Twofold coverings of the Hamming cube: a Lean-verified K(8,1,2) >= 61, and an even-n theorem improving five published bounds

Hanyu Yang — ORCID 0009-0005-0419-4070

DOI 10.5281/zenodo.22217672 (concept DOI) · Version 2.7, released 2026-09-02

Typeset rendering of `PAPER.md` at commit `b8a2c6801b0f6d0be8154d7515f4f2080ccec38c` on branch `main` of https://github.com/SeverinVisionary/oeis-a004045; rendered 2026-09-02. The text below is the source file as it stands at that commit.

Machine-checked lower bounds for binary twofold coverings of the Hamming cube

narrowing the first unknown term of OEIS A004045 to `61 ≤ K(8,1,2) ≤ 64`

Hanyu Yang — ORCID 0009-0005-0419-4070

Draft, 2026-09-03. Not submitted, and not peer-reviewed by a human. §10 states precisely what is and is not established; please read it before citing.

Generative AI was used extensively in producing this work. See Declarations for the full statement.

Abstract

A binary code `C ⊆ F_2^n` is a **twofold covering** of radius 1 if every point of `F_2^n` lies within Hamming distance 1 of at least two codewords; `K(n,1,2)` denotes the least size of such a code. We prove that for **even `n`**

> **`|C| ≥ ⌈ 3·2^(n+1) / (3n+2) ⌉`,**

by a half-page parity refinement of the covering-excess method of Johnson and van Wee. The bound strictly improves the best published lower bound for *every* even `n ≥ 6`, and in particular raises the five tabulated positions `n = 8, 10, 12, 14, 16` of Krotov and Potapov (2021) from `59, 188, 640, 2195, 7783` to `60, 192, 647, 2235, 7865`. It also gives the exact value `K(6,1,2) = 20` with no computer search, where the published value came from integer programming.

The case `n = 8` is the first unknown term of OEIS A004045, where the published record was `59 ≤ K(8,1,2) ≤ 64` — the lower bound due to Krotov and Potapov (2021). The upper bound 64 is older and is not due to a search: it appears already in the 1993 table of [HHKL], where it is the doubling `K(8,1,2) ≤ 2·K(7,1,2) = 2·32 = 64`. Östergård's 1995 tabu-search paper improves 27 upper bounds in that table, but `n = 8`, `μ = 2` is not among them. A second, computational argument refutes `|C| = 60`, and that argument is **formalised in Lean 4 with zero `sorry`s**, giving

> **`61 ≤ K(8,1,2) ≤ 64`.**

Finally, a prescribed-automorphism sweep over all 28 prime-order conjugacy classes of `Aut(Q_8) = F_2^8 ⋊ S_8` indicates that **every twofold covering of `F_2^8` with at most 63 codewords is asymmetric** — it has trivial automorphism group — which constrains where any improvement of the upper bound can live. This last result is on a weaker footing than the others: 4 of the 28 classes are machine-certified and the remaining 24 are floating-point solver verdicts.

Everything is reproducible: `./reproduce.sh` runs twelve self-asserting checks using the Python standard library, apart from one step that needs `scipy` for small LP control cases, and exits non-zero if any fails.

In one paragraph, for the non-specialist

Imagine the 256 possible eight-bit strings. Choose a small set of them — call these *sentinels* — so that every one of the 256 strings is within one bit-flip of at least **two** sentinels. Two, not one, so that the system still works if a sentinel fails. How few sentinels suffice? Nobody knows exactly. The answer has only ever been pinned to a range: at most 64, a value already tabulated in 1993 by a one-line doubling argument and never improved since, and at least 59, raised to that value in 2021. It is a listed open value in the on-line encyclopedia of integer sequences. This paper raises the lower end to 61, so the answer is now known to be 61, 62, 63 or 64. The improvement comes from a short counting argument that also improves infinitely many other cases, plus one computational step that a proof assistant checks, so it does not rest on trusting the author or a solver. The upper end, 64, is untouched, and closing the remaining gap is open.

Contributions, and what is actually new

Ranked by what a reader is likely to care about. "Novelty" is stated conservatively; see §7 for the full prior-art discussion, including a preprint we could not locate.

