PALEY GRAPHS OF CAPTURE DEPTH SEVEN: THE TWO-BASEPOINT CAPTURE DEPTH BEYOND $p = 250$

KOREA SUPERINTELLIGENCE LABS

Abstract. To a prime p and a subgroup $E \leq \mathbb{Z}_p^\times$ containing -1, Marashdeh attaches a nested chain of subspaces $V_0 \subseteq V_1 \subseteq \cdots$ of $\mathbb{C}^p$, generated from the two point masses δ_0, δ_1 by the class matrices of the cyclotomic scheme and the diagonal idempotents of the basepoint partition, and calls the least d for which V_d contains a point mass δ_z with $z \notin \{0,1\}$ the *capture depth* $d(p,E)$; one capture, at any depth, forces every circulant graph of order p with multiplier group E to have no quantum symmetry. His table of depths covers all 214 pairs with $p \leq 250$; along the Paley graphs P_p it reads $1,2,3,4,5,6$ on $p = 5$, 13, 17, 29–61, 73–181, 193–241, and he asks whether the Paley depth is unbounded. We compute the first values beyond that range: $d(P_{521}) = 7$, the first capture depth seven on record, and $d(P_{509}) = 6$; every Paley prime $257 \leq p \leq 509$ has depth at most 6; and 521 is the least Paley prime of depth exceeding six, the list of the 45 Paley primes below 521 being itself part of the theorem. Each statement is certified two-sidedly and exactly over $\mathbb{Q}$: membership in V_d by an explicit integer combination of generator-word vectors, non-membership in V_{d-1} by a graded spanning certificate for V_4 together with one integer functional whose adjoint-word images annihilate it. The certificate checkers are proved sound against the definition of the filtration and the checks are carried out by a compiled evaluator; everything is machine-checked in Lean 4 with no `sorry`. Outside the formal development, and labelled as such, a modular computation reproduces the source's whole table, extends the Paley row to $p \leq 1621$ (depth 7 on 521–1453, depth 8 from 1481), and records that $\dim V_j = 2 \cdot 3^{j-1} + 4$ independently of p wherever the filtration has not saturated, so that the depth grows like $\log_3 p$.

1. Introduction

The objects. Fix a prime $p \geq 5$ and a subgroup $E \leq \mathbb{Z}_p^\times$ with $-1 \in E$; put $r = (p-1)/|E|$, let $C_1 = E, C_2, \ldots, C_r$ be the cosets of E in $\mathbb{Z}_p^\times$ and $C_0 = \{0\}$. The *class matrices* act on $\mathbb{C}^p = \mathbb{C}[\mathbb{Z}_p]$ by $(T_s v)(x) = \sum_{c \in C_s} v(x-c)$, so $T_0 = I$, and their span $\mathfrak{B}$ is the Bose–Mesner algebra of the cyclotomic scheme. Marashdeh [1] fixes the two *basepoints* 0 and 1 and forms the *two-basepoint Terwilliger algebra* $\mathfrak{T}(0,1)$ generated by $\mathfrak{B}$ and the diagonal idempotents $E_s^*(a) = \mathrm{diag}(\chi_{a+C_s})$, $a \in \{0,1\}$, $0 \leq s \leq r$, together with the module $W(0,1) = \mathfrak{T}(0,1)\delta_0 + \mathfrak{T}(0,1)\delta_1$ [1, Definition 4.1]. The products $E_s^*(0)E_t^*(1)$ are the indicators of the blocks $B_{s,t} = C_s \cap (1 + C_t)$ of the *basepoint partition* [1, Definition 4.2]; write $\mathfrak{D}$ for the algebra of diagonal matrices constant on those blocks. The *convolution filtration* is, in the source's words (equation (4) there),

$$V_0 := \mathrm{span}\{\delta_0, \delta_1\}, \qquad V_d := \mathfrak{D}\,\mathfrak{B}\,V_{d-1} \quad (d \geq 1),$$

so that V_d increases with d and $W(0,1) = \bigcup_d V_d$. The quantity this paper is about is [1, Definition 5.14], quoted verbatim: "The *capture depth* $d(p,E)$ is the least $d \geq 1$ for which V_d contains a point mass δ_z with $z \notin \{0,1\}$". We write P_p for the Paley graph, the case $E = \mathrm{QR}$ of the quadratic residues, $r = 2$, $p \equiv 1 \pmod 4$, and $d(P_p)$ for $d(p, \mathrm{QR})$.

Why the depth matters. The depth is a finite piece of linear algebra, but what it certifies is operator-algebraic. Theorem 4.7 of [1] (Theorem A of its introduction) states that if $W(0,1) = \mathbb{C}^p$ then every circulant graph of prime order p with multiplier group $E \lneq \mathbb{Z}_p^\times$ has no quantum symmetry, $\mathrm{Qut}(X) = \mathrm{Aut}(X)$; and its capture dichotomy, Theorem 5.8 (Theorem B), says that the set of z with δ_z in the reachable module $\mathcal{R}(0,1) \supseteq W(0,1)$ is either $\{0,1\}$ or all of $\mathbb{Z}_p$, so that (Corollary 5.9 there) *one* point mass δ_z with $z \notin \{0,1\}$ in $W(0,1)$, at any depth, already gives $\mathcal{R}(0,1) = \mathbb{C}^p$ and quantum rigidity. Chassaniol's orbital criterion [4] is identified as the depth-one case (Corollary 5.10 there).

That bridge to $\mathrm{Qut}(X) = \mathrm{Aut}(X)$ is cited, not reproved and not formalised here: this paper is about $d(p, E)$ as a function on the Paley family, which is a statement about finite-dimensional subspaces of $\mathbb{Q}^p$ and nothing else.

Where the source stops. Theorem 8.1 of [1] (Theorem F of its introduction is Theorem 8.1 together with Corollary 8.2; the numbering throughout is that of the compiled e-print, see the bibliography) reads verbatim: "For every prime $5 \le p \le 250$ and every proper subgroup $E \le \mathbb{Z}_p^\times$ with $-1 \in E$ one has $W(0,1) = \mathbb{C}^p$. The number of such pairs (p, E) is 214." Table 1 of its Appendix A lists $d(p, E)$ and the least captured vertex for all 214 pairs; §8.2 there observes that every pair of depth at least four is a Paley graph and prints the Paley row

p	5	13	17	29–61	73–181	193–241
$d(P_p)$	1	2	3	4	5	6

The source computes no pair with $p > 250$, in the paper or in its data archive [2] (see §1), and the growth of that row is the evidence behind the open question of its §9.1, the subsection "What bounded depth cannot do" (which shares its number with Conjecture 9.1, the two counters being independent), quoted verbatim with the cross-references resolved to the compiled numbering: "More generally, if the Paley depth is unbounded — as the data suggest — then no criterion at a fixed depth can settle Conjecture 9.1, and in particular no strengthening of Corollary 5.10 from depth one to depth d, for d fixed, can do so." Conjecture 9.1 is the capture conjecture (for every $p \ge 5$ and proper $E \ni -1$ some δ_z, $z \notin \{0,1\}$, lies in $\mathcal{R}(0,1)$) and Conjecture 9.3 its harmonic form ($\delta_{2^{-1}} \in \mathcal{R}(0,1)$); both remain open.

One more sentence of the source governs the form of everything below. Its Lemma 8.3 shows that a *spanning* statement may be proved modulo a prime q (a family of integer vectors whose reductions span $\mathbb{F}_q^N$ spans $\mathbb{Q}^N$), and the paragraph after it says, verbatim, that "a capture is a *membership* assertion, for which the one-sided argument of Lemma 8.3 is unavailable: a vector may lie in the reduction of a lattice modulo q without lying in its rational span." A capture depth therefore has to be computed in exact arithmetic, in both directions, and that is what the certificates of this paper do.

What is proved. All theorems below are about the Paley case $E = \mathrm{QR}$, over $\mathbb{Q}$ (§2 explains why nothing is lost against $\mathbb{C}$).

