(s,\chi)/L(s,\chi)$. At a zero ρ of multiplicity m, that logarithmic derivative contributes $-m/(s-\rho)$ in its principal part. Centering removes the terms at $s = 1$. The residues at a zero of the family take the form

$$-\frac{S}{\rho(\rho-1)}, \qquad S > 0,$$

where S is a positive weighted sum of multiplicities, including the conjugate-character terms used to express real parts. Since all contributions at a fixed ρ have the same complex factor, they cannot cancel. In particular the residue is positive at a real zero in $(0,1)$. This sign matters because the argument uses an upper bound on a real-valued sum. `poleData.lean`

From a window bound to nonvanishing

Let $0 < \theta < 1$ and $\delta > 0$. The criterion assumes that for every lower threshold there exists a larger Y such that $G_m(y) \le Cy^\theta$ on the whole interval $[Y, Y^{1+\delta}]$. After $u = \log y$, high derivatives of the Laplace transform insert the kernel $u^k \exp(-\sigma u)/k!$. The proof chooses k so this kernel concentrates inside $[\log Y, (1+\delta)\log Y]$ and bounds the tails. Cauchy estimates compare the derivative bound with the growth forced by a nearby pole. The argument handles the real-pole case using the positive residue. The conclusion is nonvanishing for the family on $\operatorname{Re} s > \theta$, with the region $\operatorname{Re} s \ge 1$ handled by existing theorems. `landauTails.lean` `landauWindow.lean` `ZeroFreeOfRpowWindow.lean`

4 What the arithmetic argument must supply

For the cyclic level $\mathcal{L}$ of each primitive character, the argument requires families of windows on which

$$G_{\mathcal{L}}(x) \le x^{1/2+(e_D+e_L+7)\varepsilon}.$$

The final assembly uses the power $\lambda = 3/2$ for the windows. The constants e_D, e_L are chosen before the quantifier over all sufficiently small $\varepsilon > 0$. For a candidate zero with real part $\beta > 1/2$, the code chooses a θ strictly between $1/2$ and $\min(\beta, 1)$ and makes $(e_D + e_L + 7)\varepsilon$ small enough to apply the analytic criterion. The thresholds and the family may depend on ε. A bound at isolated points would leave the window hypothesis unmet. `TopWin.lean` `ZeroFreeAllOfTopG.lean` `TopPlug.lean`

The bridge from arithmetic data to the bound

The theorem that produces the quantitative bound is `UpperTheory.Comparison₂.core_gm`. It takes two inputs at scale x.

1. `EAtH` relates a level's primitive characters to data on an auxiliary arithmetic host and controls comparison errors.
2. `CoreAtH` records a witness with the identities, size bounds, and error estimates needed by the prime-sum argument.

The proof matches the weighted sum to the host expression, applies the chain estimate `Deduction.ch_top` and the comparison `Deduction.ch_core`, and absorbs fixed factors into the spare powers of x^{ε}. The family-level theorem chooses a suitable member for each x in a window. The analytic theorem then consumes the resulting inequality and the window quantifiers.

`CoreAtH.lean` `CoreGm.lean` `WindowInequalityOfArtinInputs.lean` `LevelGmLeOnFamilyScales.lean`

How the family requirements are assembled

`zeroFreeAll_of_topG` separates the existence of hosts from the construction of compatible estimates over a family. The supply argument `supplyChi` combines a cohomology window for a primitive character with admissible arithmetic classes. The other input, `TopG`, produces the family-level estimates under that supply hypothesis and the relevant cohomology bounds. `topDesWin` collects the design-existence, divisibility, rank and global compatibility hypotheses used to obtain `ChainTied` witnesses on the windows. These witnesses feed `CoreAtH`. This is a useful boundary for reviewing whether each downstream estimate applies to the hosts that are actually supplied. `SupplyChi.lean` `HostOfField.lean` `DesWin.lean`