#	Contribution	Before	After	Verified by	Novelty
1	`K(8,1,2) ≥ 61`, eliminating two of the six candidate values for the first unknown term of A004045	`≥ 59` (2021)	`≥ 61`	**Lean 4, zero `sorry`s**, audited axiom trace	The *bound* is new — no published lower bound exceeds 59. The ingredients (Delsarte nonnegativity, Farkas certificates) are standard, and covering codes have been formalised in Lean before ([Flo1], [Flo2] — ordinary coverings only).
2	Theorem 1: `K(n,1,2) ≥ ⌈3·2^(n+1)/(3n+2)⌉` for all even `n`	—	improves 5 tabulated positions and infinitely many more	half-page proof; `excess_theorem.py`	Not found in the primary sources we checked, **and** we give a *positive* argument (§7) that the standard method provably cannot produce it. Not certified new: see the Chen–Li disclosure.
3	`K(6,1,2) = 20` with no computer search	value known, by integer programming	same value, by hand	`bipartite_split.py`	The value is not new; the *search-free proof* is.

#	Contribution	Before	After	Verified by	Novelty
4	Every twofold covering of `F_2^8` with `≤ 63` words is asymmetric	not addressed	asymmetric	28 prime-order classes infeasible; 4 machine-certified, 24 floating-point	Method is standard (prescribed automorphisms, Kramer–Mesner). The statement for these parameters appears to be new but is of a routine kind.
5	Proposition 4: the method extends to all even `μ`, but wins only at `μ = 2`	—	—	`mu_generalization.py`	A negative result about our own method.

Is any of this a breakthrough? Not in method. Theorem 1 is elementary — that is its point, and it is also a reason to suspect someone may have noticed it before. What is unusual here is the *standard of verification*: the headline inequality `61 ≤ K(8,1,2)` is not asserted on the strength of a solver log or of the author's care, but reduced to Lean's kernel plus four definitions a reader can check by eye — and which the author has checked (§8). For a quantity whose literature consists largely of uncertified integer-programming runs, that is the contribution most likely to outlast the numbers.

What this paper does not do. It does not touch the upper bound, which remains 64, the 1993 doubling bound; and it does not close A004045(8). It has had no peer review; the author has read the four Lean definitions the result rests on (§8), but no human has read the proofs.

---

1. Introduction

Let `F_2^n` be the binary Hamming cube and `B(x) = {y : d(x,y) ≤ 1}` the closed ball of radius one, so `|B(x)| = n+1`. A set `C ⊆ F_2^n` is a **μ-fold covering** of radius 1 if every `x ∈ F_2^n` satisfies `|B(x) ∩ C| ≥ μ`. Write `K(n,1,μ)` for the smallest size of such a `C`. Codewords are **distinct**: `C` is a set, not a multiset. (Allowing repeats gives a different quantity, written `K̃` by Hämäläinen, Honkala, Kaikkonen and Litsyn [HHKL], and the arguments below use the set hypothesis essentially.)

`K(n,1,2)` is the double-domination number of the hypercube `Q_n` and is sequence A004045 in the OEIS, whose first unknown term is `K(8,1,2)`. Before this note the record was

```
59 ≤ K(8,1,2) ≤ 64,
```

the lower bound due to Krotov and Potapov [KP, Theorem 6] and the upper bound to [HHKL, Table 5], via `K(n+1,r,μ) ≤ 2K(n,r,μ)`.

The standard tool for lower bounds here is the **covering excess** of Johnson and van Wee: count how much the balls around codewords overlap, and convert the surplus into a bound. [HHKL] extended the method to multiple coverings. Our observation is that for **even n** a parity constraint is available that their excess term does not see, and that it is exactly the case their method misses (§7).

Results

Theorem 1 (§3). For even `n`, `K(n,1,2) ≥ ⌈3·2^(n+1)/(3n+2)⌉`.

Theorem 2 (§4). For every even `n ≥ 6`, the bound of Theorem 1 is strictly greater than [KP, Theorem 6] at `μ = 2`, ceilings included.

Corollary 3 (§5). `K(6,1,2) = 20`, with no search; and `K(8,1,2) ≥ 60`.