Theorem A (Theorems 4.1 and 4.2). $d(P_{521}) = 7$ and $d(P_{509}) = 6$. More precisely, for $p = 521$ some point mass δ_z with $z \notin \{0,1\}$ lies in V_7 and none lies in V_6; for $p = 509$ some such point mass lies in V_6 and none lies in V_5.

Theorem B (Theorems 4.3 and 4.4). Every one of the 21 primes $p \equiv 1 \pmod 4$ with $257 \le p \le 509$ has $d(P_p) \le 6$; for the 24 Paley primes $p \le 241$ the same certificates re-derive the upper half of the source's Paley row. The primes $p \equiv 1 \pmod 4$ with $5 \le p < 521$ are exactly 45 listed primes, each of which is captured by depth 6, while P_{521} is not; hence 521 is the least Paley prime of capture depth exceeding six, and the least of capture depth exactly seven.

Theorem C (Proposition 5.1). For $p = 5$, 13, 17, 29, 37, 41, 53, 61, 73 the capture depths are 1, 2, 3, 4, 4, 4, 4, 4, 5, certified two-sidedly from the definition of the filtration. These are the entries of Table 1 of [1].

Theorem C is the reason to believe the reading of the definitions behind Theorems A and B, and it was run first: a filtration transcribed with the wrong blocks, the wrong classes or the wrong seeds would not return the published nine values. Section 5 adds the complementary evidence that neither half of the certificate format is vacuous.

What the computation is. A capture depth $d(P_p) = d$ is two inequalities, and the two halves have different shapes (Section 3). The upper half, $\delta_z \in V_d$, is a list of *generator words*: vectors $\chi_B \odot T_s u$ built level by level from δ_0, δ_1, together with an integer combination of them equal to $m\,\delta_z$ with $m \ne 0$. The lower half, $\delta_z \notin V_{d-1}$ for every $z \notin \{0,1\}$, needs an upper bound on a space, and is a *graded*

spanning certificate — a store of integer vectors and a chain of sizes $2 = k_0 \le k_1 \le \cdots \le k_t$ such that every generator image of a stored vector indexed below k_j is an explicit rational combination of stored vectors indexed below k_{j+1}, so that $V_t \subseteq \mathrm{span(store)}$ — together with a single integer functional f all of whose images under k adjoint generator steps are orthogonal to the store, so that $f \perp V_{t+k}$, and which is nonzero at every vertex outside $\{0, 1\}$. The split $d - 1 = t + k$ is what makes $p = 521$ affordable: with $t = 4$ the store has 58 vectors at both $p = 509$ and $p = 521$, and $k = 2$ costs $27^2 = 729$ adjoint words. Both halves are checked in exact integer arithmetic by a compiled evaluator, and the checkers are proved sound against the definition of V_d (Propositions 3.2 and 3.4); those proofs, the monotonicity of the filtration and the adjoint identity behind the lower half are ordinary reasoning, carried out in full and machine-checked (Section 8).

The table below says, for each Paley prime $p \le 521$, which half of $d(P_p)$ is certified here and where the other half comes from.

range of p	upper bound $d(P_p) \le d$	lower bound $d(P_p) \ge d$
$p \le 73$ (9 primes)	exact, here (Prop. 5.1)	exact, here (Prop. 5.1)
$89 \le p \le 241$ (15 primes)	exact, here (Thm. 4.3)	exact, [1, Table 1]
$257 \le p \le 461$ (20 primes)	exact, here (Thm. 4.3)	modular only (Remark 6.2)
509, 521	exact, here (Thms. 4.2, 4.1)	exact, here (Thms. 4.2, 4.1)

Provenance. The values $d(P_{509}) = 6$ and $d(P_{521}) = 7$, and the minimality of 521, are not in [1]: its Theorem 8.1, Table 1 and §8.2 stop at $p \le 250$, and its data archive [2] was downloaded and opened on 7 September 2026 — the file `depths.json` there holds exactly 214 rows, each a quadruple $(p, |E|, d(p, E), \min Z(p, E))$, with largest $p = 241$, and the script `compute_depths.py` loops `for p in primerange(5, 251)`, so no larger sweep exists in the archive either. Its 24 Paley rows agree, in both the depth and the least captured vertex, with the computation of §6. The abstract page of arXiv:2609.00462 showed version 1 only on that date. What else was searched, on the same date: OpenAlex resolves the work by its DataCite DOI to a record with zero citations, and a search for works citing it returns none; the arXiv full-text search for "capture depth" together with "Paley" returns no entry; and in a local full-text corpus of arXiv papers the phrase "capture depth" occurs only in computer-vision papers, "two-basepoint" only in Heegaard–Floer papers, and the only texts pairing a Paley graph with quantum symmetry are the source itself and Schmidt's [5], which records (verbatim, with his two citation keys resolved to the present bibliography) that "It was already shown in [6] that the Paley graph P_9 has no quantum symmetry and in [4] that P_{13} and P_{17} have no quantum symmetry" — rigidity statements, never a depth. The oldest paper the source cites for the problem, Banica–Bichon–Chenevier [3], proves (in the source's words) that a circulant p-graph of type $k = |E|$ "has no quantum symmetry whenever $p > 6^{\varphi(k)}$"; for a Paley graph $k = (p-1)/2$, so that threshold never applies, and the paper prints no depths. Chassaniol [4] settles all types $k \le 8$, again a rigidity statement. We read these negatives as "not found", not as a proof of novelty; the positive part of the claim is that the source is days old, has no citing work, and computes nothing beyond $p = 250$.

Remark 1.1 (a consequence through the source's theorems; not part of the formal development). By Corollary 5.9 of [1], applied to $E = \mathrm{QR}$, each capture certificate of Theorems A and B certifies on its own that the corresponding Paley graph has no quantum symmetry. The captures at the 22 Paley primes $257 \le p \le 521$ are therefore 22 Paley graphs, beyond the $p \le 241$ of [1, Corollary 8.2], for which $\mathrm{Qut}(P_p) = \mathrm{Aut}(P_p)$. This uses the source's Theorems 4.7 and 5.8, which we have not formalised; nothing below depends on it.

2. Preliminaries

2.1. **Classes, blocks, generators.** Throughout, $p \equiv 1 \pmod 4$ is prime and $E = \mathrm{QR}$. For $x \in \mathbb{Z}_p$ put $\mathrm{cls}(x) = 0, 1, 2$ according as $x = 0$, x is a nonzero square, x is a non-square, so $C_s = \{x : \mathrm{cls}(x) = s\}$. Since $p \equiv 1 \pmod 4$, -1 is a square and $-C_s = C_s$. The blocks of the basepoint partition are labelled by $(\mathrm{cls}(x), \mathrm{cls}(x-1))$; the block $B_{0,1} = \{0\}$ and $B_{1,0} = \{1\}$ are singletons, and the four blocks $B_{s,t} = C_s \cap (1 + C_t)$ with $s, t \in \{1, 2\}$ have sizes $(p-5)/4$ for $(s, t) = (1, 1)$ and $(p-1)/4$ otherwise —

the cyclotomic numbers of order two. Thus there are six blocks for $p \geq 13$ and five for $p = 5$, where $B_{1,1}$ is empty; for $p = 5$ all five blocks are singletons and $d(P_5) = 1$ by [1, Lemma 5.13].

We work over $\mathbb{Q}$. For $s \in \{0, 1, 2\}$ and a block label b the maps

$$T_s v = \Big(x \mapsto \sum_{y \in C_s} v(x - y)\Big), \qquad \chi_b \odot v = \big(x \mapsto v(x) \text{ if } x \in B_b, \text{ else } 0\big)$$

are $\mathbb{Q}$-linear on $\mathbb{Q}^p = \mathbb{Q}^{\mathbb{Z}_p}$, and the 27 *generators* are $g_{s,b} = \chi_b \odot T_s$. (The label b runs over all nine pairs in $\{0, 1, 2\}^2$, as it does in the formal development; the three pairs that label no block give the zero map, so only 18 of the 27 generators are nonzero, and none of the counts below depends on which convention is used.) The filtration over $\mathbb{Q}$ is

$$V_0 = \operatorname{span}_{\mathbb{Q}}\{\delta_0, \delta_1\}, \qquad V_{n+1} = \sum_{s,b} g_{s,b}(V_n),$$

and we say p is *captured at depth* n, written $\mathrm{Cap}_p(n)$, if $\delta_z \in V_n$ for some $z \in \mathbb{Z}_p \setminus \{0, 1\}$.