5 The two arithmetic branches

Auxiliary fields and the cohomology window

`IsChiField` describes the auxiliary field through an embedding into a cyclotomic field and a subgroup condition involving the given character and two quadratic conditions. `WinPair` prescribes the relative sizes and congruence conditions on auxiliary primes q, d. The field bound `VVle` demands finiteness of the relevant lengths and bounds the weighted sums over finite prime sets, with weights $\log \ell$. Here the length is that of a localization kernel in degree-one Galois cohomology for the twist by $-e$. A nontrivial point for review is that the field and local data actually constructed for the prime-sum estimate meet the conditions of this bound. `ExistsPageN.lean` `Statement.lean` `supplyOfAdmOf.lean`

The proposition `VWinAt` supplies a window for each sufficiently small ε and admissible choice of $e(x)$. Its pointwise target `VAt` uses a cutoff $\lfloor x^{401} \rfloor$ in the specialization `VWin`, with cohomology bound $\exp(x^{\varepsilon}/2)$. The route through `vwin_of_row` combines local–global compatibility, base change, automorphy inputs and several counting bounds. The proof packages these inputs before applying `vwin_of_counts`. `VwinOfRow.lean` `WindowBoundOfLiteratureLevel.lean` `JointWinOfCounts.lean`

Theta matching, covolumes and divisibility

The other branch constructs the hypotheses used for the prime-sum comparison. The assembly passes theta-integral agreement and local matching, a covolume bound, a local Langlands existence result, and the cohomology window to `every_character_of_theta_integrals_local_matching`. Inside that route the supplied results are used to construct the required nonempty family hypotheses. At the design stage, `divinN_of_hypV` turns a cohomology input into divisibility data using additional link and rank hypotheses. The window bound depends on this compatibility as well as on analytic error estimates.

`EveryCharacterOfLiteratureOfVWin.lean` `SiegelEisensteinSecondTermHypotheses.lean` `DivinNOfHypV.lean`

Where to start a deeper review. Check the proof of the local–global compatibility row in the form used by `vwin_of_row`, then follow one concrete character all the way from `VWinAt` to its `CoreAtH` witness. This tests whether the arithmetic constructions supply the same data and uniformity that the analytic comparison consumes.

6 Reading order and verification evidence

Start from the actual proof terms in `Main.lean`. The principal route passes through `allChi` in `MainTheorems.lean` to `UpperTheory.FinalTheorems.everyCharacter` in `MainConclusion.lean`. The final declaration supplies `galoisLocalGlobalStrong_recParameter` from the theta-lift and Galois-representation development. Follow the named declarations rather than treating every file imported by a module as a direct mathematical premise. `RowSrc.lean`

Question	Files to open
What is being proved?	`Main.lean`, `Statements.lean`
How is the upper bound used?	`ZeroFreeOfRpowWindow.lean`, `TopWin.lean`
Where is the upper bound assembled?	`CoreGm.lean`, `ZeroFreeAllOfTopG.lean`
Where are both branches joined?	`MainConclusion.lean`, `EveryCharacterOfLiteratureOfVWin.lean`

What has been checked

The isolated reduction job `157173` on Yokota completed with exit status zero. Lean checked the two conditional deductions from `Claim.HalfPlane` to RH and Dirichlet GRH. The axiom audit covered `Claim.prim_reflect`, `Claim.strip`, `Claim.zeta`, `Claim.factor_ne_zero`, `Formalism.VW.rh_of_right`, `Deduction.zeroFree_of_rpow_window` and $\texttt{UpperTheory.Comparison}_2\texttt{.formB}$. Each reported `propext`, `Classical.choice` and `Quot.sound`. The full job `157166` was still compiling the import closure of `Main` when these notes were drafted. Its recorded procedure checks the top-level statements and transitive axioms and then runs `leanchecker -fresh Main`. Reduction job output Reduction audit source Full job status

Reproducibility

The public archive consists of fourteen concatenated segments. The private verification repository preserves 175,187 submitted Lean source files and the three project configuration files. The toolchain is Lean `v4.33.1`, and the Mathlib pin is `0df444a360eaa60ab8c11dca51a86af692955474`. Segment checksums Full build script

The evidence supports the analytic reductions under their stated inputs. The decisive pending check is whether the submitted arithmetic constructions supply the window estimate for every required character.