Proposition 4 (§6). For even `n` and even `μ`, `K(n,1,μ) ≥ ⌈μ(μ+1)2^n / ((μ+1)(n+1) − 1)⌉`. At `μ ≥ 4` this is weaker than [KP]; the method wins only at `μ = 2`.

Theorem 5 (§8). `K(8,1,2) ≥ 61`. Formalised in Lean 4.

Theorem 6 (§9). Every twofold covering `C ⊆ F_2^8` with `|C| ≤ 63` has trivial automorphism group in `Aut(Q_8) = F_2^8 ⋊ S_8`.

2. Notation

For `C ⊆ F_2^n` put

```
c(y) = |B(y) ∩ C|,     g(y) = c(y) − 2,     E = Σ_{y ∈ F_2^n} g(y).
```

If `C` is a twofold covering then `g ≥ 0`. Counting incidences `(y, z)` with `z ∈ C` and `y ∈ B(z)` in two ways gives `Σ_y c(y) = (n+1)|C|`, hence

```
E = (n+1)|C| − 2^(n+1).                                          (2.1)
```

Let `S = {y : g(y) ≥ 1}` be the **over-covered set** and `s = |S|`. Since `g` is a non-negative integer vanishing off `S`,

```
s ≤ E.                                                            (2.2)
```

We use the ball-intersection numbers of the cube: for `x ≠ z`,

```
|B(x) ∩ B(z)| = 2 if d(x,z) ∈ {1,2},  and 0 if d(x,z) ≥ 3,     (2.3)
```

with `|B(x) ∩ B(x)| = n+1`. (For `d = 1` the common points are `x` and `z`; for `d = 2` they are the two words between them.)

3. Theorem 1 and its proof

> **Theorem 1.** Let `n` be even and let `C ⊆ F_2^n` be a set with `|B(x) ∩ C| ≥ 2` for every `x ∈ F_2^n`. Then `|C| ≥ 3·2^(n+1)/(3n+2)`, and hence `K(n,1,2) ≥ ⌈3·2^(n+1)/(3n+2)⌉`.

Proof. **Step 1 (parity).** Fix `x ∈ C` and sum the coverage over its ball. Exchanging the order of summation and using (2.3),

```
Σ_{y ∈ B(x)} c(y) = Σ_{z ∈ C} |B(x) ∩ B(z)| = (n+1) + 2·N₂(x),
```

where `N₂(x) = #{z ∈ C : 1 ≤ d(x,z) ≤ 2}`. Because `n` is even, `n+1` is odd, so this sum is **odd**. It is a sum of `n+1` terms each at least 2, so it is at least `2(n+1)`; being odd it is at least `2(n+1) + 1`. Therefore

```
Σ_{y ∈ B(x)} g(y) ≥ 1,
```

so every codeword `x` has at least one `y ∈ B(x)` with `g(y) ≥ 1`, i.e. with `y ∈ S`.

Step 2 (incidence count). Count pairs `(x, y)` with `x ∈ C`, `y ∈ S`, `y ∈ B(x)`. By Step 1 each `x ∈ C` occurs at least once, and each `y ∈ S` occurs exactly `|B(y) ∩ C| = c(y)` times. Hence

```
|C| ≤ Σ_{y ∈ S} c(y) = Σ_{y ∈ S} (g(y) + 2) = E + 2s.
```

Step 3. By (2.2), `s ≤ E`, so `|C| ≤ 3E`. Substituting (2.1),

```
|C| ≤ 3(n+1)|C| − 3·2^(n+1)   ⟹   3·2^(n+1) ≤ (3n+2)|C|. ∎
```

The parity step is load-bearing. For odd `n` the conclusion is false. In `F_2^3` the code `C = {000, 001, 110, 111}` has `c(x) = 2` for every `x`, so it is a *perfect* twofold covering with `E = 0`, and `|C| = 4 = K(3,1,2)`, whereas the formula would demand `⌈48/11⌉ = 5`. (`odd_n_witness.py`.)

The set hypothesis is load-bearing. With repeated codewords the diagonal term in Step 1 need not have the stated parity.