Lemma 2.1 (the filtration over $\mathbb{Q}$ is the filtration over $\mathbb{C}$)**.** *Let $V_d^{\mathbb{C}}$ denote the space of equation (4) of* [1] *for $E = \mathrm{QR}$. Then $V_d^{\mathbb{C}} = \mathbb{C} \otimes_{\mathbb{Q}} V_d$, and for every z, $\delta_z \in V_d^{\mathbb{C}}$ iff $\delta_z \in V_d$.*

Proof. $\mathfrak{D}$ is spanned by the block idempotents and $\mathfrak{B}$ by T_0, T_1, T_2, so $\mathfrak{D}\mathfrak{B}V_{d-1}^{\mathbb{C}}$ is the complex span of the vectors $\chi_b \odot T_s v$, $v \in V_{d-1}^{\mathbb{C}}$, and induction on d gives the first claim. A rational vector lies in the complex span of rational vectors iff it lies in their rational span (the two spans have the same dimension over their respective fields, and the rational span embeds in the complex one), which is the second. This lemma is proved on paper; the formal development takes the rational filtration as its definition. □

Lemma 2.2 (monotonicity)**.** *$V_n \subseteq V_{n+1}$ for all n; hence $\mathrm{Cap}_p(m)$ implies $\mathrm{Cap}_p(n)$ for $m \leq n$, and $d(P_p) = d$ if and only if $\mathrm{Cap}_p(d)$ and not $\mathrm{Cap}_p(d-1)$.*

Proof. $T_0 = I$ and $\sum_b \chi_b \odot v = v$, so $v = \sum_b g_{0,b} v \in V_{n+1}$ for $v \in V_n$. The last claim is the definition of the least $d \geq 1$ together with monotonicity. (Since $\mathrm{Cap}_p(0)$ is false, the case $d = 1$ is included.) □

2.2. **The adjoint identity.** Let $\langle f, v\rangle = \sum_x f(x)v(x)$ on $\mathbb{Q}^p$ and $g_{s,b}^* = T_s \circ (\chi_b \odot -)$.

Lemma 2.3 (adjoint identity)**.** *For all $f, v \in \mathbb{Q}^p$, $\langle f, g_{s,b} v\rangle = \langle g_{s,b}^* f, v\rangle$.*

Proof. $\chi_b \odot -$ is symmetric for $\langle \cdot, \cdot \rangle$ trivially, and T_s is symmetric because

$$\langle f, T_s v\rangle = \sum_{y \in C_s} \sum_x f(x)v(x - y) = \sum_{y \in C_s} \sum_x f(x + y)v(x) = \langle T_s f, v\rangle,$$

the middle step being the substitutions $y \mapsto -y$ (which uses $-C_s = C_s$) and $x \mapsto x + y$. The only arithmetic input is that -1 is a square modulo p. □

Say f *kills* V_n if $\langle f, v\rangle = 0$ for all $v \in V_n$. Two consequences drive the lower certificates: if $g_{s,b}^* f$ kills V_n for all 27 generators then f kills V_{n+1} (since V_{n+1} is spanned by the $g_{s,b} v$ and $\langle f, g_{s,b} v\rangle = \langle g_{s,b}^* f, v\rangle = 0$); and if f kills V_n and $f(z) \neq 0$ then $\delta_z \notin V_n$ (since $\langle f, \delta_z\rangle = f(z)$).

3. The certificates and their soundness

All certificate data are integer vectors of length p, indexed by $\mathbb{Z}_p$ through $x \mapsto x \bmod p$, and lists of integers; the generators act on them by the integer versions of T_s and $\chi_b \odot -$.

Definition 3.1 (upper certificate)**.** A *generator-word tree* of d levels is a list of d lists of triples (i, s, b). Starting from the store $u_0 = \delta_0$, $u_1 = \delta_1$, each triple of level j appends $u = g_{s,b} u_i$ to the store, where i indexes an earlier entry. An *upper certificate for* $\mathrm{Cap}_p(d)$ is such a tree together with $z \in \mathbb{Z}_p \setminus \{0, 1\}$, an integer $m \neq 0$ and a list of pairs (i, c_i) with $\sum_i c_i u_i = m\,\delta_z$ as integer vectors.

Proposition 3.2 (soundness of the upper certificate)**.** *If an upper certificate for $\mathrm{Cap}_p(d)$ passes — that is, the integer identity $\sum_i c_i u_i = m\,\delta_z$ holds for the store built from its tree — then $\mathrm{Cap}_p(d)$.*

Proof. By induction on the level, every stored vector after j levels lies in V_j: the seeds lie in V_0, a stored vector lies in every later V_n by Lemma 2.2, and $g_{s,b}u_i \in V_{j+1}$ for $u_i \in V_j$ by definition of V_{j+1}. Hence $m\delta_z \in V_d$, and $\delta_z = m^{-1}(m\delta_z) \in V_d$ as $m \neq 0$. □

Definition 3.3 (graded spanning certificate and annihilator)**.** Let $\mathrm{vs} = (v_0, v_1, \dots)$ be a store of integer vectors with $v_0 = \delta_0$, $v_1 = \delta_1$, and let $2 = k_0 \le k_1 \le \cdots \le k_t$. A *graded spanning certificate for V_t* assigns to every $i < k_j$ $(0 \le j < t)$ and every generator (s, b) a nonzero integer e and a list of pairs (i', c) with all $i' < k_{j+1}$ such that

$$e \cdot g_{s,b} v_i = \sum c\, v_{i'} \qquad \text{as integer vectors.}$$

An *annihilator of order k* for the store is an integer vector f such that every one of the 27^k words $g^*_{s_1,b_1} \cdots g^*_{s_k,b_k} f$ is orthogonal to every stored vector, and $f(z) \neq 0$ for every $z \in \mathbb{Z}_p \setminus \{0, 1\}$.

Proposition 3.4 (soundness of the lower certificate)**.** *If a graded spanning certificate for V_t and an annihilator f of order k for its store both pass, then no point mass δ_z with $z \notin \{0, 1\}$ lies in V_{t+k}; that is, $\neg\mathrm{Cap}_p(t + k)$.*

Proof. Write S_j for the $\mathbb{Q}$-span of $v_0, \dots, v_{k_j - 1}$. $V_0 \subseteq S_0$ since the seeds are δ_0, δ_1. If $V_j \subseteq S_j$ then V_{j+1}, spanned by the $g_{s,b}v$ with $v \in V_j$, is contained in S_{j+1}: $g_{s,b}$ is linear, and for each $i < k_j$ the certificate line gives $g_{s,b}v_i = e^{-1} \sum c\, v_{i'} \in S_{j+1}$. So $V_t \subseteq S_t$, which lies in the span of the store. Every word of length k applied to f is orthogonal to the store, hence kills V_t; by the adjoint identity (Lemma 2.3) and induction on k, f kills V_{t+k}. Finally $f(z) \neq 0$ off $\{0, 1\}$ keeps every such δ_z out of V_{t+k}. □

Remark 3.5 (why the certificate is graded). One might hope to certify $\delta_z \notin V_{d-1}$ with an ungraded object: a subspace $S \ni \delta_0, \delta_1$ closed under all 27 generators and containing no other point mass, or dually a functional f with $f(0) = f(1) = 0$ whose line is stable under all adjoint generators. Either object bounds the *whole* filtration at once, and so cannot exist at a prime where a capture occurs: such an S contains every V_d, hence the captured δ_z; and such an f is orthogonal to V_0 and therefore, by the adjoint identity and induction, to every V_d, so that $f(z) = \langle f, \delta_z \rangle = 0$ for the captured z. Whatever bounds V_{d-1} without bounding V_d must therefore be graded. The grading — t primal levels checked against a store, k adjoint levels checked against the same store — is what keeps both halves finite and small: at $p = 509$ and $p = 521$ the store at $t = 4$ has 58 vectors, the chain certificate is $27 \cdot (2 + 6 + 10 + 22) = 1080$ identities, and the adjoint side costs 27^k convolutions of length p. The ungraded alternative at $p = 521$ would need a 488-vector store and its closure.

The certificates were *found* by a modular search (a row-echelon saturation of the filtration over $\mathbb{F}_q$, $q = 1\,000\,003$, which chooses which generator words to try) and then re-derived exactly in rational arithmetic before being handed to the checker; the search enters no proof. Membership over $\mathbb{F}_q$ neither implies nor is implied by membership over $\mathbb{Q}$, which is the point of the source's paragraph after Lemma 8.3, and is why the search is not the certificate.

4. The Paley family beyond $p = 250$

Theorem 4.1. *Let $p = 521$. Some point mass δ_z with $z \notin \{0, 1\}$ lies in V_7, and no point mass δ_z with $z \notin \{0, 1\}$ lies in V_6. Hence $d(P_{521}) = 7$.*

Proof, and what was computed. Both halves are certificates of Section 3, checked by the compiled evaluator; the passage from the checks to the statement is Propositions 3.2 and 3.4 with Lemma 2.2. *Upper half*: a generator-word tree of seven levels with $4, 4, 12, 36, 108, 322, 33$ nodes (521 stored vectors including the two seeds) and an integer combination of them equal to $m\,\delta_2$, where m is a positive integer of 210 decimal digits; the check is one array equality of length 521. *Lower half*: the same tree truncated at level four (store of 58 vectors, sizes $k_j = 2, 6, 10, 22, 58$), the 1080 chain identities, and one integer functional pushed through $k = 2$ adjoint levels — $27^2 = 729$ words, each checked orthogonal to all 58 stored vectors — and nonzero at every $z \notin \{0, 1\}$; so $t + k = 6$ and $\neg\mathrm{Cap}_{521}(6)$.

The certified captured vertex is $z = 2$ (the formal statement is existential; the certificate exhibits δ_2). □

Theorem 4.2. *Let $p = 509$. Some point mass δ_z with $z \notin \{0, 1\}$ lies in V_6, and no point mass δ_z with $z \notin \{0, 1\}$ lies in V_5. Hence $d(P_{509}) = 6$.*

Proof, and what was computed. As for Theorem 4.1. Upper half: a tree of six levels with 4, 4, 12, 36, 108, 321 nodes (487 stored vectors) and a combination equal to $m\,\delta_2$ with m of 212 decimal digits. Lower half: the level-four truncation (58 vectors, $k_j = 2$, 6, 10, 22, 58), the chain identities, and one functional pushed through $k = 1$ adjoint level (27 words); so $t + k = 5$ and $\neg\mathrm{Cap}_{509}(5)$. □

Theorems 4.1 and 4.2 are the first capture depths computed beyond the source's range, and 7 is the first depth seven on record: the source's Paley row, and its whole table, stop at 6.

Theorem 4.3. *Let $e(p) = 1, 2, 3, 4, 5, 6$ for $p = 5$; 13; 17; $29 \le p \le 61$; $73 \le p \le 181$; $193 \le p \le 509$ respectively. For every prime $p \equiv 1 \pmod 4$ with $5 \le p \le 509$ — the 45 primes*

$$5, 13, 17, 29, 37, 41, 53, 61, 73, 89, 97, 101, 109, 113, 137, 149, 157, 173, 181, 193, 197, 229, 233, 241,$$
$$257, 269, 277, 281, 293, 313, 317, 337, 349, 353, 373, 389, 397, 401, 409, 421, 433, 449, 457, 461, 509$$

— some point mass δ_z with $z \notin \{0, 1\}$ lies in $V_{e(p)}$. In particular $d(P_p) \le 6$ for each of the 21 primes $257 \le p \le 509$, and $d(P_p) \le e(p)$, the entry of the source's Paley row, for each of the 24 primes $p \le 241$.

Proof, and what was computed. Forty-five upper certificates (Definition 3.1), one per prime, each an $e(p)$-level generator-word tree with a combination equal to $m_p\delta_2$, $m_p \neq 0$, checked as one array equality of length p; Proposition 3.2 converts each into $\mathrm{Cap}_p(e(p))$. For $p \le 241$ the level $e(p)$ is the tabulated depth, so the certificate re-derives the "$\le$" half of Table 1; for the 21 primes past the source's range it is the depth the modular search reports (Remark 6.2), and the certificate makes the upper bound exact. These certificates say nothing about lower bounds: for $89 \le p \le 241$ the lower half of $d(P_p)$ is the source's exact computation, and for $257 \le p \le 461$ it is modular evidence only. □

Theorem 4.4. (i) *The primes p with $5 \le p < 521$ and $p \equiv 1 \pmod 4$ are exactly the 45 primes listed in Theorem 4.3.*

(ii) *For each of those 45 primes, $\mathrm{Cap}_p(6)$; and $\mathrm{Cap}_{521}(7)$ while $\neg\mathrm{Cap}_{521}(6)$.*

Consequently 521 is the least prime $p \equiv 1 \pmod 4$ with $d(P_p) > 6$, and the least with $d(P_p) = 7$.

Proof, and what was computed. (i) is a finite enumeration: the list of $n < 521$ with $5 \le n$, $n \equiv 1 \pmod 4$ and n prime, computed by the evaluator and compared with the displayed list. (ii) is Theorem 4.3 raised to level 6 by Lemma 2.2, together with Theorem 4.1. The whole statement, (i) and (ii) together, is one theorem of the formal development, so the minimality does not rest on the source's table — or on any reading of a table — at any prime. For the last sentence: $d(P_p) \le 6$ for every Paley prime $p < 521$ by (ii) and Lemma 2.2, and $d(P_{521}) = 7$. □

Corollary 4.5 (the Paley row, extended and graded by what certifies it)**.** *For the Paley primes $p \le 521$,*

p	5	13	17	29–61	73–181	193–241	257–509
$d(P_p)$	1	2	3	4	5	6	≤ 6 *(and* $= 6$ *at* 509*)*

$d(P_{521}) = 7$,

where the entries for $p \le 241$ combine Theorem 4.3 with the source's Table 1 (both halves being this paper's for the nine primes of Proposition 5.1), and the entries at 509 and 521 are Theorems 4.2 and 4.1.

5. The published entries reproduced, and the controls

Proposition 5.1. *For $p = 5, 13, 17, 29, 37, 41, 53, 61, 73$ one has $\mathrm{Cap}_p(d_p)$ and $\neg\mathrm{Cap}_p(d_p - 1)$ with $d_p = 1, 2, 3, 4, 4, 4, 4, 5$ respectively; that is, $d(P_p) = d_p$. These are the nine Paley entries of Table 1 of* [1] *with $p \le 73$.*

Proof, and what was computed. Nine two-sided certificates. The upper halves are trees of d_p levels with combinations equal to $m_p\delta_2$, $m_p = 1, 1, 1, 3, 2, 8, 69, 4020, 988$. The lower halves are graded spanning certificates for V_{d_p-1} with store sizes k_j equal to (2); $(2, 6)$; $(2, 6, 9)$; $(2, 6, 9, 17)$; $(2, 6, 10, 21)$ for $p = 37, 41, 53$; $(2, 6, 10, 22)$ for $p = 61$; and $(2, 6, 10, 22, 57)$ for $p = 73$, each with an annihilator of order $k = 0$ (for $p = 5$ the store is the two seeds and the certificate says only that no third point mass lies in V_0). The values themselves are due to [1]; what is ours is the certified two-sided reproduction from the definition of the filtration. □

A certificate format can be vacuous in two ways: a checker that accepts corrupted data, or a lower bound that is not tight. The following controls, all at $p = 29$ where Proposition 5.1 gives $d(P_{29}) = 4$, close both.