4. Theorem 2: strict dominance for every even n ≥ 6

[KP, Theorem 6] gives, with `μ ≡ τ (mod 2)`, `τ ∈ {0,1}`,

```
(a) K(n,1,μ) ≥ 2^n(μn + 3μ + τ)/(n(n+4))     for n ≡ 0 (mod 4),
(c) K(n,1,μ) ≥ 2^n(μn + μ + τ)/(n(n+2))      for n ≡ 2 (mod 4).
```

Write `L(n) = 6·2^n/(3n+2)` for the bound of Theorem 1 at `μ = 2`, and `KP(n)` for the corresponding right-hand side above with `μ = 2, τ = 0`.

> **Theorem 2.** For every even `n ≥ 6`, `⌈L(n)⌉ > ⌈KP(n)⌉`.

Proof. Cross-multiplying gives the exact identities

```
L − KP_a = 2^n(2n − 12) / [n(n+4)(3n+2)]      (n ≡ 0 mod 4),
L − KP_c = 2^n(2n −  4) / [n(n+2)(3n+2)]      (n ≡ 2 mod 4),
```

so the sign is decided by `2n − 12` and `2n − 4`: `L > KP` for all even `n ≥ 6`. (At `n = 4`, `KP` is the larger real number, but both ceilings are 7, so no integer bound is lost.)

The real inequality does not immediately give the integer one, and the gap is genuinely small where it matters: it is `8/15` at `n = 6` and `16/39` at `n = 8`, both **below 1**. Those two cases are checked directly (`20 > 19`, `60 > 59`). For `n = 10` the gap is `64/15 > 1`. For even `n ≥ 12`, using `2n − 12 ≥ n/2` (valid for `n ≥ 8`) and `n(n+4)(3n+2) ≤ 8n³` (valid for `n ≥ 4`),

```
L − KP ≥ 2^n(n/2)/(8n³) = 2^n/(16n²) > 1,
```

since `2^n > 16n²` for all `n ≥ 11` and `2^n/n²` is increasing. A gap exceeding 1 forces `⌈L⌉ ≥ L > KP + 1 > ⌈KP⌉`. ∎

Asymptotics. Both bounds are `~2^(n+1)/n`, so their ratio tends to 1 and the improvement looks negligible. It is not: the *additive* gap is `~(2/3)·2^n/n²`, which is unbounded. Over even `n ∈ [6, 200]` the improvement survives the ceiling at **98 of 98** values.

n	published lower bound	Theorem 1	gain
6	20 (exact, [Seu] by IP)	**20**	0 (matched search-free)
8	59 [KP]	**60**	+1
10	188	**192**	+4
12	640	**647**	+7
14	2195	**2235**	+40
16	7783	**7865**	+82

The `n ≥ 8` entries are exactly the `μ = 2` positions [KP] print, and each of their printed lower bounds equals their formula value, so no tabulated entry is stronger than the formula (`dominance.py` asserts this).

5. Consequences at small n

`K(6,1,2) = 20`. Theorem 1 gives `⌈384/20⌉ = 20`, and a 20-word twofold covering of `Q_6` exists. The published value is also 20 but was obtained by integer programming and exhaustive search [Seu]; Theorem 1 proves it in one line. Likewise the weight-parity refinement of §8 gives `K(4,1,2) = 8` without search, where [HHKL, Theorem 1] used a weight-distribution case analysis.

`K(8,1,2) ≥ 60`. Theorem 1 gives `⌈1536/26⌉ = 60`, one above the published 59. This package establishes the same bound by two further independent routes:

1. an exact-rational SCIP → VIPR certificate refuting `|C| = 59`, accepted by the standalone checker `viprchk` (3.59 GB), archived at doi:10.5281/zenodo.22217672;
2. agreement of two independent floating-point solvers (evidence, not a certificate).

Hence `60 ≤ K(8,1,2) ≤ 64`.