Proposition 5.2 (negative controls)**.** *At $p = 29$:*

(i) *changing one entry of the annihilator that certifies $\neg\mathrm{Cap}_{29}(3)$ makes the orthogonality check fail;*
(ii) *that same annihilator is* not *orthogonal to the level-four store (the full generator-word tree of the upper certificate), so the checker would refuse a certificate for $\neg\mathrm{Cap}_{29}(4)$ built from it — as it must, $\mathrm{Cap}_{29}(4)$ being true;*
(iii) *corrupting one coefficient of the graded spanning certificate makes the chain check fail;*
(iv) *the upper certificate with the wrong multiplier ($4\delta_2$ in place of $3\delta_2$) is rejected.*

Proof, and what was computed. Four evaluations of the checkers on modified data, each returning *false* (or, in (iv), an array inequality). Together with $d(P_{29}) = 4$ they refute a too-small claim ($\neg\mathrm{Cap}_{29}(3)$ is certified, $\neg\mathrm{Cap}_{29}(2)$ would be a weaker lower bound) and a too-large one ($\neg\mathrm{Cap}_{29}(4)$ cannot be certified from the same data), and show that neither checker accepts perturbed input. □

6. Outside the formal development

Everything in this section was computed by the modular saturation mentioned in Section 3 (a C program working over $\mathbb{F}_q$, from the definitions: a reduced row-echelon basis of V_j is maintained, the frontier is expanded by the 27 generators, and a capture is detected as a unit row of the echelon form at a coordinate outside $\{0, 1\}$), or by a short exact rational computation of our own. None of it is a theorem of this paper.

Remark 6.1 (Table 1 reproduced; computed, not proved here). The saturation was run for all 214 pairs (p, E) of [1, Table 1] — depth and least captured vertex — under the three moduli $q = 1\,000\,003$, $99\,999\,989$ and $2\,147\,483\,629$, with 214 agreements out of 214 each time, and agrees row by row with the file `depths.json` of the source's archive [2]. This was done before anything beyond $p = 250$ was computed. The source computed its table in exact integer arithmetic; ours is a modular reproduction, so for the 190 non-Paley pairs and for the lower halves at $89 \le p \le 241$ it is a consistency check, not an independent proof.

Remark 6.2 (the Paley row to $p \le 1621$; modular, not exact). The same saturation along the Paley primes gives

p	5	13	17	29–61	73–181	193–509	521–1453	1481–1621
$d(P_p) \bmod q$	1	2	3	4	5	6	7	8

with the three moduli above agreeing on all of $p \le 617$ and the continuation to 1621 run at $q = 1\,000\,003$ alone. In every run the least captured vertex was $z = 2$. These are values of the filtration reduced modulo q; by the source's paragraph after Lemma 8.3 they neither imply nor are implied

by the rational values, and the exactly certified members of the range are exactly those of Section 4. What the modular row adds to the theorems is a prediction: the next transition, $7 \to 8$, sits between 1453 and 1481, and Section 7 prices its exact certification.

Remark 6.3 (the dimension law; observed, not proved). The saturation also prints $\dim_{\mathbb{F}_q} V_j$, and those numbers are the same for every Paley prime large enough for the space not to have saturated:

$$\dim V_j = 2,\ 6,\ 10,\ 22,\ 58,\ 166,\ 490,\ 1462,\ \dots \qquad (j = 0, 1, 2, \dots),$$

that is $\dim V_j = 2 \cdot 3^{j-1} + 4$ for $j \geq 1$, independently of p. The filtration grows by a factor of three per level, so the depth grows like $\log_3 p$, and each transition of the Paley row sits where p passes the corresponding generic dimension:

depth transition	between the Paley primes	generic $\dim V_j$ at the lower level
$1 \to 2$	$5 \to 13$	6
$2 \to 3$	$13 \to 17$	10
$3 \to 4$	$17 \to 29$	22
$4 \to 5$	$61 \to 73$	58
$5 \to 6$	$181 \to 193$	166
$6 \to 7$	$509 \to 521$	490
$7 \to 8$	$1453 \to 1481$	1462

The rule is not "capture as soon as the generic dimension exceeds p": near a transition the dimensions fall below the generic values, and the boundary is decided by arithmetic. At $p = 509$ the level-six space has dimension $487 < 509$ and does contain a point mass; at $p = 521$ it has dimension 488 and does not (and at $p = 1453$ and 1481 the level-seven dimensions are 1444 and 1456, against the generic 1462). The two certified trees of Theorems 4.2 and 4.1 have exactly those sizes — 487 and 488 stored vectors at level six — and their chain certificates prove $\dim_{\mathbb{Q}} V_4 \leq 58$ at both primes. For $p \leq 73$ we recomputed the exact rational dimensions from the definitions: $(2, 5)$ for $p = 5$; $(2, 6, 9, 12, 13)$ for 13; $(2, 6, 9, 14, 17)$ for 17; $(2, 6, 9, 17, 29)$ for 29; $(2, 6, 10, 21, *)$ for $37, 41, 53$; $(2, 6, 10, 22, *)$ for 61; and $(2, 6, 10, 22, 57, 73)$ for 73 — so the small primes deviate from the law before saturating, and the store sizes k_j of the certificates of Proposition 5.1 are exactly these dimensions. We have no proof of the law, and no proof even that $\dim V_j \to \infty$ with j uniformly in p; either would answer the source's question of §9.1 about unbounded depth, and we regard this as the theoretical target the data point at.

7. Cost, and what is not attempted

The times below are measurements on one machine, quoted only to locate the frontier of the method as built. Certificate generation in exact rational arithmetic took about 9 minutes at $p = 521$, about 7 at $p = 509$, and 8 minutes for the 45 upper certificates together (from 3 seconds at $p = 257$ to 46 seconds at $p = 461$). Checking: the two soundness modules 10 and 8 seconds; the small and middle two-sided certificates ($p \leq 73$) 9 and 14 seconds; the controls 11 seconds; the four upper-certificate modules 20, 58, 270 and 213 seconds; $d(P_{509}) = 6$ in 136 seconds and $d(P_{521}) = 7$ in 207 seconds; the minimality theorem, which only assembles the others, 7 seconds. The modular searches: all 214 pairs of Table 1 under three moduli in 4 seconds, the Paley extension to 617 under three moduli in 15 seconds, and the continuation to 1621 in 5 minutes. For the whole run that produced this development, including all elaboration overhead, our run ledger records 84 processor-minutes.

The next depth. The modular row puts the first depth eight at $p = 1481$ (Remark 6.2). The figures in this subsection are extrapolations from the measured rate of the checker; none of the runs they describe was carried out. At that rate (about $2 \cdot 10^6$ term-operations per second) the upper certificate — a tree of 1481 nodes, each a convolution of length p over a class of size 740 — is about 13 minutes; the lower certificate at $t = 4$, $k = 3$ needs $27^3 = 19\,683$ adjoint words and about three hours, past what we allow a single check; at $t = 5$, $k = 2$ the adjoint side is 729 words again (about 7 minutes) but the level-five store has 166 vectors, so the chain certificate grows from about 1 100 to about 4 500

identities with about 170 coefficients each, roughly ten times the data, about 25 minutes of chain checking, and a rational generation pass about $(166/58)^2 \approx 8$ times the $p = 521$ one. So $d(P_{1481}) = 8$ is affordable only at $t = 5$ and only if the data are split across modules; it is priced, not attempted.

Other multiplier groups. Only the Paley case is formalised: the class function is hard-wired to the quadratic residues. Generalising to arbitrary $E \ni -1$ is a one-parameter change (classes from a coset table) and would let the other 190 pairs of Table 1 be certified by the same two shapes of certificate; there is no new mathematics in it. The ideal negative control is also outside the formalised case: for $E = \mathbb{Z}_p^\times$, the complete graph, the source's Remark 5.12 notes that $W(0,1)$ is three-dimensional and nothing outside $\{0,1\}$ is ever captured, and a subspace closed under all adjoint generators and orthogonal to δ_0, δ_1 would certify non-capture at every depth at once. The controls of Proposition 5.2 serve instead.

The harmonic point. The source's Conjecture 9.3 singles out the vertex 2^{-1} and its Remark 8.4 records 2^{-1} captured at minimal depth in all 24 Paley pairs with $p \leq 241$. Our certificates exhibit δ_2 throughout, and the modular search reports only the least captured vertex, not the capture set $Z(p, E)$; certifying $\delta_{2^{-1}} \in V_d$ at 509 and 521 is the same upper certificate with a different target and was not done.

8. Verification

Propositions 3.2 and 3.4, Theorems 4.1, 4.2, 4.3 and 4.4 and Propositions 5.1 and 5.2, together with Lemmas 2.2 and 2.3, are formalised and machine-checked in Lean 4 [7] against Mathlib [8]; their statements are reproduced verbatim in Appendix A. The development has twelve modules and 10 291 lines, almost all of them certificate data (whitespace-separated integers in string literals, parsed by decoders that carry no proof obligation: the checkers are sound for whatever the decoders return). Two modules are the reasoning layer — the filtration as a $\mathbb{Q}$-submodule of $\mathbb{Z}_p \to \mathbb{Q}$, monotonicity, the adjoint identity, a bridge from integer arrays to rational vectors, the two checkers and their soundness theorems — and contain no evaluation; the other ten hold the data and one compiled evaluation (`native_decide`) per check. There are 202 theorem and lemma declarations. Lemma 2.1 is proved on paper only, and Section 6 and the pricing of Section 7 are outside the formal development and labelled where they occur. There is no `sorry`. An axiom audit (`#print axioms` on all 86 declarations that carry a result of this paper) gives the following. Every one of them rests on `propext`, `Quot.sound` and `Classical.choice` and on nothing else beyond the evaluation axioms of its own checks; `sorryAx` occurs nowhere. The two soundness theorems, and with them the whole reasoning layer, use *only* those three. An upper certificate adds two evaluation axioms and a lower one four, so a two-sided depth theorem carries six; the enumeration of the 45 primes is one evaluation; the 45-fold conjunction behind Theorem 4.4 carries $2 \cdot 45 + 6 = 96$, namely two per upper certificate together with the six of $p = 521$; and each control carries one of its own. Every evaluation is a finite integer identity: an array equality, a Boolean checker, or a list-of-primes equality. The repository is private: repository reference to be added at submission.

One engineering fact is worth recording, because it decided whether the computation was affordable at all. The checker's convolution was first written with its summation index in $\mathbb{Z}/p\mathbb{Z}$, and a batch of upper certificates for $137 \leq p \leq 277$ did not finish in eight minutes. For a *variable* p the ring structure on $\mathbb{Z}/p\mathbb{Z}$ is built by recursion on p, so every subtraction, cast and value extraction in the inner loop is a structure projection in the interpreter, and a per-coordinate list allocation compounds it. Rewriting the same convolution over natural-number indices, with one four-line bridge lemma $(x - y) \bmod p = (x + p - y) \bmod p$ for $0 \leq x, y < p$, took that batch to 58 seconds and the $p \leq 73$ module from 59 to 14 seconds — a factor of ten or more, and the difference between the $p = 521$ certificate fitting in 207 seconds and an estimate of about 90 minutes. The transferable rule is not to do arithmetic in $\mathbb{Z}/p\mathbb{Z}$ for variable p inside an evaluated hot loop.

References

[1] M. F. Marashdeh, *Two-basepoint Terwilliger algebras and the quantum symmetry of prime-order circulants*, arXiv:2609.00462v1 [math.OA], submitted 31 August 2026 (the abstract page listed v1 only on 7 September 2026). All statement numbers cited above are those of the e-print as compiled from the arXiv source: Definition 4.1 (two-basepoint Terwilliger algebra and module), Definition 4.2 (basepoint partition), Theorem 4.7 (Criterion; Theorem A of the introduction), Theorem 5.8 (Capture dichotomy; Theorem B), Corollary 5.9 (one point mass suffices), Corollary 5.10 (Chassaniol's criterion is depth one; Corollary C), Remark 5.12 (the complete graph), equation (4) (the convolution filtration), Lemma 5.13, Definition 5.14 (capture depth), Theorem 8.1 (Theorem F), Corollary 8.2, Lemma 8.3, §8.2 "Capture depths", Remark 8.4 (the harmonic point), Conjecture 9.1 (capture conjecture; Conjecture G), the subsection §9.1 "What bounded depth cannot do" (whose number coincides with that of the conjecture only because the two counters are independent), Conjecture 9.3 (harmonic capture), Appendix A and its Table 1.

[2] M. F. Marashdeh, *Verification scripts for: Quantum rigidity of prime-order circulants via two-basepoint Terwilliger algebras*, software, Zenodo, 2026; `doi:10.5281/zenodo.22182428` is the all-versions DOI, and the version consulted here is 1.1, `doi:10.5281/zenodo.22182483`, published 1 September 2026.

[3] T. Banica, J. Bichon and G. Chenevier, *Graphs having no quantum symmetry*, Ann. Inst. Fourier (Grenoble) **57** (2007), no. 3, 955–971; `doi:10.5802/aif.2282`; arXiv:math/0605257.

[4] A. Chassaniol, *Study of quantum symmetries for vertex-transitive graphs using intertwiner spaces*, preprint, arXiv:1904.00455 (2019); no journal version is listed on the abstract page as of September 2026.

[5] S. Schmidt, *On the quantum symmetry of distance-transitive graphs*, Adv. Math. **368** (2020), 107150; `doi:10.1016/j.aim.2020.107150`; arXiv:1906.06537, from whose full text the sentence quoted in §1 is taken.

[6] T. Banica and J. Bichon, *Quantum automorphism groups of vertex-transitive graphs of order* ≤ 11, J. Algebraic Combin. **26** (2007), no. 1, 83–105; `doi:10.1007/s10801-006-0049-9`. Cited here only as the reference to which Schmidt's quoted sentence attributes P_9.

[7] L. de Moura and S. Ullrich, *The Lean 4 theorem prover and programming language*, in: A. Platzer and G. Sutcliffe (eds.), Automated Deduction — CADE 28, Lecture Notes in Computer Science **12699**, Springer, Cham, 2021, pp. 625–635; `doi:10.1007/978-3-030-79876-5_37`.

[8] The mathlib Community, *The Lean mathematical library*, in: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020), ACM, 2020, pp. 367–381; `doi:10.1145/3372885.3373824`.

Appendix A. The formal statements

The Lean statements corresponding to the results above, verbatim (statements only; proofs, docstrings and attributes omitted), preceded by the definitions they use. Here `p` is a natural number with a `NeZero` instance, `ZMod p` is $\mathbb{Z}_p$, `Fin 3` indexes the classes, a block label is a pair in `Fin 3 × Fin 3`, `V p n` is V_n as a $\mathbb{Q}$-submodule of `ZMod p → ℚ`, `Pi.single z 1` is δ_z, and `Captured p n` is $\mathrm{Cap}_p(n)$. Certificate data enter through `Array Int` (integer vectors indexed by `ZMod.val`); `cosetN p s` is the class C_s as a list of residues, `blkTab p` the table of block codes, and `toQ` reads an integer array as an element of `ZMod p → ℚ`. The hypothesis `hsq` of the lower certificate is that -1 is a square, `hcl` that the class lists supplied to the checker are the true classes.