6. Even multiplicities, and why the method wins only at μ = 2

Carrying `μ` through the proof, Step 1 needs the ball sum `(n+1) + 2N₂(x)` to exceed `μ(n+1)` by rounding, which happens exactly when `μ(n+1)` is even — for even `n`, exactly when `μ` is **even**. The same three steps then give:

> **Proposition 4.** For even `n` and even `μ`, `K(n,1,μ) ≥ μ(μ+1)2^n / ((μ+1)(n+1) − 1)`.

At `μ = 2` this is Theorem 1. At `n = 8`: `μ = 4` gives 117 against [KP]'s 118; `μ = 6` gives 174 against 176; `μ = 8` gives 231 against 235. **So this is not a uniformly better bound.** It wins at `μ = 2` and loses at every larger even `μ`, with the gap widening. Any statement of these results must say so.

7. Relation to previous work

The excess method is due to Johnson and to van Wee; van Wee's 1988 paper treats *ordinary* coverings `K(n,R)` and its even-`n`, radius-one consequence is `K(n,1) ≥ 2^n/n`. It should **not** be cited as proving the present bound.

[HHKL] extended excess counting to multiple coverings. Their Theorem 6 reads

```
K(n,r,μ) ≥ (μ(n+1−k) + ε)2^n / ((n+1−k)V(n,r) + ε V(n,r−1)),
ε := (r+1)⌈μ(n+1)/(r+1)⌉ − μ(n+1),
```

and `ε` is their entire gain over the sphere-covering bound. At `r = 1`, `ε = 0` exactly when `μ(n+1)` is even — for even `n`, exactly when `μ` is **even** — and the bound then collapses *identically* to the sphere bound. This is why their Corollary 2 is stated only for "`μ ≤ n` odd and `n` even": their parity term has nothing to say at even `μ`. Consistently, **no `μ = 2` entry in their own table is marked as coming from Theorem 6**; their `μ = 2` values are the sphere bound at `n = 6, 10, 12, 16` and their weight-split inequalities at `n = 8, 14`. (`hhkl_theorem6.py` verifies the collapse as exact rational equality for all even `n ≤ 40` and even `μ < n`.)

The present argument uses a *different* parity — the ball sum at a **codeword**, odd whenever `n` is even, for every `μ` — and therefore fires precisely where theirs vanishes.

Formalisation. Covering codes have been formalised in Lean 4 before, and recently. [Flo1] gives certificate predicates for upper bounds, lower bounds and exact values of `K_q(n,r)`, with an end-to-end checking workflow for bounds transcribed from the literature. [Flo2] goes further and settles an exact value, `K_8(4,2) = 23`, in Lean — a lower bound argument combining fiber counting with two Lean-checked LRAT refutations of stored CNF instances, packaged as a proof-carrying artifact. That is methodologically the nearest neighbour to §8: same proof assistant, same use of checked SAT refutations, same artifact-first posture.

Both works treat **ordinary** covering codes (`μ = 1`) only. So the claim we can defend for §8 is narrower than "the first formal treatment of covering codes", and narrower than "the first Lean-certified covering-code value": it is that we are not aware of a prior formalisation of *multiple* covering codes, nor of a formal proof establishing a bound that **improves** the published literature rather than reproducing or settling a value within it.

[Seu] is not a general-theorem paper: it improves 57 individual positions by integer programming and exhaustive search. Its `n = 8`, `μ = 2` entry is 58 and is unmarked, i.e. copied from the tables of [CHLL], so it did not improve that cell.

Priority disclosure — and it is sharper than "we could not find it". [HHKL] cites *two* forthcoming Chen–Li papers. Reference [29], "*New lower bounds for binary covering codes*", was published as [LC] in 1994; we have read it, and it treats **ordinary** coverings `K(n,R)` only, introducing a "multiexcess" generalisation of van Wee's method. It states that "in a forthcoming paper [6], this idea is applied to multiple covering codes" — and its reference [6] is [HHKL] itself, so the two groups cite each other as forthcoming.

The paper we cannot locate is [HHKL, ref. 2], *W. Chen and D. Li, "Lower bounds for multiple covering codes*", a **different** work that appears never to have been published. What matters is where [HHKL] places it. Immediately after its Theorem 6 and Corollary 2 — Corollary 2 being the odd-`μ`, even-`n` case, which is precisely the case complementary to ours — [HHKL] writes:

> "Chen and Li have independently presented similar definitions and results, and also many further results, see [2]."

So the risk is not that some unrelated preprint might overlap. It is that a located, named, unpublished manuscript is described by [HHKL] as containing similar results *plus many further ones*, at exactly the point in the argument that this paper improves. Whether those further results include the even-`μ` case is unknown to us. We therefore claim only that we have **not found** this specialisation in the published

literature, and we do not claim it is new. Separately, [KP] already use Delsarte nonnegativity on a covering code's own distance distribution, so that ingredient is standard for this table.

8. Theorem 5: `K(8,1,2) ≥ 61`, formalised in Lean 4

Splitting `F_2^n` by weight parity gives two more valid rows, `M_e + n·M_o ≥ 2^n` and `M_o + n·M_e ≥ 2^n`, whose per-half excesses kill `K(4,1,2) = 7` and `K(6,1,2) = 19` outright (`bipartite_split.py`), and reduce the `n = 8`, `M = 60` case to a weight-balanced `(30,30)` code whose odd half is not a translate of its even half.

`m61_refutation.py` closes `M = 60` completely. The chain combines Theorem 1's Step 1 with a Delsarte/sum-of-squares bound on the distance distribution, a second-moment identity forcing excess *concentration*, an integrality argument forcing a fully-covered word, and a weighted layer count that ends in a Farkas contradiction (demand 2658 against supply 2553). Delsarte nonnegativity is used in sum-of-squares form, which avoids having to formalise the MacWilliams transform, and it is applied to the code's own distance distribution — valid for an arbitrary subset of `F_2^n`, so no structure is assumed.

8.1 The formal statement

The argument is formalised in Lean 4 with mathlib, with **no `sorry` and no `native_decide`**. The headline theorem and its audited axiom trace:

```
le_card_of_isDoubleCover : ∀ (C : Finset V), IsDoubleCover C → 61 ≤ C.card
  depends on axioms: [propext, Classical.choice, Quot.sound]
```

Those three axioms are the ordinary foundations of mathlib; the absence of `sorryAx` is what matters. Reproduce with

```
cd lean && lake exe cache get && lake build   # ~4 minutes warm
lake build Mcov.Audit                          # axiom trace, all 39 declarations
lake build Mcov.Sanity                         # independent non-vacuity checks
```

8.2 Why the formalisation is not vacuous

A formal statement can be true because it says nothing. Three checks, in `Mcov/Sanity.lean`, written separately from the development, rule that out: `Fintype.card V = 256` (the space really is `F_2^8`), `univ_isDoubleCover` (the predicate is **satisfiable**, so the theorem is not vacuously true), and `empty_not_isDoubleCover` (nor is it trivially true). All are axiom-clean.

The step from "no 60-word twofold covering exists" to `K(8,1,2) ≥ 61` is monotonicity: a subset of a twofold covering that is too small cannot itself be one. This is stated as `cov_mono` and `isDoubleCover_mono`. It is easy, and it is exactly the step an informal write-up is most likely to leave implicit — two independent reviews flagged it as missing before it was formalised.

8.3 What a sceptical reader should check

The trusted base is Lean's kernel plus the **four** definitions that the theorem statement mentions, transitively:

```
abbrev V : Type := Finset (Fin 8)
def dist (x y : V) : ℕ := (symmDiff x y).card
def cov (C : Finset V) (v : V) : ℕ := (C.filter (fun c => dist c v ≤ 1)).card
def IsDoubleCover (C : Finset V) : Prop := ∀ v : V, 2 ≤ cov C v
```

`Mcov/Basic.lean` defines eleven further objects (`g`, `E`, `S`, `a`, `chi`, `W`, `f`, `ballPt`, `hval`, `gz`, `m`), but none of them occurs in the statement of `le_card_of_isDoubleCover`. They are proof machinery: if one of them were wrong, the development would either fail to typecheck or still prove the stated theorem, because the kernel checks the proof term against the stated type. So they are outside the trust boundary, and a reader need not check them. The four above are the whole of what human judgement has to settle; everything after that is the kernel's problem.