`Core.lean` (the filtration; no evaluation)

```
def cls (x : ZMod p) : Fin 3 :=
  if x = 0 then 0 else if ∃ r : ZMod p, x = r * r then 1 else 2
def blkOf (x : ZMod p) : Fin 3 × Fin 3 := (cls p x, cls p (x - 1))
def univL : List (ZMod p) := (List.range p).map (fun n : ℕ => (n : ZMod p))
def cosetL (s : Fin 3) : List (ZMod p) := (univL p).filter (fun y => cls p y = s)
def cosetN (s : Fin 3) : List ℕ := (cosetL p s).map ZMod.val
def cosetF (s : Fin 3) : Finset (ZMod p) := (cosetL p s).toFinset
def conv (s : Fin 3) (v : ZMod p → ℚ) : ZMod p → ℚ := fun x => ∑ y ∈ cosetF p s, v (x - y)
def maskv (b : Fin 3 × Fin 3) (v : ZMod p → ℚ) : ZMod p → ℚ :=
  fun x => if blkOf p x = b then v x else 0
def genv (s : Fin 3) (b : Fin 3 × Fin 3) (v : ZMod p → ℚ) : ZMod p → ℚ :=
  maskv p b (conv p s v)
def adjv (s : Fin 3) (b : Fin 3 × Fin 3) (v : ZMod p → ℚ) : ZMod p → ℚ :=
  conv p s (maskv p b v)
def genL (s : Fin 3) (b : Fin 3 × Fin 3) : (ZMod p → ℚ) →ₗ[ℚ] (ZMod p → ℚ) :=
  (maskL p b).comp (convL p s)
def V : ℕ → Submodule ℚ (ZMod p → ℚ)
  | 0 => Submodule.span ℚ {Pi.single 0 1, Pi.single 1 1}
  | (n + 1) => ⨆ s : Fin 3, ⨆ b : Fin 3 × Fin 3, (V n).map (genL p s b)
def Captured (n : ℕ) : Prop := ∃ z : ZMod p, z ≠ 0 ∧ z ≠ 1 ∧ Pi.single z (1 : ℚ) ∈ V p n
lemma V_mono {m n : ℕ} (h : m ≤ n) : V p m ≤ V p n
lemma Captured.mono {m n : ℕ} (h : m ≤ n) (hc : Captured p m) : Captured p n
def ip (f v : ZMod p → ℚ) : ℚ := ∑ x : ZMod p, f x * v x
```

```
lemma ip_genv (hsq : ∃ c : ZMod p, (-1 : ZMod p) = c * c) (s : Fin 3) (b : Fin 3 × Fin 3)
    (f v : ZMod p → ℚ) : ip f (genv p s b v) = ip (adjv p s b f) v
def Kills (f : ZMod p → ℚ) (n : ℕ) : Prop := ∀ v ∈ V p n, ip f v = 0
lemma Kills.succ (hsq : ∃ c : ZMod p, (-1 : ZMod p) = c * c) {f : ZMod p → ℚ} {n : ℕ}
    (h : ∀ (s : Fin 3) (b : Fin 3 × Fin 3), Kills (adjv p s b f) n) : Kills f (n + 1)
def toQ (a : Array Int) : ZMod p → ℚ := fun x => ((a.getD x.val 0 : ℤ) : ℚ)
def convA (l : List ℕ) (a : Array Int) : Array Int :=
  Array.ofFn (n := p) fun i : Fin p =>
    l.foldl (fun acc y => acc + a.getD (((i : ℕ) + p - y) % p) 0) 0
def maskA (b : Fin 3 × Fin 3) (bt : Array Nat) (a : Array Int) : Array Int :=
  Array.ofFn (n := p) fun i : Fin p =>
    if bt.getD (i : ℕ) 0 = blkCode b then a.getD (i : ℕ) 0 else 0
def genA (l : List ℕ) (b : Fin 3 × Fin 3) (bt : Array Nat) (a : Array Int) : Array Int :=
  maskA p b bt (convA p l a)
def adjA (l : List ℕ) (b : Fin 3 × Fin 3) (bt : Array Nat) (a : Array Int) : Array Int :=
  convA p l (maskA p b bt a)
def comboA (cs : List (Nat × Int)) (vs : Array (Array Int)) : Array Int :=
  Array.ofFn (n := p) fun i : Fin p =>
    (cs.map (fun c => c.2 * (vs.getD c.1 #[]).getD (i : ℕ) 0)).sum
def scaleDeltaA (m : Int) (z : ZMod p) : Array Int :=
  Array.ofFn (n := p) fun i : Fin p => if (i : ℕ) = z.val then m else 0
def dotA (f a : Array Int) : Int := ∑ i ∈ Finset.range p, f.getD i 0 * a.getD i 0
```

`Cert.lean` (the checkers and their soundness; Propositions 3.2 and 3.4)

```
def spanAll (vs : Array (Array Int)) : Submodule ℚ (ZMod p → ℚ) :=
  Submodule.span ℚ (Set.range fun i : ℕ => toQ p (vs.getD i #[]))
def scaleArr (m : Int) (a : Array Int) : Array Int :=
  Array.ofFn (n := p) fun i : Fin p => m * a.getD (i : ℕ) 0
def seedA : Array (Array Int) := #[scaleDeltaA p 1 0, scaleDeltaA p 1 1]
def stepA (cl : Fin 3 → List ℕ) (bt : Array Nat) (acc : Array (Array Int))
    (spec : List (Nat × Fin 3 × (Fin 3 × Fin 3))) : Array (Array Int) :=
  spec.foldl (fun a t => a.push (genA p (cl t.2.1) t.2.2 bt (acc.getD t.1 #[]))) acc
def buildA (cl : Fin 3 → List ℕ) (bt : Array Nat) :
    List (List (Nat × Fin 3 × (Fin 3 × Fin 3))) → Array (Array Int) → Array (Array Int)
  | [], acc => acc
  | sp :: rest, acc => buildA cl bt rest (stepA p cl bt acc sp)
theorem captured_of_combo {cl : Fin 3 → List ℕ} (hcl : ∀ s, cl s = cosetN p s)
    (specs : List (List (Nat × Fin 3 × (Fin 3 × Fin 3)))) (z : ZMod p) (hz0 : z ≠ 0)
    (hz1 : z ≠ 1) (m : Int) (hm : m ≠ 0) (cs : List (Nat × Int))
    (hid : comboA p cs (buildA p cl (blkTab p) specs (seedA p)) = scaleDeltaA p m z) :
    Captured p specs.length
def entryOK (cl : Fin 3 → List ℕ) (bt : Array Nat) (vs : Array (Array Int)) (k' : ℕ)
    (cert : Array (Int × List (Nat × Int))) (i : ℕ) (s : Fin 3) (b : Fin 3 × Fin 3) : Bool :=
  let c := cert.getD (i * 27 + s.val * 9 + b.1.val * 3 + b.2.val) (0, [])
  (c.1 != 0) && (c.2.all fun x => decide (x.1 < k')) &&
    (scaleArr p c.1 (genA p (cl s) b bt (vs.getD i #[])) == comboA p c.2 vs)
def levelOK (cl : Fin 3 → List ℕ) (bt : Array Nat) (vs : Array (Array Int)) (k k' : ℕ)
    (cert : Array (Int × List (Nat × Int))) : Bool :=
  (List.range k).all fun i =>
    (List.finRange 3).all fun s =>
      (List.finRange 3).all fun b1 =>
        (List.finRange 3).all fun b2 => entryOK p cl bt vs k' cert i s (b1, b2)
def chainOK (cl : Fin 3 → List ℕ) (bt : Array Nat) (vs : Array (Array Int)) :
    List ℕ → List (Array (Int × List (Nat × Int))) → Bool
  | k :: k' :: ks, c :: cs => levelOK p cl bt vs k k' c && chainOK cl bt vs (k' :: ks) cs
  | [_], [] => true
  | _, _ => false
def seedOK (vs : Array (Array Int)) (k : ℕ) : Bool :=
  decide (2 ≤ k) && (vs.getD 0 #[] == scaleDeltaA p 1 0) && (vs.getD 1 #[] == scaleDeltaA p 1 1)
def adjStepL (cl : Fin 3 → List ℕ) (bt : Array Nat) (ws : List (Array Int)) :
    List (Array Int) :=
  ws.flatMap fun a => gensL.map fun g => adjA p (cl g.1) g.2 bt a
def adjIterL (cl : Fin 3 → List ℕ) (bt : Array Nat) : ℕ → List (Array Int) →
    List (Array Int)
  | 0, ws => ws
  | (k + 1), ws => adjIterL cl bt k (adjStepL p cl bt ws)
def annOK (vs : Array (Array Int)) (ws : List (Array Int)) : Bool :=
  ws.all fun a => (List.range vs.size).all fun i => dotA p a (vs.getD i #[]) == 0
def nonzeroOK (f : Array Int) : Bool :=
  (List.range p).all fun i => (i == 0) || (i == 1) || (f.getD i 0 != 0)
theorem not_captured_of_annihilator {cl : Fin 3 → List ℕ}
    (hsq : ∃ c : ZMod p, (-1 : ZMod p) = c * c)
    (hcl : ∀ s, cl s = cosetN p s) (vs : Array (Array Int)) (k₀ : ℕ) (ks : List ℕ)
```

```
    (cs : List (Array (Int × List (Nat × Int)))) (f : Array Int) (k : ℕ)
    (hseed : seedOK p vs k₀ = true)
    (hchain : chainOK p cl (blkTab p) vs (k₀ :: ks) cs = true)
    (hann : annOK p vs (adjIterL p cl (blkTab p) k [f]) = true)
    (hnz : nonzeroOK p f = true) :
    ¬ Captured p (ks.length + k)
```

`Depth521.lean`, `Depth509.lean` (Theorems 4.1 and 4.2; namespaces `P521`, `P509`)

```
theorem captured_521 : Captured 521 7
theorem not_captured_521 : ¬ Captured 521 6
theorem depth_521 : Captured 521 7 ∧ ¬ Captured 521 6
theorem captured_509 : Captured 509 6
theorem not_captured_509 : ¬ Captured 509 5
theorem depth_509 : Captured 509 6 ∧ ¬ Captured 509 5
```

`Small.lean`, `Mid.lean` (Proposition 5.1; namespaces `P5`, …, `P73`; each `depth_p` is the conjunction of the two statements before it)

```
theorem captured_5 : Captured 5 1          theorem not_captured_5 : ¬ Captured 5 0
theorem captured_13 : Captured 13 2        theorem not_captured_13 : ¬ Captured 13 1
theorem captured_17 : Captured 17 3        theorem not_captured_17 : ¬ Captured 17 2
theorem captured_29 : Captured 29 4        theorem not_captured_29 : ¬ Captured 29 3
theorem captured_37 : Captured 37 4        theorem not_captured_37 : ¬ Captured 37 3
theorem captured_41 : Captured 41 4        theorem not_captured_41 : ¬ Captured 41 3
theorem captured_53 : Captured 53 4        theorem not_captured_53 : ¬ Captured 53 3
theorem captured_61 : Captured 61 4        theorem not_captured_61 : ¬ Captured 61 3
theorem captured_73 : Captured 73 5        theorem not_captured_73 : ¬ Captured 73 4
theorem depth_5 : Captured 5 1 ∧ ¬ Captured 5 0
theorem depth_13 : Captured 13 2 ∧ ¬ Captured 13 1
theorem depth_17 : Captured 17 3 ∧ ¬ Captured 17 2
theorem depth_29 : Captured 29 4 ∧ ¬ Captured 29 3
theorem depth_37 : Captured 37 4 ∧ ¬ Captured 37 3
theorem depth_41 : Captured 41 4 ∧ ¬ Captured 41 3
theorem depth_53 : Captured 53 4 ∧ ¬ Captured 53 3
theorem depth_61 : Captured 61 4 ∧ ¬ Captured 61 3
theorem depth_73 : Captured 73 5 ∧ ¬ Captured 73 4
```

`UpperA.lean`–`UpperD.lean` (Theorem 4.3; namespaces `U5`, …, `U509`)

```
theorem captured_u5 : Captured 5 1        theorem captured_u193 : Captured 193 6
theorem captured_u13 : Captured 13 2      theorem captured_u197 : Captured 197 6
theorem captured_u17 : Captured 17 3      theorem captured_u229 : Captured 229 6
theorem captured_u29 : Captured 29 4      theorem captured_u233 : Captured 233 6
theorem captured_u37 : Captured 37 4      theorem captured_u241 : Captured 241 6
theorem captured_u41 : Captured 41 4      theorem captured_u257 : Captured 257 6
theorem captured_u53 : Captured 53 4      theorem captured_u269 : Captured 269 6
theorem captured_u61 : Captured 61 4      theorem captured_u277 : Captured 277 6
theorem captured_u73 : Captured 73 5      theorem captured_u281 : Captured 281 6
theorem captured_u89 : Captured 89 5      theorem captured_u293 : Captured 293 6
theorem captured_u97 : Captured 97 5      theorem captured_u313 : Captured 313 6
theorem captured_u101 : Captured 101 5    theorem captured_u317 : Captured 317 6
theorem captured_u109 : Captured 109 5    theorem captured_u337 : Captured 337 6
theorem captured_u113 : Captured 113 5    theorem captured_u349 : Captured 349 6
theorem captured_u137 : Captured 137 5    theorem captured_u353 : Captured 353 6
theorem captured_u149 : Captured 149 5    theorem captured_u373 : Captured 373 6
theorem captured_u157 : Captured 157 5    theorem captured_u389 : Captured 389 6
theorem captured_u173 : Captured 173 5    theorem captured_u397 : Captured 397 6
theorem captured_u181 : Captured 181 5    theorem captured_u401 : Captured 401 6
                                          theorem captured_u409 : Captured 409 6
                                          theorem captured_u421 : Captured 421 6
                                          theorem captured_u433 : Captured 433 6
                                          theorem captured_u449 : Captured 449 6
                                          theorem captured_u457 : Captured 457 6
                                          theorem captured_u461 : Captured 461 6
                                          theorem captured_u509 : Captured 509 6
```

`Least.lean` (Theorem 4.4)

```
theorem paley_primes_below_521 :
    (List.range 521).filter (fun p => decide (5 ≤ p) && (p % 4 == 1) && decide (Nat.Prime p)) =
      [5, 13, 17, 29, 37, 41, 53, 61, 73, 89, 97, 101, 109, 113, 137, 149, 157, 173, 181, 193,
       197, 229, 233, 241, 257, 269, 277, 281, 293, 313, 317, 337, 349, 353, 373, 389, 397, 401,
       409, 421, 433, 449, 457, 461, 509]
theorem least_paley_depth_seven :
    (Captured 5 6 ∧ Captured 13 6 ∧ Captured 17 6 ∧ Captured 29 6 ∧ Captured 37 6 ∧
      Captured 41 6 ∧ Captured 53 6 ∧ Captured 61 6 ∧ Captured 73 6 ∧ Captured 89 6 ∧
      Captured 97 6 ∧ Captured 101 6 ∧ Captured 109 6 ∧ Captured 113 6 ∧ Captured 137 6 ∧
```

```
      Captured 149 6 ∧ Captured 157 6 ∧ Captured 173 6 ∧ Captured 181 6 ∧ Captured 193 6 ∧
      Captured 197 6 ∧ Captured 229 6 ∧ Captured 233 6 ∧ Captured 241 6 ∧ Captured 257 6 ∧
      Captured 269 6 ∧ Captured 277 6 ∧ Captured 281 6 ∧ Captured 293 6 ∧ Captured 313 6 ∧
      Captured 317 6 ∧ Captured 337 6 ∧ Captured 349 6 ∧ Captured 353 6 ∧ Captured 373 6 ∧
      Captured 389 6 ∧ Captured 397 6 ∧ Captured 401 6 ∧ Captured 409 6 ∧ Captured 421 6 ∧
      Captured 433 6 ∧ Captured 449 6 ∧ Captured 457 6 ∧ Captured 461 6 ∧ Captured 509 6) ∧
      Captured 521 7 ∧ ¬ Captured 521 6
```

(In the source file the list and the conjunction are written one item per line; the line breaks above are the only change.)

`Controls.lean` (Proposition 5.2; namespace `Controls`, with `P29.VS`, `P29.FA`, `P29.KS`, `P29.CS`, `P29.certS0`, `P29.upperC`, `P29.specsU` the $p = 29$ data and `store29 := buildA 29 P29.CL (blkTab 29) P29.specsU (seedA 29)`)

```
theorem corrupt_annihilator_fails :
    annOK 29 P29.VS [P29.FA.set! 5 7] = false
theorem annihilator_fails_one_level_up :
    annOK 29 store29 [P29.FA] = false
theorem corrupt_chain_fails :
    chainOK 29 P29.CL (blkTab 29) P29.VS (2 :: P29.KS)
      (((decodeCert (parseInts P29.certS0)).set! 0 (1, [(0, 5)])) :: P29.CS.drop 1)
      = false
theorem wrong_scalar_fails :
    comboA 29 P29.upperC store29 ≠ scaleDeltaA 29 4 2
```