The author has now read these four definitions (2026-09-03), against an audit worksheet that pairs each with its plain reading, worked examples, and the rival readings that would change the object (`= 2` and `≤ 2` for `2 ≤`, `∃` for `∀`). As a cross-check the four were transliterated into ordinary code and run exhaustively at `n = 4`, where they reproduce the published `K(4,1,2) = 8` in both directions: no 7-word twofold covering exists, and an 8-word one does. Definitions that did not encode the intended object would not land on a published value.

9. Theorem 6: every `≤ 63` code is asymmetric

`Aut(Q_8) = F_2^8 ⋊ S_8` (order `256 · 8! = 10 321 920`) acts by `x ↦ π(x) ⊕ t`. Any non-identity element has a power of prime order, and a code invariant under a group is invariant under each of its subgroups, so it suffices to rule out invariance under prime-order elements. The prime orders available are 2, 3, 5, 7, and the corresponding signed-cycle-type classes number 28 — verified rather than assumed, by enumerating all 185 signed cycle types of `B_8` and checking exactly 28 have prime order.

Restricting to codes invariant under `⟨g⟩` turns the 256-variable problem into one variable per orbit. All 28 classes are infeasible at `|C| ≤ 63`. An independent 746-class cyclic sweep (`prescribed.py`), a separate implementation, agrees on every verdict; its accounting closes exactly (742 swept plus 3 dropped by an orbit filter is all 745 non-identity classes).

Caveat, stated plainly. Only **4 of the 28** are machine-certified — the order-3, -5 and -7 classes, via VeriPB cutting planes or exact-rational VIPR. The 24 order-2 classes are floating-point solver verdicts; both certification routes blow up on them, one passing 850 MB of proof on an instance a floating-point solver settles in seconds. So Theorem 6 is on a materially weaker footing than Theorem 5, and should be read as such.

`symmetry_parity.py` narrows what remains. An order-2 `g = (π, t)` acts freely exactly when its signed type has `c > 0`, since then `t ∉ im(1+π)` and `x ⊕ π(x) = t` is unsolvable; every orbit then has size 2, so an invariant code has **even** size. That is 20 of the 24, and combined with Theorem 5 it forces `|C| = 62` exactly — one equality instance instead of an inequality spanning three sizes. Note this makes the narrowed route *depend* on Theorem 5, whereas the original `≤ 63` instances do not; both are kept.

Why it matters. Any code witnessing `K(8,1,2) ≤ 63`, if one exists, is completely asymmetric. Every classical construction in this area is symmetric, which explains why the upper bound has not moved since 1995, and tells a future search to break symmetry aggressively rather than prescribe it.

10. What is and is not established

Established, by hand — Theorems 1 and 2, Corollary 3, Proposition 4, and `K(8,1,2) ≥ 60`. Each is reproduced by a self-asserting script (`./reproduce.sh`, twelve checks, standard library except for one `scipy` control step), and `K(8,1,2) ≥ 60` additionally by a machine-checkable exact-rational certificate accepted by `viprchk`.

Established, machine-verified — Theorem 5, `K(8,1,2) ≥ 61`, by a Lean 4 development with zero `sorry`s and an audited axiom trace. Hence `61 ≤ K(8,1,2) ≤ 64`.

Established, but only partly certified — Theorem 6. 4 of 28 classes are machine-certified; the remaining 24 rest on floating-point solver verdicts.

Not established — the *novelty* of Theorem 1 (§7): Krotov–Potapov already use Delsarte nonnegativity on a covering code's own distance distribution, and the Chen–Li preprint is unlocated. Nothing here touches the upper bound, which remains 64. And while the author has audited the four Lean definitions the result depends on, **no human has reviewed the proofs**, which rest on Lean's kernel alone.

Declarations

Use of generative AI

Generative AI was used extensively in producing this work. Large language models were used for literature search, for proposing the `M = 60` argument of §8, for writing most of the code and the Lean development, and for adversarial review of the results.

Tools. Anthropic Claude (Opus 5) for most of the code, the Lean development and drafting; Anthropic Fable 5.1 for pre-publication audit; OpenAI GPT-5.6 (ChatGPT, and Codex `gpt-5.6-terra`) and DeepSeek v4 for independent review and prior-art search. Review transcripts are archived in `reviews/`.

No AI system is an author of this work. The sole author is Hanyu Yang, who directed the work, verified the results, and takes full responsibility for the content, including any part drafted or proposed with AI assistance.

This is disclosed prominently rather than in a footnote because it bears on how the claims should be read — and because the package is built around the consequence. Machine reasoning is not treated as evidence anywhere in it. The review transcripts in `reviews/` are archived as provenance only. Every load-bearing claim is one of: an argument short enough to check by hand; a self-asserting script; a machine-checkable certificate; or, for `K(8,1,2) ≥ 61`, a Lean 4 development whose axiom trace is audited. The reason for formalising rather than seeking machine review is recorded in `lean/README.md`: two machine reviews called the argument sound while between them making four errors on it, and catching *disjoint* defects.

Human review

No human has reviewed the proofs, including for the results this paper presents as established, and the paper has had no peer review.

The **definitions** are a separate matter, and the distinction is the load- bearing one. A development with zero `sorry`s and a clean axiom trace proves the formal statement *from the formal definitions*; it is not evidence that those definitions encode the intended object, and that is the one link with no machine on it. For `K(8,1,2) ≥ 61` the trust boundary is four definitions — `V`, `dist`, `cov`, `IsDoubleCover` — the

transitive closure of what the theorem statement mentions (§8). **The author read all four on 2026-09-03** and confirmed each encodes its intended meaning, aided by the `n = 4` reproduction of the published `K(4,1,2) = 8`. The remaining eleven definitions in `Mcov/Basic.lean` are proof machinery outside that boundary.

So: the definitions have been checked by a person; the proofs have not, and rest on Lean's kernel.

Competing interests

The author declares no competing interests.

Funding

No funding was received for this work.

Data and code availability

All code, certificates, logs and the Lean development are openly available.

- Repository: https://github.com/SeverinVisionary/oeis-a004045
- Archive: DOI 10.5281/zenodo.22217672 (concept DOI; resolves to the current version)

Code is MIT-licensed; prose, tables and certificates are CC BY 4.0. Results are reproduced by `./reproduce.sh` (twelve self-asserting checks, about a minute, non-zero exit on any failure); the Lean development is a separate build of about four minutes.

References

- [CHLL] G. Cohen, I. Honkala, S. Litsyn, A. Lobstein. *Covering Codes*. North-Holland, 1997.
- [HHKL] H. O. Hämäläinen, I. S. Honkala, M. K. Kaikkonen, S. Litsyn. Bounds for binary multiple covering codes. *Des. Codes Cryptogr.* **3** (1993) 251–275. doi:10.1007/BF01388486
- [KP] D. S. Krotov, V. N. Potapov. On multifold packings of radius-1 balls in Hamming graphs. *IEEE Trans. Inform. Theory* **67**(6) (2021) 3585–3598. doi:10.1109/TIT.2020.3046260, arXiv:1902.00023
- [LC] D. Li, W. Chen. New lower bounds for binary covering codes. *IEEE Trans. Inform. Theory* **40**(4) (1994) 1122–1129. doi:10.1109/18.335963
- [Flo1] A. Florath. Formal foundations and proof-carrying certificates for q-ary covering codes in Lean 4. arXiv:2606.09600, 2026.
- [Flo2] A. Florath. A Lean-certified proof of `K_8(4,2) = 23`. arXiv:2606.16688, 2026.
- [Seu] E. A. Seuranen. New lower bounds for multiple coverings. *Des. Codes Cryptogr.* **45** (2007) 91–94. doi:10.1007/s10623-007-9089-y
- [vW] G. J. M. van Wee. Improved sphere bounds on the covering radius of codes. *IEEE Trans. Inform. Theory* **34** (1988) 237–245. doi:10.1109/18.2632
- OEIS Foundation. Sequence A004045. https://oeis.org/A004045
- H. Yang. Machine-checked bounds on K(8,1,2), the first unknown term of OEIS A004045. Zenodo, 2026. doi:10.5281/zenodo.22217672 (concept DOI; always resolves to the current version)